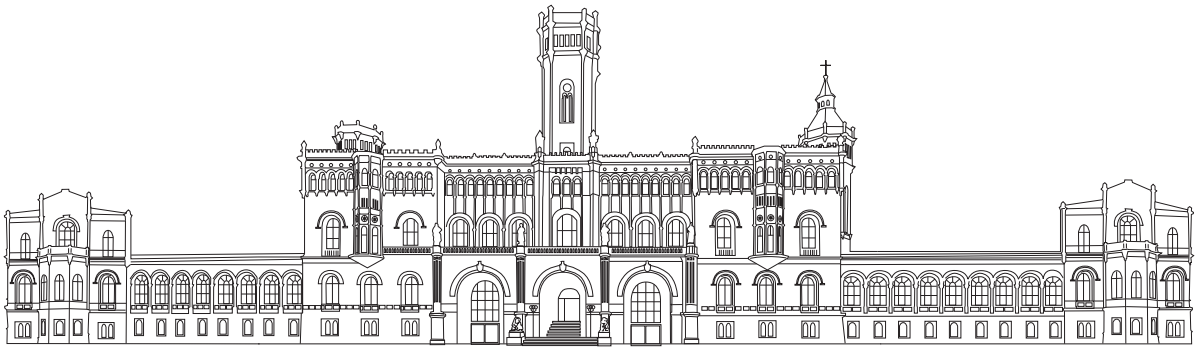


Gottfried Wilhelm Leibniz University Hanover
Institute for Theoretical Physics

Quantum Optimization Algorithms for the Traveling Salesman Problem

In partial fulfillment of the requirement
for the degree of the Master of Science.

Submitted by
Marvin Schwiering,
March 13, 2024.

First Examiner:	Prof. Dr. Tobias J. Osborne
Second Examiner:	Prof. Dr. Klemens Hammerer

Abstract

In this thesis, we build upon the results of [Koß+23], which describes a hardcoded variational quantum algorithm for the Open-Shop Scheduling Problem, containing the Traveling Salesman Problem as a special case. Using a group-theoretical framework, we construct such a hardcoded algorithm that requires only $\Theta(n \log n)$ angles to guarantee finding all possible solutions, which is shown to be asymptotically optimal within the framework. For problem sizes of computational interest, this results in a powerful reduction of parameters. Further, the concrete implementation on quantum hardware is discussed.

Contents

Chapter 1.

Introduction

Optimization — the task of finding a configuration that either minimizes or maximizes a special value — is a fundamental facet of being. In modern civilization, it permeates every aspect of economical, societal, and political life: The financial fate of a company can be decided by its optimization of supply chains, product manufacturing, or search engine results [cf. GY15; AF17; BB18]. Politicians, for better or worse, often make decisions that optimize their country's GDP, their party's election results, or their personal payouts from lobbyists [cf. CG07; Yac15; GO21; Mor24]. Optimization can save energy, time, and lives [cf. Con+13; ZCL15; WZ22]. Indeed, one can contend for optimization to predate civilization for billions of years. Evolution can be understood extraordinarily well by mathematically modeling it as an optimization problem [cf. Pap+23, p. 1; Gre23].[1] More fundamentally even, optimization can be argued to crucially seep through the fabrics of nature itself: The principle of stationary action, from which for example the laws of classical mechanics, quantum mechanics, and general relativity can be derived [cf. Str15, pp. 83–84; SN20, pp. 115–118; Str13, pp. 84–87], postulates that all trajectories realizable in our universe are exactly those whose 'action' is stationary[2] — that is: locally minimized or maximized. Look around you close enough, and you will see the fingerprints of optimization problems everywhere.[3]

One of the field's most illustrious examples is the Traveling Salesman Problem (TSP). Simplistically, when given a list of cities and the distances between them, it asks to find the shortest path that visits each city exactly once before returning to the origin. Its premise is easy, deceptively so: The Traveling Salesman Problem turns out to be one of the most studied optimization problems in history [App+06; MN22, p. 37], and the question of whether there exists an efficient algorithm to solve it is equivalent to determining

[1] Interestingly, evolutions performs so well at improvement that its relation to optimization has become a two-way street: large classes of optimization algorithms, called evolutionary algorithms or genetic algorithms, work by mimicking the mechanisms of evolution [cf. SD08; YG10; SMY02]. This is part of a broader connection between biology and computer science, most notably continued in Machine Learning [cf. Arb02; Nie15; Kra23].

[2] Mathematically speaking, the task of finding stationary trajectories is the problem of finding the optimum of a function $F : \{f : \mathbb{R}^m \to \mathbb{R}^n\} \to \mathbb{R}$ and belongs to the realm of variational calculus. This is analogous but not entirely identical to finding the optimum of a function $F : \mathbb{R}^n \to \mathbb{R}$, home in the realm of multidimensional calculus. Interestingly, neural networks and their training can be understood as a method that enables solving the former by mapping it to the latter.

[3] In admiration of the enthralling writing of Ananyo Bhattacharya [Bha21, p. xiv].

whether $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$, widely considered one of the greatest unsolved problems in mathematics and computer science [For09, p. 80; NR16, pp. 1–4; NC10, p. 40]. Moreover, its applications range almost as wide as those of optimization in general. Beyond logistics, the Traveling Salesman Problem finds use in molecular biology, astronomy, crystallography, and electrical engineering — amongst others [cf. Iva00; Hit+03; BMB01; BS89; Mag86; App+06, pp. 59–80].

As technology progresses, the catalog of tools available to developers, engineers, and designers to tackle such problems grows. For computation in particular, one of this decade's most prominent advances is the emergence of quantum computers [cf. Aru+19; Ach+23; Cas23; Bao+23; HLC23]. The field arose from pioneering work in the 1980s and 1990s [Ben80; Fey82; Deu85; Sho94; Gro96]: Consider a search space $\mathcal{S}$ of n elements $x \in \mathcal{S}$. Provided access to a map $f : \mathcal{S} \to \{0, 1\}$, the task is to find the unique element of $\mathcal{S}$ satisfying $f(x) = 1$. This is akin to being tasked to find a ball hidden under one of n cups while being allowed to check under one cup at a time. If no further information is given about $\mathcal{S}$ or f (i.e. the cups), there exists no better classical strategy than to evaluate $f(x)$ for random solutions $x \in \mathcal{S}$ not yet inspected. In the best case, one gets lucky and identifies the solution with the first attempt; in the worst case, all possible solutions have to be examined before finding the correct one. On average, one will need $n/2$ guesses — that is: the average number of function evaluations required by the brute-force algorithm finding the solution scales linearly with n. We abbreviate this fact by saying that the algorithm's runtime is $O(n)$. Arguably, one of the key insights that shaped the field of quantum optimization was the seminal realization of Lov Grover that quantum mechanics allows for an algorithm that is faster than that [Gro96]. The quantum search algorithm, commonly also referenced by the name of its inventor, has both a best- and worst-case runtime (as measured by the number of function evaluations) of $O(\sqrt{n})$, which is quadratically faster than the classically optimal brute-force algorithm. Putting it in terms of our previous analogy: Quantum mechanics allows for a way to determine under which cup the ball is hidden without *ever* having to check under all cups.[4]

Sadly, the stunning generality of the quantum search algorithm is also its downside. Virtually all problems of scientific or economic interest possess structure that can be exploited to design algorithms whose performance far exceeds that of a brute force algorithm.[5] Yet, Grover's insights helped spark the search for other search and optimization algorithms relying on the distinctive behavior of quantum mechanical information

[4] While this analogy certainly has its flaws, I do think it conveys well how incredibly surprising the discovery was (and still is). For instance, Michael Nielsen recalls his initial encounter with it as follows: "When I first heard about the quantum search algorithm I thought it sounded impossible. I just couldn't imagine any way it could be true" [MN19a].

[5] A notable exception with economic implications is bitcoin mining, which involves searching inputs k to a cryptographic hash function h whose hash values $h(k)$ satisfy certain criteria. Cryptographic hash functions are designed with the intent of allowing no methodology for determining the key k yielding a given hash $h(k)$ faster than brute-force search. Cf. [Sat20, pp. 291–292].

and operations. In 2000, the introduction of the Quantum Adiabatic Algorithm by Edward Farhi et al. provided another way to solve search problems using inherently quantum mechanical principles [Far+00]. More than a decade later, the same authors proposed the Quantum Approximate Optimization Algorithm, which can be understood as a Trotterized variant of the adiabatic algorithm. Along with results from Alberto Peruzzo et al. [Per+14], it spawned the investigation of so-called variational quantum algorithms (VQAs) [cf. Cer+21]. These algorithms constitute a broad and active field of research that this thesis can be considered a part of. Of increasing interest also became the search for quantum speed-ups of preexisting and more refined classical algorithms. Quantum walks were shown to be quadratically [cf. NV00; Aha+01], at times even exponentially faster than their classical counterparts [cf. CFG02; Kem05]. A quantum backtracking algorithm achieving an almost-quadratic speed-up over its classical counterpart was proposed by Ashley Montanaro [Mon16b], motivating numerous refinements, as well as generalizations to quantum branch-and-bound algorithms [cf. AK17; AGJ19; Mon20; Cha+22]. Subsequently, Andris Ambainis et al. proposed a quantum variant of a dynamic programming algorithm [Amb+19]. Its application to tackling the Traveling Salesman Problems yields an algorithm solving it in time $\tilde{O}(1.728^n)$, which is strictly better than the best known classical algorithm [cf. Abb+23, p. 24].[6]

Overview and Contributions of This Thesis

In this thesis, we build upon the results of Gereon Koßmann et al. [Koß+23], which describe a hardcoded variational quantum algorithm for the Open-Shop Scheduling Problem, containing the Traveling Salesman Problem as a special case.[7]

(i) In Chapter 2, we detail the mathematical theories that underpin this work. After fixing general notations (Section 2.1), an introduction to the theory of groups is provided, and results later built upon are established (Section 2.2). Finally, elementary concepts of graph theory are examined (Section 2.3).

(ii) In Chapter 3, fundamental notions of optimization problems (Section 3.1) and the Traveling Salesman Problem (Section 3.2) are reviewed, specifically with respect to quantum computational considerations. Then, we reexamine the analysis of the feasibility-preserving structure of the TSP's encoding by [Koß+23], upon which the hardcoded VQA is based (Section 3.3).

(iii) In Chapter 4, a summary of quantum computational objects and ideas later required is provided (Section 4.1), followed by an inspection of how the discussed group-theoretic considerations can be mapped to quantum algorithms (Section 4.2).

[6] $\tilde{O}(g(n))$ hides factors polynomial in n.

[7] While this thesis focuses on the Traveling Salesman Problem, the improvements on [Koß+23] actually all hold for the Open-Shop Scheduling Problem as well.

(iv) In Chapter 5, we propose the following improvements to the work of [Koß+23]:

- We construct a family of hardcoded mixing unitaries requiring only $O(n \log n)$ parameters to reach all feasible solutions (Definition 5.12 and Theorem 5.16), which is an almost-quadratic reduction[8] of the $O(n^2)$ parameters required[9] for the mixing family presented in [Koß+23].
- We prove that all families of hardcoded mixing unitaries constructed within the ansatz class of [Koß+23] will also require at least $O(n \log n)$ parameters, i.e. we have exhausted the considered mathematical framework by finding its asymptotically optimal solution for the construction of such families (Proposition 5.8 and Theorem 5.17).
- We discuss a technical error made by [Koß+23] in the transfer of mathematical objects to quantum circuits and outline a way of resolving it (Theorem 5.25).

These improvements were, in particular, made possible by contributions of Benjamin Sambale in personal correspondence about the problem; please refer to Chapter 5 for a proper acknowledgment thereof.

[8] $(n \log n)^{2-\epsilon} \in O(n^2)$ for every arbitrarily small $\epsilon > 0$. That is because $\lim_{n\to\infty} (\log n)^{2-\epsilon}/n^{\epsilon} = 0$ implies $(\log n)^{2-\epsilon} \in O(n^{\epsilon})$ which itself implies $(n \log n)^{2-\epsilon} = n^{2-\epsilon} \log^{2-\epsilon}(n) \in O(n^{2-\epsilon} n^{\epsilon}) = O(n^2)$.

[9] As of the writing of the thesis, the current version of [Koß+23] claims that $n(n-1)^2/2$ parameters are required by their ansatz. However, an absolutely minimal adjustment of their argument yields a family of mixing unitaries with $n(n-1)/2 \in O(n^2)$ parameters (cf. Remark 5.3). Due to that, I personally deem the latter a fairer point of comparison.

Mathematical Prelude: Selected Tools

In this chapter, we plan to establish the mathematical foundation on which this work rests. Section 2.1 begins by consolidating notations of the general mathematical framework. While this step is imperative, it can also be dull at times. Which is why skimming this section and returning to it upon potential confusion later on is advisable. Section 2.2 lies at the heart of this chapter, and represents a self-contained introduction to group theory. All results of this thesis that can be considered novel are group-theoretic ones that have been translated to quantum computation, and all statements proven in that section will be used in subsequent parts of this thesis. Finally, Section 2.3 concludes this chapter by presenting a swift introduction to the notion of graphs, as both the Traveling Salesman Problem and the preceding work of [Koß+23; Koß22] are inherently graph-theoretic in nature.

All concepts, ideas, and conclusions in this chapter stem from mature and established theories, and can be found in standard textbooks on linear algebra [Wal21; Wal22; NC10, pp. 61–79], groups [Rob96; Sag01; Sam17], and graphs [Die17; HHM08].

2.1. General Notations

As usual, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ denote, respectively, the set of natural, integer, rational, real, and complex numbers (with $0 \notin \mathbb{N}$); further, $\mathbb{K}$ is defined to be an element of $\{\mathbb{R}, \mathbb{C}\}$. Occasionally, for $n \in \mathbb{N}$, the case $n = 1$ will be referred to as trivial. For $m, n \in \mathbb{R}$, $[m..n]$ denotes the integer interval from m to n, i.e. $[m..n] = [m, n] \cap \mathbb{Z}$. If $m \in \mathbb{N}$ is trivial, $[n]$ is used as a shorthand notation for $[m..n] = [1..n]$.

Arbitrary sets will usually be denoted by M or N. If M and N are such arbitrary sets, then $\mathrm{Map}(M, N)$ designates the set of all maps from M to N and $\mathrm{Bij}(M, N)$ designates the set of all bijective maps from M to N. In the event that $M = N$, $\mathrm{Map}(M, N)$ and $\mathrm{Bij}(M, N)$ are abbreviated by $\mathrm{Map}(M)$ and $\mathrm{Bij}(M)$. Customarily, δ_{ij} denotes the Kronecker-delta, i.e. the map $\delta : M \times M \to \{0, 1\}, (i, j) \mapsto \delta_{ij}$ satisfying $\delta_{ij} = 1$ if and only if $i = j$.

The power set of a set M, denoted $\mathcal{P}(M)$, is the set of all subsets of M. A family of sets over M is a subset of the power set of M; it is called pairwise disjoint if any two

of its elements are either equal or disjoint. Furthermore, a partition of M is a family of sets P over M that (i) is pairwise disjoint, (ii) does not contain the empty set, and (iii) whose union is the set M. An element of a partition is called a block. Provided M is non-empty and finite, the size of a partition P is the number $|P| \in \mathbb{N}$ and the minimum block size of P is the number $\min\{|B| : B \in P\} \in \mathbb{N}$.

In continuation, a partial order $\leq$ on a set M is a relation on M that is (i) reflexive ($m \leq m$ holds for all $m \in M$), (ii) transitive ($m_1 \leq m_2$ and $m_2 \leq m_3$ imply $m_1 \leq m_3$ for all $m_1, m_2, m_3 \in M$), and (iii) antisymmetric (if $m_1 \leq m_2$ and $m_2 \leq m_1$, then $m_1 = m_2$ for all $m_1, m_2 \in M$). Additionally, a total order $\leq$ on M is a partial order that is (iv) total (for all $m_1, m_2 \in M$, $m_1 \leq m_2$ or $m_2 \leq m_1$ holds). If $\leq$ is a partial (or total) order on M, then the tuple $(M, \leq)$ is called a partially (or totally) ordered set.

Moreover, for a set M, a finite sequence of length $n \in \mathbb{N}$ is formally defined as a map $x : [n] \to M$ and an infinite sequence is formally defined as a map $x : \mathbb{N} \to M$. But of course, these maps are associated with the ('infinite') tuple $(x_1, x_2, \dots)$ where $x_i = x(i)$, and sequences will be denoted by $(x_i)_{i\in[n]} \subset M$ or $(x_i)_{i\in\mathbb{N}} \subset M$. If $(x_i)_{i\in[n_1]} \subset M$ and $(y_i)_{i\in[n_2]} \subset N$ are two finite sequences, then $(x_i)^\frown(y_i)$ signifies their concatenation given by $(x_1, \dots, x_{n_1}, y_1, \dots, y_{n_2}) \subset M \cup N$.

A piece of non-standard nomenclature is highlighted in the following ...

Definition 2.1 (Enumeration Functions)
An **enumeration function** of a finite set M is a bijection $\mathcal{E} : M \to [|M|]$.

As established, if V and W are vector spaces over the same field F, then $\otimes$ denotes their tensor product, as well as the tensor product of vectors $v \in V$ and $w \in W$. Comparably, if $M_1 \in \mathrm{Mat}(m, n; F)$ and $M_2 \in \mathrm{Mat}(m, n; F)$ are matrices of the same shape $m \times n$ defined over the same field F, then $\odot$ denotes their Hadamard product, as well as the Hadamard product of vectors $v \in F^n \cong \mathrm{Mat}(n, 1; F)$ and $w \in F^n \cong \mathrm{Mat}(n, 1; F)$. If $p \in \mathbb{R}_{\geq 1}$, the p-norm $\|\cdot\|_p$ of $v = \sum_{i=1}^n v_i e_i \in \mathbb{K}^n$ is given by

$$\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p}$$

where e_i is the i-th canonical basis vector

$$e_i = (\underbrace{0, \dots, 0}_{i-1 \text{ times}}, 1, \overbrace{0, \dots, 0}^{n-i \text{ times}}) \in \mathbb{K}^n.$$

Further, if V and W are vector spaces, then $\mathcal{L}(V, W)$ denotes the set of linear operators from V to W and $\mathrm{GL}(V, W)$ denotes the set of invertible linear operators from V to W, abbreviated $\mathcal{L}(V)$ and $\mathrm{GL}(V)$ respectively if $V = W$. If $\mathcal{H}$ is a Hilbert space, then $U(\mathcal{H})$ denotes the set of unitary operators on $\mathcal{H}$ and $\mathrm{Herm}(\mathcal{H})$ denotes the set of Hermitian operators on $\mathcal{H}$. $\mathrm{Herm}_{\geq 0}(\mathcal{H})$ is the subset of Hermitian operators that is component-wise

positive. Of the many algebraic structures that can be defined upon these sets, two are of particular importance on the following: Both $\mathrm{GL}(V)$ and $U(\mathcal{H})$ form groups with respect to the composition of operators, and, by abuse of notation, $\mathrm{GL}(V)$ and $U(\mathcal{H})$ will interchangeably denote both the sets and the associated groups.

Two important but non-standard notations are again highlighted:

Definition 2.2 (Bits and Qubits)
Let $n \in \mathbb{N}$.

(i) The n-**bit space** is the set $\mathbb{b}^n$ where $\mathbb{b} = \{0, 1\}$.

(ii) The n-**qubit space** is the Hilbert space $\mathfrak{q}^{\otimes n}$ where $\mathfrak{q} \cong \mathbb{C}^2$.

An element of the n-bit space is called a bitstring. If $I \subset [n]$, then b_I denotes the bitstring of length n whose i-th component b_i satisfies $b_i = 1$ if and only if $i \in I$. The computational basis of the 1-qubit space is a fixed orthonormal basis whose vectors are denoted by $|0\rangle$ and $|1\rangle$; for the n-qubit space, the computational basis is the n-fold tensor product of $\{|0\rangle, |1\rangle\}$. As is common in quantum mechanics and quantum computation, tensor products are occasionally omitted, particularly for tensor products of vectors where their omission is most obvious. For the computational basis in particular, we write the tensor product of multiple vectors as one vector whose label lists the labels of the respective factors. For instance, the computational basis vector $|0\rangle \otimes |1\rangle \otimes |0\rangle$ of $\mathfrak{q}^{\otimes 3}$ is alternatively written as $|0\rangle |1\rangle |0\rangle$ or $|010\rangle$. Important to remember at this point it that there exists a canonical map from the n-bit space to the computational basis of the n-qubit space, given by $b \mapsto |b\rangle$. Or, put differently, the n-bit space acts as an index set for the computational basis of the n-qubit space, i.e. $\{|0\rangle, |1\rangle\}^{\otimes n} = \{|b\rangle : b \in \mathbb{b}^n\}$. A coordinate subspace of $\mathfrak{q}^{\otimes n}$ is a subspace $V = \mathrm{span}\{|b\rangle : b \in B\} \leq \mathfrak{q}^{\otimes n}$ with $B \subset \mathbb{b}^n$. If $B = \varnothing$ or $B = \mathbb{b}^n$, then the coordinate subspace is called trivial. A coordinate subspace V is said to be left invariant by a map $f : \mathfrak{q}^{\otimes n} \to \mathfrak{q}^{\otimes n}$ if $f(V) \subset V$. Finally, two other orthonormal bases $\{|\pm\rangle\}$ and $\{|\pm i\rangle\}$ of $\mathfrak{q}$ are defined as

$$|\pm\rangle = \frac{|0\rangle \pm |1\rangle}{\sqrt{2}} \quad \text{and} \quad |\pm i\rangle = \frac{|0\rangle \pm i\,|1\rangle}{\sqrt{2}}.$$

2.2. Elementary Group Theory

2.2.1. Fundamental Notions

Groups are a simple and elegant yet powerful concept. Their modern definition only dates back as far as the 19th century [Sti10, p. 383], while the underlying concept is, in some sense, as old as numbers themselves. Integers and their addition, as well as reals and their multiplication, are some of the simplest constructs in mathematics that constitute a group. And while plenty of mundane objects of everyday life, such as the

behavior of a clock or the manipulations of a Rubik's cube, are also modeled by them, their application in pure math extends far beyond basic arithmetic. Groups describe or are essential to vector spaces, matrix multiplication, transformations of algebraic objects, properties of differential operators on manifolds [cf. AS68], or the bizarrely complicated proof regarding the simple fact that, for $n > 2$, no three integers satisfy the equation $x^n + y^n = z^n$ — also known as Fermat's Last Theorem [Wil95].

In physics, computer science, and quantum computing, occurrences of groups are found in abundance as well. Conservation laws, such as the conservation of energy of an isolated system, or the corresponding fundamental symmetries of spacetime can best be understood in a group-theoretic language. Regarding material sciences, symmetries of a crystal are frequently most elegantly described using groups [cf. Hil86; Bor12]. Shor's algorithm [Sho94], most notably known to the general public for potentially breaking internet encryption in the future [Mon23; Böc14; Nau23], can be understood as reducing the problem of prime factorization to the hidden subgroup problem — before proposing an efficient method to solve the latter [Mon16a, p. 2].

Our interest is motivated by results of [Koß+23] that connect groups to the construction of quantum algorithms. The majority of Chapter 5 is spent in the search for sequences of group elements that construct quantum algorithms more favorably than the one considered in [Koß+23].

Definition 2.3 (Groups)
A **group** is a tuple $(\mathcal{G}, *)$ where $\mathcal{G}$ is a set and $*$ is a map $* : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ that satisfies the following conditions.

($\mathcal{G}1$) Associativity: For all $g_1, g_2, g_3 \in \mathcal{G}$ it holds that $(g_1 * g_2) * g_3 = g_1 * (g_2 * g_3)$.

($\mathcal{G}2$) Existence of a neutral element: There exists an element $e \in \mathcal{G}$, called the **neutral element** of $\mathcal{G}$, that satisfies $g * e = e * g = g$.

($\mathcal{G}3$) Existence of an inverse element: For all $g \in \mathcal{G}$ there exists an element $g^{-1} \in \mathcal{G}$, called the **inverse element** of g, that satisfies $g * g^{-1} = g^{-1} * g = e$.

A group is called **abelian** if it also satisfies the following condition.

($\mathcal{G}4$) Commutativity: For all $g_1, g_2 \in \mathcal{G}$ it holds that $g_1 * g_2 = g_2 * g_1$.

Plenty of propositions with mathematically trivial yet noteworthy statements exist that we will not reproduce thoroughly here. To state the most obvious example, the neutral element of a group is unique, which justifies calling it *the* neutral element in the first place. (Correspondingly, the inverse of any given element is also unique.) Another fact regularly used tacitly in this paper includes the demands placed on a tuple to be considered a group containing redundancy: It suffices to require that the neutral and inverse element are 'left neutral' and 'left inverse' (i.e. satisfy $e * g = g$ and $g^{-1} * g = e$) or, analogously, 'right neutral' and 'right inverse' — the other direction then

is implied. Lastly, it is noteworthy that the notation shall adhere to a group by $(\mathcal{G}, *)$ before eventually converting to the commonly used abuse of notation of simply referring to a group $(\mathcal{G}, *)$ by $\mathcal{G}$ and to a product $g_1 * g_2$ by $g_1 g_2$.

Definition 2.4 (Involutions)
An **involution of a group** $(\mathcal{G}, *)$ is an element $g \in \mathcal{G}$ satisfying $g * g = e$.

Frequently, an involution is defined as a map $f : M \to M$ on a set M that satisfies $f \circ f = \mathrm{id}_M$. It is to note that Definition 2.4 is a generalization of that: Since any such involution $f : M \to M$ is necessarily a bijection, it is an element of the *symmetric group* on M, which will be introduced in Section 2.2.4. Then, $f \in \mathrm{Sym}(M)$ satisfies the group-theoretic definition if and only if it satisfies the set-theoretic definition.

Definition 2.5 (Product Groups)
The **(direct) product** of two groups $(\mathcal{G}, *)$ and $(\mathcal{K}, \bullet)$ is the group $(\mathcal{L}, \star)$ satisfying $\mathcal{L} = \mathcal{G} \times \mathcal{K}$ and $(g_1, k_1) \star (g_2, k_2) = (g_1 * g_2, k_1 \bullet k_2)$.

2.2.2. Subgroups

Virtually all classes of mathematical objects admit the concept of a subobject of the same class — as well as an investigation of these subobjects. Both groups and graphs are no exception in this regard. The concept of a subgroup is defined in the following, and subsequently two important criteria for proving a subset to be a subgroup are proven in the following.

Definition 2.6 (Subgroups)
A **subgroup of a group** $(\mathcal{G}, *)$ is a group $(\mathcal{K}, \bullet)$ satisfying $\mathcal{K} \subset \mathcal{G}$ and $\bullet$ being the restriction of $*$ to $\mathcal{K} \times \mathcal{K}$. In that case, we write $\mathcal{K} \leq \mathcal{G}$. We denote by $\mathrm{Sub}(\mathcal{G})$ the set of all subgroups of $\mathcal{G}$.[10]

Proposition 2.7 (Subgroup Criteria [cf. KS98, p. 4])
Let $(\mathcal{G}, *)$ be a group. Let $\mathcal{K}_1, \mathcal{K}_2 \subset \mathcal{G}$ be non-empty; further, let $\mathcal{K}_2$ be finite.

(i) $\mathcal{K}_1 \leq \mathcal{G}$ if and only if $k_1 * k_2^{-1} \in \mathcal{K}_1$ for all $k_1, k_2 \in \mathcal{K}_1$.

(ii) $\mathcal{K}_2 \leq \mathcal{G}$ if and only if $k_1 * k_2 \in \mathcal{K}_2$ for all $k_1, k_2 \in \mathcal{K}_2$.

[10] This set exists within ZF set theory [cf. BM75, pp. 52–61] because it can be constructed from the power set of $\mathcal{G}$ (which exists due to the axiom of power set) with the axiom of restricted comprehension (called axiom of subset in [BM75]).

Proof

(i) We only prove the implication $\Leftarrow$ as the converse implication is trivially true. Associativity obviously holds. The existence of a neutral element follows from $e = k * k^{-1} \in \mathcal{K}_1$ for any $k \in \mathcal{K}_1$, which is non-empty by assumption. Closure of the map and the existence of an inverse element follow from $k_1, k_2 \in \mathcal{K}_1$ implying $k_2^{-1} = e * k_2^{-1} \in \mathcal{K}_1$ and therefore also $k_1 * k_2 = k_1 * (k_2^{-1})^{-1} \in \mathcal{K}_1$.

(ii) We again only prove the non-trivial implication: First, note that the left multiplication map $l_{g_1} : \mathcal{G} \to \mathcal{G}, g_2 \mapsto g_1 * g_2$ is bijective for all $g_1 \in \mathcal{G}$ because $l_{g_1}^{-1}$ is given by $l_{g_1^{-1}}$. Now, if $k_1 \in \mathcal{K}_2$, then the restriction of l_{k_1} to $\mathcal{K}_2$, which we denote by $\tilde{l}_{k_1}$, maps from $\mathcal{K}_2$ to $\mathcal{K}_2$ by the assumption that $k_1, k_2 \in \mathcal{K}_2$ implies $k_1 * k_2 \in \mathcal{K}_2$. Further, $\tilde{l}_{k_1}$ is also injective (because any restriction of an injective map is) and surjective (because any injective map from a finite set to a finite set of the same cardinality is). Therefore, $\tilde{l}_{k_1}^{-1}$ exists and, in particular, is equal to the restriction of $l_{k_1}^{-1}$ to $\mathcal{K}_2$.[11] Clearly,

$$e = k_1^{-1} * k_1 = l_{k_1^{-1}}(k_1) = l_{k_1}^{-1}(k_1) = \tilde{l}_{k_1}^{-1}(k_1) \in \mathcal{K}_2.$$

Hence,

$$k_1^{-1} = k_1^{-1} * e = l_{k_1^{-1}}(e) = l_{k_1}^{-1}(e) = \tilde{l}_{k_1}^{-1}(e) \in \mathcal{K}_2.$$

Since $k_1 \in \mathcal{K}_2$ was arbitrary, $k \in \mathcal{K}_2$ always implies $k^{-1} \in \mathcal{K}_2$ for all $k \in \mathcal{K}_2$. Combining this with the initial assumption leads to the conditions of part (i) being fulfilled. Thus, $\mathcal{K}_2 \leq \mathcal{G}$. □

A few fundamental properties of subgroups — the properties that, in fact, motivate the use of '$\leq$' to denote subgroups in the first place — can be summarized in a concise manner:

Proposition 2.8 (Being a Subgroup is a Partial Order)
If $(\mathcal{G}, *)$ is a group, then $(\mathrm{Sub}(\mathcal{G}), \leq)$ is a partially ordered set.

Proof

Obviously, every group is a subgroup of itself because every set is a subset of itself. Analogously, two groups being subgroups of each other implies that they are the same group because two sets being subsets of each other implies that they are the same set. Finally, transitivity holds due to a similar argument. □

[11] Let $k \in \mathcal{K}_2$. Since $\tilde{l}_{k_1}$ is a bijection on $\mathcal{K}_2$, there exists $k' \in \mathcal{K}_2$ such that $k = \tilde{l}_{k_1}(k') = l_{k_1}(k')$. Then, $\tilde{l}_{k_1}^{-1}(k) = \tilde{l}_{k_1}^{-1}(\tilde{l}_{k_1}(k')) = k' = l_{k_1}^{-1}(l_{k_1}(k')) = l_{k_1}^{-1}(k)$. Finally, $l_{k_1}^{-1}|_{\mathcal{K}_2} = \tilde{l}_{k_1}^{-1}$ follows from the fact that $k \in \mathcal{K}_2$ was arbitrary.

2.2.3. Group Morphisms

Definition 2.9 (Group Morphisms)
Let $(\mathcal{G}, *)$ and $(\mathcal{K}, \bullet)$ be groups.

(i) A **group homomorphism** from $(\mathcal{G}, *)$ to $(\mathcal{K}, \bullet)$ is a mapping $\phi : \mathcal{G} \to \mathcal{K}$ such that $\phi(g_1 * g_2) = \phi(g_1) \bullet \phi(g_2)$ for all $g_1, g_2 \in \mathcal{G}$. We denote by $\mathrm{Hom}(\mathcal{G}, \mathcal{K})$ the set of all group homomorphisms from $(\mathcal{G}, *)$ to $(\mathcal{K}, \bullet)$.

(ii) A **group isomorphism** from $(\mathcal{G}, *)$ to $(\mathcal{K}, \bullet)$ is a bijective group homomorphism ϕ from $(\mathcal{G}, *)$ to $(\mathcal{K}, \bullet)$ such that ϕ^{-1} is a group homomorphism from $(\mathcal{K}, \bullet)$ to $(\mathcal{G}, *)$. We denote by $\mathrm{Iso}(\mathcal{G}, \mathcal{K})$ the set of all group isomorphisms from $(\mathcal{G}, *)$ to $(\mathcal{K}, \bullet)$.

(iii) A **group automorphism** of $(\mathcal{G}, *)$ is a group isomorphism from $(\mathcal{G}, *)$ to $(\mathcal{G}, *)$. We denote by $\mathrm{Aut}(\mathcal{G})$ the set of all group automorphisms of $(\mathcal{G}, *)$.

(iv) $(\mathcal{G}, *)$ and $(\mathcal{K}, \bullet)$ are called **isomorphic** if there exists a group isomorphism from $(\mathcal{G}, *)$ to $(\mathcal{K}, \bullet)$. In that case, we write $\mathcal{G} \cong \mathcal{K}$.

In general, the term *homomorphism* denotes a map between mathematical objects of the same type (groups, algebras, categories, etc.) that 'preserves the structure' of that construction. *Isomorphism* then refers to homomorphisms possessing an inverse map that, in turn, is itself a homomorphism. The condition of the inverse being a homomorphism is indeed frequently necessary. [12] Whereas occasionally, requiring its existence is sufficient to imply that. For groups, the latter is the case:

Proposition 2.10 (Bijective Homomorphisms are Isomorphisms)
Let ϕ be a group homomorphism from $(\mathcal{G}, *)$ to $(\mathcal{K}, \bullet)$. The following statements are equivalent:

(i) ϕ is bijective.

(ii) ϕ is a group isomorphism.

Proof
Let ϕ be a bijective homomorphism. We want to show that $\phi^{-1}(k_1 \bullet k_2) = \phi^{-1}(k_1) * \phi^{-1}(k_2)$ for all $k_1, k_2 \in \mathcal{K}$: Since ϕ is bijective, there exist $g_1, g_2 \in \mathcal{G}$ such that $\phi(g_i) = k_i$ and $\phi^{-1}(k_i) = g_i$. Since ϕ is a homomorphism, $\phi(g_1 * g_2) = \phi(g_1) \bullet \phi(g_2)$. Applying ϕ^{-1} to both sides now yields the desired result. □

[12] For example, a topological isomorphism (more often called a homeomorphism) is a continuous map f between topological spaces that possesses an inverse map f^{-1} that also is continuous. f^{-1} being continuous is equivalent to f being an open map, which is, in general, indeed not implied by f being continuous and bijective [cf. Wal14, pp. 22–23].

Crucially, the property of 'preserving the structure' is also preserved when multiple homomorphisms are composed. In other words:

Proposition 2.11 (Composing Homom. Yields Homom.)
If ϕ is a group homomorphism from $(\mathcal{G}, *)$ to $(\mathcal{K}, \bullet)$ and ψ is a group homomorphism from $(\mathcal{K}, \bullet)$ to $(\mathcal{L}, \star)$, then $\psi \circ \phi$ is a group homomorphism from $(\mathcal{G}, *)$ to $(\mathcal{L}, \star)$.

Proof
Clearly, for $g_1, g_2 \in \mathcal{G}$, $\psi(\phi(g_1 * g_2)) = \psi(\phi(g_1) \bullet \phi(g_2)) = \psi(\phi(g_1)) \star \psi(\phi(g_2))$. □

The *kernel* and the *image* of a homomorphism are defined before a set of statements fundamental to the exploration of groups is proven. One should be aware that, when working with different groups $(\mathcal{G}, *)$ and $(\mathcal{K}, \bullet)$, the neutral elements will be denoted by $e_{\mathcal{G}}$ and $e_{\mathcal{K}}$ respectively.

Definition 2.12 (Kernel and Image)
Let ϕ be a homomorphism from $(\mathcal{G}, *)$ to $(\mathcal{K}, \bullet)$.

(i) The **kernel** of ϕ is the set $\ker \phi = \{g \in \mathcal{G} : \phi(g) = e_{\mathcal{K}}\} \subset \mathcal{G}$.

(ii) The **image** of ϕ is the set $\operatorname{Im} \phi = \{\phi(g) \in \mathcal{K} : g \in \mathcal{G}\} \subset \mathcal{K}$.

Proposition 2.13 (Fundamental Statements About Homomorphisms)
Let ϕ be a homomorphism from $(\mathcal{G}, *)$ to $(\mathcal{K}, \bullet)$. Further, let $n \in \mathbb{N}$ and $\mathcal{G}' \subset \mathcal{G}$.

(i) If $g_1, \ldots, g_n \in \mathcal{G}$ and $z_1, \ldots, z_n \in [\text{-}1 \,..\, 1]$, then $\phi\Big(\prod_{i=1}^{n} g_i^{z_i}\Big) = \prod_{i=1}^{n} \phi(g_i)^{z_i}$.

(ii) If $g \in \mathcal{G}$ is an involution, then $\phi(g) \in \mathcal{K}$ is an involution.

(iii) $\ker \phi \leq \mathcal{G}$ and $\operatorname{Im} \phi \leq \mathcal{K}$.

(iv) If $\mathcal{G}' \leq \mathcal{G}$, then $\phi(\mathcal{G}') \leq \mathcal{K}$.

Proof ([Cf. Rob96, pp. 18, 20])

(i) Defining $h_i = g_i^{z_i}$ for all $i \in [n]$ yields $\phi(g_1^{z_1} * \ldots * g_n^{z_n}) = \phi(h_1 * \ldots * h_n) = \phi(g_1^{z_1}) \bullet \ldots \bullet \phi(g_n^{z_n})$ by using the defining property of a group homomorphism $n-1$ times for the factors h_1 and $(h_2 * \ldots * h_n)$, h_2 and $(h_3 * \ldots * h_n)$, etc. Furthermore, $\phi(g_1^{z_1}) \bullet \ldots \bullet \phi(g_n^{z_n}) = \phi(g_1)^{z_1} \bullet \ldots \bullet \phi(g_n)^{z_n}$ follows from a simple proof by cases: If $z_i = 0$, then multiplying both sides of $\phi(e_{\mathcal{G}}) = \phi(e_{\mathcal{G}} * e_{\mathcal{G}}) = \phi(e_{\mathcal{G}}) \bullet \phi(e_{\mathcal{G}})$ with $\phi(e_{\mathcal{G}})^{-1}$ implies $\phi(g_i^{z_i}) = \phi(e_{\mathcal{G}}) = e_{\mathcal{K}} = \phi(g_i)^{z_i}$. If $z_i = -1$, then $\phi(g_i^{z_i}) = \phi(g_i)^{z_i}$ is implied by $\phi(g^{-1}) \bullet \phi(g) = \phi(g^{-1} * g) = \phi(e_{\mathcal{G}}) = e_{\mathcal{K}}$. If $z_i = 1$, then, more trivially, $\phi(g_i^{z_i}) = \phi(g_i) = \phi(g_i)^{z_i}$.

(ii) If $g \in \mathcal{G}$ satisfies $g * g = e_{\mathcal{G}}$, then $\phi(g) \bullet \phi(g) = \phi(g * g) = \phi(e_{\mathcal{G}}) = e_{\mathcal{K}}$.

(iii) Recalling Proposition 2.7(i), note that $\ker\phi$ is non-empty because any group homomorphism maps the neutral element to the neutral element by part (i). Now, consider that $g_1, g_2 \in \ker\phi$ implies $g_1 * g_2^{-1} \in \ker\phi$ due to $\phi(g_1 * g_2^{-1}) = \phi(g_1 * g_2^{-1}) \bullet e_{\mathcal{K}} = \phi(g_1 * g_2^{-1}) \bullet \phi(g_2) = \phi(g_1 * g_2^{-1} * g_2) = \phi(g_1) = e_{\mathcal{K}}$. Thus, $\ker\phi \leq \mathcal{G}$.

Similarly, let $k_1, k_2 \in \operatorname{Im}\phi$. By definition, there exist $g_1, g_2 \in \mathcal{G}$ such that $\phi(g_i) = k_i$ for $i \in [2]$. Then, $k_1 \bullet k_2^{-1} \in \operatorname{Im}\phi$ follows from $k_1 \bullet k_2^{-1} = \phi(g_1) \bullet \phi(g_2)^{-1} = \phi(g_1) \bullet \phi(g_2^{-1}) = \phi(g_1 * g_2^{-1})$, where we used the fact that any group homomorphism maps the inverse of an element to the inverse of its image by part (i).

(iv) Define the map $\psi : \mathcal{G}' \to \mathcal{K}, \psi(g) = \phi(g)$, then ψ is a group homomorphism from $\mathcal{G}'$ to $\mathcal{K}$ since it satisfies $\psi(g_1 * g_2) = \phi(g_1 * g_2) = \phi(g_1) \bullet \phi(g_2) = \psi(g_1) \bullet \psi(g_2)$ for all $g_1, g_2 \in \mathcal{G}'$. Then, $\phi(\mathcal{G}') = \psi(\mathcal{G}') = \operatorname{Im}\psi \leq \mathcal{K}$ follows from (iii). □

Part (i) of this proposition naturally generalizes to $z_i \in \mathbb{Z}$ by virtue of $n \in \mathbb{N}$ being arbitrary. Additionally, if ϕ is injective, part (ii) implies the converse of its own statement. This is due to the fact that ϕ then is a bijective group homomorphism from $(\mathcal{G}, *)$ to $(\phi(\mathcal{G}), \bullet)$ by part (iii), which, by Proposition 2.10, is an isomorphism; hence, part (ii) also holds for the group homomorphism ϕ^{-1}.

2.2.4. The Symmetric Group

In this subsection, we aim to define a special class of groups, called *symmetric groups*. Within group theory, these are of heightened importance due to multiple factors. Perhaps most importantly, Cayley's theorem states that every group is isomorphic to some subgroup of a symmetric group.[13] Therefore, the study of groups can, in some sense, be reduced to the study of symmetric groups. Moreover, symmetric groups are crucial to the definition of group actions and group representations. These not only represent a cornerstone of group theory. They are studied for their own sake within the realm of representation theory, for example due to their importance in quantum physics. For this thesis, symmetric groups are important because the study of feasibility-preserving operations on optimization problems will naturally reduce to the study of these groups.

[13] While we do not make use of this fundamental result, we deemed it at least worth a footnote to convey how elementary its proof is: As seen in the proof of Proposition 2.7(ii), the left multiplication map l_g is bijective for all $g \in \mathcal{G}$. Therefore, it is a permutation on $\mathcal{G}$ and the map $\phi : g \mapsto l_g$ is a group homomorphism due to $l_{g_1}(l_{g_2}(g_3)) = g_1 * g_2 * g_3 = l_{g_1 * g_2}(g_3)$. With a small effort, it can be demonstrated that l_{g_1} and l_{g_2} coincide if and only if $g_1 = g_2$, i.e. ϕ is injective and therefore bijective onto its image. By Propositions 2.10 and 2.13(iv), ϕ is a isomorphism from $\mathcal{G}$ to $\phi(\mathcal{G}) \leq \operatorname{Sym}(\mathcal{G})$.

Definition and Proposition 2.14 (Symmetric Groups)
Let M be a set and let $n \in \mathbb{N}$.

(i) The **symmetric group on** M is the group $\mathrm{Sym}(M) := (\mathrm{Bij}(M), \circ)$.

(ii) The **symmetric group of order** n is the group $S_n := \mathrm{Sym}([n])$.

(iii) A **permutation** is an element of a symmetric group.

Proof
The mapping $\circ$ is a binary operation on $\mathrm{Bij}(M)$ because the composition of two bijections is again a bijection. The composition of mappings is always associative. The neutral element of $\mathrm{Sym}(M)$ is given by id_M. In addition to that, the inverse of $f \in \mathrm{Bij}(M)$ concerning $\circ$ exists by definition because f is bijective (and, in particular, is always an element of $\mathrm{Bij}(M)$). Hence, $\mathrm{Sym}(M)$ is a group. Since this shows that $\mathrm{Sym}(M)$ is a group for all sets M, it is in particular true for $M = [n]$ for all $n \in \mathbb{N}$. □

Multiple ways of denoting permutations will be of use to us. If the set M is finite, i.e. can be written as $\{m_1, \ldots, m_{|M|}\}$, then a permutation σ on M is fully defined by listing all elements m of M paired with the element $\sigma(m)$ to which it gets mapped, for example in a matrix,

$$\sigma = \begin{pmatrix} m_1 & \ldots & m_{|M|} \\ \sigma(m_1) & \ldots & \sigma(m_{|M|}) \end{pmatrix}.$$

This is called the **two-line notation** of σ. In general, there will not exist a canonical method of numbering the $|M|$ elements of M; however, for certain choices of M such a method might exist. Certainly, for $M = [n]$, it would be an obvious choice to let the number $i \in [n]$ be the i-th element m_i; then, to denote

$$\sigma = \begin{pmatrix} 1 & \ldots & n \\ \sigma(1) & \ldots & \sigma(n) \end{pmatrix},$$

we write

$$\sigma = \begin{pmatrix} \sigma(1) & \ldots & \sigma(n) \end{pmatrix},$$

which is called the **one-line notation** of σ. It remains noteworthy, however, that in certain situations, the two-line notation is useful even for $M = [n]$, as not ordering the elements can be a convenient intermediary step. For instance, the inverse of any permutation σ, e.g.

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix} \in S_4,$$

is obtained simply by switching the rows of its two-line notation, i.e.

$$\sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 3 & 4 & 2 \\ 1 & 2 & 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix}.$$

The third, and possibly most valuable notation, is the **cycle notation**. By virtue of $[n]$ being finite, for an arbitrary $i \in [n]$, there must exist a $k \in [n]$ such that $\sigma^k(i) = i$. Assuming that multiple such $k \in [n]$ exist, let k denote the smallest; then, the sequence $(i, \sigma(i), \ldots, \sigma^k(i))$ is called a k**-cycle**. Picking one of the $n - k$ elements not contained in the cycle determines a different cycle of σ; repeating this procedure until no such elements remain determines its cycle notation. For instance,

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix} = (1)(2, 3, 4).$$

Cycles of trivial length are frequently omitted, i.e. $(1)(2, 3, 4) = (2, 3, 4)$. As with the two-line notation, the inverse of a permutation given in its cycle notation can be constructed in a simple manner — namely by reserving the order of all its cycles. For example

$$\sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix} = (1)(2, 4, 3).$$

A crucial property of group automorphisms whose proof we delayed is that they themselves form a group with respect to the composition of maps. Specifically, the automorphisms $\mathrm{Aut}(\mathcal{G})$ of a group $\mathcal{G}$ form a subgroup of the symmetric group $\mathrm{Sym}(\mathcal{G})$ on the set of elements of $\mathcal{G}$.

Proposition 2.15 (Group Automorphisms Form a Subgroup)
If $(\mathcal{G}, *)$ is a group, then $(\mathrm{Aut}(\mathcal{G}), \circ) \leq \mathrm{Sym}(\mathcal{G})$.

Proof
First, $\mathrm{Aut}(\mathcal{G}) \subset \mathrm{Sym}(\mathcal{G})$ is trivially true as any automorphism $\phi \in \mathrm{Aut}(\mathcal{G})$ is required to be a bijection $\phi : \mathcal{G} \to \mathcal{G}$. Second, $\mathrm{Aut}(\mathcal{G}) \leq \mathrm{Sym}(\mathcal{G})$ is implied by Proposition 2.7(i) as follows: Let $\phi, \psi \in \mathrm{Aut}(\mathcal{G})$, then $\psi^{-1} \in \mathrm{Aut}(\mathcal{G})$ follows by definition. Due to Proposition 2.11, $\phi \circ \psi^{-1}$ is a bijective group homomorphism from $(\mathcal{G}, *)$ to $(\mathcal{G}, *)$. By Proposition 2.10, $\phi \circ \psi^{-1} \in \mathrm{Aut}(\mathcal{G})$. □

If $\mathcal{G}$ is a group, then $\mathrm{Aut}(\mathcal{G})$ will, by abuse of notation, denote both the set of automorphisms as well as the associated group. Importantly, any bijection f between sets M and N canonically induces an isomorphism between the symmetric groups of those sets:

Definition and Proposition 2.16 (The Induced Isomorphism ϕ_f)
Let M and N be sets and let $f : M \to N$ be a bijection, then

$$\begin{aligned} \phi_f : \mathrm{Sym}(M) &\to \mathrm{Sym}(N) \\ \sigma &\mapsto f \circ \sigma \circ f^{-1} \end{aligned}$$

is a group isomorphism from $\mathrm{Sym}(M)$ to $\mathrm{Sym}(N)$.

Proof
ϕ_f is a homomorphism because

$$f \circ (\sigma_1 \circ \sigma_2) \circ f^{-1} = f \circ \sigma_1 \circ \mathrm{id}_M \circ \sigma_2 \circ f^{-1} = f \circ \sigma_1 \circ f^{-1} \circ f \circ \sigma_2 \circ f^{-1}$$

implies that $\phi_f(\sigma_1\sigma_2) = \phi_f(\sigma_1)\phi_f(\sigma_2)$ holds. Further, ϕ_f is surjective (if $\pi \in \mathrm{Sym}(N)$, then $\sigma = f^{-1} \circ \pi \circ f \in \mathrm{Sym}(M)$ satisfies $\phi_f(\sigma) = \pi$) and injective (if $\phi_f(\sigma_1) = \phi_f(\sigma_2)$, then $f \circ \sigma_1 \circ f^{-1} = f \circ \sigma_2 \circ f^{-1}$ implies $\sigma_1 \circ f^{-1} = \sigma_2 \circ f^{-1}$ and subsequently $\sigma_1 = \sigma_2$). Thus, ϕ_f is a group isomorphism by Proposition 2.10. □

Intuitively, it is obvious that S_n contains $n!$ elements, as the task of counting the number of bijections corresponds to the task of determining the number of ways in which n objects can be assigned to n positions: The first object has n possible positions, the second item has $n-1$ possible positions, etc, resulting in $n \cdot (n-1) \cdot \ldots \cdot 1 = n!$ possibilities.

Definition 2.17 (Transpositions)
Let $n \in \mathbb{N}$ and $i, j \in [n]$.

(i) A **transposition in** S_n is a permutation $\tau_{ij} = (i, j) \in S_n$. We denote by T_n the set of all transpositions in S_n.

(ii) An **adjacency transposition in** S_n is a transposition τ_{ij} satisfying $j = i+1$. We denote by T_n^a the set of all adjacency transpositions in S_n.

Despite, in general, not forming a subgroup of S_n, thereby failing to possess many mathematically desirable features, (adjacency) transpositions are of crucial value. Mathematically speaking, they are the simplest permutations acting in a non-trivial manner, able to be utelized to 'decompose' arbitrary permutations.[14] Physically, adjacency transpositions will later be mapped to quantum gates that act on adjacent quantum bits or quantum registers, which is valuable when working with quantum computers possessing limited qubit connectivity.

We define a special class of subgroups of the symmetric group that will play a role in proving one of the key results of Chapter 5 (Proposition 5.10 and Proposition 5.11 are

[14] S_n itself is the smallest subgroup of S_n that contains T_n or T_n^a, i.e. any permutation can be written as a product of transpositions.

crucially used in the proof of the central Theorem 5.16) before moving on to discussing group actions and group representations.

Definition and Proposition 2.18 (Fixed-Point Subgroups)
Let $n \in \mathbb{N}$.

(i) The **fixed-point subgroup**[15] **of** $F \subset [n]$ is the subgroup

$$S_n^F = \{\sigma \in S_n \ : \ \forall f \in F : \sigma(f) = f\} \leq S_n.$$

(ii) The **fixed-point subgroup of** $f \in [n]$ is the subgroup $S_n^f = S_n^{\{f\}} \leq S_n$.[16]

Proof
According to Proposition 2.7(ii), a finite subset of a group is a subgroup if it is non-empty and closed under multiplication. Obviously, $S_n^F \subset S_n$ is finite (because S_n is) and non-empty (because $\mathrm{id} \in S_n^F$). Furthermore, if $\sigma \in S_n^F$ and $\pi \in S_n^F$, then, for all $f \in F$, one has $(\sigma \circ \pi)(f) = \sigma(\pi(f)) = \sigma(f) = f$, i.e. $\sigma\pi \in S_n^F$. Thus, $S_n^F \leq S_n$. □

2.2.5. Group Actions and Group Representations

Definition 2.19 (Group Actions)
An **action** of a group $\mathcal{G}$ on a set M is a group homomorphism ϕ from $\mathcal{G}$ to $\mathrm{Sym}(M)$. A group action is called **transitive** if, for all $m_1, m_2 \in M$, there exists an element $g \in \mathcal{G}$ satisfying $\phi(g)(m_1) = m_2$.

First of all, one should be aware that the definition here is, at times, also denoted a *left* group action, i.e. there exist more general notions of group actions. Secondly, it is of note that occasionally left group actions are also defined as maps $\phi : \mathcal{G} \times M \to M$ satisfying $\phi(e, m) = m$ and $\phi(g_1, \phi(g_2, m)) = \phi(g_1 * g_2, m)$. Both definitions of group actions induce each other. The advantage of the definition in terms of a group homomorphism is that previously proven statements can readily be applied.[17]

[15] Fixed-point subgroups are a special case of the more general concept of a *stabilizer*, which finds plentiful applications in quantum computing, predominantly in quantum error-correction [cf. Got97].

[16] The possibility of confusing the notations $S_n^f = S_n^{\{f\}}$ and $S_n^f = \underbrace{S_n \times \cdots \times S_n}_{f\text{-times}}$ can be dismissed: Nowhere in this thesis do we use the latter.

[17] The difference between these definitions parallels the difference between two common definitions of vector spaces. Often, a vector space over a field $(F, \oplus, \oslash)$ is defined as a tuple $(V, +, \cdot)$ where $(V, +)$ is an abelian group and $\cdot : F \times V \to V$ is a map satisfying certain properties. Yet, a vector space can also be defined as a tuple $(V, +, \phi)$ where $(V, +)$ is again an abelian group but ϕ is a ring homomorphism from $(F, \oplus, \oslash)$ to $(\mathrm{End}(V), \oplus', \oslash')$. Both definitions of vector spaces induce each other.

Definition 2.20 (Group Representations)
Let $\mathcal{G}$ be a group, V be a vector space, and $\mathcal{H}$ be a Hilbert space.

(i) A **representation** of $\mathcal{G}$ on V is an action ρ of $\mathcal{G}$ on V satisfying

$$\rho(\mathcal{G}) \leq \mathrm{GL}(V).$$

(ii) A **unitary representation** of $\mathcal{G}$ on $\mathcal{H}$ is a representation ρ of $\mathcal{G}$ on $\mathcal{H}$ satisfying

$$\rho(\mathcal{G}) \leq U(\mathcal{H}).$$

Before discussing the significance of these concepts, some technical remarks are considered. Firstly, the general linear group on a vector space is the group of all vector space automorphisms on that space. These maps are, in particular, bijective. Hence, $\mathrm{GL}(V)$ is a subgroup of $\mathrm{Sym}(V)$ and requiring the action ρ, which is a group homomorphism from $\mathcal{G}$ to $\mathrm{Sym}(V)$, to fulfill $\rho(\mathcal{G}) \leq \mathrm{GL}(V)$ is a well-defined condition. For unitary representations, the situation is analogous: as unitary operators are bounded linear operators that satisfy $UU^\dagger = U^\dagger U = \mathbb{I} = \mathrm{id}_{\mathcal{H}}$, they as well are automorphisms on $\mathcal{H}$; as $U(\mathcal{H})$ is a subgroup of $\mathrm{GL}(\mathcal{H})$, $U(\mathcal{H}) \leq \mathrm{Sym}(\mathcal{H})$ follows from Proposition 2.8. Second, by Proposition 2.13(iii), the condition $\rho(\mathcal{G}) \leq \mathrm{GL}(V)$ is just the requirement $\rho(\mathcal{G}) \subset \mathrm{GL}(V)$ in disguise. Correspondingly, $\rho(\mathcal{G}) \leq U(\mathcal{H})$ is also implied by $\rho(\mathcal{G}) \subset U(\mathcal{H})$.

Group representations are important to this work because they allow the mapping of permutations and other group elements to linear operators, which can be utilized in the construction of quantum algorithms. In particular, all unitary operators acting on a Hilbert space of dimension 2^n can be understood as an operation of a quantum computer on n qubits [cf. NC10].

2.3. Elementary Graph Theory

2.3.1. Fundamental Notions

The relation of graph theory and this thesis' work is a peculiar one: On the one hand, the Traveling Salesman Problem is inherently a graph-theoric problem, and the preceding work of [Koß+23; Koß22] crucially uses graphs to connect feasibility-preserving groups of classical problems to the construction of mixing families for quantum algorithms. On the other hand, due to the focus of this work, all graph-theoretic propositions found in this thesis are trival.[18] This dichotomy is the reason why an introduction to the topic is provided, but remains superficial and, at times, even omits proofs.

[18] Besides those of [Koß+23] presented and elaborated upon in Chapter 3.

Definition 2.21 (Set of Undirected Edges and Set of Directed Edges)
Let M be a set. We define the sets $\mathrm{Un}(M)$ and $\mathrm{Di}(M)$ as follows.

$$\mathrm{Un}(M) = \Big\{ \{m_1, m_2\} \ : \ m_1, m_2 \in M \wedge m_1 \neq m_2 \Big\}$$
$$\mathrm{Di}(M) = \Big\{ \ (m_1, m_2) \ : \ m_1, m_2 \in M \wedge m_1 \neq m_2 \Big\}$$

The central definition of this section, unquestionably, is the following:

Definition 2.22 (Graphs, Vertices, and Edges)
Let V be a set.

(i) An **undirected graph** is a tuple (V, E) where $E \subset \mathrm{Un}(V)$ satisfies $V \cap E = \varnothing$.

(ii) A **directed graph** is a tuple (V, E) where $E \subset \mathrm{Di}(V)$ satisfies $V \cap E = \varnothing$.

(iii) A **graph** is a tuple (V, E) that is an undirected or directed graph.

Let $G = (V, E)$ be a graph.

(iv) The set $V(G) := V$ is called the **set of vertices** (or **nodes**) of G and an element $v \in V(G)$ is called a **vertex** (or **node**) of G.

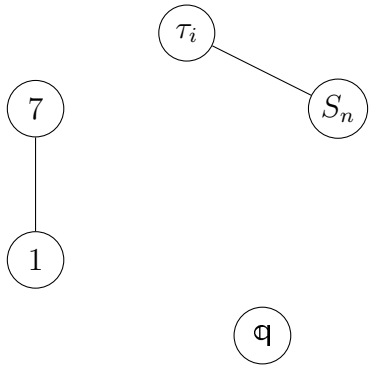
(v) The set $E(G) := E$ is called the **set of edges** of G and an element $e \in E(G)$ is called an **edge** of G. If G is an undirected graph, an edge $e = \{v_1, v_2\}$ is also denoted by v_1v_2; correspondingly, if G is a directed graph, an edge $e = (v_1, v_2)$ is also denoted by v_1v_2.

(vi) The integer $|G| := |V(G)|$ is called the **order** of G and the integer $\|G\| := |E(G)|$ is called the **size** of G.

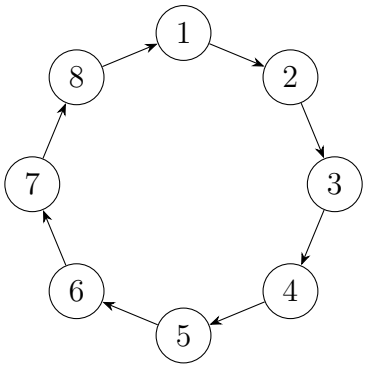
Graph theory is equipped with a dense nomenclature [cf. Die17, pp. 1–21] firmly inspired by its visual character. For example, if $v_1, v_2 \in V$ are vertices of a graph $G = (V, E)$, then $v_1v_2 \in E$ can be expressed by stating that v_1 and v_2 are *neighbours* (or *adjacent*); graphs that can be visualized as 'originating' from a vertex (called a *root*) are called *trees*; various types of subgraphs are denoted as *paths*, *walks*, or *tours*. In an attempt to establish necessary concepts quickly, in a manner compatible with the intended application, the introduction bypasses much of that nomenclature and occasionally define notions in a non-standard way (cf. Definition 2.21 and [Die17, pp. 27–28] or Definition 2.27 and [Die17, p. 3]).

Remark 2.23
Beginning now, all graphs in this thesis are assumed to be finite: the terms *undirected graph*, *directed graph*, and *graph* are to be understood as a shorthand notation for *undirected graph of finite order*, *directed graph of finite order*, and *graph of finite order*.

Graphs are commonly visualized by portraying vertices as dots or labeled circles and edges as lines connecting those dots or circles with directed edges corresponding to arrows. Figure 2.1 provides two examples.



(a) A visualization of the undirected graph $G = (V, E)$ with $V = \{1, 7, S_n, \tau_i, \mathfrak{q}\}$ and $E = \{\{1, 7\}, \{S_n, \tau_i\}\}$.



(b) A visualization of the directed graph $G = (V, E)$ with $V = [8]$ and $E = \{(i, i + 1 \bmod 8) : i \in [8]\}$.

Figure 2.1.: Visualizations of selected graphs.

2.3.2. Subgraphs

Definition 2.24 (Subgraphs, Induced Graphs, and Cycles)
Let $n \in \mathbb{N}$ satisfy $n \geq 3$.

(i) A **cycle** is a graph C satisfying $V(C) = \{v_1, \ldots, v_n\}$ and $v_i v_j \in E(C)$ if and only if $j = (i + 1) \bmod n$.

Let $G = (V, E)$ be a graph.

(ii) A **subgraph** of G is a graph $G' = (V', E')$ satisfying $V' \subset V$ and $E' \subset E$. In that case, we write $G' \subset G$. We denote by $\mathrm{Sub}(G)$ the set of all subgraphs of G.

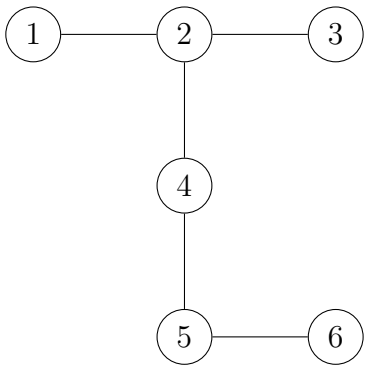
(iii) If G is an undirected graph, then the **induced graph of** $V' \subset V$ is the subgraph $G[V'] = (V, E \cap \mathrm{Un}(V))$; if G is a directed graph, then the **induced graph of** $V' \subset V$ is the subgraph $G[V'] = (V, E \cap \mathrm{Di}(V))$.

(iv) A **cycle in** G is a cycle C satisfying $C \subset G$. We denote by $\mathrm{Cyc}(G)$ the set of all cycles in G. A **Hamiltonian cycle of** G is a cycle C in G satisfying $V(C) = V(G)$. We denote by $\mathrm{Ham}(G)$ the set of all Hamiltonian cycles in G.

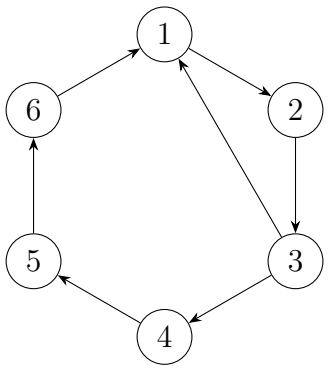
Subgraphs are of enormous importance to the Traveling Salesman Problem, as any

feasible solution to it *is* a subgraph — more specifically a Hamiltonian cycle. Analogously to subgroups, $(\mathrm{Sub}(G), \subset)$ is a partially ordered set for any graph G.

Figure 2.1b depicts the unique directed cycle of size 8. While Figure 2.2a visualizes a graph that neither is a cycle nor contains a cycle, Figure 2.2b shows a graph that is not a cycle, but contains both the non-Hamiltonian cycle $C_1 = ([3], \{(1,2),(2,3),(3,1)\})$ and the Hamiltonian cycle $C_2 = ([6], \{(i, i+1 \bmod 6) : i \in [6]\}$.



(a) A visualization of an undirected graph that contains neither a Hamiltonian nor a non-Hamiltonian cycle.



(b) A visualization of a directed graph that contains one Hamiltonian and one non-Hamiltonian cycle.

Figure 2.2.: Visualizations of selected graphs with and without cycles.

2.3.3. Graph Morphisms

Definition 2.25 (Graph Morphisms)
Let G_1 and G_2 be graphs.

(i) A **graph homomorphism** from G_1 to G_2 is a mapping $\phi : V(G_1) \to V(G_2)$ such that $e' = \phi(v_1)\,\phi(v_2) \in E(G_2)$ for all $e = v_1\,v_2 \in E(G_1)$. We denote by $\mathrm{Hom}(G_1, G_2)$ the set of all graph homomorphisms from G_1 to G_2.

(ii) A **graph isomorphism** from G_1 to G_2 is a bijective graph homomorphism ϕ from G_1 to G_2 such that ϕ^{-1} is a graph homomorphism from G_2 to G_1. We denote by $\mathrm{Iso}(G_1, G_2)$ the set of all graph isomorphisms from G_1 to G_2.

(iii) A **graph automorphism** of G_1 is a graph isomorphism from G_1 to G_1. We denote by $\mathrm{Aut}(G_1)$ the set of all graph automorphisms on G_1.

(iv) G_1 and G_2 are called **isomorphic** if there exists a graph isomorphism from G_1 to G_2. In that case, we write $G_1 \cong G_2$.

As with groups, composing graph homomorphisms again yields a graph homomorphism, and the automorphisms of a graph form a group that is a subgroup of the symmetric group of its set of vertices. In contrast to Section 2.2, however, the underlying structure G from which the group $\mathrm{Aut}(G)$ is constructed is not a group itself — a link between both theories is established.

Proposition 2.26 (Composition of Graph Homomorphisms)

(i) If ϕ is a graph homomorphism from G_1 to G_2 and ψ is a graph homomorphism from G_2 to G_3, then $\psi \circ \phi$ is a graph homomorphism from G_1 to G_3.

(ii) If $G = (V, E)$ is a graph, then $(\mathrm{Aut}(G), \circ) \leq \mathrm{Sym}(V)$.

Proof

(i) Clearly, if $v_1, v_2 \in V(G_1)$ satisfy $v_1v_2 \in E(G_1)$, then this implies that $\phi(v_1), \phi(v_2) \in V(G_2)$ satisfy $\phi(v_1)\phi(v_2) \in E(G_2)$ (because ϕ is a graph homomorphism), which itself implies that $\psi(\phi(v_1)), \psi(\phi(v_2)) \in V(G_3)$ satisfy $\psi(\phi(v_1))\psi(\phi(v_2)) \in E(G_3)$ (because ψ is a graph homomorphism).

(ii) First, $\mathrm{Aut}(G) \subset \mathrm{Sym}(V)$ is trivially true as any automorphism $\phi \in \mathrm{Aut}(G)$ is required to be a bijection $\phi : V \to V$. Second, $\mathrm{Aut}(G) \leq \mathrm{Sym}(V)$ is implied by Remark 2.23 and Proposition 2.7(ii) as follows: Let $\phi_1, \phi_2 \in \mathrm{Aut}(G)$. Then $\phi_1 \circ \phi_2$ is a bijective graph homomorphism from G to G due to part (i). Simultaneously, $\phi_1^{-1}, \phi_2^{-1} \in \mathrm{Aut}(G)$ follows directly from and definition of a graph automorphism and again implies that $(\phi_1 \circ \phi_2)^{-1} = \phi_2^{-1} \circ \phi_1^{-1}$ is a bijective graph homomorphism from G to G by part (i). Hence, $\phi_1 \circ \phi_2 \in \mathrm{Aut}(G)$. □

2.3.4. Complete Graphs and Weighted Graphs

Definition 2.27 (Complete Graphs, Cliques, and Cocliques)

(i) An undirected graph (V, E) is called **complete** if $E = \mathrm{Un}(V)$.

(ii) A directed graph (V, E) is called **complete** if $E = \mathrm{Di}(V)$.

Let G be a graph.

(iii) A **clique** of G is a subset $V' \subset V$ such that $G[V']$ is complete. We denote by $\mathrm{Cl}(G)$ the set of all cliques of G.

(iv) A **coclique** of G is a subset $V' \subset V$ such that $G[V']$ is of size 0. We denote by $\mathrm{Co}(G)$ the set of all cocliques of G.

Proposition 2.28 (Order and Size of a Complete Graph)
(i) If G is a complete undirected graph, then $\|G\| = |G| \cdot (|G| - 1)/2$.

(ii) If G is a complete directed graph, then $\|G\| = |G| \cdot (|G| - 1)$.

Proposition 2.29 (Number of Ham. Cycles of a Complete Graph)
(i) If G is a complete undirected graph, then $|\mathrm{Ham}(G)| = (|G| - 1)!/2$.

(ii) If G is a complete directed graph, then $|\mathrm{Ham}(G)| = (|G| - 1)!$.

Complete graphs not only allow knowledge of the number of edges and Hamiltonian cycles prior to closer inspection, but all complete graphs also naturally admit a canonical graph isomorphism to a standard complete graph of size $|G|$. Much like Cayley's theorem implies that the study of groups can be reduced to the study of symmetric groups, a sequence of propositions and accompanying discussion in Chapter 3 will argue that the solving of Traveling Salesman Problems for arbitrary graphs can be reduced to the solving of that problem for standard complete graphs.

Definition 2.30 (Standard Complete Graphs)
Let $n \in \mathbb{N}$.

(i) The **complete undirected graph of order** n is the graph $K_n^u = (V, E)$ with $V = [n]$ and $E = \mathrm{Un}(V)$.

(ii) The **complete directed graph of order** n is the graph $K_n^d = (V, E)$ with $V = [n]$ and $E = \mathrm{Di}(V)$.

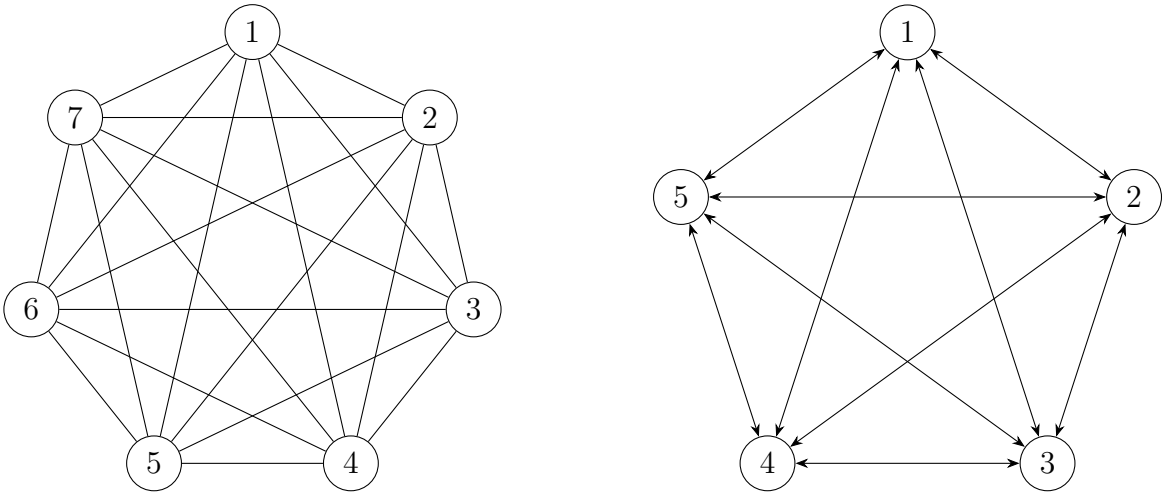
(iii) The **complete graph of order** n is the graph $K_n \in \{K_n^u, K_n^d\}$.

Proposition 2.31 (Complete Graphs are Isomorphic to K_n)
Let G be a complete graph and $\mathcal{E}$ be an enumeration function of $V(G)$.

(i) If G is a undirected graph, $\mathcal{E}$ is a graph isomorphism from G to $K_{|G|}^u$.

(ii) If G is a directed graph, $\mathcal{E}$ is a graph isomorphism from G to $K_{|G|}^d$.

Proof
Enumeration functions are bijective by definition and complete graphs by definition contain no two vertices not connected by an edge, suggesting that the defining property of a graph homomorphism is trivially true. □



(a) A visualization of the graph K_7^u.

(b) A visualization of the graph K_5^d.

Figure 2.3.: Visualizations of selected complete graphs.

Definition 2.32 (Weighted Graphs)
Let M be a set.

(i) A **M-weighted graph** G is a tuple (V, E, w) where (V, E) is a graph and w is a mapping $w : E \to M$. A **weighted graph** is an $\mathbb{R}_{\geq 0}$-weighted graph.

Let $G = (V, E, w)$ be an M-weighted graph.

(ii) The element $w(e) \in M$ is the **weight** of $e \in E(G)$, the mapping $w(G) = w$ is the **weight function** of G, and the mappings $\overline{w}$ and $\overline{\overline{w}}$ defined below are the **generalized weight functions** of G

$$\overline{w} : \mathcal{P}(E) \to \mathbb{R}_{\geq 0}, \quad E' \mapsto \overline{w}(E') = \sum_{e \in E'} w(e),$$
$$\overline{\overline{w}} : \mathrm{Sub}(G) \to \mathbb{R}_{\geq 0}, \quad G' \mapsto \overline{\overline{w}}(G') = \overline{w}\Big(E(G')\Big).$$

Remark 2.33

(i) If $G = (V, E)$ is a graph, then, by abuse of notation, the M-weighted graph (V, E, w) will also be denoted by (G, w).

(ii) If (G, w) is an M-weighted graph and $G = K_n^d$ or $G = K_n^u$, then, for $i, j \in [n]$, the weights $w(\{i, j\})$ or $w((i, j))$ will also be denoted by w_{ij}.

Computational Prelude: Optimization Problems

In this chapter, we provide an analysis of classical optimization problems that will ultimately lead to the construction of quantum algorithms solving them. In Section 3.1, we begin by discussing fundamental notions of the field. Bearing quantum computational considerations in mind, optimization problems and their encodings are introduced, interluded by a discussion of the Bachmann-Landau formalism for investigating an algorithm's time and space complexities. In Section 3.2, we formally define the Traveling Salesman Problem and afterward examine how the set of conceptually considered instances can be reasonably restricted. Based on that, the vertex-time encoding of the Traveling Salesman Problem is presented. In Section 3.3, we follow the analysis of the classical feasibility-structure of the vertex-time encoding performed by [Koß+23], culminating in the identification of two groups that act transitively on its set of feasible solutions. These will serve as a starting point for building variational quantum algorithms in subsequent chapters.

No other chapter strains our attempt to strike a healthy balance between necessary rigor and efficient pragmatism as much as this one. Admittedly, the nomenclature we develop for optimization problems, their encodings, and the constraint graph are, at times, cumbersome. Nonetheless, we deem it necessary to establish these notions meticulously to place the connection between subsequent group-theoretic results and the Traveling Salesman Problem on a reliable mathematical foundation.

3.1. Optimization Problems

3.1.1. Fundamental Notions

While intuition for what constitutes an optimization problem may be easy to find, the same cannot be said about a generally accepted definition. For instance, Stephen Boyd and Lieven Vandenberghe state that the goal of an optimization problem is to

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq b_i, \quad i = 1, \ldots, m, \end{aligned}$$

where $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and $b = (b_1, \ldots, b_m) \in \mathbb{R}^m$ are vectors and f_i is a map $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i \in [0..m]$ [BV04, p. 1]. An optimization problem would therefore formally be triple $(\{f_i\}_{i\in[0..m]}, x, b)$. As an extreme point of comparison, consider Juraj Hromkovič instead defining it to be a sextuple

$$(\Sigma_I, \Sigma_O, L, \mathcal{M}, \text{cost}, \text{goal}),$$

where Σ_I and Σ_O are alphabets (i.e. finite sets), L is a language over Σ_I (i.e. a subset of $\Sigma_I^* = \bigcup_{k\in\mathbb{N}} \Sigma_I^k$), $\mathcal{M}$ is a map $\mathcal{M} : L \to \mathcal{P}(\Sigma_O^*)$, cost is a map $\text{cost} : \bigsqcup_{x\in L} \mathcal{M}(x) \to \mathbb{R}_{\geq 0}$, and goal is a map goal $\in \{\max, \min\}$ [Hro14, p. 27]. Without delving into too much detail, regarding both of these constructions, we hope to have illustrated exemplary how many general descriptions are motivated by specific use cases. In an attempt to nevertheless discuss common notions in general terms, we aim to provide a definition that conceptually encompasses almost all classes of optimization problems without compromising much on simplicity:[19]

Definition 3.1 (Optimization Problems)

(i) An **optimization problem** is a tuple $\mathcal{O} = (\mathcal{S}, f, \mathrm{g})$, where $\mathcal{S}$ is a set, f is a map $f : \mathcal{S} \to \mathbb{R}$, and g is a map $\mathrm{g} \in \{\min, \max\}$.

Let $\mathcal{O} = (\mathcal{S}, f, \mathrm{g})$ be an optimization problem.

(ii) The set $\mathcal{S}(\mathcal{O}) = \mathcal{S}$ is called the **feasible search space** of $\mathcal{O}$ and an element of $\mathcal{S}$ is called a **feasible solution**; the set $\mathcal{S}_{\text{op}}(\mathcal{O}) = \arg \mathrm{g}_{s\in\mathcal{S}(\mathcal{O})} f$ is called the **optimal solution space** of $\mathcal{O}$ and an element of $\mathcal{S}_{\text{op}}(\mathcal{O})$ is called an **optimal solution**.[20]

(iii) The map f is called the **objective function** of $\mathcal{O}$; the map g is called the **goal function** of $\mathcal{O}$. If $\mathrm{g} = \min$, $\mathcal{O}$ is called a **minimization problem** and f is called the **cost function**; if $\mathrm{g} = \max$, $\mathcal{O}$ is called a **maximization problem** and f is called the **profit function**.

(iv) If $\mathcal{S}$ is finite, then $\mathcal{O}$ is called **finite**; if $\mathcal{S}$ is countable, then $\mathcal{O}$ is called **discrete**; if $\mathcal{S}$ is not countable, then $\mathcal{O}$ is called **continuous**.

[19] The most obvious downside of the definition provided, regards the definition of an optimization problem to be an *instance* of an optimization problem, not a *class* of optimization problems, as e.g. [Hro14, p. 27]. This renders the rigorous arguing about properties of classes of problems, such as NP-completeness, difficult to virtually impossible. We accept that concession, as our focus lies not on advancing complexity theory, but on constructing algorithms that may solve concrete instances.

[20] The term $\arg \mathrm{g}_{s\in\mathcal{S}} f$ is a unusual shorthand notation. We specify:

$$\begin{aligned} \mathrm{g} = \min \quad &\Rightarrow \quad \mathcal{S}_{\text{op}} = \arg\min_{s\in\mathcal{S}} f = \{s_{\text{op}} \in \mathcal{S} : f(s_{\text{op}}) \leq f(s)\ \forall s \in \mathcal{S}\} \\ \mathrm{g} = \max \quad &\Rightarrow \quad \mathcal{S}_{\text{op}} = \arg\max_{s\in\mathcal{S}} f = \{s_{\text{op}} \in \mathcal{S} : f(s_{\text{op}}) \geq f(s)\ \forall s \in \mathcal{S}\} \end{aligned}$$

The three key components to any optimization problem consist of a set of feasible solutions $\mathcal{S}$, a method of assigning every feasible solution $x \in \mathcal{S}$ a value $f(x)$, and the goal to find either the feasible solution with the minimal or maximal value. As an example, consider the maximum independent set problem, which asks to find a coclique V' of a graph G such that all other cocliques of G are of size equal to or smaller than V'. The maximization problem specified by a concrete graph G is given by $(\mathrm{Co}(G), |\cdot|, \max)$, where the profit function $|\cdot|$ is simply $V' \mapsto |V'|$.

Since mapping a real number to its additive inverse is a dual order automorphism on $\mathbb{R}$,[21] any maximization problem $\mathcal{O} = (\mathcal{S}, f, \max)$ induces a minimization problem $\mathcal{O}' = (\mathcal{S}, -f, \min)$ satisfying $\mathcal{S}_{\mathrm{op}}(\mathcal{O}) = \mathcal{S}_{\mathrm{op}}(\mathcal{O}')$ – and vice versa. It therefore suffices to consider either minimization or maximization. Of note is how many authors therefore artificially restrict what they consider to be an optimization problem accordingly [cf. BV04, p. 1; MN22, p. 10; Pun22, p. 2; Koß+23, p. 2].

Finally, any optimization problem has associated with it two tasks of differentiating kinds. Apart from the task of determining an optimal solution, called the search problem, there is the decision problem to consider. Given a threshold $t \in \mathbb{R}$, the decision problem asks to determine whether the optimization problem $\mathcal{O}$ possesses a feasible solution $x \in \mathcal{S}(\mathcal{O})$ such that its objective value is at least as good as t. (Depending on whether $\mathcal{O}$ is a minimization or maximization problem, this respectively denotes $f(x) \leq t$ or $t \leq f(x)$.) While an algorithm attempting to solve the search problem must return an element of $\mathcal{S}(\mathcal{O})$, an algorithm attempting to solve the decision problem must return an element of $\{\text{True}, \text{False}\} \equiv \mathbb{b}$. The algorithms discussed in the second part of this thesis focus on the former, however, the distinction is nevertheless important to outline. For example, NP-completeness is an attribute that can only be associated with decision problems, while referring to the Traveling Salesman Problem as NP-complete we do so concerning its decision problem.

3.1.2. The Bachmann-Landau Notation and Complexity Theory

When the complexity of algorithms increases, the difficulty of evaluating them commonly increases with them. This is especially true for quantum optimization algorithms, as they are designed in hopes of providing speed-ups in comparison to their classical counterparts when utilized for sufficiently large problems. However, the quantum computers with the ability to run these algorithms for such problem sizes do not yet exist. Therefore, evaluation becomes an almost purely theoretical endeavor. In the following, we introduce the Bachmann-Landau Notation (frequently called the Big O notation), a cornerstone for the analysis of algorithms without their execution [cf. Cor+22, pp. 49–63; NC10, pp. 135–138].

[21] I.e. a bijection $\phi : \mathbb{R} \to \mathbb{R}$ s.t. $x \leq y$ implies $\psi(x) \geq \psi(y)$ for $\psi = \phi$ and $\psi = \phi^{-1}$ [cf. CLM12, pp. 8–9].

Definition 3.2 (Bachmann-Landau Notation)
Let $D \subset \mathbb{R}_{\geq 0}$ be unbounded and let f and g be maps from D to $\mathbb{R}$.

(i) $f \in O(g) \quad :\Leftrightarrow \quad \exists c > 0,\, x_0 \in \mathbb{R} \quad \forall x \in D,\, x \geq x_0 :\; f(x) \leq c \cdot g(x)$

(ii) $f \in \Omega(g) \quad :\Leftrightarrow \quad \exists c > 0,\, x_0 \in \mathbb{R} \quad \forall x \in D,\, x \geq x_0 :\; f(x) \geq c \cdot g(x)$

(iii) $f \in \Theta(g) \quad :\Leftrightarrow \quad f \in O(g) \;\wedge\; f \in \Omega(g)$

The fundamental idea behind the Big O notation for the analysis of programs is to quantify the runtime or memory requirements of an algorithm in regards to important quantities – chiefly the 'size' of the problem. Afterwards, the framework is used in order to capture the 'essential' behavior of those requirements as the quantities it depends on grow. For example, suppose an algorithm receives as an argument a list of length $n \in \mathbb{N}$ and performs exactly n^3+n+42 operations before returning a result. Then, the function $f : \mathbb{N} \to \mathbb{N}$ that describes the runtime of that algorithm satisfies $f \in \Theta(n^3)$. $g(n) = n^3$ captures the 'essential behavior' of the algorithm's runtime as the problem size n grows.

Oftentimes it is standard practice to denote the Bachmann-Landau notation by $f = O(g)$. For instance, it is denoted as such in Donald Knuth's foundational "The Art of Computer Programming" [Knu97, p. 108]. While it is generally an ill-advised idea not to take Knuth's recommendation, we here do exactly that and instead write $f \in O(g)$. The associated formalism possesses parallels to both set membership and the manipulation of equations, the entirety of which is captured by neither notation. We aim to avoid incentivizing misleading arithmetic. For example, $x = O(x^3)$ and $x^2 = O(x^3)$, but obviously $x \neq x^2$.[22]

Most often, we are interested in determining an upper bound concerning the requirement of an algorithm, bestowing heightened importance to O over Ω and Θ. A task can be performed *in polynomial time* if there exists an algorithm whose runtime, with respect to the input size n of the problem f, satisfies $f \in O(n^k)$ for some $k \in \mathbb{N}$. One readily verifies that, for $c \in \mathbb{R}$ and maps f_1, f_2, g_1, g_2 defined on suitable domains, the following hold.

$$f_1 \in O(g_1) \quad \Rightarrow \quad c \cdot f_1 \in O(g_1)$$

$$f_1 \in O(g_1) \wedge f_2 \in O(g_2) \quad \Rightarrow \quad f_1 + f_2 \in O(\max\{g_1, g_2\})$$

$$f_1 \in O(g_1) \wedge f_2 \in O(g_2) \quad \Rightarrow \quad f_1 \cdot f_2 \in O(g_1 \cdot g_2)$$

There exist plenty of nuanced points worth discussing in relationship to the associated framework. Nonetheless, in the interest of keeping the scope of this thesis manageable,

[22] For comparison, [Cor+22] directly defines $O(g)$, $\Omega(g)$, and $\Theta(g)$ as the sets of functions f that satisfy the respective criteria of Definitions 3.2(i)-(iii) and therefore uses $f \in O(g)$, whereas [NC10] sidesteps the issue by using the convention of stating that "f is $O(g)$."

we shall only gloss over a few of them in a quick manner:

- **The Bachmann-Landau formalism enables evaluations and comparisons of algorithms that are independent of the computer used:** Because there exists a wealth of reasons for why different computers might execute code at different speeds, evaluating algorithms by measuring the physical time it takes for them to do just that utilizing only a single computer is bound to be an unreliable measure of quality. Determining the asymptotic behavior of the number of operations required is a more generalizable method of extrapolating the behavior of a program.

- **The Bachmann-Landau formalism does not necessarily enable evaluations and comparisons of algorithms that are independent of the *class* of computer used:** If a quantum algorithm's asymptotic runtime bound suggests an exponential speed-up over the best classical algorithm, but *all* quantum computing architectures physically realizable would require significantly longer times for an elementary operation to occur in comparison to an elementary operation on a classical computer, then the physical time required to run the quantum algorithm might nevertheless exceed the classical one's.

- **The Bachmann-Landau formalism can hide constants that are, in fact, not irrelevant:** Stating that the runtime of an algorithm f satisfies $f \in O(g)$ only ensures that there exists a threshold $t \in \mathbb{R}$ after which f is bounded by a multiple of g. Having said that, if problems of size n smaller than the threshold are considered, then an algorithm with a worse asymptotic runtime might outperform one with a better asymptotic runtime. Particularly, the threshold of improved performance might not be achievable at all by all means practical. Richard Lipton and Kenneth Regan coined the term *galactic algorithm* for "an algorithm that is wonderful in its asymptotic behavior, but is never used to actually compute anything" [LR13, p. 111], e.g. because inputs achieving the speed-up are too large to be stored on devices found on earth.[23]

- **The Bachmann-Landau formalism might not clearly define what counts as *one* operation:** While, on a fundamental level, the answer to that may vary depending on the model of computation considered [cf. NC10, p. 136], there exist simpler reasons for deviations. For instance, the assignment of a variable or the

[23] Ironically, Lipton and Regan noted in 2013 that Shor's algorithm "may or may not be a galactic algorithm" due to the uncertainty regarding the actual realization of quantum computers, but expressed the belief that "if and when practical quantum computers are built, Peter's algorithm will be one of the first algorithms run on them" [LR13, p. 110]. Now, a decade later, the situation is reversed. The realization of practical quantum computers is being worked on at an evergrowing speed, but running Shor's algorithm on a quantum computer to solve problems of real-world importance is believed to be one of the applications furthest into the future [cf. GE21].

addition of two integers is usually considered to be an elementary operation that is performed in constant time. But clearly, this is only an approximation which does not hold for sufficiently large numbers [cf. HH21]. For quantum computers in particular, the answer of what counts as *one* operation can depend on the universal gate set implemented by the concrete architecture in question [cf. NC10, pp. 188–202].

Despite its limitations, the idea of analyzing algorithms' asymptotic run-times led to transformative insights. One particularly consequential idea consist of the categorization of problems by the algorithms known to solve them: A problem is in the complexity class NP if its solutions can be verified as such in polynomial time. As an accessible example for NP problems, one may consider integer factorization. Determining whether a set of prime numbers corresponds to the set of prime factors of a large integer n is as easy as multiplying them, while determining those factors when only given n is clearly not as trivial a task. In the early 1970s, Stephen Cook proved that any NP problem can be reduced to the Boolean satisfiability problem (SAT) in polynomial time [Coo71], a property now known as NP-completeness.[24] If one devised a polynomial-time algorithm to solve SAT, *every* other NP problem could subsequently be solved in polynomial time using that same algorithm. Put differently: The Boolean satisfiability problem is at least as demanding as every other NP problem. Prominent work by Richard Karp later proved that over twenty other, well-known problems possessed that same property [Kar72]. These insights gave the field of complexity theory its modern form. As eluded to in the introduction, one of the largest unsolved problems in computer science remains the determination of whether or not P, the complexity class of all problems solvable in polynomial time, is distinct or equal to NP.

3.1.3. Encoding Optimization Problems

The most abstract formulation of a problem may certainly be of help regarding its treatment and the search for a solution. However, running an algorithm requires a precise set of statements to perform, including concrete definitions of variable types as well as which rules it is to be transformed by. Modern classical computers may have advanced to the point of producing high-level programming languages that are far removed from hardware considerations and implement data types that parallel the notion of an abstract graph, but these are rarely the languages producing the fastest-running code. Moreover, the programming of quantum computers continues to remain, by all means, a mostly low-level endeavor, with hardware considerations playing a significant role [cf. Dom+23, p. 2].[25] The customary approach to tackling optimization problems on a quantum computer is to formulate them in terms of several variables, often binary ones $b_k \in \mathbb{b}$, which may need to satisfy certain constraints c_i, before then mapping these

[24] Leonid Levin independently proved an equivalent result for the tiling problem [Lev73].
[25] Some high-level quantum programming languages do exist, for example *Silq* [Bic+20].

bits to qubits utilizing the canonical map $b \mapsto |b\rangle$. This procedure is called *encoding* an optimization problem. The encoding that will be used in the subsequent work is the following:

Definition 3.3 (Binary Combinatorial Optimization Problems)

(i) A **binary combinatorial optimization problem** $\mathcal{C}$ is a tuple (p, q, c, f, g) where p and q are natural numbers $p \in \mathbb{N}$ and $q \in \mathbb{N}_0$, c is a map $c : \mathbb{b}^p \to \mathbb{b}^q$, f is a map $f : \mathbb{b}^p \to \mathbb{R}$, and g is a map $\mathrm{g} \in \{\min, \max\}$.

Let $\mathcal{C} = (p, q, c, f, \mathrm{g})$ be a binary combinatorial optimization problem.

(ii) The the set $\mathcal{S}_{\mathrm{tot}} = \mathbb{b}^p$ is the **total search space** of $\mathcal{C}$ and an element of $\mathcal{S}_{\mathrm{tot}}$ is called a **candidate solution**.

(iii) The i**-th constraint** of $\mathcal{C}$ is the map $c_i : \mathbb{b}^p \to \mathbb{b}, b \mapsto \langle c(b), e_i \rangle$ and a candidate solution b of $\mathcal{C}$ is said to **satisfy** c_i if $c_i(b) = 1$ or **violate** c_i if $c_i(b) = 0$. Further, $\mathcal{C}$ is said to be **constrained** if $q > 0$ and **unconstrained** if $q = 0$.[26]

(iv) The **induced optimization problem of** $\mathcal{C}$ is the optimization problem $\mathcal{O}_{\mathcal{C}} = (\mathcal{S}_{\mathcal{C}}, f, \mathrm{g})$ where $\mathcal{S}_{\mathcal{C}} = \{\, b \in \mathbb{b}^p : c(b) = \vec{1} \,\}$. The various terms defined for optimization problems apply to a binary combinatorial optimization problem if they hold for its induced optimization problem.

We wish to spend a paragraph contemplating how a formal approach to handling encodings could be defined: Assume $\mathcal{C}$ to be a 'specified' optimization problem, for example, an integer linear program or the binary combinatorial optimization problem defined above. Then $\mathcal{C}$ could be said to *encode* an optimization problem $\mathcal{O}$, if there exists an 'optimization problem homomorphism' from its induced optimization problem $\mathcal{O}_{\mathcal{C}}$ to the original problem $\mathcal{O}$. Reasonable definitions of optimization problem homomorphism could include but would not be limited to a map $\phi : \mathcal{S}(\mathcal{O}_1) \to \mathcal{S}(\mathcal{O}_2)$ satisfying $f_1(s) = f_2(\phi(s))$ or a map $\phi : \mathcal{S}(\mathcal{O}_1) \to \mathcal{S}(\mathcal{O}_2)$ satisfying $s \in \mathcal{S}_{\mathrm{op}}(\mathcal{O}_1) \Leftrightarrow \phi(s) \in \mathcal{S}_{\mathrm{op}}(\mathcal{O}_2)$. When defining encodings, it would be sensible to demand surjectivity of $\phi : \mathcal{S}(\mathcal{O}_{\mathcal{C}}) \to \mathcal{S}(\mathcal{O})$ in order to ensure that every feasible solution $s \in \mathcal{S}(\mathcal{O})$ of the original problem $\mathcal{O}$ possesses a viable solution $s \in \mathcal{S}(\mathcal{O}_{\mathcal{C}})$ in the specified problem $\mathcal{C}$ that corresponds with it. A 'true' encoding $\mathcal{C}$ could be one whose induced problem $\mathcal{O}_{\mathcal{C}}$ is 'isomorphic' to $\mathcal{O}$, i.e. where ϕ is bijective. The above is purely speculative, however. As there does not exist a commonly accepted mathematical framework for a rigorous treatment of these procedures, most authors do not spend time proving the equivalence of their quantum computational encoding and the original problem.[27] (And, in the interest of focusing this chapter's efforts on more deserving endeavors in Section 3.3, we will not either.) Nevertheless, we do believe (i) that the above outline provides the right intuition for

[26] Here, $\langle \cdot, \cdot \rangle$ denotes the standard scalar product $\langle v, w \rangle = \sum v_i w_i$ for $v = \sum v_i e_i$ and $w = \sum w_i e_i$.
[27] As an example, consider [Luc14], which first provided Ising formulations of many NP-problems.

the high-level ideas behind encodings and (ii) that the ability to rigorize these notions is essential to a holistically mathematical treatment.

While the encoding of an optimization problem is not unique it can differ in multiple aspects. Most conspicuously, encodings can vary in the intuition used to assign meaning to variables. For example, both the Miller-Tucker-Zemlin encoding [MTZ60] and the vertex-time encoding [Luc14] of the Traveling Salesman Problem consist of a two-dimensional array of binary variables x_{ij}. In the MTZ formalism, both indices i and j of a variable x_{ij} correspond logically to a city. The same is not the case for the vertex-time formalism, in which — as the name suggests — the second index corresponds to an (imaginary) timeslot. Both encodings are fundamentally dissimilar in the manner in which they assign operational meaning to bitstrings. However, even encodings of the same problem, agreeing with one another in that regard, can differ in another integral characteristic: the manner in which they reject bitstrings that are not in correspondence with a feasible solution to the original problem. Two approaches exist:

- *Softcoding* describes the practice of not restricting an encoding's search space but modifying its objective function to discourage solutions not corresponding to feasible solutions of the original problem. The result is an unconstrained problem.

- *Hardcoding* denotes the process of restricting an encoding's search space by formulating constraints that exclude solutions not corresponding to feasible solutions of the original problem. The result is a constrained problem.

The encoding approach prevalent within quantum optimization is, undeniably so, softcoding [cf. Dom+23, p. 2; Pun22, p. vii]. Most often, an optimization problem is mapped to a quadratic unconstrained binary optimization problem (QUBO) [cf. Pun22]. Computational basis states $|b\rangle$ represent candidate solutions b, the objective function is mapped to a Hamiltonian diagonal in the computational basis, and the optimal solution (of a minimization problem) corresponds to its lowest eigenvalue eigenstate. The continued search for hardcoded mixing families in this work is motivated by findings indicating suboptimal results produced by this procedure[cf. GH19; Dam+22; BR22].

We shall conclude this section by introducing binary combinatorial optimization problems of 'scheduling-type'. In Section 3.3, we will replicate the study of feasibility-preserving permutations of [Koß+23], which crucially uses constraint graphs [Lei79]. Scheduling-type problems are important to this work because we will concretize the construction of constraint graphs concerning this class of problems.

Definition 3.4 (Scheduling-Type B.C.O.P.s [Bin22, p. 30])
Let $\mathcal{C} = (p, q, c, f, \mathrm{g})$ be a binary combinatorial optimization problem.

(i) A **one-hot constraint** of $I \subset [p]$ is a constraint c_i satisfying $c_i(b) = \zeta_I(b)$ with[28]

$$\zeta_I : \mathbb{b}^p \to \mathbb{b},\ \zeta_I(b) = \begin{cases} 1, & \text{if } \| b_I \odot b \|_1 = 1, \\ 0, & \text{else.} \end{cases}$$

(ii) A **at-most-one constraint** of $I \subset [p]$ is a constraint c_i satisfying $c_i(b) = \eta_I(b)$ with

$$\eta_I : \mathbb{b}^p \to \mathbb{b},\ \eta_I(b) = \begin{cases} 1, & \text{if } \| b_I \odot b \|_1 \leq 1, \\ 0, & \text{else.} \end{cases}$$

(iii) $\mathcal{C}$ is called **of scheduling type** if there exist exactly one $l \in [q]$ and one family of sets $\{I_1, \ldots, I_q\}$ of $[p]$ such that

(S1) $\{I_1, \ldots, I_l\}$ is a partition of $[p]$ of minimum block size 2,

(S2) for $i \in [q]$ with $i \leq l$, the constraint c_i is a one-hot constraint of I_i,

(S3) for $i \in [q]$ with $i > l$, the constraint c_i is an at-most-one constraint of I_i.

Remark 3.5
In the following, the term *scheduling encoding* is used as a shorthand notation for the term *binary combinatorial optimization problem of scheduling type*.

3.2. The Traveling Salesman Problem

3.2.1. Fundamental Notions

Without further ado, we will provide a general definition of the Traveling Salesman Problem as well as comment on its complexity before justifying the subsequent focus on problem instances induced by a weighted graph of the form (K_n, w).

Definition 3.6 (Traveling Salesman Problem)
The **Traveling Salesman Problem** specified by a weighted graph $G = (V, E, w)$ is the optimization problem $\mathrm{TSP}(G) = (\mathrm{Ham}(G), \overline{\overline{w}}, \min)$. A **tour** of G is a feasible solution of $\mathrm{TSP}(G)$.

A key characteristic to classify instances of the Traveling Salesman Problem by is whether or not they are symmetric, i.e. whether there exists a difference between a tour

[28] Recall that $b_I \in \mathbb{b}^p$ is the bitstring whose i-th component satisfies $b_i = 1$ if and only if $i \in I$; cf. page 7. Further, $\| b_1 \odot b_2 \|_1$ corresponds to the Hamming weight of the bit-wise AND of b_1 and b_2.

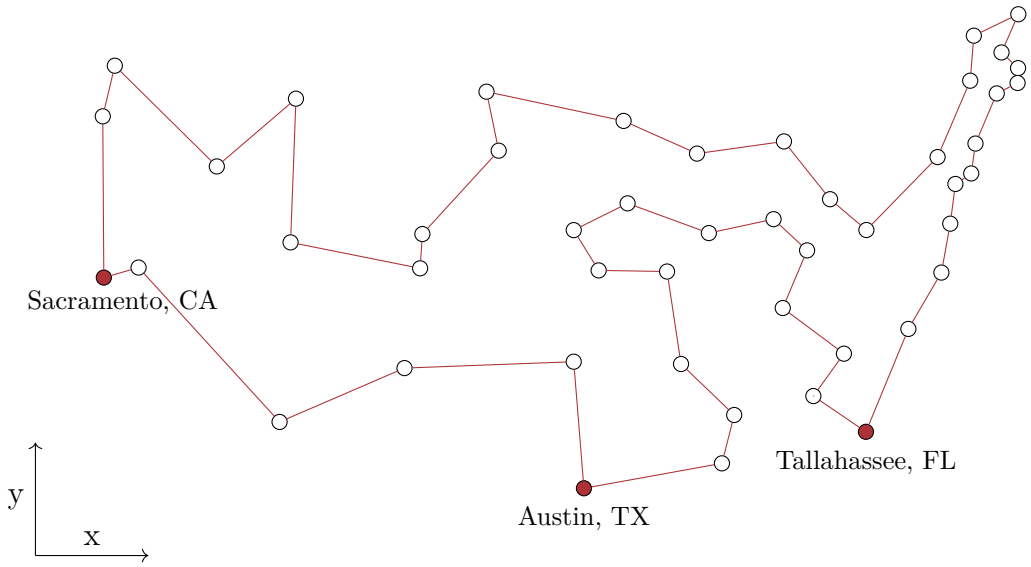


Figure 3.1.: Vis. of the TSP instance *att48* and its optimal tour from the problem library *TSPLIB* [Rei91], an influential repository of benchmarking instances [cf. App+06]. The graph's vertices are elements of $\mathbb{R}^2$ and the edge's weights are given by the metric $d(v_1, v_2) = \|v_1 - v_2\|$.[29] The instance's vertices correspond to the 48 state capitals of the contiguous United States; selected cities are marked and labeled.[30]

and its 'reverse' tour. Formally, we call a Traveling Salesman Problem by a weighted graph G symmetric if $v_1v_2 \in E$ implies $v_2v_1 \in E$ and $w(v_1v_2) = w(v_2v_1)$ for all $v_1, v_2 \in V$.

Theorem 3.7 (Karp; TSP is NP-complete)
The Traveling Salesman Problem is NP-complete.

Proof
The Hamiltonian cycle problem, i.e. the problem of determining whether a given graph possesses a Hamiltonian cycle, belongs to the list of 21 problems proven to be NP-complete in Karp's influential work [Kar72]. This implies that the TSP's search problem is NP-hard and that its decision problem is NP-complete. □

Considering that the feasible search space $\mathcal{S}(\mathrm{TSP}(G))$ of a TSP instance is given by the set of Hamiltonian cycles $\mathrm{Ham}(G)$ of G, and that the number of those cycles can increase factorially with the size $|G|$ of the graph, it should become plausible that finding the optimal solution of this seemingly trivial problem is not trivial at all. For example, Figure 3.1 visualizes a prominent problem instance: finding the shortest path that visits each of the 48 state capitals of the contiguous United States exactly once before returning

[29] This is known as a *Euclidean* TSP instance.
[30] A Euclidean transformation $R\,v + t$ was applied to all vertices $v \in \mathbb{R}^2$ to improve visual clarity.

to the origin. As the instance is symmetric and possesses 48 vertices, it possesses $(48-1)!/2$ different tours, i.e.

$$129\,311\,620\,755\,584\,090\,321\,482\,177\,576\,805\,989\,984\,598\,816\,194\,560\,000\,000\,000$$

distinct feasible solutions. Clearly, all classical algorithms that wish to explore considerable portions of the search space become swiftly impractical. A common choice is to instead use heuristics, i.e. algorithms that ensure a fast runtime but cannot guarantee finding the optimal solution, such as the Lin-Kernighan heuristic [LK73].[31] Of the exact solvers, *Concorde* is the most prominent one [App+06; MW03, p. 830]

Of considerable impact for this thesis is the fact that, with minimal concessions, it is sufficient to conceptually consider the Traveling Salesman Problem for complete weighted graphs of the form (K_n, w): Recall that our interest in the Traveling Salesman Problem originates from its ecological and scientific applications encountered by humankind. Hence, assuming the underlying graph G to be finite is an eminently reasonable restriction (and motivated Remark 2.23). Due to that, we can argue for the sufficiency of studying instances of form (K_n, w) in three steps.

(i) **It suffices to consider weighted graphs:** Since G is of finite order by assumption, it is of finite size. Therefor, the set $\{w(e) : e \in E(G)\}$ is finite and possesses a minimum $w_{\min}$. If the minimum is negative, simply adding its absolute value to all weights will not change the optimal tour. Moreover, this can be done in polynomial time, as the minimum of $O(n^2)$ weights[32] can be found in time $O(n^2)$ and then also be added to all $O(n^2)$ weights in time $O(n^2)$. The computational overhead is negligible.[33] The graph-theoretic aspect of this thought is captured in Proposition 3.8 below.[34]

(ii) **It suffices to consider complete graphs:** If G is a graph that does possess a Hamiltonian cycle, then one can add an edge without changing the optimal search space by ensuring its weight to be sufficiently large. The most trivial choice for a 'sufficiently large' weight is one that is guaranteed to exceed the total length of the *worst* preexisting tour. That is easily constructed by e.g. determining the maximum of $\{w(e) : e \in E(G)\}$. Furthermore, both finding the maximum weight as well as adding edges with sufficiently high weight can again be done in time

[31] Not to be confused with the Kernighan-Lin heuristic for partitioning graphs [KL70].

[32] Cf. Propositon 2.28 considering that incomplete graphs G must then satisfy $\|G\| < |G|\,(|G|-1)/2$ (or $\|G\| < |G|\,(|G|-1)$ respectively).

[33] Here and in the following, 'negligible' computational burden is to be understood in the context of tackling NP problems, which, by default, we consider an exponential-time endeavor.

[34] As a more general point of comparison than $\mathbb{R}$-weighted graphs would be M-weighted graphs where M is the underlying set of a totally ordered set $(M, \leq)$. However, we are not aware of any context where this case naturally occurs in a manner not describable by or reducible to an $\mathbb{R}$-weighted graph.

$O(n^2)$ with the same reasoning as before.[35] The graph-theoretic aspect of this thought is captured in Proposition 3.9 below.

(iii) **It suffices to consider standard graphs:** If G is finite and complete, then by Proposition 2.31 any enumeration function on $V(G)$ constitutes a graph isomorphism from G to $K_{|G|}$. Moreover, any isomorphism canonically induces a weight function w' on $E(K_{|G|})$ via $w' : E(K_{|G|}) \to w(E(G))$, $w(i,j) = w(\mathcal{E}^{-1}(i), \mathcal{E}^{-1}(j))$. Strikingly, such an enumeration function can not only be found but also used virtually evermore with negligible computational overhead. Hash tables, also known as associative arrays or dictionaries, are data types storing key-value pairs that, on average, allow inserting, searching, and removing an item in constant time.[36] Hence, assigning an index to n vertices can be done in time $O(n)$, and the asymptotic 'burden' of calling w' in contrast to w vanishes.

Proposition 3.8 (Mapping $\mathbb{R}$-Weighted to $\mathbb{R}_{\geq 0}$-Weighted TSP Instances)
If $G = (V, E, w)$ is an $\mathbb{R}$-weighted graph, then there exists an $\mathbb{R}_{\geq 0}$-weighted graph $G' = (V', E', w')$ satisfying

(i) $V = V'$,

(ii) $E = E'$,

(iii) $S_{\mathrm{op}}(\mathrm{TSP}(G)) = S_{\mathrm{op}}(\mathrm{TSP}(G'))$.

Proof
If $G = (V, E, w)$ is an $\mathbb{R}$-weighted graph, then $G' = (V, E, w')$ with

$$w' : E \to \mathbb{R}_{\geq 0},\ e \mapsto w(e) + |\min\{w(e) : e \in E\}|$$

is an $\mathbb{R}_{\geq 0}$-weighted graph satisfying (i)-(iii). □

[35] One drawback must be acknowledged: It might not be evident beforehand whether a given graph G contains a Hamiltonian cycle in the first place, and simply verifying this is not a desirable course of action due to the NP-completeness of the Hamiltonian cycle problem [Kar72]. If $\mathrm{Ham}(G) = \varnothing$, the mapping $G \mapsto G'$ from Proposition 3.9 does indeed change the optimal solution space. However, if G' is constructed with the procedure outlined in the proof of Proposition 3.9, then finding any feasible solution C of $\mathrm{TSP}(G')$ of length smaller than w_ϵ solves the Hamiltonian cycle problem of G affirmatively because the tour C is contained in the original graph G if and only if $\overline{\overline{w}}(C) < w_\epsilon$. Conversely, solving $\mathrm{TSP}(G')$ with provable optimality and obtaining an optimal solution C_{op} whose length exceeds w_ϵ solves the Hamiltonian cycle problem negatively.

[36] An instructive visual explanation of how this works is found in e.g. [Dow15, pp. 208–209].

Proposition 3.9 (Mapping Incomplete to Complete TSP Instances)
If $G = (V, E, w)$ is an incomplete weighted graph satisfying $\mathrm{Ham}(G) \neq \varnothing$, then there exists a complete weighted graph $G' = (V', E', w')$ satisfying

(i) $V = V'$,

(ii) $E \subset E'$,

(iii) $w = w'|_E$,

(iv) $S_{\mathrm{op}}(\mathrm{TSP}(G)) = S_{\mathrm{op}}(\mathrm{TSP}(G'))$.

Proof
We offer an outline of the proof: Define $w_\epsilon = |G| \cdot \max\{w(e) : e \in E(G)\} + \epsilon$ for $\epsilon > 0$. If G is an undirected graph, define $E' = \mathrm{Un}(V)$; else, define $E' = \mathrm{Di}(V)$. Then, $G' = (V, E', w')$ with

$$w' : E' \to \mathbb{R}_{\geq 0},\ w'(e) = \begin{cases} w(e), & \text{if } e \in E, \\ w_\epsilon, & \text{if } e \notin E, \end{cases}$$

is a complete weighted graph satisfying (i)-(iii). Now, let $C' \in \mathrm{Ham}(G') \setminus \mathrm{Ham}(G)$. Then there must exist an edge $e' \in E(C')$ such that $e' \in E(G')$ but $e' \notin E(G)$. Hence, $w(e') = w_\epsilon$ is a weak lower bound of $\overline{\overline{w}}(C')$, i.e. $w_\epsilon \leq \overline{\overline{w}}(C')$ However, by construction, w_ϵ is a strict upper bound of $\overline{\overline{w}}(C)$ for all $C \in \mathrm{Ham}(G)$, i.e. $\overline{\overline{w}}(C) < w_\epsilon$. Therefore, since $\overline{\overline{w}}(C) < \overline{\overline{w}}(C')$ for all $C \in \mathrm{Ham}(G)$ and $C' \in \mathrm{Ham}(G') \setminus \mathrm{Ham}(G)$, the graph G' satisfies (iv). □

3.2.2. Encoding the Traveling Salesman Problem

The Traveling Salesman Problem can be encoded as an integer linear program. Of the many formulations known, the most prominent ones are the Dantzig-Fulkerson-Johnson formulation and the aforementioned Miller–Tucker–Zemlin formulation [MTZ60]. Both agree with one another to the extent that they map an n-vertex TSP instance to a two-dimensional array of binary variables x_{ij}. Both i and j correspond to vertices $i, j \in [n] = V(G)$ of the considered graph G, and the variable x_{ij} satisfies $x_{ij} = 1$ if and only if the tour travels from city i to city j. To some degree, this is certainly the natural and favored way of viewing the problem. After all, the value to be minimized is the sum of the weights of the traveled edges, and in these encodings, the variables directly correspond to these values. As the name 'integer *linear* program' implies, the objective function is a map *linear* in the variables x_{ij}. For reasons eluded to more in the outlook of Chapter 6, we will restrict ourselves to the encoding prevalent within quantum optimization for the time being. The intuitive idea behind the vertex-time encoding is to introduce n imagined timeslots, and let the binary variable x_{ij} satisfy $x_{ij} = 1$ if and only

if city i is visited during timeslot j. Then, to ensure that the variables x_{ij} correspond to a Hamiltonian cycle, it suffices to require that the matrix $(x_{ij})_{1\leq i,j\leq n}$ is a permutation matrix. Put differently: Every city is visited exactly once, and at every timeslot, exactly one city is visited.

Definition 3.10 (Vertex-Time Encoding; [cf. Luc14, p. 10])
Let $n \in \mathbb{N}$ and $v, t \in [n]$. Further, let $\mathcal{E}$ be an enumeration function of $[n] \times [n]$.

(i) The v**-th vertex block** with respect to $\mathcal{E}$ is the set $\Delta_v^{\mathrm{V}} = \mathcal{E}(\{v\} \times [n]) \subset [n^2]$.

(ii) The t**-th time block** with respect to $\mathcal{E}$ is the set $\Delta_t^{\mathrm{T}} = \mathcal{E}([n] \times \{t\}) \subset [n^2]$.

(iii) The $\mathcal{E}$**-vertex-time encoding** of the Traveling Salesman Problem specified by the complete weighted graph (K_n, w) is the binary combinatorial optimization problem $\mathcal{C}_{\mathrm{TSP}}(w) = (n^2, 2n, c, f, \min)$ with

$$c : \mathbb{b}^{n^2} \to \mathbb{b}^{2n},\ c(b) = \sum_{i=1}^{n} c_i(b)\, e_i \quad \text{with} \quad c_i(b) = \begin{cases} \zeta_{\Delta_i^{\mathrm{V}}}(b), & \text{if } i \leq n, \\ \eta_{\Delta_{n-i}^{\mathrm{T}}}(b), & \text{if } i > n, \end{cases}$$

$$f : \mathbb{b}^{n^2} \to \mathbb{R},\ f(b) = \sum_{u=1}^{n} \sum_{v=1}^{n} \sum_{t=1}^{n} w_{u,v}\, b_{\mathcal{E}(u,\,t)}\, b_{\mathcal{E}(v,\,t+1 \bmod n)}$$

Proposition 3.11 (The Vertex-Time Encoding is of Scheduling Type)
Let $n \in \mathbb{N}$. Further, let (K_n, w) be a complete weighted graph and let $\mathcal{E}$ be an enumeration function of $[n]^2$. If n is non-trivial, then the $\mathcal{E}$-vertex-time encoding $\mathcal{C}_{\mathrm{TSP}}(w)$ is of scheduling type.

Proof
Clearly, $l = n \in [2n] = [q]$ and the family of sets $\{I_1, \ldots, I_q\} \subset [n] \times [n]$ with $I_i = \Delta_i^{\mathrm{V}}$ for $i \leq l$ and $I_i = \Delta_i^{\mathrm{T}}$ for $i > l$ satisfy the conditions of scheduling type B.C.O.P.s., with $|I_i| > 2$ following from n being non-trivial. Furthermore, both l and $\{I_i : i \in [m]\}$ are clearly unique since (i) l and $m - l$ must agree respectively with the number of one-hot constraints and at-most-one constraints, and (ii) I_i must be in agreement with the sets for which the one-hot constraints and at-most-one constraints are defined. □

Two remarks about the vertex-time encoding's dimension are in order. First of all, the representations of a feasible solution are degenerate by default: If an asymmetric graph is considered, then each Hamiltonian cycle of the graph has exactly n different feasible solutions of the vertex-time encoding that correspond to it.[37] For symmetric

[37] In group-theoretic terms, this corresponds to the fact that there are n different ways to write down one and the same n-cycle, i.e. $(i_1, i_2, \ldots, i_{n-1}, i_n) = (i_2, i_3, \ldots, i_n, i_1) = \ldots = (i_n, i_1, \ldots, i_{n-2}, i_{n-1})$.

graphs, that number jumps to $2n$. The former can be handled easily by fixing a certain city c_0 to be visited at a certain timeslot t_0. For a one-based indexing formalism, $c_0 = n$ and $t_0 = n$ would be a natural choice. The latter is non-trivial, and will be examined in Chapter 5. Secondly, the formulation can be adjusted to use significantly less space, requiring $\Theta(n \log_2 n)$ binary variables instead of n^2, by forming registers r_i of $\log_2 n$ binary variables that store the city visited at timeslot i (or, equivalently, by grouping binary variables into registers r_i of $\log_2 n$ variables that store at what timeslot the city i is visited). For simplicity, we will conceptually remain with the n^2-formalism. Nevertheless, it requires emphasis that all results work excellently for the other formalism as well. When converting the n^2-variant to the $n \log n$-variant, one of the subgroups acting transitively identified in Section 3.3 remains. Additionally, the decompositions of S_n derived in Chapter 5 can be applied for that subgroup to construct a mixing family. Issues of preserving feasibility and traversing the entire feasible space in the $n \log_2 n$-formalism become insignificant when using hardcoded mixing families.

3.3. Feasibility-Structure of the Vertex-Time Encoding

This section marks the return to more mathematical considerations, whose foundation was laid in the preceding chapter and with Definitions 3.3, 3.4, and 3.10. The goal is to identify transitive group actions on the feasible solution set $\mathcal{S}(\mathcal{C})$ of the $\mathcal{E}$-vertex-time encoding $\mathcal{C}$ by following the developments of [Koß+23]. First, the constraint graph is defined with respect to scheduling encodings, and fundamental constructions and ideas are discussed. Then, the group of feasibility-preserving permutations of the $\mathcal{E}$-vertex-time encoding's constraint graph is derived, and two subgroups acting transitively on the constraint graph's feasible search space are identified. Last, using the correspondence of feasible solutions of the constraint graph and feasible solutions of the encoding it represents, the transitive action of these subgroups on the encoding's feasible search space is established.

3.3.1. The Constraint Graph

Definition and Proposition 3.12 (Constraint Graphs)
Let $\mathcal{C} = (p, q, c, f, \mathrm{g})$ be a scheduling encoding with $l \in [q]$ and $\{I_i : i \in [q]\} \subset \mathcal{P}([p])$ satisfying the scheduling-type conditions.

(i) The **constraint graph** of $\mathcal{C}$ is the tuple $\mathrm{C}_\mathcal{C} = (V, E, \iota, l)$ where V and E are sets and ι is a map $\iota : [p] \to V$ satisfying the following conditions.

(C1) V is of cardinality p.

(C2) E is given by $E = \bigcup_{i\in[q]} \mathrm{Un}(\iota(I_i))$.

(C3) ι^{-1} exists and is an enumeration function on V.

Let $\mathcal{C} = (p, q, c, f, \mathrm{g})$ be a scheduling encoding and let $\mathrm{C} = (V, E, \iota, l)$ be its constraint graph.

(ii) The **total solution space** of C is the set $\mathcal{S}_{\mathrm{tot}} = \mathcal{P}(V)$ and an element of $\mathcal{S}_{\mathrm{tot}}$ is called a **candidate solution**; the **feasible solution space** of C is the set $\mathcal{S} = \{W \in \mathcal{S}_{\mathrm{tot}} : W \in \mathrm{Co}(V, E) \wedge |W| = l\}$ and an element of $\mathcal{S}$ is called a **feasible solution**.[38]

(iii) The **identification mapping** of C is the bijection $\kappa_\iota : \mathcal{S}_{\mathrm{tot}}(\mathcal{C}) \to \mathcal{S}_{\mathrm{tot}}(\mathrm{C})$ defined by $b \mapsto \iota(\{i \in [p] : b_i = 1\})$.[39]

(iv) A **feasibility-preserving map** of $\mathrm{C}_\mathcal{C}$ is a map $\phi : V \to V$ satisfying $\phi(W) \in \mathcal{S}$ for all $W \in \mathcal{S}$. The **group of all feasibility-preserving permutations** is the group $\mathcal{F}(\mathrm{C}) = \{\phi \in \mathrm{Sym}(V) : \phi \text{ preserves feasibility}\} \leq \mathrm{Sym}(V)$.

Proof
The only parts warranting a proof are part (iii) and (iv).

(iii) κ_ι is injective because $\kappa_\iota(b) = \kappa_\iota(b')$ implies $\{i \in [p] : b_i = 1\} = \{i \in [p] : b'_i = 1\}$ by bijectivity of ι, from which $b_i = 1 \Leftrightarrow b'_i = 1$ follows for all $i \in [p]$, implying $b = b'$ because all elements of b and b' only take values in $\mathbb{b} = \{0, 1\}$. κ_ι then is bijective because it is injective and $|\mathcal{S}_{\mathrm{tot}}(\mathcal{C})| = |\mathbb{b}^p| = 2^p = 2^{|V|} = |\mathcal{P}(V)| = |\mathcal{S}_{\mathrm{tot}}(\mathrm{C})|$ with $p = |V|$ following from the bijectivity of $\iota : [p] \to V$.

(iv) Noting that V is finite by definition allows the use of Proposition 2.7(ii). Clearly, $\mathcal{F}$ is non-empty as $\mathrm{id} \in \mathcal{F}$. Further, let $\phi_1, \phi_2 \in \mathcal{F}$. If $W \in \mathcal{S}_{\mathrm{tot}}$ satisfies

[38] The constraint graph model is considered by [Koß+23] chiefly for analyzing the set of feasible solutions and how to 'traverse' that space using groups of feasibility-preserving permutations, but not for analyzing the optimality of solutions. Hence, the *optimal* solution space is of no interest to us when discussing the constraint graph.

[39] Depending on what lessens notational burdens and comes more naturally in a given context, the identification mapping of $\mathrm{C} = (V, E, \iota, l)$ will interchangeably be denoted by κ_C and κ_ι.

$W \in \mathcal{S}$, then this implies that $W' = \phi_2(W) \in \mathcal{S}_{\text{tot}}$ satisfies $W' \in \mathcal{S}$ (because ϕ_2 is feasibility-preserving), which itself implies that $W'' = \phi_1(W') \in \mathcal{S}_{\text{tot}}$ satisfies $W'' \in \mathcal{S}$ (because ϕ_1 is feasibility-preserving). □

While the constraint graph is, formally, a quadruple (V, E, ι, l) and therefore not a graph per se, it obviously is associated with the graph (V, E). Notions defined in regard to graphs will be referenced with respect to constraint graphs, indicating the use of the associated graph. For example, the automorphism group of $\mathrm{C} = (V, E, \iota, l)$ is simply the automorphism group of $G = (V, E)$. When constructing the constraint graph of an $\mathcal{E}$-vertex time encoding, it is assumed that $V = [n]^2$ and $\iota = \mathcal{E}^{-1}$ unless explicitly stated otherwise. For visual intuition regarding the construction and structure of the constraint graph, as well as the correspondence of feasible solutions of $\mathrm{C}_{\mathcal{C}_{\mathrm{TSP}}(w)}$ to solutions of the original problem $\mathrm{TSP}(G)$, we refer to Figure 3.2.

Proposition 3.13 ($\mathcal{S}(\mathcal{C})$ Corresponds to $\mathcal{S}(\mathrm{C})$)
Let $\mathcal{C}$ be a scheduling encoding and let C be its constraint graph. If $b \in \mathcal{S}_{\text{tot}}(\mathcal{C})$ and $W \in \mathcal{S}_{\text{tot}}(\mathrm{C})$ satisfy $W = \kappa_{\mathrm{C}}(b)$, then the following statements are equivalent.

(i) $b \in \mathcal{S}(\mathcal{C})$.

(ii) $W \in \mathcal{S}(\mathrm{C})$.

This equivalence should come as no surprise, as the constraint graph is specifically constructed to facilitate this correspondence.

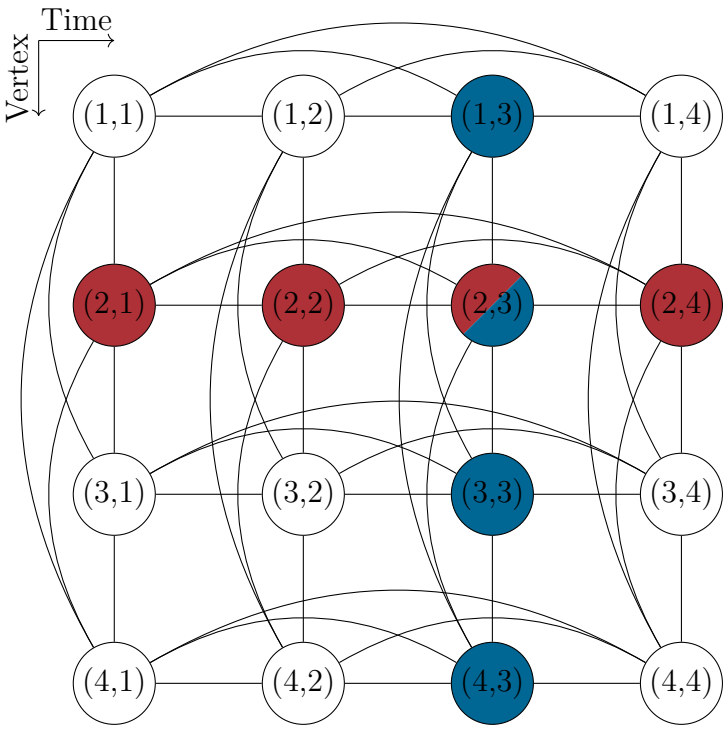
Proof
In the following, let $\mathcal{C}$ be given by $\mathcal{C} = (p, q, c, f, \mathrm{g})$ and let $l \in [q]$ and $\{I_i : i \in [q]\} \subset \mathcal{P}([p])$ satisfy the scheduling type conditions for $\mathcal{C}$.

(i) ⇒ (ii): Firstly, it is crucial to note that $\| b \|_1 = \sum_{k=1}^{p} b_k = l$ follows from c_i being a one-hot constraint of I_i for all $i \in [l]$ (by condition (S2) of Definition 3.4(iii)) and $\{I_i : i \in [l]\}$ being a partition of $[p]$ (by condition (S1) of Definition 3.4(iii)), implying $|\kappa_\iota(b)| = l$.

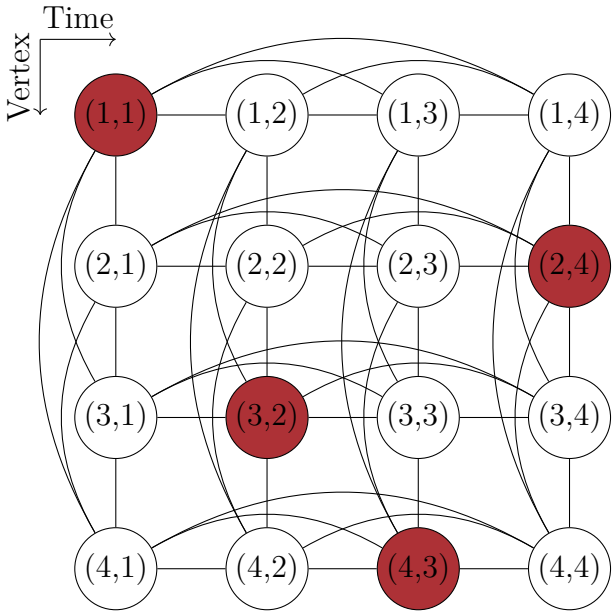
Moreover, assume that $\kappa_\iota(b) \subset V$ is not a coclique of (V, E). Then there exist $v_1, v_2 \in V$ such that $\{v_1, v_2\} \in E$. Define $k_i = \iota^{-1}(v_i)$. Since $E = \bigcup_{i \in [q]} \mathrm{Un}(\iota(I_i))$, there must exist $j \in [q]$ such that $\{v_1, v_2\} \in \mathrm{Un}(\iota(I_j))$. Additionally, the bijectivity of ι implies

$$\mathrm{Un}(\iota(I_j)) = \iota\Big(\Big\{\{k, k'\} : k, k' \in I_j \wedge k \neq k\Big\}\Big),$$

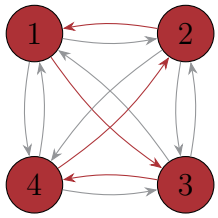
from which it then also follows that $\{k_1, k_2\} \in \{\, \{k, k'\} : k, k' \in (I_j) \wedge k \neq k \,\}$. Hence, the feasible solution b violates the constraint c_j, regardless of whether c_j is a one-hot or at-most-one constraint of I_j – contradiction! Thus, $W \in \mathcal{S}(\mathrm{C})$.



(a) Vis. of the constraint graph $\mathrm{C}_{\mathcal{C}}$. The set $\mathcal{E}^{-1}(\Delta_2^{\mathrm{V}}) \subset V$ corresponding to the 2nd vertex block (red) forms a clique due to the one-hot constraint $c_2 = \zeta_{\Delta_2^{\mathrm{V}}}$ and the set $\mathcal{E}^{-1}(\Delta_3^{\mathrm{T}}) \subset V$ corresponding to the 3rd time block (blue) forms a clique due to the at-most-one constraint $c_7 = \eta_{\Delta_3^{\mathrm{T}}}$.



(b) Vis. of the constraint graph $\mathrm{C}_{\mathcal{C}}$. The feasible solution $W = \{(1,1),(3,2),(4,3),(2,4)\} \in \mathcal{S}$ (red) forms a coclique of order 4.



(c) Vis. of K_4^d. W corresponds to the H. cycle with $E = \{(1,3),(3,4),(4,2),(2,1)\}$ (red).

Figure 3.2.: Let $G = (K_4^d, w)$ be a weighted graph and let $\mathcal{C} = (p, q, c, f, \mathrm{g})$ be its $\mathcal{E}$-vertex-time encoding with $p = 16$ and $q = 8$. Let $\mathrm{C}_{\mathcal{C}} = (V, E, \mathcal{E}^{-1}, l)$ be the constraint graph of $\mathcal{C}$ with $V = [4]^2$ and $l = 4$. Figure 3.2a illustrates the construction of the constraint graph; Figures 3.2b and 3.2c illustrate the correspondence between feasible solutions of $\mathrm{C}_{\mathcal{C}}$ and feasible solutions of $\mathrm{TSP}(G) \equiv \mathcal{O}_{\mathcal{C}}$.

(i) $\Leftarrow$ (ii): Assume b does not satisfy the at-most-one constraint c_j, i.e. $\| b \odot b_{I_j} \|_1 > 1$. Then there exist $k_1, k_2 \in I_j, k_1 \neq k_2$ satisfying $b_{k_1} = 1$ and $b_{k_2} = 1$, which implies that $\iota(k_1), \iota(k_2) \in \kappa_\iota(b)$. Now,

$$\{\iota(k_1), \iota(k_2)\} \in \iota\Big(\Big\{\{k, k'\} : k, k' \in I_j \wedge k \neq k'\Big\}\Big) = \mathrm{Un}(\iota(I_j)) \subset E$$

yields a contradiction to $W \in \mathcal{S}(\mathrm{C})$.

Assume b does not satisfy the one-hot constraint c_j, i.e. $\| b \odot b_{I_j} \|_1 \neq 1$. Without loss of generality, assume $\| b \odot b_{I_j} \|_1 = 0$ (because $\| b \odot b_{I_j} \|_1 > 1$ yields a contradiction with the same reasoning as above, implying the truth of (i) $\Leftarrow$ (ii)). Now,

$$\|b\|_1 = \sum_{k=1}^{p} b_k = \sum_{i=1}^{l} \sum_{k \in I_i} b_k = \sum_{\substack{i=1, \\ i \neq j}}^{l} \underbrace{\sum_{k \in I_i} b_k}_{\leq 1} + \underbrace{\sum_{k \in I_j} b_k}_{=0} < l$$

yields a contradiction to $W \in \mathcal{S}(\mathrm{C})$. Here, the equality $\sum_{k=1}^{p} b_k = \sum_{i=1}^{l} \sum_{k \in I_i} b_k$ follows from condition (S1) of Definition 3.4(iii) and $\sum_{k \in I_i} b_k \leq 1$ follows from the fact that b either satisfies c_i or satisfies $\| b \odot b_{I_i} \|_1 = 0$ (because $\| b \odot b_{I_i} \|_1 > 1$ yields a contradiction with the same reasoning as above, implying the truth of (i) $\Leftarrow$ (ii)). $\| b \|_1 < l$ implying $|\kappa_\iota(b)| < l$ nevertheless yields a contradiction to $W \in \mathcal{S}(\mathrm{C})$.

Thus, $b \in \mathcal{S}(\mathcal{C})$. □

3.3.2. Transitive Group Actions on $\mathcal{S}(\mathrm{C})$

Definition and Proposition 3.14 (The Group Action Ψ)
Let $\mathrm{C} = (V, E, \iota, l)$ be a constraint graph, then a group action Ψ_C of $\mathrm{Sym}(V)$ on $\mathcal{S}_{\mathrm{tot}}(\mathrm{C})$ is defined by

$$\begin{aligned} \Psi_\mathrm{C} : \mathrm{Sym}(V) &\to \mathrm{Sym}(\mathcal{S}_{\mathrm{tot}}(\mathrm{C})) \\ \sigma &\mapsto [W \mapsto \sigma(W)] \end{aligned}$$

Proof
The definition's validity is mostly self-evident: Let $\sigma \in \mathrm{Sym}(V)$ be arbitrary. If $W \subset V$, i.e. if $W \in \mathcal{P}(V) = \mathcal{S}_{\mathrm{tot}}(\mathrm{C})$, then $\sigma(W) \subset V$, i.e. then $\sigma(W) \in \mathcal{P}(V) = \mathcal{S}_{\mathrm{tot}}(\mathrm{C})$. Moreover, the map $W \mapsto \sigma(W)$ is injective (if $\sigma(W_1) = \sigma(W_2)$, then $\sigma^{-1}(\sigma(W_1)) = \sigma^{-1}(\sigma(W_2))$, implying $W_1 = W_2$) and surjective (if $W' \in \mathcal{P}(V) = \mathcal{S}_{\mathrm{tot}}(\mathrm{C})$, then $W = \sigma^{-1}(W') \in \mathcal{P}(V) = \mathcal{S}_{\mathrm{tot}}(\mathrm{C})$ satisfies $\sigma(W) = W'$) because σ is injective and surjective. Ψ_C being a homomorphism follows from $\sigma_1(\sigma_2(W)) = (\sigma_1 \circ \sigma_2)(W)$. Thus, Ψ_C is a group isomorphism by Proposition 2.10. □

This construction more generally produces a group action of $\mathrm{Sym}(M)$ on $\mathcal{P}(M)$ for an arbitrary set M. But unlike the group isomorphism $\phi_f : \mathrm{Sym}(M) \to \mathrm{Sym}(N)$ induced

by a bijection $f : M \to N$ (Definition 2.16), we will only ever utilize this for a constraint graph's vertex set.

Proposition 3.15 ($\mathcal{F}$ Acts on $\mathcal{S}$)
Let C be a constraint graph, then Ψ_{C} is a group action of $\mathcal{F}(\mathrm{C})$ on $\mathcal{S}(\mathrm{C})$.

Proof
The most effort required to prove this proposition lies in concretizing the exact statement. While we hope the succinct phrasing of the proposition to be clear, it is restated here more extensively: The map Ψ'_{C} given by

$$\begin{aligned}\Psi'_{\mathrm{C}} : \mathcal{F}(\mathrm{C}) &\to \mathrm{Sym}(\mathcal{S}(\mathrm{C})) \\ \phi &\mapsto [W \mapsto \phi(W)]|_{\mathcal{S}(\mathrm{C})}\end{aligned}$$

is a group action of $\mathcal{F}(\mathrm{C})$ on $\mathcal{S}(\mathrm{C})$, i.e. there are two different restrictions occurring: The map Ψ_{C} is restricted to the set $\mathcal{F}(\mathrm{C})$, and the map $\Psi_{\mathrm{C}}(\phi)$ to which $\phi \in \mathcal{F}$ is mapped is restricted to the set $\mathcal{S}(\mathrm{C})$.

Ψ_{C} restricted to $\mathcal{F}(\mathrm{C})$ is a group homomorphism because $\mathcal{F}(\mathrm{C})$ is a subgroup of $\mathrm{Sym}(V)$ by Definition and Proposition 3.12(iv). $\Psi_{\mathrm{C}}(\phi)$ restricted to $\mathcal{S}$ is a bijective because $\Psi_{\mathrm{C}}(\phi)$ is, and maps from $\mathcal{S}$ to $\mathcal{S}$ because ϕ is feasibility-preserving. □

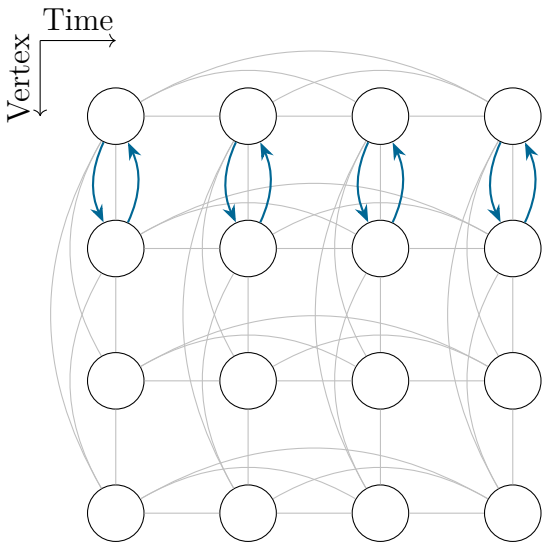
Theorem 3.16 (Koßmann et al. $\mathcal{F}(\mathrm{C}_{\mathcal{C}})$ for the Vertex-Time Encoding)
Let $\mathcal{C}$ be a binary combinatorial optimization problem and let (K_n, w) be a complete weighted graph for non-trivial $n \in \mathbb{N}$.

(i) If $\mathcal{C}$ is of scheduling type, then $\mathrm{Aut}(\mathrm{C}_{\mathcal{C}}) = \mathcal{F}(\mathrm{C}_{\mathcal{C}})$.

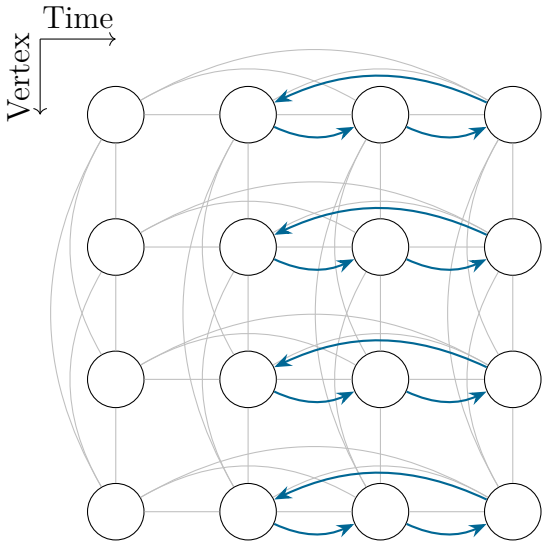
(ii) If $\mathcal{C}$ is the $\mathcal{E}$-vertex-time encoding of (K_n, w), then $\mathrm{Aut}(\mathrm{C}_{\mathcal{C}}) \cong (S_n \times S_n) \rtimes S_2$.

Theorem 3.16 compiles a significant chunk of non-trivial results of [Koß+23] (and the theses [Bin22] and [Koß22] from which it resulted) in a manner appropriate to our objectives. For its proof, we refer to those works. As far as technicalities are concerned, we merely make a few remarks about both parts:

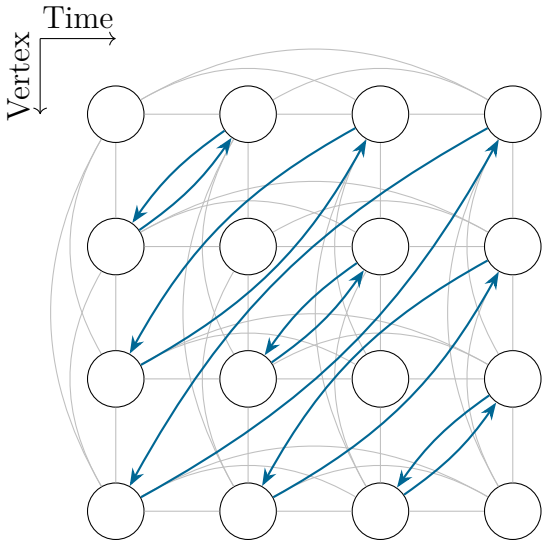
(i) To begin with, note that the inclusion $\mathrm{Aut}(\mathrm{C}) \leq \mathcal{F}(\mathrm{C})$ holds for any 'constraint graph' $(V, E, \mathcal{S})$, i.e. any graph (V, E) that possesses a notion of a solution set $\mathcal{S} \subset \mathcal{P}(V)$. The hierarchy $\mathrm{Aut}(\mathrm{C}) \leq \mathcal{F}(\mathrm{C}) \leq \mathrm{Sym}(V)$ is therefore always true. For the inclusion $\mathcal{F}(\mathrm{C}) \leq \mathrm{Aut}(\mathrm{C})$ to hold, it suffices to merely require that for all vertices $v_1, v_2 \in V$ with $v_1v_2 \notin E$ there exists a solution set $W \in \mathcal{S}$ with $v_1, v_2 \in W$. This is satisfied by all constraint graphs following Definition 3.12, but would also be satisfied for much broader classes.



(a) Visualization of the action of the subgroup $(S_4 \times \{\mathrm{id}\}) \rtimes \{\mathrm{id}\} \leq \mathrm{Aut}(\mathrm{C}_{\mathcal{C}})$ at the example of $(\tau_1 \times \mathrm{id}) \rtimes \mathrm{id}$ (blue). Interpreted as an action on the solution set $\mathcal{S}$, the permutation exchanges the timeslots at which the 1st and 2nd city are 'visited'.

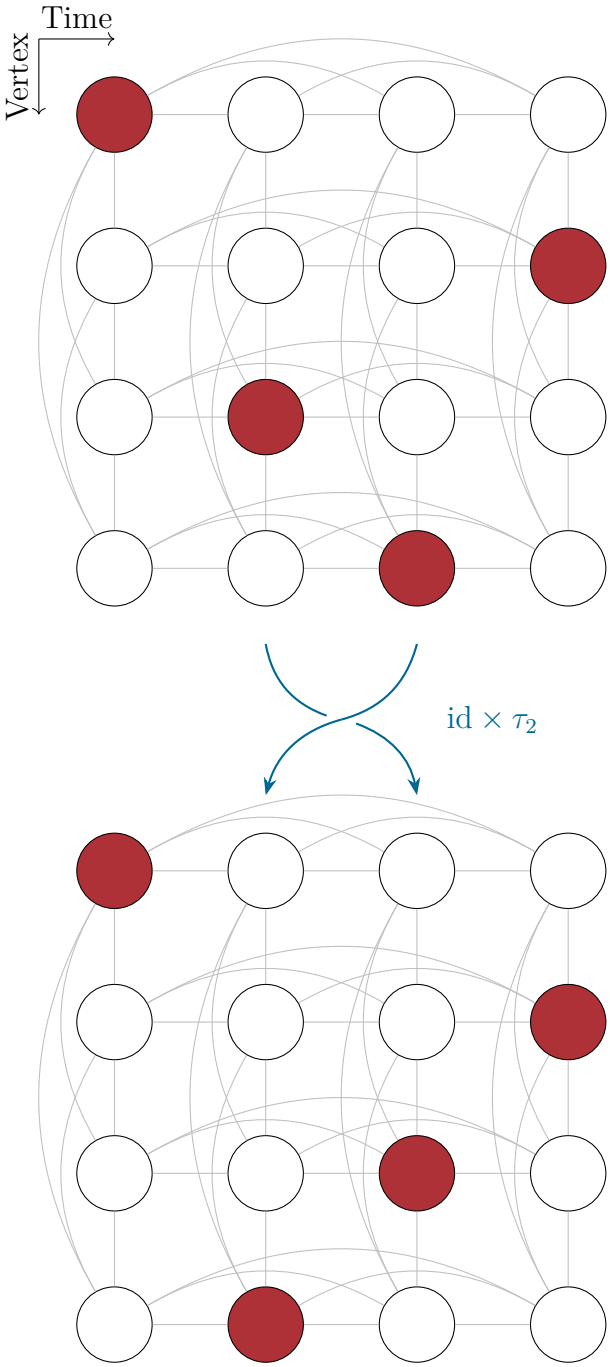


(b) Visualization of the action of the subgroup $(\{\mathrm{id}\} \times S_4) \rtimes \{\mathrm{id}\} \leq \mathrm{Aut}(\mathrm{C}_{\mathcal{C}})$ at the example of $(\mathrm{id} \times (2,3,4)) \rtimes \mathrm{id}$ (blue). Interpreted as an action on the solution set $\mathcal{S}$, the permutation cyclically permutes the cities 'visited' during the 2nd, 3rd, and 4th timeslot.

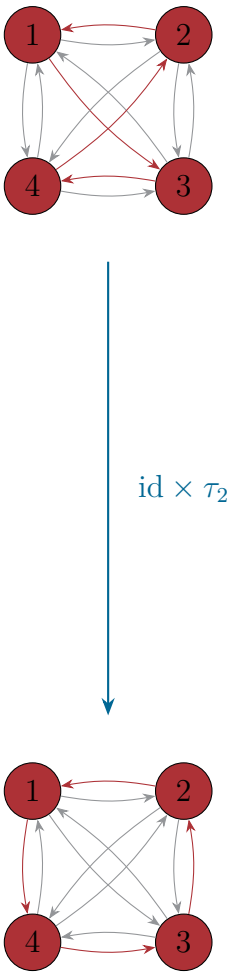


(c) Visualization of the action of the subgroup $(\{\mathrm{id}\} \times \{\mathrm{id}\}) \rtimes S_2 \leq \mathrm{Aut}(\mathrm{C}_{\mathcal{C}})$ at the example of $(\mathrm{id} \times \mathrm{id}) \rtimes \tau_1$ (blue). Interpreted as an action on the solution set $\mathcal{S}$, the permutation effectively interchanges the meaning of 'position' and 'time'.

Figure 3.3.: Let $G = (K_4^d, w)$ be a weighted graph, $\mathcal{C}$ be its $\mathcal{E}$-vertex-time encoding, and $\mathrm{C}_{\mathcal{C}}$ be the constraint graph of $\mathcal{C}$. Figures 3.3a, 3.3b, and 3.3c visualize the action of three major subgroups of $\mathrm{Aut}(\mathrm{C}_{\mathcal{C}})$ using example permutations.



(a) Visualization of the action of $\mathrm{id} \times \tau_2$ on $\mathrm{C}_{\mathcal{C}}$. All vertices of the 2nd time block $\mathcal{E}^{-1}(\Delta_2^{\mathrm{T}})$ are swapped with the vertices of the 3rd time block $\mathcal{E}^{-1}(\Delta_3^{\mathrm{T}})$ corresp. to the same city block.



(b) Visualization of the action of $\mathrm{id} \times \tau_2$ on $\mathrm{Ham}(G) \equiv \mathcal{S}$. The cities 'visited' 2nd and 3rd are swapped.

Figure 3.4.: Let $G = (K_4^d, w)$ be a weighted graph, $\mathcal{C}$ be its $\mathcal{E}$-vertex-time encoding, and $\mathrm{C}_{\mathcal{C}}$ be the constraint graph of $\mathcal{C}$. Figures 3.4a and 3.4b illustrate the action of $(\{\mathrm{id}\} \times S_4) \rtimes \{\mathrm{id}\} \leq \mathcal{F}$ at the example of $(\mathrm{id} \times \tau_2) \rtimes \mathrm{id}$ (abbr. $\mathrm{id} \times \tau_2$).

(ii) Secondly, [Koß+23] determines $\mathrm{Aut}(\mathrm{C}_{\mathcal{C}})$ for the broader class of Open-Shop Scheduling Problems $\mathrm{OSSP}(J, M, T)$. A TSP instance with n cities is given by the special case $\mathrm{OSSP}(n, 1, n)$, and, for $M \cdot T = J$, the automorphism group is given by $\mathrm{Aut}(\mathrm{C}_{\mathrm{OSSP}}) = (S_J \times S_J) \rtimes S_2$. Here, the symbol '$\rtimes$' denotes the semidirect product of groups [cf. Rob96, pp. 27–28]. While the permutation $(\mathrm{id} \times \mathrm{id}) \rtimes \tau_1$ does represent an intriguing action on the solution set (cf. Figure 3.3c), we will mostly ignore it due to lack of necessity (cf. Proposition 3.17), which is the reason why the introduction of the semidirect product was omitted in Section 2.2. Finally, note that the isomorphism between $\mathrm{Aut}(\mathrm{C}_{\mathcal{C}})$ and $(S_n \times S_n) \leq (S_n \times S_n) \rtimes S_2$ is simply id_V provided that the constraint graph $\mathrm{C}_{\mathcal{C}}$ of the $\mathcal{E}$-vertex-time encoding uses $V = [n]^2$ and $\iota = \mathcal{E}^{-1}$.

To gain insight regarding the behavior of the developed constructions, we again refer to visualizations: Figure 3.3 explores the behavior of three major subgroups of $\mathrm{Aut}(\mathrm{C}_{\mathcal{C}}) = (S_n \times S_n) \rtimes S_2$ for the $\mathcal{E}$-vertex-time encoding $\mathcal{C}$. Figure 3.4 examines how these subgroup's behavior translates to transformations of Hamiltonian cycles of the original problem $\mathrm{TSP}(G)$. We conclude with the following ...

Definition and Proposition 3.17 (Vertex-Group and Time-Group for C)
Let $\mathcal{C}$ be the $\mathcal{E}$-vertex-time encoding of the complete weighted graph (K_n, w) and let C be its constraint graph.

(i) The **C-vertex-group** $S_n^{\mathrm{V}} = (S_n \times \{\mathrm{id}\}) \rtimes \{\mathrm{id}\} \leq \mathcal{F}$ acts transitively on $\mathcal{S}(\mathrm{C})$ via Ψ_{C}.

(ii) The **C-time-group** $S_n^{\mathrm{T}} = (\{\mathrm{id}\} \times S_n) \rtimes \{\mathrm{id}\} \leq \mathcal{F}$ acts transitively on $\mathcal{S}(\mathrm{C})$ via Ψ_{C}.

Proof
Follows from [Koß+23, p. 5]. □

3.3.3. Transitive Group Actions on $\mathcal{S}(\mathcal{C})$

With S_n^{V} and S_n^{T}, we have identified two subgroups of $\mathrm{Sym}(V)$ that act transitively on the constraint graph's set of feasible solutions $\mathcal{S}(\mathrm{C})$. Since V and $[p]$ are in bijection via the indexing map ι, and $\mathcal{S}(\mathrm{C})$ and $\mathcal{S}(\mathcal{C})$ are in bijection via the identification map κ_ι, we can directly utilize this to identify two subgroups of S_p that act transitively on $\mathcal{S}(\mathcal{C})$.

Corollary 3.18 (The Induced Isomorphisms ϕ_ι and ϕ_{κ_ι})
Let $\mathcal{C} = (p, q, c, f, \mathrm{g})$ be a scheduling encoding and let C be its constraint graph.[40]

(i) ϕ_ι is a group isomorphism from S_p to $\mathrm{Sym}(V)$.

(ii) ϕ_{κ_ι} is a group isomorphism from $\mathrm{Sym}(\mathcal{S}(\mathcal{C}))$ to $\mathrm{Sym}(\mathcal{S}(\mathrm{C}))$.

Proof

(i) Follows from Definition and Proposition 2.16 by condition (C3) of Definition 3.12(i).

(ii) Follows from Definition and Proposition 2.16 by Definition and Proposition 3.12(iii), since $\kappa_\iota|_{\mathcal{S}(\mathcal{C})}$ being a bijection from $\mathcal{S}(\mathcal{C})$ to $\mathcal{S}(\mathrm{C})$ follows from Definition 3.12(iii) and Proposition 3.13. □

Definition and Proposition 3.19 (The Group Action Φ_n)
Let $n \in \mathbb{N}$, then the group action Φ_n of S_n on $\mathbb{b}^n$ is defined by

$$\begin{aligned} \Phi_n : S_n &\to \mathrm{Sym}(\mathbb{b}^n) \\ \sigma &\mapsto \left(\delta_{\sigma^{-1}(i),\,j}\right)_{1\leq i,j\leq n} \end{aligned}$$

and satisfies $\Phi_n(\sigma)(b_1,\ldots,b_n) = \left(b_{\sigma^{-1}(1)},\ldots,b_{\sigma^{-1}(n)}\right)$ for all $\sigma \in S_n$ and $b \in \mathbb{b}^n$.

Before moving on to the proof, consider that we will occasionally omit the subscript if the integer $n \in \mathbb{N}$ is clear from the context (as in the following, for example, where it is clear that $n \in \mathbb{N}$ is arbitrary).

Proof
We prove that Φ is (i) well-defined, (ii) a homomorphism, and (iii) satisfies the stated relation.

(i) Let $\sigma \in S_n$. $\Phi(\sigma)$ is a bijection because it is a permutation matrix and therefore, in particular, an element of the orthogonal group of order n:

$$\begin{aligned} \left(\Phi(\sigma)\cdot\Phi(\sigma)^T\right)_{ik} &= \sum_{j=1}^{n}\left(\Phi(\sigma)\right)_{ij}\left(\Phi(\sigma)^T\right)_{jk} = \sum_{j=1}^{n}\Phi(\sigma)_{ij}\Phi(\sigma)_{kj} \\ &= \sum_{j=1}^{n}\delta_{\sigma^{-1}(i),\,j}\delta_{\sigma^{-1}(k),\,j} = \delta_{\sigma^{-1}(i),\,\sigma^{-1}(k)} = \delta_{ik} \end{aligned}$$

Hence, $\Phi(\sigma)$ is a bijection from $\mathbb{R}^n$ to $\mathbb{R}^n$, implying that its restriction to $\mathbb{b}^n$ is a bijection from $\mathbb{b}^n$ to $\Phi_\sigma(\mathbb{b}^n)$. Moreover, $\Phi_\sigma(\mathbb{b}^n) \subset \mathbb{b}^n$ follows from

$$\left(\Phi(\sigma)\;b\right)_i = \sum_{j=1}^{n}\Phi(\sigma)_{ij}\;b_j = \sum_{j=1}^{n}\delta_{\sigma^{-1}(i),\,j}\;b_j = b_{\sigma^{-1}(i)} \in \mathbb{b}$$

and implies $\Phi_\sigma(\mathbb{b}^n) = \mathbb{b}^n$ due to a cardinality argument by the bijectivity of Φ.

[40] The map of (ii) is more accurately be denoted by $\phi_{\left(\kappa_\iota|_{\mathcal{S}(\mathcal{C})}\right)}$. The restriction was omitted for readability.

(ii) Let $\sigma_1, \sigma_2 \in S_n$, then $\Phi(\sigma_1)\ \Phi(\sigma_2) = \Phi(\sigma_1\ \sigma_2)$ is proven component-wise as follows.

$$\begin{aligned}\Big(\Phi(\sigma_1) \cdot \Phi(\sigma_2)\Big)_{ik} &= \sum_{j=1}^{n} \Phi(\sigma_1)_{ij}\ \Phi(\sigma_2)_{jk} = \sum_{j=1}^{n} \delta_{\sigma_1^{-1}(i),\,j}\ \delta_{\sigma_2^{-1}(j),\,k} \\ &= \sum_{j=1}^{n} \delta_{\sigma_1^{-1}(i),\,j}\ \delta_{j,\,\sigma_2(k)} = \delta_{\sigma_1^{-1}(i),\,\sigma_2(k)} = \delta_{\sigma_2^{-1}(\sigma_1^{-1}(i)),\,k} \\ &= \delta_{(\sigma_1\circ\sigma_2)^{-1}(i),\,k} = \Big(\Phi(\sigma_1 \circ \sigma_2)\Big)_{ik}\end{aligned}$$

(iii) $\Phi_\sigma(b_1, \ldots, b_n) = \Big(b_{\sigma^{-1}(1)}, \ldots, b_{\sigma^{-1}(n)}\Big)$ is implied by the last equation of part (i) □

Proposition 3.20 (Φ Corresponds to Ψ)
Let $\mathcal{C} = (p, q, c, f, \mathrm{g})$ be a scheduling encoding and let $\mathrm{C} = (V, E, \iota, l)$ be its constraint graph, then $\Phi_p = \phi_{\kappa_\iota}^{-1} \circ \Psi_\mathrm{C} \circ \phi_\iota$.

Proof
Follows from [Bin22, pp. 34–36] by Remark 3.22, appended to the end of this chapter.□

Definition and Proposition 3.21 (Vertex-Group and Time-Group for $\mathcal{C}$)
Let $\mathcal{C}$ be the $\mathcal{E}$-vertex-time encoding of the complete weighted graph (K_n, w).

(i) The **$\mathcal{C}$-vertex-group** $\mathrm{S}_n^\mathrm{V} = \phi_\mathcal{E}(S_n^\mathrm{V}) \leq S_{n^2}$ acts transitively on $\mathcal{S}(\mathcal{C})$ via Φ.

(ii) The **$\mathcal{C}$-time-group** $\mathrm{S}_n^\mathrm{T} = \phi_\mathcal{E}(S_n^\mathrm{T}) \leq S_{n^2}$ acts transitively on $\mathcal{S}(\mathcal{C})$ via Φ.

Proof
Follows readily from Definition and Proposition 3.17 using Proposition 3.13 and Proposition 3.20. □

There exists a canonical isomorphism between symmetric group S_n and the C-vertex group $S_n^\mathrm{V} = (S_n \times \{\mathrm{id}\}) \rtimes \{\mathrm{id}\}$ as well as the C-time group $S_n^\mathrm{T} = (\{\mathrm{id}\} \times S_n) \rtimes \{\mathrm{id}\}$. The isomorphism between S_n^V and S_n^V as well as between S_n^T and S_n^T is equally canonical by definition. Hence, we will occasionally not differentiate between these groups when it eases nomenclature (cf. the proof of Theorem 5.2).

Remark 3.22 (Connection to [Bin22, pp. 33–36])
The Definitions and Propositions 3.14, 3.19, 3.20, 4.6, and 4.7 of this work exhaustively reproduce the results of [Bin22, pp. 33–36]. Apart from the fact that our work is built on these results, which motivates their introduction, this is done for the following reason:

We believe there to be an error in the original work that is ultimately inconsequential but still wish not to propagate. Specifically, we believe [Bin22, pp. 33–36] contains two

mistakes that 'cancel out' in such a manner that the overall construction nevertheless functions correctly: First, the map Φ from [Bin22, p. 34] is actually an antihomomorphism, i.e. $\Phi(\sigma_1\sigma_2) = \Phi(\sigma_2)\Phi(\sigma_1)$ holds. As a minimal counterexample to the homomorphism claim, consider $\sigma_1 = \tau_1 \in S_3$ and $\sigma_2 = \tau_2 \in S_3$ with $\sigma_1\sigma_2 = (1,2,3)$ resulting in

$$\Phi(\sigma_1\sigma_2)(z_1, z_2, z_3) = (z_{(\sigma_1\sigma_2)(1)}, z_{(\sigma_1\sigma_2)(2)}, z_{(\sigma_1\sigma_2)(3)}) = (z_2, z_3, z_1)$$

and

$$\begin{aligned}\Phi(\sigma_2)(z_1, z_2, z_3) &= (z_{\sigma_2(1)}, z_{\sigma_2(2)}, z_{\sigma_2(3)}) = (z_1, z_3, z_2) = (z'_1, z'_2, z'_3),\\ \Phi(\sigma_1)(z'_1, z'_2, z'_3) &= (z'_{\sigma_1(1)}, z'_{\sigma_1(2)}, z'_{\sigma_1(3)}) = (z'_2, z'_1, z'_3) = (z_3, z_1, z_2),\end{aligned}$$

i.e. $\Phi(\sigma_1\sigma_2)(z) = (z_2, z_3, z_1) \neq (z_3, z_1, z_2) = [\Phi(\sigma_1) \circ \Phi(\sigma_2)](z)$. The map defined by us in Definition 3.19 is the composition of Φ from [Bin22, p. 34] with the antihomomorphism $\sigma \mapsto \sigma^{-1}$, producing a homomorphism. Second, we believe the proof of $\Pi = \tilde{\kappa} \circ \Phi$ from [Bin22, p. 35] to be erroneous. Particularly, $\kappa(z_{\tau(I)})$ is not equal to $(\kappa \circ \Phi(\tau))(z_I)$ but to $(\kappa \circ \Phi(\tau^{-1}))(z_I)$ because $z_{\tau(I)}$ is not equal to $\Phi(\tau)(z_I)$ but to $\Phi(\tau^{-1})(z_I)$. As a minimal counterexample to the equality of the first two terms, consider $\tau = (1,2,3)$ and $I = \{3\}$ with $z_I = (0,0,1) = (z_1, z_2, z_3)$ resulting in

$$\Phi(\tau)(z_I) = \Phi(\tau)(z_1, z_2, z_3) = (z_{\tau(1)}, z_{\tau(2)}, z_{\tau(3)}) = (z_2, z_3, z_1) = (0,1,0),$$

$$z_{\tau(I)} = z_{\tau(\{3\})} = z_{\{\tau(3)\}} = z_{\{1\}} = (1,0,0).$$

On the contrary, $z_{\tau(I)} = \Phi(\tau^{-1})(z_I)$ is shown as follows: The i-th component of

$$\Phi(\tau^{-1})(z_I) = \Phi(\tau^{-1})\Big((z_I)_1, \dots, (z_I)_N\Big) = \Big((z_I)_{\tau^{-1}(1)}, \dots, (z_I)_{\tau^{-1}(N)}\Big)$$

is given by $(z_I)_{\tau^{-1}(i)}$ and satisfies

$$(z_I)_{\tau^{-1}(i)} = 1 \;\Leftrightarrow\; \tau^{-1}(i) \in I \;\Leftrightarrow\; i \in \tau(I) \;\Leftrightarrow (z_{\tau(I)})_i = 1$$

as well as

$$(z_I)_{\tau^{-1}(i)} = 0 \;\Leftrightarrow\; \tau^{-1}(i) \notin I \;\Leftrightarrow\; i \notin \tau(I) \;\Leftrightarrow\; (z_{\tau(I)})_i = 0.$$

Thus, $z_{\tau(I)} = \Phi(\tau^{-1})(z_I)$.

However, if Φ from [Bin22, p. 34] is redefined as the map from Definition 3.19, i.e. as the composition of Φ from [Bin22, p. 34] and $\sigma \mapsto \sigma^{-1}$, then the equality $\Pi = \tilde{\kappa} \circ \Phi$ from [Bin22, p. 35] holds, the diagrams from [Bin22, p. 36] commute, and subsequent results such as [Bin22, Theorem 4.7] nevertheless remain true (cf. its proof from [Bin22, p. 36]). Thus, we can build upon and invoke these results.[41]

[41] Thanks to Lennart Binkowski for pointing out that the definition of Φ from [Bin22, p. 34] is flawed.

Physical Prelude: Quantum Computation

In this chapter, we close the gap between optimization problems and transitive group actions on their feasible search spaces on the one hand, and quantum algorithms on the other. Beginning in Section 4.1, we briefly review fundamental notions of quantum computing. By no means shall this serve as a comprehensive or self-contained initiation to the topic; for that, we refer to the many introductions written on the subject [cf. LaP21; Zyg18; Sch19; Won22; MN19b], chiefly [NC10]. A rough overview of the most prominent quantum algorithms is provided by [Mon16a]; for quantum *optimization* algorithms specifically, a recent and holistic overview is given by [Abb+23]. Instead, similarly to Section 2.1, we merely recapitulate the most important concepts subsequently used. Then, in Section 4.2, we inspect the origin and functionality of quantum optimization algorithms belonging to the alternating operator ansatz class. The explanation of these algorithms is repeated so often within the field that we again restrict ourselves to the briefest of overviews, and refer to the original works [Far+00; FGG14; Had+19] or to the many reviews thereof [cf. CK19; Cer+21; Ble+23] for more details.[42] Afterward, we elaborate on how [Koß+23] connects group-theoretic considerations to the construction of an algorithm of this class. In particular, rigorous definitions for various concepts are provided.

4.1. Quantum Gates and Quantum Circuits

As is common in quantum computation [cf. NC10], we denote the Pauli operators by X, Y, and Z. The eigenbases of X, Y, and Z are respectively given by the $\{|\pm\rangle\}$, $\{|\pm i\rangle\}$, and the computational basis, implying that their matrix representations with respect to the computational basis are

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

[42] As this is a thesis in physics, I chose to put more focus on elaborating the topics *not* directly concerning physics, such as group theory and or encoding optimization problems, in contrast to e.g. quantum mechanics and quantum computation.

The rotation operators around the X-, Y-, and Z-axis are the operators

$$R_X(\theta) = e^{-i\theta X/2}, \qquad R_Y(\theta) = e^{-i\theta Y/2}, \qquad R_Z(\theta) = e^{-i\theta Z/2}.$$

When visualizing a qubit with the Bloch sphere, the states on its surface found along the x-, y-, and z-coordinate axes correspond respectively to these eigenbases of X, Y, and Z. The action of a Pauli operator corresponds to a $\pi/2$ rotation around its respective axis, and the action of a rotation operator corresponds to a θ rotation around its axis. The Hadamard,[43] phase, and T-gate are the operators whose matrix representations with respect to the computational basis are

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \qquad S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}, \qquad T = \begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}.$$

If $U \in U(\mathfrak{q}^{\otimes m})$ is an arbitrary unitary, then $U_k \in U(\mathfrak{q}^{\otimes(m\cdot n)})$ is the tensor product of U with a suitable number of identity operators $\mathbb{I} \in U(\mathfrak{q}^{\otimes m})$, i.e.

$$U_k = \underbrace{\mathbb{I} \otimes \ldots \otimes \mathbb{I}}_{(k-1)\text{ times}} \otimes U \otimes \overbrace{\mathbb{I} \otimes \ldots \otimes \mathbb{I}}^{(n-k)\text{ times}}.$$

n is assumed to be clear from the context. Also, the controlled-U operators or controlled-U gates are the unitary operators $C_1U_2 \in \mathfrak{q}^{\otimes(1+k)}$ and $C_2U_1 \in \mathfrak{q}^{\otimes(k+1)}$ given by

$$C_1U_2 = |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes U, \qquad C_2U_1 = \mathbb{I} \otimes |0\rangle\langle 0| + U \otimes |1\rangle\langle 1|.$$

Notably, this naturally generalizes to index tuples other than $(1,2)$ or $(2,1)$ if the operator is embedded in a larger Hilbert space but only acts non-trivially on the specified subspaces. If no indices are specified, then the index tuple is assumed to be $(1,2)$, i.e. $CU = C_1U_2$. Moreover, the CNOT gate is a longhand notation for the CX gate, as the action of the Pauli operator X on a computational basis state corresponds to a bit flip.

Important multi-qubit gates are the SWAP-gate, the Toffoli-gate C^2NOT, and the Fredkin-gate CSWAP given by

$$\text{SWAP} = |00\rangle\langle 00| + |10\rangle\langle 01| + |01\rangle\langle 10| + |11\rangle\langle 11|$$

$$C^2\text{NOT} = (|0\rangle\langle 0|)^{\otimes 2} \otimes \mathbb{I} + (|1\rangle\langle 1|)^{\otimes 2} \otimes X$$

$$C\text{SWAP} = |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes \text{SWAP}$$

One standard methodology of concretizing quantum algorithms is by drawing their quantum circuits, which is the quantum computational equivalent of a Boolean circuit. Figure 4.1 below visualizes an example circuit.

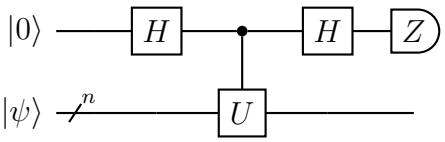


Figure 4.1.: A quantum circuit that performs the so-called Hadamard test with respect to the unitary $U \in U(\mathfrak{q}^{\otimes n})$ for a state $|\psi\rangle$. Specifically, the circuit is equivalent to performing the unitary $H_1 \cdot C_1 U_2 \cdot H_1$ on the state $|0\rangle |\psi\rangle \in \mathfrak{q}^{\otimes(1+n)}$ followed by a measurement of the first qubit in the eigenbasis of Z. The Hadamard test can be used to obtain an estimate of $\operatorname{Re} \langle\psi| U |\psi\rangle$.[44]

4.2. The Quantum Alternating Operator Ansatz

4.2.1. The Algorithm and Its Origin

As discussed in the introduction to this thesis, the Quantum Adiabatic Algorithm [Far+00] provides a method to solve search problems using inherently quantum mechanical procedures. Assume H_B is a Hamiltonian whose ground state is easy to construct and let H_P be a Hamiltonian whose ground state encodes the solution to a search problem. The algorithm then simulates the evolution of a system whose time-dependent Hamiltonian H interpolates between H_B and H_P, i.e.

$$H(t) = \left(1 - \frac{t}{T}\right) H_B + \frac{t}{T} H_P,$$

before measuring the system's state to obtain the desired result. Provided the evolution occurs slowly enough, i.e. provided that T is large enough compared to the spectral gap between the smallest eigenvalue and the second smallest eigenvalue of $H(t)$, the adiabatic theorem ensures that the system will remain in or close to its ground state.

The Quantum Approximate Optimization Algorithm [FGG14] can be understood as a variant of the Quantum Adiabatic Algorithm that trades in convergence guarantees for computational speed by Trotterizing the Hamiltonian's evolution p times: The evolution of two Hamiltonians, the mixing Hamiltonian B and the cost Hamiltonian C, is applied in an alternating fashion for times β_i and γ_i. C encodes an objective function f to be optimized as an operator diagonal in the computational basis, i.e. $C |b\rangle = f(b) |b\rangle$, whereas B is given $B = \sum_{i=1}^{n} X_i$. Respectively, these Hamiltonians loosely correspond to H_P and H_B. The alternating evolution is done to prepare the state

$$|\vec{\gamma}, \vec{\beta}\rangle = \left(\prod_{i=1}^{p} e^{-i\gamma_i C} e^{-i\beta_i B}\right) |\text{in}\rangle$$

[44] This and all following quantum circuits were drawn using the *quantikz* package for LaTeX [cf. Kay23].

where $|\text{in}\rangle$ is the algorithm's initial state $|\text{in}\rangle = |+\rangle^{\otimes n}$. β_i and γ_i are referred to as *angles*. One considerable strength of the original algorithm was the ability to identify angles beforehand that, for the considered maximum cut problem, were able to guarantee a certain approximation ratio, i.e. a lower bound was given for the quotient of the quality of the obtained solution and the quality of the optimal solution. In the general case, the angles are not treated as parameters to be calculated beforehand but rather as continuously varying[45] parameters. In a feedback loop, a classical optimization algorithm is tasked to determine angles optimizing $\langle\vec{\gamma},\vec{\beta}|\, C\,|\vec{\gamma},\vec{\beta}\rangle$, which is repeatedly estimated by using e.g. Hadamard test techniques or error-mitigating versions thereof [cf. Cle+98; Koc21; Hug+21]. Eventually, $|\vec{\gamma},\vec{\beta}\rangle$ is again prepared and then measured to obtain a solution.

The Quantum Alternating Operator Ansatz proposed by Stuart Hadfield et al. [Had+19] generalizes the ideas behind the Quantum Approximate Optimization Algorithm to provide an abstract paradigm for the design of variational quantum algorithms. Chiefly, instead of fixing components of the algorithm with specific definitions, it alternatively provided conditions that these components must satisfy. In particular, this was done with a heightened conceptual consideration of constraint optimization problems. For example, the mixing Hamiltonian is not fixed to be $\sum_i X_i$ but instead defined as a family of operators satisfying certain criteria that ensure they (i) preserve and (ii) explore the subspace of feasible solutions.

4.2.2. Phase Separators and Mixing Families

After providing a high-level synopsis of the development of and ideas behind the Quantum Alternating Operator Ansatz, we now rigorously define various parts of the algorithm. For that, we closely follow [Had+19] and [Koß+23]. We begin by defining the objective Hamiltonian. Note that the Quantum Alternating Operator Ansatz replaces the objective Hamiltonian C from the Quantum Approximate Optimization Algorithm with the notion of a phase separator Hamiltonian (cf. Definition 4.3) and that the objective Hamiltonian H_f defined below is just one candidate satisfying the conditions posed on a phase separator (cf. Corollary 4.4).

Definition and Proposition 4.1 (Objective Hamiltonian)
The **objective Hamiltonian mapping of order** $n \in \mathbb{N}$ is the map H defined by

$$\begin{aligned} H : \mathrm{Map}(\mathfrak{b}^n, \mathbb{R}) &\to \mathrm{Herm}(\mathfrak{q}^{\otimes n}) \\ f &\mapsto H_f = \sum_{b\in\mathfrak{b}^n} f(b)\,|b\rangle\,\langle b| \end{aligned}$$

[45] Hence the name *variational* quantum algorithm.

Proof
Clearly, H_f is a linear map for all $f \in \mathrm{Map}(\mathbb{b}^n, \mathbb{R})$ because it is the weighted sum of linear maps $|b\rangle\langle b|$, i.e. $H(\mathrm{Map}(\mathbb{b}^n, \mathbb{R})) \subset \mathcal{L}(\mathfrak{q}^{\otimes n})$. Further, by the definition of H, the matrix representation of H_f with respect to the computational basis is a diagonal matrix with real entries and therefore Hermitian. Thus, $H(\mathrm{Map}(\mathbb{b}^n, \mathbb{R})) \subset \mathrm{Herm}(\mathfrak{q}^{\otimes n})$. □

As indicated, such mappings of maps on bitstrings to Hamiltonians diagonal in the computational basis are a standard tool within quantum optimization [cf. FGG14, p. 2; Had+19, p. 4; Dom+23, p. 1; Koß+23, p. 4]. Next, we wish to concretize e.g. what it means for a mixing family to *explore* or *preserve* the subspace of feasible solutions. For that, we first translate concepts discussed for binary combinatorial optimization problems to multi-qubit Hilbert spaces.

Definition 4.2 (Quantum Search Space)
Let $\mathcal{C} = (p, q, c, f, \mathrm{g})$ be a binary combinatorial optimization problem.

(i) The **total quantum search space of** $\mathcal{C}$ is the Hilbert space

$$\mathcal{Q}_{\mathrm{tot}}(\mathcal{C}) = \mathfrak{q}^{\otimes p}.$$

(ii) The **feasible quantum search space of** $\mathcal{C}$ is the Hilbert space

$$\mathcal{Q}(\mathcal{C}) = \mathrm{span}\{\, |b\rangle \in \mathcal{Q}_{\mathrm{tot}}(\mathcal{C}) : b \in \mathcal{S}(\mathcal{C}) \,\}.$$

(iii) The **optimal quantum search space of** $\mathcal{C}$ is the Hilbert space

$$\mathcal{Q}_{\mathrm{op}}(\mathcal{C}) = \mathrm{span}\{\, |b\rangle \in \mathcal{Q}(\mathcal{C}) : b \in \mathcal{S}_{\mathrm{op}}(\mathcal{C}) \,\}.$$

If the associated binary combinatorial optimization problem is clear from the context, $\mathcal{Q}_{\mathrm{tot}}(\mathcal{C})$, $\mathcal{Q}(\mathcal{C})$, and $\mathcal{Q}_{\mathrm{op}}(\mathcal{C})$ are respectively abbreviated by $\mathcal{Q}_{\mathrm{tot}}$, $\mathcal{Q}$, and $\mathcal{Q}_{\mathrm{op}}$.

Definition 4.3 (Phase Separators)
Let $\mathcal{C} = (p, q, c, f, g)$ be a binary combinatorial optimization problem.

(i) A **phase separator Hamiltonian** of $\mathcal{C}$ is an operator $H \in \mathrm{Herm}(\mathcal{Q}_{\mathrm{tot}})$ that satisfies the following conditions.

(P1) H is diagonal in the combinational basis.

(P2) The eigenspace of $H|_{\mathcal{Q}}$ corresponding to the smallest eigenvalue is $\mathcal{Q}_{\mathrm{op}}$.

(ii) The **parameterized phase seperator** corresponding to a phase separator Hamiltonian H of $\mathcal{C}$ is the operator $U_P(H, \gamma) = e^{-i\gamma H}$.

Corollary 4.4 (Objective Hamiltonians are Phase Separators)
If $\mathcal{C} = (p, q, c, f, g)$ is a binary combinatorial optimization problem, then the objective Hamiltonian H_f is a phase separator Hamiltonian of $\mathcal{C}$.

Definition 4.5 (Mixing Families)
Let $\mathcal{C} = (p, q, c, f, g)$ be a binary combinatorial optimization problem and let $m \in \mathbb{N}$.

(i) A linear operator $A \in \mathcal{L}(\mathcal{Q}_{\text{tot}})$ is called **feasibility-preserving** with respect to $\mathcal{C}$ if it satisfies $A|b\rangle \in \mathcal{Q}$ for all $|b\rangle \in \mathcal{Q}$. A family $\{A_i\}_{i\in[m]}$ of linear operators $A_i \in \mathcal{L}(\mathcal{Q}_{\text{tot}})$ is called **feasibility-preserving** with respect to $\mathcal{C}$ if all operators A_i are feasibility-preserving with respect to $\mathcal{C}$. A family $\{A_i\}_{i\in[m]}$ of linear operators $A_i \in \mathcal{L}(\mathcal{Q}_{\text{tot}})$ is called **problem-mixing** with respect to $\mathcal{C}$ if any coordinate subspace of $\mathcal{Q}$ left invariant under all A_i is trivial.

(ii) A **mixing family** of $\mathcal{C}$ is a family $\{H_i\}_{i\in[m]}$ of operators $H_i \in \text{Herm}_{\geq 0}(\mathcal{Q}_{\text{tot}})$ that is feasibility-preserving and problem-mixing with respect to $\mathcal{C}$.

(iii) The **parameterized simultaneous mixer** corresponding to a mixing family $\{H_i\}_{i\in[m]}$ of $\mathcal{C}$ is the operator

$$U_M(\{H_i\}, \beta) = e^{-i\beta \sum_{i\in[m]} H_i}.$$

(iv) The **parameterized sequential mixer** corresponding to a mixing family $\{H_i\}_{i\in[m]}$ of $\mathcal{C}$ and $\sigma \in S_m$ is the operator

$$U_{M,\sigma}(\{H_i\}, \beta) = \prod_{i\in[m]} e^{-i\beta H_{\sigma(i)}}.$$

The purpose of a mixing family is to construct a mixer that allows the algorithm to explore the subspace of feasible solutions: For all pairs of feasible states $|b_1\rangle, |b_2\rangle \in \mathcal{Q}$ there exist angles β_i such that the repeated application of the mixer generates an overlap between these states, i.e. $\langle b_1| U_M(\{H_i\}, \beta_r) \dots U_M(\{H_i\}, \beta_1) |b_2\rangle \neq 0$. Further, the mixer *only* explores the subspace of feasible solutions, i.e. if the quantum computer is initialized in a state $|\text{in}\rangle \in \mathcal{Q}$, then it will not leave $\mathcal{Q}$. A phase separator Hamiltonian, by virtue of being diagonal in the computational basis, does not explore the feasible subspace (albeit it trivially preserves feasibility). Instead, the application of the parameterized phase separator provides flexibility in weighing different paths of exploration.

Crucially, the central idea of [Koß+23] is to transfer the preceding analysis of the classical feasibility-structure of binary combinatorial optimization problems to the construction of mixing families. A permutation π acting on the total solution space $\mathcal{S}_{\text{tot}}(\mathcal{C})$ is mapped to a unitary operator acting on $\mathcal{Q}_{\text{tot}}(\mathcal{C})$ via $\pi \cdot |b\rangle = |\pi b\rangle$. If the permutation acting on $\mathcal{S}_{\text{tot}}(\mathcal{C})$ is feasibility-preserving, then so is its corresponding operator acting on

$\mathcal{Q}_{\text{tot}}(\mathcal{C})$. If the group action is transitive on $\mathcal{S}(\mathcal{C})$, then for all $b_1, b_2 \in \mathcal{S}(\mathcal{C})$ there exists a permutation π such that the corresponding operator satisfies $\pi \cdot |b_1\rangle = |b_2\rangle$. Now, taking suitable matrix logarithms of these unitary operators yields a mixing family of $\mathcal{C}$ [cf. Koß+23, p. 7; Bin22, p. 36]. However, one can do better. If the permutations are involutions, then by Proposition 2.13(ii) the corresponding operators are Hamiltonians (cf. Section 5.3) and themselves constitute a mixing family. Now, mapping all permutations to operators independently is infeasible, as the vertex-group $\mathrm{S}_N^{\mathrm{V}}$ and the time-group $\mathrm{S}_n^{\mathrm{T}}$ both contain factorially many elements.[46] But any permutation can be written as the product of polynomially many fixed, involutory permutations $\sigma_1, \ldots, \sigma_m$. These σ_i constitute a decomposition of the symmetric group, and if these are mapped to mixers separately, then one can ensure that there exist angles such that any solution is reachable, i.e. such that for all $b_1, b_2 \in \mathcal{S}(\mathcal{C})$ there exist angles β_i such that

$$U(\sigma_1, \ldots, \sigma_m; \beta_1, \ldots, \beta_m) \, |b_1\rangle \propto |b_2\rangle \, .$$

In particular, these is no longer a need for phase-separators!

For the rest of this chapter, we merely provide rigorous proofs of the ideas we have just elaborated. Specifically, we first define the group homomorphism λ from $\mathrm{Sym}(\mathbb{b}^p) = \mathrm{Sym}(\mathcal{S}_{\text{tot}})$ to $U(\mathfrak{q}^{\otimes p}) = U(\mathcal{Q}_{\text{tot}})$ that realizes the group action $\pi \cdot |b\rangle = |\pi b\rangle$ (Definition and Proposition 4.6). Then, we define the unitary representation $\Lambda = \lambda \circ \Phi$, i.e. the group homomorphism Λ from S_p to $U(\mathfrak{q}^{\otimes p}) = U(\mathcal{Q}_{\text{tot}})$ (Definition and Proposition 4.7). Using Λ, the $\mathcal{C}$-vertex group $\mathrm{S}_n^{\mathrm{V}} \leq S_{n^2} = S_p$ and the $\mathcal{C}$-time group $\mathrm{S}_n^{\mathrm{T}} \leq S_{n^2} = S_p$ act on the feasible quantum search space $\mathcal{Q}$. With that, we will be set up to, in the next chapter, search for decompositions of $S_n \cong \mathrm{S}_n^{\mathrm{V}}$ or $S_n \cong \mathrm{S}_n^{\mathrm{T}}$ that are more advantageous to construct mixers with. In particular, we will then give more rigorous definitions of what was meant above with a 'decomposition' of the symmetric group (Definitions 5.1 and 5.18) and how to construct the operators $U(\sigma_1, \ldots, \sigma_m; \beta_1, \ldots, \beta_m)$ (Theorem 5.2).

Definition and Proposition 4.6 (The Homomorphism λ)
Let $n \in \mathbb{N}$, then the group homomorphism λ from $\mathrm{Sym}(\mathbb{b}^n)$ to $U(\mathfrak{q}^{\otimes n})$ is defined by

$$\begin{aligned} \lambda : \mathrm{Sym}(\mathbb{b}^n) &\to U(\mathfrak{q}^{\otimes n}) \\ \pi &\mapsto \sum\nolimits_{b,\, b' \in \mathbb{b}^n} \delta_{\pi^{-1}(b),\, b'} \, |b\rangle \langle b'| \end{aligned}$$

and satisfies $\lambda_\pi |b\rangle = |\pi b\rangle$ for all $\pi \in \mathrm{Sym}(\mathbb{b}^n)$ and $b \in \mathbb{b}^n$.

As a technical remark, one may note that until now we denoted bitstrings by $b \in \mathbb{b}^n$ such that their i-th coefficient could be denoted by $b_i \in \mathbb{b}$. In some of what follows, we will not need to 'access' individual coefficients while simultaneously using more than

[46] In general, one readily verifies that any group acting transitively on a set must have at least the same cardinality as said set. Since the Traveling Salesman Problem has factorially many solutions (cf. Proposition 2.29), this scaling is unavoidable.

two bitstrings. Hence, we will occasionally use $b_1, b_2, \ldots$ to denote bitstrings and not bits. If that is the case, it will be clear.

Proof
We prove that λ is (i) well-defined, (ii) a homomorphism, and (iii) satisfies the stated relation.

(i) Let $\pi \in \mathrm{Sym}(\mathbb{b}^n)$. Clearly, $\lambda(\pi)$ is a linear operator on $\mathfrak{q}^{\otimes n}$ because it is the weighted sum of linear operators $|b\rangle\langle b'|$. Further, $\lambda(\pi)$ is unitary, as is shown by a direct calculation:

$$\begin{aligned}
\Big(\lambda(\pi)\Big)\Big(\lambda(\pi)\Big)^\dagger &= \lambda(\pi)\textstyle\sum_{b_3,\,b_4\in\mathbb{b}^n}\Big(\delta_{\pi^{-1}(b_3),\,b_4}\,|b_3\rangle\langle b_4|\Big)^\dagger \\
&= \lambda(\pi)\textstyle\sum_{b_3,\,b_4\in\mathbb{b}^n}\Big(\delta_{\pi^{-1}(b_3),\,b_4}\,|b_4\rangle\langle b_3|\Big) \\
&= \Big(\textstyle\sum_{b_1,\,b_2\in\mathbb{b}^n}\delta_{\pi^{-1}(b_1),\,b_2}\,|b_1\rangle\langle b_2|\Big)\Big(\textstyle\sum_{b_3,\,b_4\in\mathbb{b}^n}\delta_{\pi^{-1}(b_3),\,b_4}\,|b_4\rangle\langle b_3|\Big) \\
&= \textstyle\sum_{b_1,\ldots,b_4\in\mathbb{b}^n}\delta_{\pi^{-1}(b_1),\,b_2}\;\delta_{\pi^{-1}(b_3),\,b_4}\,|b_1\rangle\langle b_2\,|\,b_4\rangle\langle b_3| \\
&= \textstyle\sum_{b_1,\ldots,b_4\in\mathbb{b}^n}\delta_{\pi^{-1}(b_1),\,b_2}\;\delta_{\pi^{-1}(b_3),\,b_4}\;\delta_{b_2,\,b_4}\,|b_1\rangle\langle b_3| \\
&= \textstyle\sum_{b_1,\,b_2,\,b_3\in\mathbb{b}^n}\delta_{\pi^{-1}(b_1),\,b_2}\;\delta_{\pi^{-1}(b_3),\,b_2}\,|b_1\rangle\langle b_3| \\
&= \textstyle\sum_{b_1,b_3\in\mathbb{b}^n}\delta_{\pi^{-1}(b_1),\,\pi^{-1}(b_3)}\,|b_1\rangle\langle b_3| \\
&= \textstyle\sum_{b_1,b_3\in\mathbb{b}^n}\delta_{b_1,\,b_3}\,|b_1\rangle\langle b_3| \\
&= \textstyle\sum_{b\in\mathbb{b}^n}|b\rangle\langle b| \\
&= \mathbb{I}
\end{aligned}$$

(ii) Let $\pi_1, \pi_2 \in \mathrm{Sym}(\mathbb{b}^n)$, then

$$\begin{aligned}
\lambda(\pi_1)\lambda(\pi_2) &= \Big(\textstyle\sum_{b_1,\,b_2\in\mathbb{b}^n}\delta_{\pi_1^{-1}(b_1),\,b_2}\,|b_1\rangle\langle b_2|\Big)\Big(\textstyle\sum_{b_3,\,b_4\in\mathbb{b}^n}\delta_{\pi_2^{-1}(b_3),\,b_4}\,|b_3\rangle\langle b_4|\Big) \\
&= \textstyle\sum_{b_1,\ldots,b_4\in\mathbb{b}^n}\delta_{\pi_1^{-1}(b_1),\,b_2}\;\delta_{\pi_2^{-1}(b_3),\,b_4}\;\delta_{b_2,b_3}\,|b_1\rangle\langle b_4| \\
&= \textstyle\sum_{b_1,\,b_2,\,b_4\in\mathbb{b}^n}\delta_{\pi_1^{-1}(b_1),\,b_2}\;\delta_{\pi_2^{-1}(b_2),\,b_4}\,|b_1\rangle\langle b_4| \\
&= \textstyle\sum_{b_1,\,b_2,\,b_4\in\mathbb{b}^n}\delta_{\pi_1^{-1}(b_1),\,b_2}\;\delta_{b_2,\,\pi_2(b_4)}\,|b_1\rangle\langle b_4|
\end{aligned}$$

$$= \sum_{b_1,\, b_4 \in \mathbb{b}^n} \delta_{\pi_1^{-1}(b_1),\, \pi_2(b_4)} \, |b_1\rangle \, \langle b_4|$$

$$= \sum_{b_1,\, b_4 \in \mathbb{b}^n} \delta_{\pi_2^{-1}(\pi_1^{-1}(b_1)),\, b_4} \, |b_1\rangle \, \langle b_4|$$

$$= \sum_{b_1,\, b_4 \in \mathbb{b}^n} \delta_{(\pi_1 \circ \pi_2)^{-1}(b_1),\, b_4} \, |b_1\rangle \, \langle b_4|$$

$$= \lambda(\pi_1 \circ \pi_2)$$

(iii) Let $\pi \in \mathrm{Sym}(\mathbb{b}^n)$ and let $b \in \mathbb{b}^n$, then

$$\lambda_\pi \, |b\rangle = \sum_{b_1,\, b_2 \in \mathbb{b}^n} \delta_{\pi^{-1}(b_1),\, b_2} \, |b_1\rangle \, \langle b_2 \,|\, b\rangle = \sum_{b_1,\, b_2 \in \mathbb{b}^n} \delta_{\pi^{-1}(b_1),\, b_2} \; \delta_{b_2,\, b} \, |b_1\rangle$$

$$= \sum_{b_1 \in \mathbb{b}^n} \delta_{\pi^{-1}(b_1),\, b} \, |b_1\rangle = \sum_{b_1 \in \mathbb{b}^n} \delta_{b_1,\, \pi(b)} \, |b_1\rangle = |\pi b\rangle$$ □

Definition and Proposition 4.7 (The Unitary Representation Λ)
Let $n \in \mathbb{N}$, then the unitary group representation Λ of S_n on $\mathfrak{q}^{\otimes n}$ is defined by

$$\begin{aligned} \Lambda : S_n &\to U(\mathfrak{q}^{\otimes n}) \\ \sigma &\mapsto (\lambda \circ \Phi)(\sigma) \end{aligned}$$

and satisfies $\Lambda(\sigma) \bigotimes_{i=1}^n |\psi_i\rangle = \bigotimes_{i=1}^n |\psi_{\sigma^{-1}(i)}\rangle$ for all $\sigma \in S_n$ and $|\psi_1\rangle, \ldots, |\psi_n\rangle \in \mathfrak{q}$.

Proof
Λ being a unitary group representation of S_n on $\mathfrak{q}^{\otimes n}$ follows directly from Proposition 2.11 due to the definitions of Φ and λ. Now, let $\sigma \in S_n$ and $|\psi_i\rangle \in \mathfrak{q}$ for $i \in [n]$. Define $\psi_{i,b}$ to be the basis coefficients of $|\psi_i\rangle$ with respect to the computational basis, i.e. let $|\psi_i\rangle = \psi_{i,0} \, |0\rangle + \psi_{i,1} \, |1\rangle$ for all $i \in [n]$. Recall that the tensor product is bilinear, implying that

$$\bigotimes_{i=1}^n |\psi_i\rangle = \bigotimes_{i \in [n]} \Big(\sum_{b \in \mathbb{b}} \psi_{i,b} \, |b\rangle \Big) = \bigotimes_{i \in [n]} \Big(\sum_{b_i \in \mathbb{b}} \psi_{i,b_i} \, |b_i\rangle \Big)$$

$$= \sum_{b \in \mathbb{b}^n} \bigotimes_{i \in [n]} \psi_{i,b_i} \, |b_i\rangle = \sum_{b \in \mathbb{b}^n} \Big(\prod_{i=1}^n \psi_{i,b_i} \Big) \, |b\rangle$$

holds. Define $\psi_b = \prod_{i=1}^n \psi_{i,b_i}$ and note that $\prod_{i=1}^n \psi_{\sigma^{-1}(i),b_i} = \prod_{i=1}^n \psi_{i,b_{\sigma(i)}}$ follows from the commutativity of scalar multiplication. Then, $\Lambda(\sigma) \bigotimes_{i=1}^n |\psi_i\rangle = \bigotimes_{i=1}^n |\psi_{\sigma^{-1}(i)}\rangle$ follows from the following calculation.

$$\Lambda(\sigma) \bigotimes_{i=1}^n |\psi_i\rangle = \lambda(\pi) \bigotimes_{i \in [n]} |\psi_i\rangle$$

$$= \Big(\sum_{b,\, \bar{b} \in \mathbb{b}^n} \delta_{\pi^{-1}(b),\, \bar{b}} \; |b\rangle \, \langle \bar{b}| \Big) \bigotimes_{i \in [n]} |\psi_i\rangle$$

$$
\begin{aligned}
&= \Big(\sum\nolimits_{b,\,\bar{b}\in\mathbb{b}^n} \delta_{\pi^{-1}(b),\,\bar{b}}\ |b\rangle\,\langle\bar{b}| \Big) \sum\nolimits_{b'\in\mathbb{b}^n} \psi_{b'}\,|b'\rangle \\
&= \sum\nolimits_{b,\,\bar{b},\,b'\in\mathbb{b}^n} \delta_{\pi^{-1}(b),\,\bar{b}}\ \psi_{b'}\,|b\rangle\,\langle\bar{b}\,|\,b'\rangle \\
&= \sum\nolimits_{b,\,\bar{b},\,b'\in\mathbb{b}^n} \delta_{\pi^{-1}(b),\,\bar{b}}\ \delta_{\bar{b},\,b'}\ \psi_{b'}\,|b\rangle \\
&= \sum\nolimits_{b,\,b'\in\mathbb{b}^n} \delta_{\pi^{-1}(b),\,b'}\ \psi_{b'}\,|b\rangle \\
&= \sum\nolimits_{b\in\mathbb{b}^n} \psi_{\pi^{-1}(b)}\,|b\rangle \\
&= \sum\nolimits_{b\in\mathbb{b}^n} \Big(\prod\nolimits_{i=1}^{n} \psi_{i,\,(\pi^{-1}\,b)_i} \Big)\,|b\rangle \\
&= \sum\nolimits_{b\in\mathbb{b}^n} \Big(\prod\nolimits_{i=1}^{n} \psi_{i,\,b_{\sigma(i)}} \Big)\,|b\rangle \\
&= \sum\nolimits_{b\in\mathbb{b}^n} \Big(\prod\nolimits_{i=1}^{n} \psi_{\sigma^{-1}(i),\,b_i} \Big)\,|b\rangle \\
&= \sum\nolimits_{b\in\mathbb{b}^n} \Big(\prod\nolimits_{i=1}^{n} \psi_{\sigma^{-1}(i),\,b_i} \Big) \bigotimes\nolimits_{i\in[n]} |b_i\rangle \\
&= \sum\nolimits_{b\in\mathbb{b}^n} \bigotimes\nolimits_{i\in[n]} \psi_{\sigma^{-1}(i),\,b_i}\,|b_i\rangle \\
&= \bigotimes\nolimits_{i\in[n]} \Big(\sum\nolimits_{b_i\in\mathbb{b}} \psi_{\sigma^{-1}(i),\,b_i}\,|b_i\rangle \Big) \\
&= \bigotimes\nolimits_{i\in[n]} \Big(\sum\nolimits_{b\in\mathbb{b}} \psi_{\sigma^{-1}(i),\,b}\,|b\rangle \Big) \\
&= \bigotimes\nolimits_{i\in[n]} |\psi_{\sigma^{-1}(i)}\rangle
\end{aligned}
$$

□

Quantum Alternating Operator Ansatz for the Traveling Salesman Problem

In this Chapter, we connect prior work to obtain a framework for mapping group-theoretical considerations to variational quantum algorithms for the Traveling Salesman Problem. In Section 5.1, we begin by studying asymmetric instances. *Asymmetric*, in this context, is to be understood as *not necessarily symmetric*, i.e. the approach works for both directed and undirected graphs. The section's findings constitute the central results of this thesis. In Section 5.2, we investigate possible improvements for the special case of symmetric graphs, but only obtain partial results. In Section 5.3 we discuss the concrete decomposition of group-theoretical mixers into standard quantum gates before, in Section 5.4, elaborating on the implications of this chapter's developments.

5.1. Mixing Families for Asymmetric Instances

Immediately, crucial notions and a central theorem are presented.

Definition 5.1 (Ordered Span and Complete Sequences)
Let $n \in \mathbb{N}$ and let $\Pi = (\pi_i)_{i\in[m]} \subset S_n$ be a finite sequence of length $m \in \mathbb{N}$.

(i) The **ordered span** of Π is the set $\langle\!\langle \Pi \rangle\!\rangle = \{\pi_m^{b_m} \dots \pi_1^{b_1} : b_i \in \mathbb{b}\} \subset S_n$.

(ii) The sequence Π is called **complete in** $\mathcal{G} \leq S_n$ if and only if $\mathcal{G} = \langle\!\langle \Pi \rangle\!\rangle$.

Theorem 5.2 (Mixers Induced by a Complete Sequence of Involutions)
Let $\mathcal{C}$ be the $\mathcal{E}$-vertex-time encoding of the complete weighted graph (K_n, w). Further, let the finite sequence $\Pi = (\pi_i)_{i\in[m]} \subset S_n$ of length $m \in \mathbb{N}$ be complete in S_n.

If all elements of Π are involutions, then for all feasible solutions $b_1, b_2 \in \mathcal{S}(\mathcal{C})$ of $\mathcal{C}$ there exists $\beta \in [0, 2\pi)^m$ satisfying

$$|b_2\rangle \propto e^{-i\beta_m \Lambda(\pi_i)} \dots e^{-i\beta_1 \Lambda(\pi_1)} |b_1\rangle .$$

Proof
For the purpose of this proof, we enumerate the elements of Π by $j \in [m]$ to better differentiate between the index $j \in [m]$ and the imaginary unit $i = (0,1) \in \mathbb{C}$. Further, note that $\pi \in \mathbb{R}$ with $\pi \approx 3.141592\ldots$ whereas $\pi_j \in S_n$.

The group action Φ_{n^2} of $\mathrm{S}_n^\star \in \{\mathrm{S}_n^\mathrm{V}, \mathrm{S}_n^\mathrm{T}\}$ on $\mathcal{S}(\mathcal{C})$ is transitive by Definition and Proposition 3.21, implying the existence of $\sigma \in \mathrm{S}_n^\star$ satisfying $b_2 = \Phi(\sigma)(b_1)$. As $\Pi \subset S_n \cong \mathrm{S}_n^\star$ is complete in $S_n \cong \mathrm{S}_n^\star$, there exist $b'_1, \ldots, b'_m \in \mathbb{b}$ satisfying

$$\sigma = \pi_m^{b'_m} \ldots \pi_1^{b'_1}$$

which implies

$$b_2 = \Phi(\sigma)(b_1) = \left[\Phi\left(\pi_m^{b'_m}\right) \ldots \Phi\left(\pi_1^{b'_1}\right)\right](b_1).$$

Since π_j is an involution for all $j \in [m]$, $\Lambda(\pi_j)$ is also an involution for all $j \in [m]$ by Proposition 2.13(ii). For arbitrary $\beta_j \in \mathbb{R}$, $\Lambda(\pi_j)$ being an involution then implies that

$$e^{-i\beta_j \Lambda(\pi_j)} = \cos(\beta_j)\, \mathbb{I} - i \sin(\beta_j)\, \Lambda(\pi_j).$$

holds. If β_j is now defined to be $\beta_j = (\pi\, b'_j)/2$ for all $j \in [m]$, then $e^{-i\beta_j \Lambda(\pi_j)}$ is given by

$$e^{-i\beta_j \Lambda(\pi_j)} = (-i)^{b'_j}\, \Lambda(\pi_j)^{b'_j} = (-i)^{b'_j}\, \Lambda\Big(\pi_j^{b'_j}\Big).$$

Therefore, $|b_2\rangle \propto e^{-i\beta_m \Lambda(\pi_m)} \ldots e^{-i\beta_1 \Lambda(\pi_1)} |b_1\rangle$ follows from the calculation below.

$$\begin{aligned}
e^{-i\beta_m \Lambda(\pi_m)} \ldots e^{-i\beta_1 \Lambda(\pi_1)} |b_1\rangle &= \left(\prod_{j=1}^{m} (-i)^{b'_j}\right) \Lambda\Big(\pi_m^{b'_m}\Big) \ldots \Lambda\Big(\pi_1^{b'_1}\Big) |b_1\rangle \\
&= \left(\prod_{j=1}^{m} (-i)^{b'_j}\right) \lambda\Big(\Phi\Big(\pi_m^{b'_m}\Big)\Big) \ldots \lambda\Big(\Phi\Big(\pi_1^{b'_1}\Big)\Big) |b_1\rangle \\
&= \left(\prod_{j=1}^{m} (-i)^{b'_j}\right) \lambda\Big[\Phi\Big(\pi_m^{b'_m}\Big) \ldots \Phi\Big(\pi_1^{b'_1}\Big)\Big] |b_1\rangle \\
&= \left(\prod_{j=1}^{m} (-i)^{b'_j}\right) \left|\left[\Phi\left(\pi_m^{b'_m}\right) \ldots \Phi\left(\pi_1^{b'_1}\right)\right](b_1)\right\rangle \\
&= \left(\prod_{j=1}^{m} (-i)^{b'_j}\right) |\Phi(\sigma)(b_1)\rangle \\
&= \left(\prod_{j=1}^{m} (-i)^{b'_j}\right) |b_2\rangle
\end{aligned}$$

□

Theorem 5.2 integrates results of Chapter 2, Chapter 3, and Chapter 4 to state clearly how the concepts outlined in Definition 5.1 translate to the construction of variational

quantum algorithms.[47] It provides the link between all preceding considerations and the developments that are to follow. In particular, for the remainder of this section (and for a majority of Section 5.2), we are content with remaining in the mental framework of searching for complete sequences of involutions, knowing that they equate to variational algorithms. Chiefly, we aim to minimize their length, as it corresponds to the number of angles required by the corresponding quantum algorithm.

We begin by incorporating the results of [Koß+23] into the framework of Definition 5.1:

Remark 5.3 ([Koß+23] Requires $O(n^2)$ Mixing Parameters)
When decomposing the permutations $\sigma \in S_J$ with $J \in \mathbb{N}$, Koßmann et al. writes that "there is a well-known representation as a product of at most $J\,(J-1)/2$ transpositions." Further, any transposition can be written "as a product of some of the $J-1$ adjacency transpositions $\tau_1 = (1,2), \ldots, \tau_{J-1} = (J-1, J)$" [Koß+23, p. 8]. By concatenation of these techniques, an arbitrary permutation is decomposed into the product of $J\,(J-1)^2/2$ adjacency transpositions. While this construction isn't wrong, it might possibly be misleading. And, in particular, it can be enhanced easily: Any permutation of n elements can be written as the product of at most $n-1$ arbitrary transpositions τ_{ij} (each of which can indeed be represented as the product of at most $n-1$ adjacency transpositions). Or, it can directly be written as the product of $n\,(n-1)/2$ adjacency transpositions τ_i. The factors of the $O(n^2)$-decomposition already are adjacency transpositions, and the composition of both is not required. In particular, the order of the possible factors of the $O(n^2)$-representation can be fixed — it constitutes a sequence complete in S_n. Moreover, all its elements are involutions by virtue of being transpositions.

Definition 5.4 (The Sequence Δ)
Let $n \in \mathbb{N}$ and $i \in [n]$. The sequences $\Gamma_{n,i} \subset T_n^a$ and $\Delta_n \subset T_n^a$ are defined as follows.[48]

$$\Gamma_{n,i} = (\tau_{i-1}, \tau_{i-2}, \ldots, \tau_1)$$

$$\Delta_n = \Gamma_{n,1} \frown \Gamma_{n,2} \frown \ldots \frown \Gamma_{n,n-1} \frown \Gamma_{n,n}$$

Proposition 5.5 (The Length of Δ_n)
If $n \in \mathbb{N}$, then $|\Delta_n| = n\,(n-1)/2$.

Proof

$$|\Delta_n| = \sum_{i=1}^{n} |\Gamma_{n,i}| = \sum_{i=1}^{n} (i-1) = \frac{n\,(n-1)}{2} \qquad \square$$

[47] Importantly, by $e^{-i\beta_i\,\Lambda(\pi_i)} = \cos(\beta_i)\,\mathbb{I} - i\sin(\beta_i)\,\Lambda(\pi_i)$ it follows that for $\beta_i = \pi/4$ the prepared state is in a (non-uniform) superposition of computational basis states $|b\rangle \in \mathcal{Q}(\mathcal{C})$, i.e. in a superposition of all feasible solutions.

[48] Remember that $(x_i)^\frown(y_i)$ denotes the concatenation of finite sequences (x_i) and (y_i); see page 6.

Proposition 5.6 (Δ_n is Complete)
Δ_n is complete in S_n for all $n \in \mathbb{N}$.

As the proof of this is well-known within group theory,[49] we instead offer computational intuition for this result: Suppose you are being given a list (or a sequence) or $n \in \mathbb{N}$ elements of a totally ordered set, $(a_1, \ldots, a_n)$. Being given the opportunity to apply the transposition τ_1 ensures the ability to sort the subsequence of the first two elements: If $a_2 < a_1$, apply τ_1; otherwise, do nothing. Now, if $(a'_1, \ldots, a'_n)$ is the resulting list satisfying $a'_1 \leq a'_2$, then the opportunity to potentially apply τ_2 followed by the potential application of τ_1 ensures that the subsequence of the first three elements can be sorted: If $a_3 < a_1$, apply both τ_2 and τ_1; if $a_1 \leq a_3 < a_2$, apply τ_2; otherwise, do nothing. This procedure can be iterated until the whole sequence is sorted, requiring at most $n\,(n-1)/2$ adjacency transpositions. Notably, it is logically equivalent (albeit not identical) to the bubble-sort algorithm.

Having recast the construction of [Koß+23] into a complete sequence, we turn the exploration of new ones. Specifically, we start by defining and proving lower bounds L_S and L_T for the length of complete sequences. These will guide subsequent investigations and enable the assessment of obtained results.

Definition 5.7 (The Maps L_S and L_T)
The maps L_S and L_T are defined as follows.

$$L_S : \mathbb{N} \to \mathbb{R},\ n \mapsto \log_2(n!)$$

$$L_T : \mathbb{N} \to \mathbb{R},\ n \mapsto n\,(n-1)/2$$

Proposition 5.8 (Minimal Length of Complete Sequences)
Let $n \in \mathbb{N}$. Further, let $\Pi_1 \subset S_n$ and $\Pi_2 \subset T_n^a$ be finite sequences.

(i) If $\Pi_1 \subset S_n$ is complete in S_n, then $|\Pi_1| \geq L_S(n)$.

(ii) If $\Pi_2 \subset T_n^a$ is complete in S_n, then $|\Pi_2| \geq L_T(n)$.

Proof
(i) If $\Pi_1 \subset S_n$ is complete in S_n, then $n! = |S_n| = |\langle\!\langle \Pi_1 \rangle\!\rangle| \leq |\mathfrak{b}^{|\Pi_1|}| = 2^{|\Pi_1|}$, implying

$$|\Pi_1| \geq \log_2(n!)$$

because $\log_2$ is monotonically increasing.

[49] For example, [Sam17, pp. 19–20] gives a conceptually identical task as an exercise to the reader.

(ii) We offer an outline of the proof: Using the concept of 'inversions', one can prove that the permutation $\sigma \in S_n$ given by

$$\sigma = \begin{pmatrix} 1 & \dots & n \\ n & \dots & 1 \end{pmatrix}$$

cannot be written as the product of less than $n\,(n-1)/2$ adjacency transpositions.[50] Further, if $\Pi_2 \subset T_n^a$, then any element of the ordered span $\langle\!\langle \Pi_2 \rangle\!\rangle$ is the product of at most $|\Pi_2|$ adjacency transpositions. Hence, if $\Pi_2 \subset T_n^a$ is complete in S_n, then $\sigma \in S_n = \langle\!\langle \Pi_2 \rangle\!\rangle$ implies $|\Pi_2| \geq n\,(n-1)/2$. □

Remarkably, part (ii) implies that the sequence Δ_n is of optimal length, i.e. no further improvements can be made for the special case of adjacency transpositions. Part (i), however, suggests that an asymptotic reduction in length might be possible. The remainder of this section will now consist almost entirely of mathematical Definitions, Propositions, and Theorems that realize that reduction: In Definition 5.12, we define the sequence Ξ_n. After proving Proposition 5.13 about its length (corresponding to the number of angles of the associated VQA) and Proposition 5.14 about the number of transpositions it contains (corresponding to the number of gates of the associated VQA; cf. Section 5.3), we show that it consists of involutions (Proposition 5.15), is complete in S_n (Theorem 5.16), and is of asymptotically optimal length (Theorem 5.17). Before, we define the map R (Definition 5.9), which is essential to the Definition of Ξ_n, and prove that it is an isomorphism (Proposition 5.10) that 'preserves completeness' (Proposition 5.11).

Definition 5.9 (The Map R)
Let $n \in \mathbb{N}$. Define R to be the mapping $R : S_n \to S_{n+1}^1$, $R(\sigma) = \hat{\sigma}$ with $\hat{\sigma}$ defined below.

$$\hat{\sigma}(i) = \begin{cases} 1, & \text{if } i = 1 \\ \sigma(i-1)+1, & \text{if } i > 1 \end{cases}$$

The definition $\hat{\sigma}(i) = \sigma(i-1)+1$ for $i > 1$ likely looks more convoluted than it actually is. If a permutation $\sigma \in S_n$ is given in its cycle notation, for example $\tau_i = (i, i+1)$, then the permutation $R(\sigma) \in S_{n+1}^1$ is obtained by simply increasing every number in every cycle by 1, i.e. $R(\tau_i) = (i+1, i+2) = \tau_{i+1}$.

Proposition 5.10 (R is an Isomorphism)
R is a group isomorphism from S_n to S_{n+1}^1 for all $n \in \mathbb{N}$.

[50] The proof of Proposition 5.19(ii) contains more information on inversions.

Proof
Homomorphism: Let $\sigma, \pi \in S_n$ and define $\gamma = \sigma \circ \pi$. We differentiate the cases $i = 1$ and $i > 1$.

$$\begin{aligned}
i = 1 \quad \Rightarrow \quad R(\sigma \circ \pi)(i) &= R(\gamma)(1) = \hat{\gamma}(1) = 1 = \hat{\sigma}(1) = \hat{\sigma}(\hat{\pi}(1)) = [\hat{\sigma} \circ \hat{\pi}](1) \\
&= [R(\sigma) \circ R(\pi)](i) \\
i > 1 \quad \Rightarrow \quad R(\sigma \circ \pi)(i) &= R(\gamma)(i) = \hat{\gamma}(i) \overset{i \geq 1}{=} \gamma(i-1) + 1 = [\sigma \circ \pi](i-1) + 1 \\
&= \sigma(\pi(i-1)) + 1 = \sigma(\pi(i-1) + 1 - 1) + 1 \\
&\overset{i \geq 1}{=} \sigma(\hat{\pi}(i) - 1) + 1 \overset{\hat{\pi}(i) > 1}{=} \hat{\sigma}(\hat{\pi}(i)) = [\hat{\sigma} \circ \hat{\pi}](i) \\
&= [R(\sigma) \circ R(\pi)](i)
\end{aligned}$$

In the penultimate line, $\hat{\pi}(i) > 1$ was concluded by using $\hat{\pi}(i) \overset{i \geq 1}{=} \pi(i-1)+1 \geq 1+1 > 1$. Now, since $[R(\sigma \circ \pi)](i) = [R(\sigma) \circ R(\pi)](i)$ for all $i \in [n]$, it follows that $R(\sigma \circ \pi) = R(\sigma) \circ R(\pi)$. Since $\sigma, \pi \in S_n$ are arbitrary, it follows that R is a homomorphism.

Bijection: We argue that R is a bijection because the mapping $P : S_{n+1}^1 \to S_n$, $P(\sigma) = \check{\sigma}$, where $\check{\sigma}$ is given by $\check{\sigma}(i) = \sigma(i+1) - 1$, satisfies $R \circ P = \mathrm{id}_{S_{n+1}^1}$ and $P \circ R = \mathrm{id}_{S_n}$.

(i) $\forall \sigma \in S_{n+1}^1 : P(\sigma) \in S_n$: Let $\sigma \in S_{n+1}^1$ and define the map $f : \{2, \ldots, n+1\} \to \{1, \ldots, n\}$, $f(x) = x - 1$. Now, $P(\sigma) \in S_n$ follows from the fact that

$$P(\sigma) = f \ \circ \ \sigma|_{\{2,\ldots,n+1\}} \ \circ \ f^{-1}$$

where

$$\begin{aligned}
\sigma|_{\{2,\ldots,n+1\}} &\in \mathrm{Bij}\Big(\{2, \ldots, n+1\}, \{2, \ldots, n+1\}\Big), \\
f &\in \mathrm{Bij}\Big(\{2, \ldots, n+1\}, \{1, \ldots, n\}\Big).
\end{aligned}$$

(ii) $P \circ R = \mathrm{id}_{S_n}$: Let $\sigma \in S_n$ and define $\pi = R(\sigma)$. Let $i \in [n]$. Then,

$$\begin{aligned}
[(F \circ R)(\sigma)](i) = [F(R(\sigma))](i) = [F(\pi)](i) = \check{\pi}(i) = \pi(i+1) - 1 = \hat{\sigma}(i+1) - 1 \\
= \Big[\hat{\sigma}(i+1)\Big] - 1 \overset{i+1>1}{=} \Big[\sigma\Big((i+1) - 1\Big) + 1\Big] - 1 = \sigma(i).
\end{aligned}$$

Since $[(P \circ R)(\sigma)](i) = \sigma(i)$ for all $i \in [n]$, it follows that $(P \circ R)(\sigma) = \sigma$. Since $\sigma \in S_n$ is arbitrary, it follows that $P \circ R = \mathrm{id}_{S_n}$.

(iii) $R \circ P = \mathrm{id}_{S_{n+1}^1}$: Let $\sigma \in S_{n+1}^1$ and define $\pi = P(\sigma)$. Let $i \in [n+1]$. We differentiate

the cases $i = 1$ and $i > 1$.

$$\begin{aligned} i = 1 \quad &\Rightarrow \quad [(R \circ P)(\sigma)](i) = [R(P(\sigma))](1) = [R(\pi)](1) = \hat{\pi}(1) = 1 = \sigma(i) \\ i > 1 \quad &\Rightarrow \quad [(R \circ P)(\sigma)](i) = [R(P(\sigma))](i) = [R(\pi)](i) = \hat{\pi}(i) \overset{i \geq 1}{=} \pi(i-1) + 1 \\ &\qquad\qquad = \check{\sigma}(i-1) + 1 = \sigma\Big((i-1)+1\Big) - 1 + 1 = \sigma(i) \end{aligned}$$

Since $[(R \circ P)(\sigma)](i) = \sigma(i)$ for all $i \in [n+1]$, it follows that $(P \circ R)(\sigma) = \sigma$. Since $\sigma \in S_{n+1}^1$ is arbitrary, it follows that $R \circ P = \mathrm{id}_{S_{n+1}^1}$.

Thus, R is an isomorphism by Proposition 2.10. □

Proposition 5.11 (*R* Maps Complete Sequences to Complete Sequences) Let $n \in \mathbb{N}$. If $\Pi \subset S_n$ is complete in $\mathcal{G} \leq S_n$, then $R(\Pi) \subset S_{n+1}$ is complete in $R(\mathcal{G}) \leq S_{n+1}$.

Proof
So let $\Pi \subset S_n$ be a finite sequence, given by $\Pi = (\pi_1, \ldots, \pi_p)$ where $p = |\Pi| \in \mathbb{N}$, that is complete in some subgroup $\mathcal{G}$ of S_n.

(i) $R(\mathcal{G}) \leq S_{n+1}$: Since $R : S_n \to S_{n+1}^1$ is a group homomorphism, $\mathcal{G}$ being a subgroup of S_n implies that $R(\mathcal{G})$ is a subgroup of S_{n+1}^1 by Proposition 2.13(iv). Furthermore, since being a subgroup is a transitive relation by Proposition 2.8, $R(\mathcal{G}) \leq S_{n+1}^1$ and $S_{n+1}^1 \leq S_{n+1}$ imply $R(\mathcal{G}) \leq S_{n+1}$.

(ii) $\langle R(\Pi) \rangle \subset R(\mathcal{G})$: Let $\sigma \in \langle R(\Pi) \rangle$, then there exists $b \in \mathbb{b}^p$ satisfying $\sigma = R(\pi_p)^{b_p} \ldots R(\pi_1)^{b_1}$. Define $\tilde{\sigma} = \pi_p^{b_p} \ldots \pi_1^{b_1}$, then $\tilde{\sigma} \in \mathcal{G}$ follows from the fact that Π is complete is $\mathcal{G}$. Thus,

$$\sigma = R(\pi_p)^{b_p} \ldots R(\pi_1)^{b_1} \overset{2.13(\mathrm{i})}{=} R(\pi_p^{b_p} \ldots \pi_1^{b_1}) = R(\tilde{\sigma}) \in R(\mathcal{G}).$$

(iii) $\langle\!\langle R(\Pi) \rangle\!\rangle \supset R(\mathcal{G})$: Let $\sigma \in R(\mathcal{G})$, then there exists $\tilde{\sigma} \in \mathcal{G}$ such that $\sigma = R(\tilde{\sigma})$. Since Π is complete in $\mathcal{G}$, there exists $b \in \mathbb{b}^p$ satisfying $\tilde{\sigma} = \pi_p^{b_p} \ldots \pi_1^{b_1}$. Thus,

$$\sigma = R(\tilde{\sigma}) = R(\pi_p^{b_p} \ldots \pi_1^{b_1}) \overset{2.13(\mathrm{i})}{=} R(\pi_p)^{b_p} \ldots R(\pi_1)^{b_1} \in \langle\!\langle R(\Pi) \rangle\!\rangle.$$ □

Before continuing, I want to highlight the valuable and appreciated contributions of Benjamin Sambale. Upon being presented with an outline of the group-theoretic problem, he produced the definition of Ξ_n along with a first proof of its completeness. In particular, said proof recognized that $|\Xi_n| \sim n \log n$. Afterward, I rigorized certain parts of the proof (resulting in e.g. Propositions 5.10 and 5.11) and fixed a mathematically significant but conceptually inconsequential mistake (resulting in the addition of step (i) of the proof of Theorem 5.16 presented below).

Definition 5.12 (Sambale; the Sequence Ξ_n)

(i) The mappings I and J are defined as follows.

$$I : \mathbb{N} \to \mathbb{Z}, \qquad x \mapsto \lceil \log_2(x) \rceil$$

$$J : \mathbb{N}^2 \to \mathbb{Z},\ (x, y) \mapsto \min\{x - 2^{y-1},\ 2^{y-1}\}$$

(ii) For $n \in \mathbb{N}$ and $i \in [I(n)]$, the permutation $\xi_{n,i} \in S_n$ and the sequence $\Xi_n \subset S_n$ are defined as follows.

$$\xi_{n,i} = \prod_{j=1}^{J(n,i)} (j,\ j + 2^{i-1})$$

$$\Xi_n = \begin{cases} \varnothing, & \text{if } n = 1, \\ R(\Xi_{n-1}) ^\frown (\xi_{n,1}, \ldots, \xi_{n,I(n)}), & \text{if } n > 1 \end{cases}$$

Proposition 5.13 (Sambale; Number of Permutations in Ξ_n)

(i) $|\Xi_n| = \sum_{n'=1}^{n} I(n')$ for all $n \in \mathbb{N}$.

(ii) $|\Xi_n| \in \Theta(n \log_2 n)$.

Proof

(i) $|\Xi_n| = \sum_{n'=1}^{n} I(n')$ follows immediately from the definition of Ξ_n when considering that $|\Xi_{n'}| = |R(\Xi_{n'})|$ and $I(1) = \lceil \log_2(1) \rceil = 0 = |\varnothing|$.

(ii) $|\Xi_n| \sim n \log_2 n$ follows from $|\Xi_n| = \sum_{n'=1}^{n} I(n') = \sum_{n'=1}^{n} \lceil \log_2(n') \rceil$ with the help of $\lim_{n\to\infty} n / \log_2(n!) = 0$ and $\lim_{n\to\infty} n \log_2 n / \log_2(n!) = 1$ (Stirling's approximation), and implies $|\Xi_n| \in \Theta(n \log_2 n)$.[51] That is because the two limits imply both $\log_2(n!) \sim n \log_2 n$ and $\log_2(n!) + n \sim n \log_2 n$ — and $|\Xi_n|$ is bounded from below by the former and bounded from above by the latter.

$$|\Xi_n| = \sum_{n'=1}^{n} \lceil \log_2(n') \rceil \geq \sum_{n'=1}^{n} \log_2(n') = \log_2 \left(\prod_{n'=1}^{n} n' \right) = \log_2(n!)$$

$$\begin{aligned} |\Xi_n| = \sum_{n'=1}^{n} \lceil \log_2(n') \rceil < \sum_{n'=1}^{n} (\log_2(n') + 1) &= \sum_{n'=1}^{n} \log_2(n') \ + \sum_{n'=1}^{n} 1 \\ &= \log_2 \left(\prod_{n'=1}^{n} n' \right) + \ n \\ &= \log_2(n!) + n \end{aligned}$$

□

[51] $f \sim g$ is a shorthand notation for $\lim_{n\to\infty} f(n)/g(n) = 1$.

Proposition 5.14 (Number of Transpositions in Ξ_n)
(i) $\forall n \in \mathbb{N}\, \forall i \in [I(n)] :\ J(n,i) = n - 2^{i-1} \ \Leftrightarrow\ i = I(n)$

(ii) $\forall n \in \mathbb{N} :\ \sum_{n'=1}^{n} \sum_{i=1}^{I(n')} \sum_{j=1}^{J(n',i)} 1 = \frac{n\,(n-1)}{2}$

Proof

(i) Clearly, $J(n,i) = \min\{n - 2^{i-1}, 2^{i-1}\} = n - 2^{i-1}$ if and only if $n - 2^{i-1} \leq 2^{i-1}$. Noting that the latter condition is equivalent to $\log_2(n) \leq i$, consider the following:

(i.i) If $i < I(n)$, then $i \leq I(n) - 1 = \lceil \log_2(n) \rceil - 1 < (\log_2(n) + 1) - 1 = \log_2(n)$, implying $J(n,i) \neq n - 2^{i-1}$.

(i.ii) If $i = I(n)$, then $i = \lceil \log_2(n) \rceil \geq \log_2(n)$, implying $J(n,i) = n - 2^{i-1}$.

Case (i.i) proves the implication

$$\neg\Big[J(n,i) = n - 2^{i-1}\Big] \quad \Leftarrow \quad \neg\Big[i = I(n)\Big],$$

which is equivalent to its contraposition

$$J(n,i) = n - 2^{i-1} \quad \Rightarrow \quad i = I(n).$$

Case (i.ii) proves the implication

$$J(n,i) = n - 2^{i-1} \quad \Leftarrow \quad i = I(n).$$

In culmination, we proved the equivalence

$$J(n,i) = n - 2^{i-1} \quad \Leftrightarrow \quad i = I(n).$$

(ii) Recall that the finite geometric series for $m \in \mathbb{N}$ and $r \in \mathbb{R} \setminus \{0\}$ is given by $\sum_{k=1}^{m} r^{k-1} = \frac{1-r^m}{1-r}$ and that the triangular number of $m \in \mathbb{N}$ is given by $\sum_{k=1}^{m} k = \frac{m\,(m+1)}{2}$. Keeping these relations, as well as our result from part (i), in mind, we can evaluate the sum in a straightforward manner:

$$\begin{aligned}
\sum_{n'=1}^{n} \sum_{i=1}^{I(n')} \sum_{j=1}^{J(n',i)} 1 &= \sum_{n'=1}^{n} \sum_{i=1}^{I(n')} J(n',i) \\
&= \sum_{n'=1}^{n} \left(\sum_{i=1}^{I(n')-1} (2^{i-1}) + \left(n' - 2^{I(n')-1}\right) \right) \\
&= \sum_{n'=1}^{n} \left(\frac{1 - 2^{I(n')-1}}{1-2} + \left(n' - 2^{I(n')-1}\right) \right) \\
&= \sum_{n'=1}^{n} \left(2^{I(n')-1} - 1 + \left(n' - 2^{I(n')-1}\right) \right)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{n'=1}^{n} (n' - 1) \\
&= \sum_{n'=1}^{n} n' - \sum_{n'=1}^{n} 1 \\
&= \frac{n\,(n-1)}{2}
\end{aligned}
$$

□

Denoting $n\,(n-1)/2$ to be the number of transpositions 'contained' in the sequence Ξ_n is vague: What $n\,(n-1)/2$ counts is the total number of two-cycles found in $\Xi_n = (R^{n-2}(\xi_{2,1}),\, \ldots,\, R(\xi_{n-1,I(n-1)}),\, \xi_{n,1} \ldots, \xi_{n,I(n)})$, where the permutation $\xi_{m,i}$ is given by

$$\xi_{m,i} = \prod_{j=1}^{J(m,i)} (j,\, j + 2^{i-1})$$

following Definition 5.12. Any of these transpositions $\tau_{ij} \in S_n$ will be mapped to n transpositions $\tau_{kl} \in \mathrm{S}_n^\star \leq S_{n^2}$ with $\mathrm{S}_n^\star \in \{\mathrm{S}_n^\mathrm{V}, \mathrm{S}_n^\mathrm{T}\}$, and each of these will be mapped to a SWAP gate by Λ (cf. Section 5.3). Hence, the number of SWAP gates needed to require all of the unitary operators corresponding the sequence's elements is $\Theta(n^3)$. In particular, it is exactly equal to the corresponding number for Δ_n. (Note that $|\Delta_n| = n\,(n-1)/2$ and that each element in Δ_n is a transposition.) Thereby, the mixing family corresponding to Ξ_n will require the same number of gates to implement it as the mixing family corresponding to Δ_n. Any reduction is purely in the number of mixing parameters for the classical solver to optimize over. Finally, we proceed to prove the key results regarding Ξ_n:

Proposition 5.15 ($\xi_{n,i}$ is Involutive)
$\xi_{n,i}$ is an involution for all $n \in \mathbb{N}$ and $i \in [I(n)]$.

Proof
A permutation is an involution if and only if it can be written as a product of disjoint two-cycles [Sag01, p. 2; Yan+13, p. 2]. Fortunately, $\xi_{n,i}$ is already defined as a product of two-cycles, so the only thing left to do is prove that the cycles $(j,\, j + 2^{i-1})$ in $\xi_{n,i}$ are pairwise disjoint (i.e. act non-trivially on disjoint subsets of $[n]$).

Let $n \in \mathbb{N}$ and $i \in [I(n)]$. Further, let $j_1, j_2 \in [J(n,i)]$. Clearly, the intersection of $\{j_1, j_1 + 2^{i-1}\}$ and $\{j_2, j_2 + 2^{i-1}\}$ can only be non-empty for $j_1 \neq j_2$ if $j_1 = j_2 + 2^{i-1}$ or $j_2 = j_1 + 2^{i-1}$. Both of these statements are, however, false:

$$
\begin{aligned}
j_1 \leq J(n,i) = \min\{n - 2^{i-1}, 2^{i-1}\} \leq 2^{i-1} < 1 + 2^{i-1} \leq j_2 + 2^{i-1} \\
j_2 \leq J(n,i) = \min\{n - 2^{i-1}, 2^{i-1}\} \leq 2^{i-1} < 1 + 2^{i-1} \leq j_1 + 2^{i-1}
\end{aligned}
$$

Hence, $\xi_{n,i}$ is an involution for all $n \in \mathbb{N}$ and $i \in [I(n)]$. □

Theorem 5.16 (Sambale; Ξ_n is Complete)
Ξ_n is complete in S_n for all $n \in \mathbb{N}$.

Proof
Proof by induction on $n \in \mathbb{N}$.

Induction base: Let $n = 1$, then by definition $\langle\!\langle \Xi_1 \rangle\!\rangle = \langle\!\langle \varnothing \rangle\!\rangle = \{\mathrm{id}\} = S_1$.[52]

Induction step: Let $n > 1$ and assume that Ξ_{n-1} is complete in S_{n-1}. Clearly, the inclusion $\langle\!\langle \Xi_n \rangle\!\rangle \subset S_n$ is obvious. We proceed to prove $\langle\!\langle \Xi_n \rangle\!\rangle \supset S_n$: Let $\sigma \in S_n$. Now, let $s \in \mathbb{b}^{I(n)}$ be the binary representation[53] of the number $\sigma(1) - 1 \in [0\,..\,n-1]$, i.e. let $s_i \in \mathbb{b}$ satisfy $\sigma(1) - 1 = \sum_{i=1}^{I(n)} s_i 2^{i-1}$. Define $\pi_s = \xi_{n,I(n)}^{s_{I(n)}} \dots \xi_{n,1}^{s_1}$. We conclude the proof in two steps.

(i) $\pi_s^{-1}\sigma \in S_n^1$: We prove $\Big(\xi_{n,I}^{s_I} \dots \xi_{n,1}^{s_1}\Big)(1) = \sum_{i=1}^{I} s_i 2^{i-1} + 1$ for all $I \in [I(n)]$ by (finite) induction on $I \in [I(n)]$.[54] Then, the case $I = I(n)$ implies $\pi_s(1) = \sigma(1)$ from which $\pi_s^{-1}\sigma \in S_n^1$ follows because of

$$1 = \pi_s^{-1}\Big(\pi_s(1)\Big) = \pi_s^{-1}\Big(\sigma(1)\Big) = \Big(\pi_s^{-1} \circ \sigma\Big)(1).$$

(i.i) Induction base: Let $I = 1$, then $\Big(\xi_{n,I}^{s_I} \dots \xi_{n,1}^{s_1}\Big)(1) = \xi_{n,1}^{s_1}(1)$ and $\sum_{i=1}^{I} s_i 2^{i-1} + 1 = s_1 + 1$. Furthermore, $\xi_{n,1} = (1,2)$ because $J(n,1) = \min\{n - 2^0, 2^0\} = 1$. We differentiate the cases $s_1 = 0$ and $s_1 = 1$ to obtain the desired result:

$$\begin{aligned} s_1 = 0 \quad &\Rightarrow \quad s_1 + 1 = 1 = \mathrm{id}(1) = \Big(1,2\Big)^0(1) = \Big(1,2\Big)^{s_1}(1) \\ s_1 = 1 \quad &\Rightarrow \quad s_1 + 1 = 2 = \Big(1,2\Big)(1) = \Big(1,2\Big)^{s_1}(1) \end{aligned}$$

(i.ii) Induction step: Let $I > 1$ and assume that $\Big(\xi_{n,I-1}^{s_{I-1}} \dots \xi_{n,1}^{s_1}\Big)(1) = \sum_{i=1}^{I-1} s_i 2^{i-1} + 1$ is true. Furthermore, assume that $s_I = 1$ since otherwise there is nothing left to prove. If we define $i_0 = \sum_{i=1}^{I-1} s_i 2^{i-1} + 1$, then the following equations show that we are required to prove $\xi_{n,I}(i_0) = i_0 + 2^{I-1}$.

[52] If this induction base appears questionably arbitrary, consider that the sequence $\Xi_2 = (\tau_1)$ yields the ordered palette $\langle\!\langle \Xi_2 \rangle\!\rangle = \{\mathrm{id}, \tau_1\} = S_2$.

[53] To any computer scientist, the existence and uniqueness of such an $s \in \mathbb{b}^{I(n)} = \mathbb{b}^{\lceil \log_2(n) \rceil}$ is obvious; for a more mathematical treatment, see for example [Nat00, pp. 5–7].

[54] Note that this constitutes an abuse of notation, as both the map $I : \mathbb{N} \to \mathbb{Z}$ from Definition 5.12 as well as the number $I \in [I(n)]$ are denoted by I. However, it is always clear from the context which one is meant. Specifically, within the scope of this proof, I denotes the map if and only if it is followed by parentheses, i.e. $I(n)$.

$$\left(\xi_{n,I}^{s_I} \dots \xi_{n,1}^{s_1}\right)(1) = \xi_{n,I}^{s_I}\left(\left(\xi_{n,I-1}^{s_{I-1}} \dots \xi_{n,1}^{s_1}\right)(1)\right) \overset{\text{I.H.}}{=} \xi_{n,I}^{s_I}(i_0) \overset{S_I=1}{=} \xi_{n,I}(i_0)$$
$$\sum_{i=1}^{I} s_i 2^{i-1} + 1 = i_0 + s_I 2^{I-1} \overset{S_I=1}{=} i_0 + 2^{I-1}$$

Obviously, $i_0 \geq 1$. Further, the statements $i_0 \leq n - 2^{I-1}$ and $i_0 \leq 2^{I-1}$ are obtained as follows.

$$i_0 = \sum_{i=1}^{I-1} s_i 2^{i-1} + 1 \leq \sum_{i=1}^{I-1} 2^{i-1} + 1 = \frac{1-2^{I-1}}{1-2} + 1 = 2^{I-1} - 1 + 1 = 2^{I-1}$$
$$n \geq \sigma(1) = \sum_{i=1}^{I(n)} s_i 2^{i-1} + 1 \overset{I \leq I(n)}{\geq} \sum_{i=1}^{I} s_i 2^{i-1} + 1 = i_0 + s_I 2^{I-1} \overset{s_I=1}{=} i_0 + 2^{I-1}$$

These inequalities culminate in $1 \leq i_0 \leq \min\{n-2^{I-1}, 2^{I-1}\} = J(n,I)$, which implies that $\xi_{n,I} = \prod_{i=1}^{J(n,I)}(i,\, i+2^{I-1})$ contains the transposition (i_0, i_0+2^{I-1}). Furthermore, we know from Proposition 5.15 that $\xi_{n,I}$ is an involution and contains no other two-cycle acting non-trivially on either i_0 or $i_0 + 2^{I-1}$. Therefore, $\xi_{n,I}(i_0) = i_0 + 2^{I-1}$, implying $\left(\xi_{n,I}^{s_I} \dots \xi_{n,1}^{s_1}\right)(1) = \sum_{i=1}^{I} s_i 2^{i-1} + 1$.

(ii) $\sigma \in \langle\!\langle \Xi_n \rangle\!\rangle$: By the induction hypothesis, Ξ_{n-1} is complete in S_{n-1}. Hence, by Proposition 5.11, $R(\Xi_{n-1})$ is complete in $R(S_{n-1}) = S_n^1$, where the equivalence to S_n^1 follows from the fact that R is an isomorphism.

Therefore, since $\pi_s^{-1}\sigma \in S_n^1$ by step (i), if we rewrite the sequence $R(\Xi_{n-1})$ as $R(\Xi_{n-1}) = (x_1, \dots, x_p)$ where $p = |\Xi_{n-1}| = |R(\Xi_{n-1})|$, then there must exist $b \in \mathbb{b}^p$ such that $\pi_s^{-1}\sigma = x_p^{b_p} \dots x_1^{b_1}$. However, Ξ_n is given by

$$\Xi_n = (x_1, \dots, x_p, \xi_{n,1}, \dots, \xi_{n,I(n)}).$$

Thus, $\sigma \in \langle\!\langle \Xi_n \rangle\!\rangle$ follows due to

$$\sigma = (\pi_s \pi_s^{-1})\,\sigma = \pi_s\,(\pi_s^{-1}\sigma) = \xi_{n,I(n)}^{s_{I(n)}} \dots \xi_{n,1}^{s_1}\, x_p^{b_p} \dots x_1^{b_1}. \qquad \square$$

Theorem 5.17 ($|\Xi_n|$ is Asymptotically Optimal)
If, for all non-trivial $n \in \mathbb{N}$, $\Sigma_n \subset S_n$ is a finite sequence that is complete in S_n, then

$$\liminf_{n\to\infty} \frac{|\Sigma_n|}{|\Xi_n|} \geq 1.$$

Proof
Since the sequence $(|\Sigma_n|/|\Xi_n|)_{n\in\mathbb{N}_{\geq 2}}$ is bounded from below by the sequence $(s_n)_{n\in\mathbb{N}_{\geq 2}}$,

$$s_n = \frac{\log_2(n!)}{\log_2(n!) + n},$$

its limit inferior is bounded by below by the limit inferior of (s_n), which, because (s_n) converges, is equal to its limit of 1.

To see that $|\Sigma_n|/|\Xi_n| \geq s_n$ holds for all non-trivial $n \in \mathbb{N}$, note that $|\Sigma_n| \geq \log_2(n!)$ by Proposition 5.8(i) and $|\Xi_n| \leq \log_2(n!) + n$ by Proposition 5.13, implying $1/|\Xi_n| \geq 1/(\log_2(n!) + n)$. To see that $\lim_{n\to\infty} s_n = 1$, rewrite s_n as $s_n = 1/(a_n + b_n)$ with $a_n = \log_2(n!)/\log_2(n!)$ and $b_n = n/\log_2(n!)$. Clearly, $\lim_{n\to\infty} a_n = 1$ and $\lim_{n\to\infty} b_n = 0$. In particular, because both sequences converge, (s_n) also converges with

$$\lim_{n\to\infty} s_n = \frac{1}{\lim_{n\to\infty} a_n + \lim_{n\to\infty} b_n} = 1. \qquad \square$$

5.2. Mixing Families for Symmetric Instances

In this section, we investigate possibilities to minimize mixing families for symmetric instances of the Traveling Salesman Problem, i.e. for graphs where differentiating between a tour and its 'reverse' tour is not necessary. Requiring problem instances to satisfy special conditions often enables the construction of solutions that are more efficient than possible in the general case,[55] and symmetry is an obvious first restriction to impose on entirely arbitrary instances. Importantly, however, the results we present in this section are incomplete: we present conjectures and evidence, not theorems and proofs. Yet, we deem those incomplete considerations interesting enough to warrant a swift survey.

As hinted at before, there exist canonical identifications between the Hamiltonian cycles of a complete graph and the permutations of its vertex set.[56] One way of obtaining such an equivalence is by using the intuition of the vertex-time encoding: If there are n cities $c \in V = [n]$ to be visited in n timeslots $t \in T = [n]$, then one way to encode a solution s is with a map $s : T \to V$ that $s(t) \in V$ is the city visited at time $t \in T$ for all $t \in T$. Requiring only one city be visited at each timeslot is satisfied by virtue of every $f \in \mathrm{Map}(T, V)$ mapping each element of T to only one element of V. Requiring every city be visited is facilitated by requiring $f \in \mathrm{Map}(T, V)$ to be surjective, which is equivalent to requiring $f \in \mathrm{Map}(T, V)$ to be bijective by V and T being finite with $|T| = |V|$. Since $T = V = [n]$, every permutation $\sigma \in S_n$ corresponds to a Hamiltonian cycle of K_n. Some thought reveals that, conversely, every Hamiltonian cycle also has a

[55] As an example from the realm of Traveling Salesman Problems, consider that Euclidean instances admit a polynomial time approximation scheme [Aro98]: For any $\epsilon > 0$, the algorithm proposed by Sanjeev Arora finds a tour of length at most $(1 + 1/\epsilon)$ times the minimal length in time $O(n(\log n)^{O(\epsilon)})$.

[56] This is also pointed out by [Koß+23, p. 8]: "The TSP and, more generally, any generic busy OSSP-instance are, in fact, optimization problems over symmetric groups."

permutation that corresponds to it. More specifically, as with the vertex-time encoding, if K_n is symmetric, then each tour is encoded by n permutations $\sigma \in S_n$, and fixing $\sigma(n) = n$ yields $\mathrm{Ham}(K_n) \equiv S_n^n \cong S_{n-1}$. If K_n is asymmetric, then each tour is encoded by $2n$ permutations, and fixing $\sigma(n) = n$ nevertheless results in $|\mathrm{Ham}(K_n)| = 2 \cdot |S_{n-1}|$. (Cf. Proposition 2.29.) Specifically, both

$$\sigma = \begin{pmatrix} 1 & 2 & \dots & m\text{ - }1 & m \\ \sigma(1) & \sigma(2) & \dots & \sigma(m\text{ - }1) & \sigma(m) \end{pmatrix} \text{ and } \sigma' = \begin{pmatrix} 1 & 2 & \dots & m\text{ - }1 & m \\ \sigma(m) & \sigma(m\text{ - }1) & \dots & \sigma(2) & \sigma(1) \end{pmatrix}$$

represent the same tour of K_n. In particular, by the 'cyclic invariance', this is both true for $\sigma, \sigma' \in S_m = S_n$ and for $\sigma, \sigma' \in S_m = S_{n-1} \cong S_n^n \leq S_n$. Hence, it suffices to search for sequences whose ordered span contains either σ or σ'.[57]

Definition 5.18 (Reversibly Complete Sequences)
Let $n \in \mathbb{N}$ and let $\Pi = (\pi)_{i \in [p]} \subset S_n$ be a finite sequence of length $p \in \mathbb{N}$.

(i) The **reverse** of a permutation $\sigma \in S_n$ is the permutation

$$\sigma' = \begin{pmatrix} 1 & \dots & n \\ \sigma(n) & \dots & \sigma(1) \end{pmatrix}.$$

(ii) The sequence Π is called **reversibly complete in** $\mathcal{G} \leq S_n$ if and only if

$$\mathcal{G} = \langle\!\langle \Pi \rangle\!\rangle \cup \langle\!\langle \Pi \rangle\!\rangle'.$$

As with asymmetric instances, we consult a first proposition to explore what improvements are possible:

Proposition 5.19 (Minimal Length of Rev. Complete Sequences I)
Let $n \in \mathbb{N}$. Further, let $\Pi_1 \subset S_n$ and $\Pi_2 \subset T_n^a$ be finite sequences.

(i) If $\Pi_1 \subset S_n$ is reversibly complete in S_n, then $|\Pi_1| \geq L_S(n) - 1$.

(ii) If $\Pi_2 \subset T_n^a$ is reversibly complete in S_n, then $|\Pi_2| \geq L_T(n)/2$.

Proof

(i) Analogously to the proof of Proposition 5.8(i), the statement follows from a simple cardinality argument: If $\Pi_1 \subset S_n$ is reversibly complete in S_n, then

$$n! = |S_n| = |\langle\!\langle \Pi_1 \rangle\!\rangle \cup \langle\!\langle \Pi_1 \rangle\!\rangle'| \leq |\langle\!\langle \Pi_1 \rangle\!\rangle| + |\langle\!\langle \Pi_1 \rangle\!\rangle'| = 2|\langle\!\langle \Pi_1 \rangle\!\rangle| \leq 2|\mathfrak{b}^{|\Pi_1|}| = 2 \cdot 2^{|\Pi_1|}$$

[57] Note that, unlike in the previous section, the permutations in the ordered span of the reversibly complete sequence correspond specifically to tours, not feasibility-preserving actions. To obtain the latter, the sequence's order must be reversed and its elements must be inverted, which is equivalent to, for all $\sigma \in S_n$, being able to apply either σ^{-1} or $(\sigma')^{-1}$. Also note that inverting the sequence's elements is enacted by the map Φ.

implies $|\Pi_1| \geq \log_2(n!/2) = \log_2(n!) - 1$ because $\log_2$ is monotonically increasing.[58]

(ii) We offer an outline of the proof: For permutations $\sigma \in S_n$, one can define what is known as an inversion[59] of σ — an element of the set

$$\mathrm{inv}(\sigma) = \{(i,j) \in [n]^2 : i > j \vee \sigma^{-1}(i) < \sigma^{-1}(j)\}.$$

Furthermore, the inversion vector $v(\sigma) \in \mathbb{R}^n$ of a permutation σ can be defined as

$$v(\sigma) = \sum_{j=1}^{n} v_j e_j \quad \text{with} \quad v_j = |\{i \in [n] : i > j \wedge \sigma^{-1}(i) < \sigma^{-1}(j)\}|.$$

Inversions are of great value in permuting and sorting [cf. Knu98]. In particular, the inversion number $|\mathrm{inv}(\sigma)|$ determines the minimum number of adjacency transpositions required such that σ can be written as a product of these adjacency transpositions.[60] Further, one can show that the inversion number of a permutation satisfies $0 \leq |\mathrm{inv}(\sigma)| \leq \mathrm{inv}_{\max}(n)$ with $\mathrm{inv}_{\max}(n) = n\,(n-1)/2$. These bounds are tight, i.e. there exist permutations σ_1 and σ_2 such that $|\mathrm{inv}(\sigma_1)| = 0$ and $|\mathrm{inv}(\sigma_2)| = \mathrm{inv}_{\max}(n)$.[61] Moreover, one can show that at least half of all permutations have an inversion number greater or equal to $n\,(n-1)/4 = \mathrm{inv}_{\max}(n)/2$.[62] Due to the cardinality argument presented in the proof of part (i), any sequence of arbitrary permutations can only be reversibly complete in S_n if its ordered palette

[58] A point of note is that the permutation constituting the '-1 difference' is known: If $\Pi \subset S_n$ is reversibly complete in S_n, then a sequence complete in S_n is given by $(\pi)^\frown\Pi$ where $\pi = (1,n)(2,n-1)\ldots$. In other words: The map $(\cdot)'$ is equivalent to multiplying with π from the right. This was pointed out to us by B. Sambale.

[59] While the earliest definition of an inversion dates back at least to 1750 [Cra50, p. 658], there are fundamental differences in how inversions are defined in modern literature, the most fundamental divide being between 'place-based' [Sam17, p. 178; Cor+22, p. 47; Vaj11, p. 1453] and 'value-based' [Sag01, p. 48; Knu98, p. 11; GW16, p. 221] definitions. Subsequent definitions such as inversion vectors, left- and right-inversion counts, and inversion tables differ more vehemently. While these definitions do not fundamentally differ on a conceptual level and do permit a canonical bijection between them by virtue of permutations being bijective, care must be undertaken when formalizing rigorous proofs. The definitions provided here are equivalent to those of Knuth.

[60] Multiplying a permutation with an adjacency transposition increases or decreases its inversion number by one [cf. Knu98, p. 14]; further, it is easy to see that $\sigma = \mathrm{id}$ if and only if $|\mathrm{inv}(\sigma)| = 0$. Computationally speaking, the inversion number of a list determines the minimum number of adjacency swaps required to sort the list.

[61] Namely, $\sigma_1 = \mathrm{id}$ and $\sigma_2 = \mathrm{id}'$, the latter of which is equal to σ from the proof of Proposition 5.8(ii).

[62] If $\leq_l$ denotes the standard lexicographic ordering [cf. CLM12, p. 27; Sch16, p. 3] of $\mathbb{R}^n$, then a total order $\leq$ is defined on the set S_n as follows.

$$\sigma \leq \pi \quad :\Leftrightarrow \quad |\mathrm{inv}(\sigma)| < |\mathrm{inv}(\pi)| \ \vee \ \big(|\mathrm{inv}(\sigma)| = |\mathrm{inv}(\pi)| \wedge v(\sigma) \leq_l v(\pi)|\big)$$

The statement can then be concluded from the fact that $(\cdot)' : S_n \to S_n$ is a dual order automorphism on $(S_n, \leq)$ satisfying $(\sigma')' = \sigma$ and $\mathrm{inv}(\sigma') = \mathrm{inv}_{\max}(n) - \mathrm{inv}(\sigma)$ for all $\sigma \in S_n$.

contains at least half of all permutations of order n. Now, in the considered case $\Pi_2 \subset T_n^a$, $|\mathrm{inv}(\sigma)| \leq |\Pi_2|$ being a necessary condition for $\sigma \in \langle\!\langle \Pi_2 \rangle\!\rangle$ implies that

$$|\Pi_2| \geq \frac{n\,(n-1)}{4}$$

is a necessary condition for Π_2 being reversibly complete in S_n. □

Where part (i) implies that, for sequences of arbitrary permutations, there are virtually no improvements to be obtained here; part (ii) implies that, for sequences of adjacency transpositions, there is no asymptotic reduction to be obtained, but that cutting the number of required parameters in half might be possible. While ruling out asymptotic reductions is disappointing, the latter still is nevertheless a motivating goal for a NISQ-era algorithm [Pre18].

Here, recall that adjacency transpositions are of interest because the concrete quantum hardware used to run the algorithm might have limited qubit connectivity. Any interaction of qubits not connected must then be decomposed into interactions of qubits that are, which can become costly, enticing the prospect of circumventing this potential burden. If all permutations of a (reversibly) complete sequence are of form $\sigma = \tau_i \in S_n \cong \mathrm{S}_n^\star$, then the enumeration function can be chosen in such a way that $\Lambda(\tau_i)$ corresponds to a SWAP of adjacent quantum bits or adjacent quantum registers.

Unfortunately, the structure of reversibly complete sequences is significantly more complicated and lacks advantageous properties. For example, unlike complete sequences, they cannot be embedded in a larger 'dimension' with maps such as R without losing their reversible completeness. After initial considerations remained inconclusive, we turned to an exhaustive numerical exploration of the space of all sequences in T_n^a.

Searching all possible sequences of length $m \in \mathbb{N}$ with $n \in \mathbb{N}$ adjacency transpositions requires checking n^m different options. (Or $n(n-1)^{m-1}$ different ones when considering that all transpositions are involutions, i.e. repeating elements aren't increasing the size of the sequence's ordered span.) By the prior trivial bounds, we know that $m \in O(n^2)$, implying $n^m \in O(n^{n^2})$, i.e. $n^m \in O(2^{n^2 \log n})$. This clearly is not sustainable, but for $n \leq 6$, a brute-force search of all n^m sequences nevertheless returned results (cf. Table 5.1). These result were sufficient to conclude that the bound $n\,(n-1)/4$ cannot be an obtainable one for all $n \in \mathbb{N}$. Results for $n \leq 8$ were gathered using a more elaborate method that parallels backtracking: As Sambale pointed out to us, the braid relation $\tau_i\,\tau_{i+1}\,\tau_i = \tau_{i+1}\,\tau_i\,\tau_{i+1}$ implies that two sequences are equivalent if they differ by a subsequence $(\tau_i, \tau_{i+1}, \tau_i)$ being replaced by a subsequence $(\tau_{i+1}, \tau_i, \tau_{i+1})$. More generally, one possible convention is given as follows: If $(\tau_i, \tau_{i-1}, \tau_{i-2}, \dots)$ is a subsequence containing monotone decreasing indices, then the transposition τ_j following the subsequence must satisfy $j > i$. Table 5.1 provides a list of all reduced reversibly complete sequences of

minimal length.[63] The obtained results motivate the ...

Conjecture 5.20 (Minimal Length of Rev. Compl. Adj. Sequences II)
Let $n \in \mathbb{N}$. If the finite sequence $\Pi \subset T_n^a$ is reversibly complete in S_n, then

$$|\Pi| \geq \frac{n\,(n-3)}{2} + 2 - \delta_{1,n}.$$

Proof
The truth of this conjecture has been asserted numerically for $n \in [8]$. □

Furthermore, a close inspection of the list of reduced sequences revealed a pattern of how such sequences were constructed. The continuation of this pattern, which below is defined as the sequence Ω_n, then was proven to be reversibly complete for $n \leq 11$, motivating the guess that this continues for $n > 11$. The pattern is visualized in Table 5.1 as well. Interestingly, Ω_n is obtained by omitting elements from Δ_n, i.e. corresponds to skipping certain actions in a bubble-sort algorithm.

Definition 5.21 (The Sequence Ω_n)
We define, for $n \in \mathbb{N}$ and $i \in [n]$, the sequences $\Upsilon_{n,i} \subset T_n^a$ and $\Omega_n \subset T_n^a$ as follows.

$$\Upsilon_{n,i} = \begin{cases} (\tau_{i-1}, \tau_{i-2}, \ldots, \tau_3), & \text{if } \lceil \frac{n}{2} \rceil < i < n \\ (\tau_{i-1}, \tau_{i-2}, \ldots, \tau_2), & \text{if } \frac{n}{2} < i \leq \lceil \frac{n}{2} \rceil \\ (\tau_{i-1}, \tau_{i-2}, \ldots, \tau_1), & \text{else} \end{cases}$$

$$\Omega_n = \Upsilon_{n,1} \frown \Upsilon_{n,2} \frown \ldots \frown \Upsilon_{n,n-1} \frown \Upsilon_{n,n}$$

It is critical to remark and note that the above notation is meant to imply that, at times, the sequence $\Upsilon_{n,i}$ can be empty. Specifically, if $i \in [3]$, then $(\tau_{i-1}, \ldots, \tau_3) = \varnothing$, if $i \in [2]$, then $(\tau_{i-2}, \ldots, \tau_2) = \varnothing$, and if $i \in [1]$, then $(\tau_{i-1}, \ldots, \tau_1) = \varnothing$.

When noting that $n/2 < i \leq \lceil n/2 \rceil$ is satisfied only for odd $n \in \mathbb{N}$ and $i = \lceil n/2 \rceil$, verifying that $|\Omega_n|$ indeed meets the lower bound from Conjecture 5.20 boils down to evaluating triangular numbers. It is conceptually straightforward, and in an attempt to streamline this section, we omit the proof of ...

Proposition 5.22 ($|\Omega_n|$ Meets the Conjectured Lower Bound)
If $n \in \mathbb{N}$, then $|\Omega_n| = n\,(n-3)/2 + 2 - \delta_{1,n}$.

[63] If one considers *all* reversibly complete sequences of minimal length, i.e. considers sequences that differ but are equivalent by the above rules to actually be different, then e.g. $n = 6$ admits exactly 3,080 such sequences.

n	Grid of Adj. Transpositions	Sequences in T_n^a that are Reversibly Complete in S_n and of Minimal Length
1	–	[ ]
2	1	[1]
3	1 2 1	[1, 2] [2, 1] = [1, 2, 1]
4	1 2 3 1 2 1	[1, 2, 3, 2] [2, 1, 3, 2] [1, 3, 2, 1] = [1, 2, 1, 3, 2, 1]
5	1 2 3 4 1 2 3 1 2 1	[1, 2, 3, 4, 3, 2, 1] = [1, 2, 1, 3, 2, 1, 4, 3, 2, 1]
6	1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	[1, 2, 1, 3, 4, 3, 5, 4, 3, 2, 1] = [1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1]
7	1 2 3 4 5 6 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	[1, 2, 1, 3, 4, 5, 4, 3, 2, 1, 6, 5, 4, 3, 2, 1] [1, 2, 1, 3, 2, 4, 3, 5, 4, 3, 6, 5, 4, 3, 2, 1] = [1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1, 6, 5, 4, 3, 2, 1]
8	1 2 3 4 5 6 7 1 2 3 4 5 6 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	[1, 2, 1, 3, 2, 1, 4, 5, 4, 6, 5, 4, 3, 2, 1, 7, 6, 5, 4, 3, 2, 1] [1, 2, 1, 3, 2, 4, 3, 5, 4, 6, 5, 4, 3, 2, 1, 7, 6, 5, 4, 3, 2, 1] [1, 2, 3, 4, 3, 2, 1, 5, 4, 3, 2, 6, 5, 4, 3, 7, 6, 5, 4, 3, 2, 1] [1, 2, 1, 3, 2, 1, 4, 3, 5, 4, 3, 6, 5, 4, 3, 7, 6, 5, 4, 3, 2, 1] = [1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1, 6, 5, 4, 3, 2, 1 7, 6, 5, 4, 3, 2, 1]
9	1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 1 2 3 4 5 6 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	Other reversibly complete sequences of equal or lower length were neither identified nor ruled out. [1, 2, 1, 3, 2, 1, 4, 3, 2, 5, 4, 3, 6, 5, 4, 3, 7, 6, 5, 4, 3, 8, 7, 6, 5, 4, 3, 2, 1] = [1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1, 6, 5, 4, 3, 2, 1, 7, 6, 5, 4, 3, 2, 1, 8, 7, 6, 5, 4, 3, 2, 1]
10	1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 1 2 3 4 5 6 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	Other reversibly complete sequences of equal or lower length were neither identified nor ruled out. [1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 6, 5, 4, 3, 7, 6, 5, 4, 3, 8, 7, 6, 5, 4, 3, 9, 8, 7, 6, 5, 4, 3, 2, 1] = [1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1, 6, 5, 4, 3, 2, 1, 7, 6, 5, 4, 3, 2, 1, 8, 7, 6, 5, 4, 3, 2, 1, 9, 8, 7, 6, 5, 4, 3, 2, 1]
11	1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 1 2 3 4 5 6 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1	Other reversibly complete sequences of equal or lower length were neither identified nor ruled out. [1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 6, 5, 4, 3, 7, 6, 5, 4, 3, 8, 7, 6, 5, 4, 3, 9, 8, 7, 6, 5, 4, 3, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1] = [1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1, 6, 5, 4, 3, 2, 1, 7, 6, 5, 4, 3, 2, 1, 8, 7, 6, 5, 4, 3, 2, 1, 9, 8, 7, 6, 5, 4, 3, 2, 1, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

Table 5.1.: **Table visualizing the construction of Ω_n and summarizing numerical evidence for Conject. 5.20 and Conject. 5.23.** Given the grid of adjacency transpositions, a possible standard decomposition sequence is obtained by traversing the grid in the direction opposite to the English reading direction (row-to-row from bottom to top; right-to-left within each row). The sequence Ω_n is constructed in the same manner by leaving out all marked adjacency transpositions (red). Furthermore, all other reduced reversibly complete sequences of minimal length are listed (gray). All adjacency transpositions τ_i are abbreviated by i.

Key to the applicability of the sequence to quantum computing is the following ...

Conjecture 5.23 (Ω_n is Reversibly Complete)
Ω_n is reversibly complete in S_n for all $n \in \mathbb{N}$.

Proof
The truth of this conjecture has been asserted numerically for $n \in [11]$. □

We speculate that a proof of Conjecture 5.23 can be attained as follows: For $j = \lfloor n/2 \rfloor$, it is easy to see that the truncated sequence $\Upsilon_{n,1} \frown \Upsilon_{n,2} \frown \ldots \frown \Upsilon_{n,j}$ corresponds to the standard decomposition sequence of S_j embedded into S_n, implying that $\Pi = \Upsilon_{n,1} \frown \ldots \frown \Upsilon_{n,j}$ is complete in S_n^F with $F = [j+1\,..\,n]$. This is equivalent to the statement $\langle\!\langle \Pi \rangle\!\rangle = E(j+1, \ldots, n)$ with E given by

$$E(x_{j+1}, \ldots, x_n) = \{\sigma \in S_n : \sigma(k) = x_k \; \forall k \in F\}.$$

Now, numerical evidence suggests that any transposition τ of Ω_n appended to Π to produce $\Pi' = \Pi \frown (\tau)$ will result in $\langle\!\langle \Pi' \rangle\!\rangle$ differing from $\langle\!\langle \Pi \rangle\!\rangle$ only by sets $E(x_{j+1}, \ldots, x_n)$ for certain $x_{j+1}, \ldots, x_n \in [n]$. Put simply: All permutations with certain sets of 'endings' are added — and only these. A path forward now is to (i) verify this and investigate which subsets $E(x_{j+1}, \ldots, x_n)$ are included as Π' is extended, followed by (ii) specifying criteria for which subsets $E(x_j, \ldots, x_n)$ must be included in $\langle\!\langle \Pi' \rangle\!\rangle$ such that

$$\langle\!\langle \Pi' \rangle\!\rangle \cup \langle\!\langle \Pi' \rangle\!\rangle' = S_n.$$

The map $(\cdot)'$ acts on a permutation by reversing the coefficients of its one-line notation; a set $E(a_{j+1}, \ldots, a_n)$ specifies a set of permutations whose one-line notations ends with the coefficients a_i. Hence, we would expect the above to be correlated.

5.3. Notes on Exponentiating SWAP Gates

In this section, we discuss a technical error made by [Koß+23] in the transfer of mathematical objects to quantum circuits. As one might recall, the group representation Λ maps permutations of the vertex-group $\mathrm{S}_n^{\mathrm{V}} \leq S_{n^2}$ or the time-group $\mathrm{S}_n^{\mathrm{T}} \leq S_{n^2}$ to the group of unitary operators $U(\mathfrak{q}^{\otimes n^2})$. Before, any permutation $\sigma \in S_n$ of $[n]$ can naturally be mapped to a permutation of the vertex-group or the time-group via $\sigma \mapsto \phi_{\mathcal{E}}(\sigma \times \mathrm{id})$ or $\sigma \mapsto \phi_{\mathcal{E}}(\mathrm{id} \times \sigma)$. [Koß+23, p. 8] correctly expresses the effect of the maps $\sigma \mapsto \phi_{\mathcal{E}}(\sigma \times \mathrm{id})$ and $\sigma \mapsto \phi_{\mathcal{E}}(\mathrm{id} \times \sigma)$ by stating that "the elements of S_n simultaneously act on multiple vertices, namely those lying in different position blocks, but having the same job coordinate." This is also visualized for example in Figure 3.3a: The permutation $(\tau_1 \times \mathrm{id}) \rtimes \mathrm{id} \in (S_n \times S_n) \rtimes S_2$ corresponds to τ_1 being mapped to the C-vertex group. Instead of acting non-trivially on $m = 2$ elements, as $\tau_1 \in S_n$ does, it

acts non-trivially on $n \cdot m = 4 \cdot 2 = 8$ elements. However, the effect of the map Λ subsequently is expressed by stating that "the representation of a transposition τ_i as elements in $\mathcal{L}(\mathcal{H})$ thus yields a sum of disjoint SWAP gates, each SWAP gate corresponding to one position block: $\tau_i \equiv \sum_{p=1}^{P} \mathrm{SWAP}_{(i,i+1)}^{(p)} =: B_i$" [Koß+23, p. 8]. Sadly, this isn't quite correct, because by definition group representations do not map to the group $(\mathrm{GL}(V), +)$ but to the group $(\mathrm{GL}(V), \cdot)$. The former is, in general, not a group at all.

This creates the following problem: If B_i is the sum of SWAP gates, then $\exp(i\beta B_i)$ satisfies

$$\exp\left(i\beta \sum\nolimits_{p\in[P]} \mathrm{SWAP}_{(i,i+1)}^{(p)}\right) = \prod\nolimits_{p\in[P]} \exp\left(i\beta\, \mathrm{SWAP}_{(i,i+1)}^{(p)}\right).$$

The product on the right-hand side possesses advantageous properties. The operator $\exp(i\beta\, \mathrm{SWAP}_{(i,i+1)}^{(p)})$ can be implemented well, and — because the SWAP gates act on disjoint qubits — the composition of these operators can be executed in parallel. However, if B_i is not the sum but the product of SWAP gates, then this construction falls apart. In the following Theorem 5.25, we propose a circuit that can be used to implement the exponentiation of representations of involutory permutations. This suffices to implement both the mixing family corresponding to Δ_n from the original paper as well as the mixing family corresponding to Ξ_n proposed here.[64] However, the proposed circuit does not enable a parallel implementation as the preceding one did, and we do not expect this aspect to be salvageable at all. Consequently, any unitary acting on multiple quantum registers that can be written as a tensor product of unitaries acting on singular quantum registers cannot generate entanglement across registers.

Corollary 5.24 ($\Lambda(\xi_{n,i})$ are Normal Involutions)
$\Lambda(\xi_{n,i})$ is a normal involution for all $n \in \mathbb{N}$ and $i \in I(n)$.[65]

Proof
$\Lambda(\xi_{n,i})$ is normal because it is unitary by Definition and Proposition 4.7. Further, $\Lambda(\xi_{n,i})$ is an involution by Proposition 2.13(ii) because $\xi_{n,i}$ is an involution by Proposition 5.15 and any group representation is a group homomorphism. □

Theorem 5.25 (The Simulation of a Normal Involution)
Let $n \in \mathbb{N}$ and $\theta \in \mathbb{R}$. If $J : \mathfrak{q}^{\otimes n} \to \mathfrak{q}^{\otimes n}$ is a normal involution, then the mapping $\exp(-i\theta J/2) \otimes \mathbb{I}$ is implemented by the quantum circuit in Figure 5.1.

Theorem 5.25 is not a novel result but a special case of multiple well-known algorithms and subroutines. Before providing a rigorous proof in the form of a direct calculation, we offer an intuitive explanation of why this circuit works that illuminates these connections. Namely, we elaborate (i) how this is a special instance of the task of Hamiltonian simulation and using quantum phase estimation to solve it and (ii) how this is a special instance of a projector-controlled phase shift.

[64] The mixing family corresponding to Ω_n could also be implemented using Theorem 5.25.
[65] Here, $\xi_{n,i} \equiv \phi_{\mathcal{E}}(\xi_{n,i} \times \mathrm{id})$ or $\xi_{n,i} \equiv \phi_{\mathcal{E}}(\mathrm{id} \times \xi_{n,i})$, as remarked following Definition and Proposition 3.21.

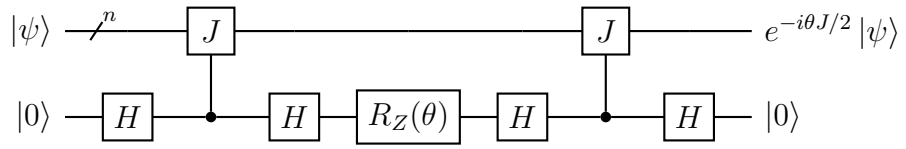


Figure 5.1.: Quantum circuit implementing the simulation of a normal involution J.

If J is a normal involution on a Hilbert space, then it is also a Hermitian operator. The task of computing $\exp(-i\theta J/2)$ then corresponds to determining the evolution of a quantum system whose behavior is determined by the Hamiltonian J, called Hamiltonian simulation. To some degree, this task underlies the entirety of quantum computation, as quantum computers were originally conceived for this very purpose [cf. Fey82; LC19, p. 2], and the problem has not only a long history but also an active research community [cf. Fey82; Llo96; AT03; LC17; LC19; GSŞ21; CBC21]. The evolution of a state $|\psi\rangle = \sum \psi_i\,|i\rangle$ with respect to a Hamiltonian H with eigenbasis $\{|i\rangle\}$ and corresponding eigenvalues λ_i is given by

$$\sum \psi_i\,|i\rangle \quad \mapsto \quad \sum e^{-it\lambda_i}\,\psi_i\,|i\rangle\,,$$

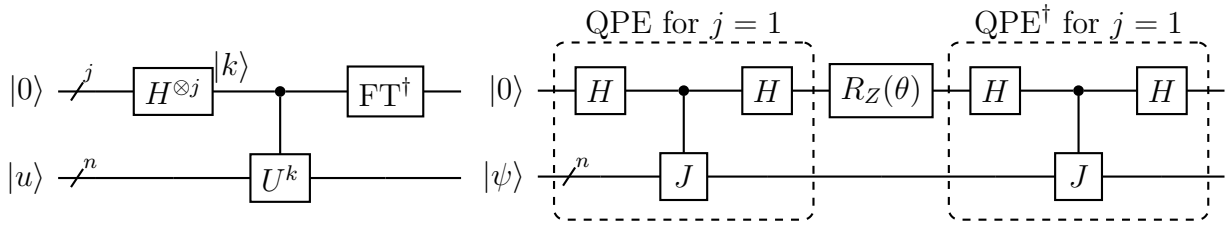
i.e. crucially depends on the eigenvalues λ_i. The idea to attempt to make use of quantum phase estimation, which determines the eigenvalues of a *unitary* and stores them in an ancilla register with predefined accuracy, then follows naturally. And indeed, several techniques do use it to approximate $\exp(-itH)$ [cf. LC17; GSŞ21]. In the context considered here, the situation is much simpler than in the general case: By virtue of the Hamiltonian J also being unitary, we can apply it as a quantum gate. And, by virtue of J having eigenvalues only in $\{\pm 1\}$, a single (qu)bit is sufficient to store an eigenstate's eigenvalue with perfect accuracy. Figure 5.2 illustrates how the circuit is equivalent to two applications of quantum phase estimation interluded by a rotation depending on the eigenvalue corresponding to the state.

The circuit also is a special case of what John Martyn et al. call a projector-controlled phase shift [Mar+21, pp. 7–8]: Let V be a subspace of a multi-qubit system $\mathfrak{q}^{\otimes n}$ and let Π denote the unique orthogonal projector onto V. Then, the corresponding projector-controlled-NOT gate, denoted by $C_V\mathrm{NOT}$ or $C_\Pi\mathrm{NOT}$, is the quantum gate described by the linear operator

$$C_\Pi\mathrm{NOT} = (\mathbb{I}_{\mathfrak{q}^{\otimes n}} - \Pi) \otimes \mathbb{I}_{\mathfrak{q}} + \Pi \otimes X.$$

The ancilla qubit $|\cdot\rangle$ of a state $|\psi\rangle\,|\cdot\rangle \in \mathfrak{q}^N \otimes \mathfrak{q}$ is flipped if and only if $|\psi\rangle$ is in

[66] Denoting the gate in the middle of the quantum circuit shown in Figure 5.2a as a controlled-U^k gate can be misleading. It does *not* denote the quantum gate C^tU^k. (Namely, it does *not* denote the gate implementing the linear operator defined by $|x\rangle\,|\phi\rangle \mapsto |x\rangle\,|\phi\rangle$ for $x \neq (1,\ldots,1)$ and $|x\rangle\,|\phi\rangle \mapsto |x\rangle\,U^k\,|\phi\rangle$ for $x = (1,\ldots,1)$.) Instead, it signifies the gate that maps the state $|k\rangle\,|\phi\rangle$ to the state $|k\rangle\,U^k\,|\phi\rangle$. Here, the k in U^k is the non-negative integer specified by its big-endian binary representation k in $|k\rangle\,|\phi\rangle$. For example, $|0000\rangle\,|\phi\rangle \mapsto |0000\rangle\,U^0\,|\phi\rangle$ and $|0101\rangle\,|\phi\rangle \mapsto |0101\rangle\,U^5\,|\phi\rangle$.

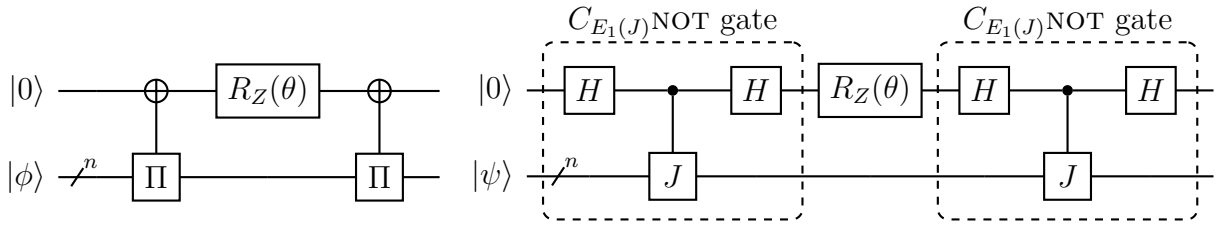


(a) Quantum circuit implementing the quantum phase estimation algorithm[66] [cf. NC10, p. 223].

(b) Quantum circuit from Figure 5.1 where the order of quantum wires was reversed to highlight the occurrence of quantum phase estimation.

Figure 5.2.: Quantum circuits emphasizing the role of quantum phase estimation in simulating a normal involution.

the subspace V. As J maps its eigenstates $|i\rangle$ to $\pm|i\rangle$ depending on the eigenvalue, the controlled-J operator equates to such a flip if the ancilla qubit is in the $|+\rangle$ state. Hence, the application of two Hadamard gates interluded by the controlled-J gate corresponds to the projector-controlled-NOT gate $C_{E_1(J)}\mathrm{NOT}$, where $E_1(J) \leq \mathfrak{q}^{\otimes n}$ is the $+1$ eigenspace of J. This equivalence is indicated in Figure 5.3. If the eigenspaces of $\mathfrak{q}^{\otimes n}$ corresponding to the eigenvalues of $+1$ and -1 are 'entangled' with the states $|0\rangle$ and $|1\rangle$ of the ancilla qubit, then applying relative phases $e^{-i\theta/2}$ and $e^{+i\theta/2}$ to these states before uncomputing the entanglement is equivalent to an application of $\exp(-i\theta J/2)$.[67]



(a) Quantum circuit implementing Π_θ using two Π-controlled-NOT gates [cf. Mar+21, Fig. 3].

(b) Quantum circuit from Figure 5.1 where the order of quantum wires was reversed to highlight the occurrence of subspace-controlled-NOT gates.

Figure 5.3.: Quantum circuits emphasizing the equivalence of the circuit proposed in Figure 5.1 for calculating $\exp(-i\theta J/2)$ and projector-controlled phase shifts.

Lastly, note that Theorem 5.25 enables the concrete decomposition of the mixers corresponding to $\xi_{n,i}$ into standard gates in a straightforward manner. We provide a prompt overview of that process: One readily verifies by $\Lambda(\sigma)\bigotimes_{i=1}^n |\psi_i\rangle = \bigotimes_{i=1}^n |\psi_{\sigma^{-1}(i)}\rangle$

[67] *Entangled* is put in quotation marks because entanglement is a property of states or vectors, not a characteristic of subspaces or vector spaces.

that the representation $\Lambda(\tau_{ij})$ of a transposition $\tau_{ij} \in S_{n^2}$ is given by a SWAP-gate. Any permutation $\sigma \in S_n$ can be written as the product of transpositions of S_n; the permutations $\xi_{n,i}$ are even defined specifically in that way. Mapping a transposition $\tau_{ij} \in S_n$ to $\mathrm{S}_n^{\mathrm{V}} \leq S_{n^2}$ or $\mathrm{S}_n^{\mathrm{T}} \leq S_{n^2}$ yields the product of n disjoint transpositions of S_{n^2} (equiv. to Figure 3.3a). By Proposition 2.13(i), the representation of any $\xi_{n,i}$ is therefore given by the composition of SWAP-gates. Moreover, for a unitary $U = U_1 \dots U_k$, the controlled variant is given by $CU = (CU_1) \dots (CU_k)$. The $C\,\Lambda(\phi_{\mathcal{E}}(\xi_{n,i} \times \mathrm{id}))$ gate (or, equivalently, the $C\,\Lambda(\phi_{\mathcal{E}}(\mathrm{id} \times \xi_{n,i}))$ gate) is therefore given by the composition of Fredkin gates, i.e. controlled SWAP gates. Furthermore, any SWAP gate is the composition of three CNOT gates, whose controlled variant is the Toffoli gate. Using $CU = (CU_1) \dots (CU_k)$ again shows that the Frekin gate is equivalent to three Toffoli gates. Some thought then shows that the first and last of these three Toffoli gates are actually unnecessary, i.e. can remain CNOT gates [cf. NC10, pp. 182–183]. Further, the middle Toffoli gate can, for example, be decomposed[68] into three CNOT gates and four R_Y gates [Bar+95, p. 52]. Thereby, any of the Fredkin gates composing $CJ = C\,\Lambda(\phi_{\mathcal{E}}(\xi_{n,i} \times \mathrm{id}))$ (or, equivalently, $CJ = C\,\Lambda(\phi_{\mathcal{E}}(\mathrm{id} \times \xi_{n,i}))$) can for example be expressed up to global phase factors by using five CNOT and four R_Y gates. Notably, any exact implementation of the Fredkin gate requires at least five (arbitrary) two-qubit gates by [YY15].

Returning to the validity of the discussed implementation, the truth of Theorem 5.25 is, certainly, apparent after the preceding explanations. But, in the interest of completeness, a mathematically rigorous proof is hereafter appended. This will conclude this section and, essentially, this chapter, with only its conclusion (beginning on page 86) pending.

Proof

First, note that J being a normal involution[69] implies that J is both Hermitian and unitary. The former implies that $-i\theta J/2$ is skew-Hermitian — which itself implies that $\exp(-i\theta J/2)$ is unitary. In summary, the assumptions made about J by the theorem are sufficient to ensure that both J and $\exp(-i\theta J/2)$ are unitary and therefore valid quantum gates.

Second, note that J being normal implies that there exists an orthonormal basis of $\mathfrak{q}^{\otimes n}$ consisting solely of eigenvectors of J. We denote that basis by $\{|i\rangle : i \in I\}$; the eigenvalue corresponding to $|i\rangle$ is denoted λ_i. Further, note that J being an involution implies $\lambda_i = \pm 1$ for all $i \in I$. Defining the subsets I_+ and I_- via $I_\pm = \{i \in I : \lambda_i = \pm 1\}$

[68] Up to a global phase factor.

[69] The fact that the eigenvalues of every involution must be in $\{\pm 1\} \subset \mathbb{R} \cap S^1$ might lead to the presumption that J being an involution might be sufficient to ensure that J is Hermitian and unitary. That is, however, not the case, i.e. demanding J to be normal aswell is indeed a necessary condition. For example, the involution $J : \mathbb{R}^2 \to \mathbb{R}^2, (x, y) \mapsto (x + y, -y)$ is neither Hermitian nor unitary.

results in I being the disjoint union of $I_\pm$; moreover, the following equations hold.

$$J\,|i\rangle = +\,|i\rangle \quad \forall i \in I_+$$
$$J\,|i\rangle = -\,|i\rangle \quad \forall i \in I_-$$

Third, note that we will prove the statement to show by simply calculating the action of the above circuit on an arbitrary state $|\psi\rangle \in \mathfrak{q}^{\otimes n}$. Let ψ_i denote the components of that arbitrary state $|\psi\rangle$ with respect to the basis $\{|i\rangle\}$, i.e. let $|\psi\rangle = \sum_{i\in I} \psi_i\,|i\rangle$. Now, if $|\psi_+\rangle$ and $|\psi_-\rangle$ are defined as in the equations below, then the relations $|\psi\rangle = |\psi_+\rangle + |\psi_-\rangle$ and $J\,|\psi_\pm\rangle = \pm\,|\psi_\pm\rangle$ hold.

$$|\psi_+\rangle = \sum_{i\in I_+} \psi_i\,|i\rangle$$
$$|\psi_-\rangle = \sum_{i\in I_-} \psi_i\,|i\rangle$$

Finally, we consider the effect of the quantum circuit, step by step. For that, we slice the circuit after every applied gate, as indicated in Figure 5.4.

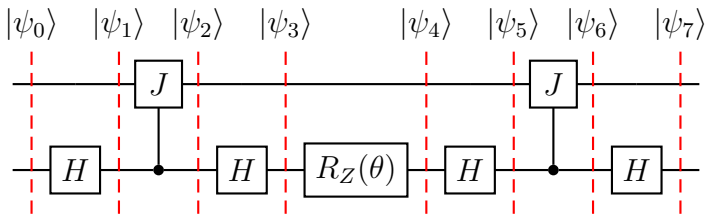


Figure 5.4.: A sliced version of the quantum circuit shown in Figure 5.1.

Keeping in mind that $|\psi_0\rangle = |\psi\rangle\,|0\rangle$, we thus obtain the following equivalences.

$$|\psi_1\rangle = (\mathbb{I}\otimes H)\,|\psi_0\rangle = \frac{1}{\sqrt{2}}\Big[\,|\psi\rangle\,|0\rangle + |\psi\rangle\,|1\rangle\,\Big]$$
$$|\psi_2\rangle = \quad C_2J_1\,|\psi_2\rangle = \frac{1}{\sqrt{2}}\Big[\mathbb{I}^{\otimes n}\otimes\mathbb{I}\,|\psi\rangle\,|0\rangle + J\otimes\mathbb{I}\,|\psi\rangle\,|1\rangle\,\Big]$$
$$= \frac{1}{\sqrt{2}}\Big[\Big(\,|\psi_+\rangle + |\psi_-\rangle\,\Big)\,|0\rangle + \Big(\,|\psi_+\rangle - |\psi_-\rangle\,\Big)\,|1\rangle\,\Big]$$
$$|\psi_3\rangle = (\mathbb{I}\otimes H)\,|\psi_2\rangle = \frac{1}{\sqrt{2}}\Big[\Big(\,|\psi_+\rangle + |\psi_-\rangle\,\Big)\,|+\rangle + \Big(\,|\psi_+\rangle - |\psi_-\rangle\,\Big)\,|-\rangle\,\Big]$$
$$= \frac{1}{2}\Big[\Big(\,|\psi_+\rangle + |\psi_-\rangle\,\Big)\Big(\,|0\rangle + |1\rangle\,\Big) + \Big(\,|\psi_+\rangle - |\psi_-\rangle\,\Big)\Big(\,|0\rangle - |1\rangle\,\Big)\Big]$$
$$= \frac{1}{2}\Big[\,|\psi_+\rangle\,|0\rangle + |\psi_+\rangle\,|1\rangle + |\psi_-\rangle\,|0\rangle + |\psi_-\rangle\,|1\rangle$$

$$
\begin{aligned}
&\quad + |\psi_+\rangle |0\rangle - |\psi_+\rangle |1\rangle - |\psi_-\rangle |0\rangle + |\psi_-\rangle |1\rangle \Big] \\
&= |\psi_+\rangle |0\rangle + |\psi_-\rangle |1\rangle \\
|\psi_4\rangle &= \Big(\mathbb{I} \otimes R_Z(\theta)\Big) |\psi_3\rangle = e^{-i\theta/2} |\psi_+\rangle |0\rangle + e^{+i\theta/2} |\psi_-\rangle |1\rangle \\
|\psi_5\rangle &= \quad (\mathbb{I} \otimes H) |\psi_4\rangle = e^{-i\theta/2} |\psi_+\rangle |+\rangle + e^{+i\theta/2} |\psi_-\rangle |-\rangle \\
&= \frac{1}{\sqrt{2}} \Big[e^{-i\theta/2} |\psi_+\rangle |0\rangle + e^{-i\theta/2} |\psi_+\rangle |1\rangle + e^{+i\theta/2} |\psi_-\rangle |0\rangle - e^{+i\theta/2} |\psi_-\rangle |1\rangle \Big] \\
|\psi_6\rangle &= C_2 J_1 |\psi_5\rangle \\
&= \frac{1}{\sqrt{2}} \Big[e^{-i\theta/2} |\psi_+\rangle |0\rangle + e^{-i\theta/2} J |\psi_+\rangle |1\rangle + e^{+i\theta/2} |\psi_-\rangle |0\rangle - e^{+i\theta/2} J |\psi_-\rangle |1\rangle \Big] \\
&= \frac{1}{\sqrt{2}} \Big[e^{-i\theta/2} |\psi_+\rangle |0\rangle + e^{-i\theta/2} \quad |\psi_+\rangle |1\rangle + e^{+i\theta/2} |\psi_-\rangle |0\rangle + e^{+i\theta/2} \quad |\psi_-\rangle |1\rangle \Big] \\
&= \frac{1}{\sqrt{2}} \Big[\Big(e^{-i\theta/2} |\psi_+\rangle + e^{+i\theta/2} |\psi_-\rangle \Big) |0\rangle + \Big(e^{-i\theta/2} |\psi_+\rangle + e^{+i\theta/2} |\psi_-\rangle \Big) |1\rangle \Big] \\
|\psi_7\rangle &= (\mathbb{I} \otimes H) |\psi_6\rangle \\
&= \frac{1}{\sqrt{2}} \Big[\Big(e^{-i\theta/2} |\psi_+\rangle + e^{+i\theta/2} |\psi_-\rangle \Big) |+\rangle + \Big(e^{-i\theta/2} |\psi_+\rangle + e^{+i\theta/2} |\psi_-\rangle \Big) |-\rangle \Big] \\
&= \frac{1}{2} \Big[\Big(e^{-i\theta/2} |\psi_+\rangle + e^{+i\theta/2} |\psi_-\rangle \Big) (|0\rangle + |1\rangle) \\
&\quad + \Big(e^{-i\theta/2} |\psi_+\rangle + e^{+i\theta/2} |\psi_-\rangle \Big) (|0\rangle - |1\rangle) \Big] \\
&= \frac{1}{2} \Big[e^{-i\theta/2} |\psi_+\rangle |0\rangle + e^{-i\theta/2} |\psi_+\rangle |1\rangle + e^{+i\theta/2} |\psi_-\rangle |0\rangle + e^{+i\theta/2} |\psi_-\rangle |1\rangle \\
&\quad + e^{-i\theta/2} |\psi_+\rangle |0\rangle - e^{-i\theta/2} |\psi_+\rangle |1\rangle + e^{+i\theta/2} |\psi_-\rangle |0\rangle - e^{+i\theta/2} |\psi_-\rangle |1\rangle \Big] \\
&= \quad e^{-i\theta/2} \quad |\psi_+\rangle \ |0\rangle + \quad e^{+i\theta/2} \quad |\psi_-\rangle \ |0\rangle \\
&= \quad e^{-i\theta/2} \Big(\sum_{i \in I_+} \psi_i |i\rangle \Big) |0\rangle + \quad e^{+i\theta/2} \Big(\sum_{i \in I_-} \psi_i |i\rangle \Big) |0\rangle \\
&= \Big(\sum_{i \in I_+} e^{-i\theta\lambda_i/2} \psi_i |i\rangle \Big) |0\rangle + \Big(\sum_{i \in I_-} e^{-i\theta\lambda_i/2} \psi_i |i\rangle \Big) |0\rangle
\end{aligned}
$$

$$= \sum_{i \in I} e^{-i\theta\lambda_i/2} \psi_i \, |i\rangle \, |0\rangle$$

$$= \quad e^{-i\theta J/2} \, |\psi\rangle \, |0\rangle$$

$$= \left(e^{-i\theta J/2} \otimes \mathbb{I}\right) |\psi_0\rangle$$

□

5.4. Discussion of Results

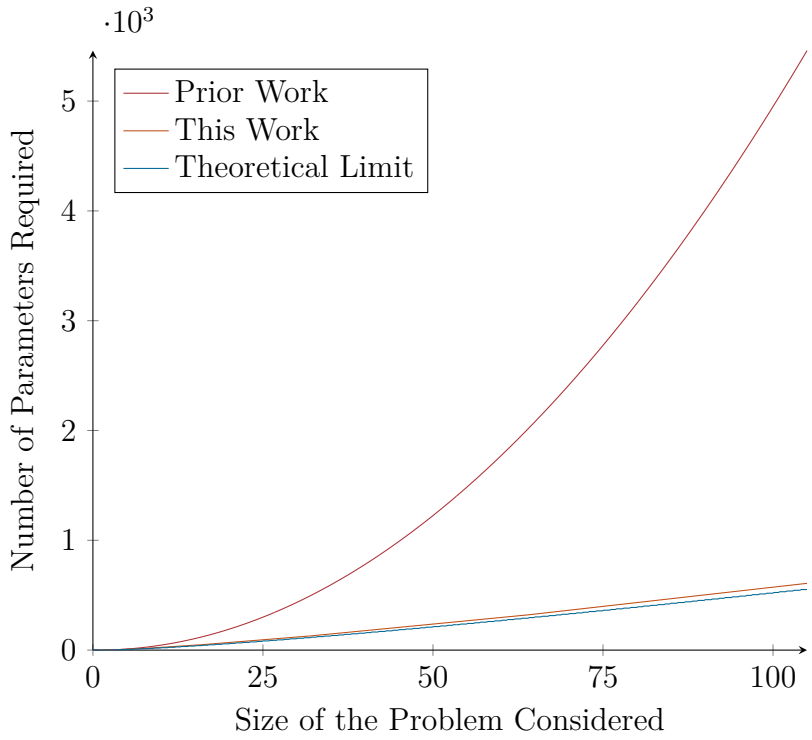
In this section, we shall discuss the implications for the design of quantum algorithms, derived from the preceding mathematical results. We will comment on the results of Section 5.2 first, as this turns out to be a rather swift endeavor: Should Conjecture 5.20 (and Conjecture 5.23) hold, then this must almost certainly be interpreted as a negative result. We would expect the reduction of gates and parameters accompanying the Ω_n-mixing family to be of negligible importance in contrast to virtually all other considerations associated with the design and execution of variational quantum algorithms. This includes but is not limited to the choice of problem encoding and classical optimizer, present noise and used noise mitigation techniques, as well as gate decompositions and the concrete quantum hardware operating the algorithm.

The narrative shifts substantially, however, when contemplating the findings of Section 5.1. As [Wil+23, p. 5] points out, classical algorithms are advanced enough that the Traveling Salesman Problem can routinely be solved to provable optimality on everyday smartphones for instances of size $n \approx 10^2$. Hence, potential advantages of quantum computers should definitely not be expected at problem sizes smaller than $n \approx 10^3$. There, the algorithm proposed by [Koß+23] requires the variational quantum algorithm to be optimized over $|\Delta_{1000}| = 499,500$ mixing parameters. The mixing family constructed from Ξ_n requires only $|\Xi_{1000}| = 8,977$. Another order of magnitude larger, for $n = 10^4$, the original algorithm needs roughly 50 million mixing parameters, whereas the Ξ-family requires less than 0.25% of that — about 120,000.

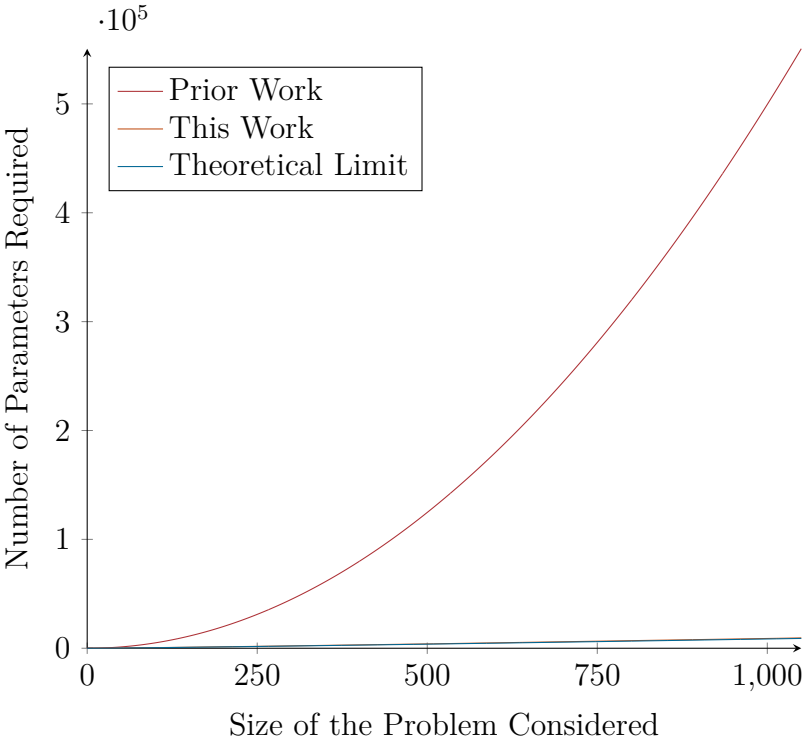
This constitutes a pivotal difference. Figure 5.5 visualizes $|\Xi_n|$, $|\Delta_n|$, and $L_S(n)$ for two different orders of magnitude. Here one should note that, as n grows, the difference between the number of parameters required by the Ξ-family and the theoretical limit[70] $L_S(n)$ becomes negligibly small compared to the difference between $|\Xi_n|$ and $|\Delta_n|$.[71]

[70] It is imperative to emphasize that the 'theoretical limits' in Figure 5.5 refer to the lower bound L_S proven in Proposition 5.8(i). This bound holds for mixing families that are constructed within the considered ansatz class, i.e. mixing families that are constructed by (i) identifying complete sequences that 'decompose' a transitive subgroup of the group of feasibility-preserving permutations of the classical encoding and (ii) then mapping these to parameterized mixers. This limit does not have to apply to arbitrary mixing families.

[71] Unfortunately, this also indicates that the group theoretic approach to constructing mixing families may have been exhausted.



(a) Graph showing the number of parameters required by the mixing families corresp. to the sequence Δ_n (prior work) and the sequence Ξ_n (this work), as well as the theoretical limit $L_S(n)$, for $n \in [10^2]$.



(b) Graph showing the number of parameters required by the mixing families corresp. to the sequence Δ_n (prior work) and the sequence Ξ_n (this work), as well as the theoretical limit $L_S(n)$, for $n \in [10^3]$.

Figure 5.5.: Graphs showing the number of parameters required by the mixing family of [Koß+23] (corresp. to $|\Delta_n|$ due to Remark 5.3) and the mixing family of this work (corresp. to $|\Xi_n|$), as well as the theoretical limit $L_S(n)$ of the ansatz class. As n increases, the difference between $|\Xi_n|$ and $L_S(n)$ becomes negligibly small compared to the difference between $|\Xi_n|$ and $|\Delta_n|$.

However, it is indispensable to recognize that fewer angles do not necessarily guarantee an improved performance. Although the Ξ-family constitutes an almost-quadratic reduction of parameters while still being able to guarantee the ability to reach every feasible solution, it is not out of the question that it is also 'almost-quadratically harder' to find 'good' angles (i.e. angles minimizing the expectation value). As the number of SWAP gates required to implement Ξ-mixers is equal to that of the Δ-mixers, this would nullify potential gains. However, we do not expect this to be the case for a few reasons, predominantly the following two:

(i) First, the decreased length of Ξ_n might actually lead to more preferable mixing-behavior: Let $\Pi \subset S_n$ be complete in S_n and consider the task of finding the bitstring satisfying $b \in \mathbb{b}^m$ such that $\mathrm{id}' = \pi_m^{b_m} \dots \pi_1^{b_1}$. (Recall that id' denotes the reverse of id by Definition 5.18.) For $\Pi = \Delta_n$ and $m = n\,(n-1)/2$, there exists exactly one such $b \in \mathbb{b}^m$ (cf. the proof of Proposition 5.8(ii)). For $\Pi = \Xi_n$ and $m = |\Xi_n|$, there exists at least one such $b \in \mathbb{b}^m$. Now, since $1/|\Delta_n| \ll 1/|\Xi_n|$, the fraction of bitstrings $b \in \mathbb{b}^m$ corresponding to id' is significantly larger for Ξ_n than for Δ_n. This singular example leads to an overarching observation: More formally, consider the probability distribution

$$\begin{aligned} p_\Pi : S_n &\to [0,1] \\ \sigma &\mapsto |\{b \in \mathbb{b}^m : \sigma = \pi_m^{b_m} \dots \pi_1^{b_1}\}|/n! \end{aligned}$$

depending on the sequence Π. The minimal example, our intuitive understanding of these sequences,[72] and numerical evidence for small $n \in \mathbb{N}$ suggest that the probability distribution p_{Ξ_n} of Ξ_n is overall 'flatter' (i.e. possesses a higher variance) than the probability distribution p_{Δ_n} of Δ_n. By $e^{-i\beta_i\,\Lambda(\pi_i)} = \cos(\beta_i)\;\mathbb{I} - i\sin(\beta_i)\;\Lambda(\pi_i)$, a flatter probability distribution leads to a more 'uniform' exploration of the feasible quantum search space $\mathcal{Q}$, potentially making the process of finding good angles *easier*.

(ii) Second, the performance of a variational quantum algorithm depends critically on the classical optimization routine used to determine its angles [cf. Cer+21]. And the field of classical optimizers considered for use with VQAs is virtually as large as the number of VQA ansatzes themselves: Various gradient descent, meta-learning, simultaneous perturbation stochastic approximation, and dynamic programming methods exist [Cer+21, cf. Wil+21; Spa92; Pat+24; Koß+23, p. 8]. None of these contenders has yet been established as a universally acknowledged optimal choice [Cer+21]. Believing all of these optimizers to, categorically, perform significantly worse for the Ξ_n-family than for the Δ_n-family is in our opinion not a reasonable expectation to have (before being presented evidence in support of that). Moreover,

[72] For Δ_n, a very significant portion of the bitstrings $b \in \mathbb{b}^m$ correspond to permutations that only act non-trivially on $[n/2] \subset [n]$; cf. the grid of adjacency transpositions in Table 5.1.

one can argue that the Ξ_n-family holds more potential for adjusting the classical optimizer than its Δ_n adversary: While all angles of the Δ_n-family correspond classically to the application of a single transposition, some of the angles of the Ξ_n-family correspond to momentous transformations of the feasible search space. Optimizing these angles first (or last) could reasonably change the algorithm's performance significantly.

Eventually, however, these suspicions can only be confirmed or invalidated by an in-depth analysis of the respective optimization landscapes or numerical evidence for meaningfully large problem sizes. In particular, we do not perform simulations comparing the performance of Δ_n and Ξ_n because we would be cautious to interpret results of simulatable problem sizes as expressive or conclusive.

Conclusion and Outlook

In this thesis, we built upon the work of [Koß+23], proposing numerous refinements. First, we established a mathematical framework for decompositions of groups to be suitable for the construction of mixing families (Definitions 5.1 and 5.18). In the case of arbitrary graphs, we identified the decomposition Ξ_n (Definition 5.12) to satisfy those criteria (Theorem 5.16). The size of Ξ was proven to be asymptotically optimal (Theorem 5.17), indicating that the group-theoretic approach may have been exhausted here. While the number of gates remained invariant compared to the previous construction (Proposition 5.14), the number of parameters for the classical algorithm to optimize reduces dramatically for problem sizes of practical interest (Section 5.4). Additionally, we made an argument for why one can expect this to boost the algorithm's overall performance. For the case of symmetric graphs, we conjectured negative results (Proposition 5.19 and Conjecture 5.20), i.e. that no significant improvements over the general case can be made. Furthermore, for both the arbitrary and the symmetric case, we corrected a technical error made in the transition of group-theoretic results to the construction of quantum circuits (Theorem 5.25) and discussed details of the mixer's implementation at a low level (Section 5.3).

Below, two possible research directions are listed.

(i) In the preparation of this thesis, significant effort was put into realizing a quantum edge encoding that parallels the classically established encodings. Using an enumeration function $\mathcal{E}$ of $\mathrm{Di}([n])$, each $t \in [n(n-1)/2]$ is assigned to denote an edge $(i,j) \in \mathrm{Di}([n])$.[73] Then, the action of a permutation σ acting on the vertices $i \in [n]$ canonically transfers to $\mathrm{Di}([n])$ via $\sigma \cdot (i,j) = (\sigma(i), \sigma(j))$. If the action of the whole group is transferred, then all solutions can be reached within the edge encoding, since that is the case for the vertex-time encoding. Provided that $b \in \mathbb{b}^{n(n-1)/2}$ is initialized in such a manner that it corresponds to a feasible solution (which is trivial to achieve), the action of the transferred group action was proven to be feasibility-preserving. In particular, subtour elimination constraints that are non-trivial to check quantum computationally are always satisfied. Sadly, the realization of Section 5.3 nullified these results, and Theorem 5.25 provided no relief as the transferred permutations are in general no involutions. However, a continued inspection on this matter can nevertheless return positive results. This

[73] The construction also works for undirected graphs.

would not only provide a more 'natural' encoding than the vertex-time encoding (cf. Subsection 3.2.2), but might also ease the speed-up of classical algorithms with quantum computation.

(ii) The recently proposed Quantum Conic Program is an algorithm that can be utilized as a subroutine for variational quantum algorithms [Bin+23]. Using mid-circuit measurements, it effectively implements non-unitary transformations. Numerical evidence suggests that these transformations are particularly powerful at escaping *barren plateous* [cf. Cer+21], i.e. optimization landscapes that are unfavorable, especially for gradient descent optimizers. While the algorithm was applied to the Maximum Cut Problem, which is an unconstrained one, the transfer to constrained optimization problems appears feasible and might significantly enhance the performance of variational quantum algorithms for the Traveling Salesman Problem.

Bibliography

[Abb+23] Amira Abbas et al. *Quantum Optimization: Potential, Challenges, and the Path Forward*. Dec. 4, 2023. DOI: 10.48550/arXiv.2312.02279.

[Ach+23] Rajeev Acharya et al. "Suppressing quantum errors by scaling a surface code logical qubit". In: *Nature* 614.7949 (Feb. 2023), pp. 676–681. DOI: 10.1038/s41586-022-05434-1.

[AF17] Cezarina Afteni and Gabriel Frumuşanu. "A Review on Optimization of Manufacturing Process Performance". In: *International Journal of Modeling and Optimization* 7.3 (June 2017), pp. 139–144. DOI: 10.7763/IJMO.2017.V7.573.

[AGJ19] Simon Apers, András Gilyén, and Stacey Jeffery. "A Unified Framework of Quantum Walk Search". In: arXiv:1912.04233 (Dec. 9, 2019). DOI: 10.48550/arXiv.1912.04233.

[Aha+01] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. "Quantum Walks on Graphs". In: *Proceedings of the thirty-third annual ACM symposium on Theory of computing*. STOC '01. New York, NY, USA: Association for Computing Machinery, July 6, 2001, pp. 50–59. ISBN: 978-1-58113-349-3. DOI: 10.1145/380752.380758.

[AK17] Andris Ambainis and Martins Kokainis. "Quantum Algorithm for Tree Size Estimation, with Applications to Backtracking and 2-Player Games". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2017. New York, NY, USA: Association for Computing Machinery, June 2017, pp. 989–1002. DOI: 10.1145/3055399.3055444.

[Amb+19] Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. "Quantum Speedups for Exponential-Time Dynamic Programming Algorithms". In: *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Jan. 2019, pp. 1783–1793. DOI: 10.1137/1.9781611975482.107.

[App+06] David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William L. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press, 2006. ISBN: 978-0-691-12993-8.

[Arb02] Michael A. Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Second Edition. Cambridge, MA: MIT Press, Nov. 22, 2002. ISBN: 978-0-262-01197-6. DOI: 10.7551/mitpress/3413.001.0001.

[Aro98] Sanjeev Arora. "Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems". In: *Journal of the ACM* 45.5 (Sept. 1, 1998), pp. 753–782. DOI: 10.1145/290179.290180.

[Aru+19] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (Oct. 2019), pp. 505–510. DOI: 10.1038/s41586-019-1666-5.

[AS68] M. F. Atiyah and I. M. Singer. "The Index of Elliptic Operators: I". In: *Annals of Mathematics* 87.3 (1968), pp. 484–530. DOI: 10.2307/1970715.

[AT03] Dorit Aharonov and Amnon Ta-Shma. "Adiabatic Quantum State Generation and Statistical Zero Knowledge". In: *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. STOC '03. New York, NY, USA: Association for Computing Machinery, June 9, 2003, pp. 20–29. ISBN: 978-1-58113-674-6. DOI: 10.1145/780542.780546.

[Bao+23] Yicheng Bao, Scarlett S. Yu, Loïc Anderegg, Eunmi Chae, Wolfgang Ketterle, Kang-Kuen Ni, and John M. Doyle. "Dipolar spin-exchange and entanglement between molecules in an optical tweezer array". In: *Science* 382.6675 (Dec. 7, 2023), pp. 1138–1143. DOI: 10.1126/science.adf8999.

[Bar+95] Adriano Barenco et al. "Elementary gates for quantum computation". In: *Physical Review A* 52.5 (Nov. 1, 1995), pp. 3457–3467. DOI: 10.1103/PhysRevA.52.3457.

[BB18] Ravneet Singh Bhandari and Ajay Bansal. "Impact of Search Engine Optimization as a Marketing Tool". In: *Jindal Journal of Business Research* 7.1 (June 1, 2018), pp. 23–36. DOI: 10.1177/2278682117754016.

[Ben80] Paul Benioff. "The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines". In: *Journal of Statistical Physics* 22.5 (May 1, 1980), pp. 563–591. DOI: 10.1007/BF01011339.

[Bha21] Ananyo Bhattacharya. *The Man from the Future: The Visionary Life of John von Neumann*. UK: Penguin Books, Oct. 7, 2021. ISBN: 978-0-241-39886-9.

[Bic+20] Benjamin Bichsel, Maximilian Baader, Timon Gehr, and Martin Vechev. "Silq: A High-Level Quantum Language with Safe Uncomputation and Intuitive Semantics". In: *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI 2020. New York, NY, USA: Association for Computing Machinery, June 11, 2020, pp. 286–300. DOI: 10.1145/3385412.3386007.

[Bin+23] Lennart Binkowski, Gereon Koßmann, Tobias J. Osborne, René Schwonnek, and Timo Ziegler. "From barren plateaus through fertile valleys: Conic extensions of parameterised quantum circuits". In: arXiv:2310.04255 (Oct. 6, 2023). DOI: 10.48550/arXiv.2310.04255.

[Bin22] Lennart Binkowski. "Constraint Graph Model Analysis of the Quantum Alternating Operator Ansatz". Master Thesis. Hanover, Germany: Leibniz University Hanover, Aug. 26, 2022.

[Ble+23] Kostas Blekos, Dean Brand, Andrea Ceschini, Chiao-Hui Chou, Rui-Hao Li, Komal Pandya, and Alessandro Summer. "A Review on Quantum Approximate Optimization Algorithm and its Variants". In: arXiv:2306.09198 (June 26, 2023). DOI: 10.48550/arXiv.2306.09198.

[BM75] Donald W. Barnes and John M. Mack. *An Algebraic Introduction to Mathematical Logic*. Graduate Texts in Mathematics 22. New York, NY: Springer, 1975. ISBN: 978-1-4757-4489-7. DOI: 10.1007/978-1-4757-4489-7.

[BMB01] Christopher A. Bailey, Timothy W. McLain, and Randal W. Beard. "Fuel-Saving Strategies for Dual Spacecraft Interferometry Missions". In: *The Journal of the Astronautical Sciences* 49.3 (Sept. 1, 2001), pp. 469–488. DOI: 10.1007/BF03546233.

[Böc14] Hanno Böck. "Verschlüsselung: Sicher trotz Quantencomputern". In: *Die Zeit* (Oct. 5, 2014). Böck2014. URL: https://www.zeit.de/digital/datenschutz/2014-10/post-quanten-kryptographie (visited on 02/21/2024).

[Bor12] Walter Borchardt-Ott. *Crystallography: An Introduction*. Springer, 2012. ISBN: 978-3-642-16452-1. DOI: 10.1007/978-3-642-16452-1.

[BR22] Jack S. Baker and Santosh Kumar Radha. "Wasserstein Solution Quality and the Quantum Approximate Optimization Algorithm: A Portfolio Optimization Case Study". In: arXiv:2202.06782 (Feb. 14, 2022). DOI: 10.48550/arXiv.2202.06782.

[BS89] Robert G Bland and David F Shallcross. "Large travelling salesman problems arising from experiments in X-ray crystallography: A preliminary report on computation". In: *Operations Research Letters* 8.3 (June 1, 1989), pp. 125–128. DOI: 10.1016/0167-6377(89)90037-0.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge, UK: Cambridge University Press, Mar. 8, 2004. ISBN: 978-0-521-83378-3. URL: https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf (visited on 07/04/2023).

[Cas23] Davide Castelvecchi. "IBM releases first-ever 1,000-qubit quantum chip". In: *Nature* 624.7991 (Dec. 4, 2023), pp. 238–238. DOI: 10.1038/d41586-023-03854-1.

[CBC21] Laura Clinton, Johannes Bausch, and Toby Cubitt. "Hamiltonian simulation algorithms for near-term quantum hardware". In: *Nature Communications* 12.1 (Aug. 17, 2021), p. 4989. DOI: 10.1038/s41467-021-25196-0.

[Cer+21] M. Cerezo et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (Sept. 2021), pp. 625–644. DOI: 10.1038/s42254-021-00348-9.

[CFG02] Andrew M. Childs, Edward Farhi, and Sam Gutmann. "An Example of the Difference Between Quantum and Classical Random Walks". In: *Quantum Information Processing* 1.1 (Apr. 1, 2002), pp. 35–43. DOI: 10.1023/A:1019609420309.

[CG07] Nauro F. Campos and Francesco Giovannoni. "Lobbying, corruption and political influence". In: *Public Choice* 131.1 (Apr. 1, 2007), pp. 1–21. DOI: 10.1007/s11127-006-9102-4.

[Cha+22] Shouvanik Chakrabarti, Pierre Minssen, Romina Yalovetzky, and Marco Pistoia. "Universal Quantum Speedup for Branch-and-Bound, Branch-and-Cut, and Tree-Search Algorithms". In: arXiv:2210.03210 (Oct. 6, 2022). DOI: 10.48550/arXiv.2210.03210.

[CK19] Jaeho Choi and Joongheon Kim. "A Tutorial on Quantum Approximate Optimization Algorithm (QAOA): Fundamentals and Applications". In: 2019 International Conference on Information and Communication Technology Convergence. Jeju, Korea, Oct. 2019, pp. 138–142. DOI: 10.1109/ICTC46691.2019.8939749.

[Cle+98] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. "Quantum algorithms revisited". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (Jan. 8, 1998), pp. 339–354. DOI: 10.1098/rspa.1998.0164.

[CLM12] Nathalie Caspard, Bruno Leclerc, and Bernard Monjardet. *Finite Ordered Sets: Concepts, Results and Uses*. Encyclopedia of Mathematics and its Applications 144. Cambridge University Press, Jan. 26, 2012. ISBN: 978-1-107-01369-8.

[Con+13] Miguel Constantino, Xenia Klimentova, Ana Viana, and Abdur Rais. "New insights on integer-programming models for the kidney exchange problem". In: *European Journal of Operational Research* 231.1 (Nov. 16, 2013), pp. 57–68. DOI: 10.1016/j.ejor.2013.05.025.

[Coo71] Stephen A. Cook. "The Complexity of Theorem-Proving Procedures". In: *Proceedings of the third annual ACM symposium on Theory of computing*. STOC '71. New York, NY, USA: Association for Computing Machinery, May 3, 1971, pp. 151–158. ISBN: 978-1-4503-7464-4. DOI: 10.1145/800157.805047.

[Cor+22] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. 4th edition. Cambridge, Massachusett: The MIT Press, Apr. 5, 2022. ISBN: 978-0-262-04630-5.

[Cra50] Gabriel Cramer. *Introduction à l'Analyse des Lignes Courbes Algébriques*. Genève: Chez les Frères Cramer et C. Philibert, 1750.

[Dam+22] Wim van Dam, Karim Eldefrawy, Nicholas Genise, and Natalie Parham. "Quantum Optimization Heuristics with an Application to Knapsack Problems". In: arXiv:2108.08805 (Feb. 3, 2022). DOI: 10.48550/arXiv.2108.08805.

[Deu85] David Deutsch. "Quantum theory, the Church–Turing principle and the universal quantum computer". In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (July 8, 1985), pp. 97–117. DOI: 10.1098/rspa.1985.0070.

[Die17] Reinhard Diestel. *Graph Theory*. Vol. 173. Graduate Texts in Mathematics. Berlin, Heidelberg: Springer, 2017. ISBN: 978-3-662-53622-3. DOI: 10.1007/978-3-662-53622-3.

[Dom+23] Federico Dominguez, Josua Unger, Matthias Traube, Barry Mant, Christian Ertler, and Wolfgang Lechner. "Encoding-independent optimization problem formulation for quantum computing". In: *Frontiers in Quantum Science and Technology* 2 (Sept. 7, 2023). DOI: 10.3389/frqst.2023.1229471.

[Dow15] Allen B. Downey. *Think Python: How to Think Like a Computer Scientist*. 2nd ed. O'Reilly Media, Dec. 2, 2015. URL: https://greenteapress.com/wp/think-python-2e/.

[Far+00] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. "Quantum Computation by Adiabatic Evolution". In: arXiv:quant-ph/0001106 (Jan. 28, 2000). DOI: 10.48550/arXiv.quant-ph/0001106.

[Fey82] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6 (June 1982), pp. 467–488. DOI: 10.1007/BF02650179.

[FGG14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A Quantum Approximate Optimization Algorithm". In: arXiv:1411.4028 (Nov. 14, 2014). DOI: 10.48550/arXiv.1411.4028.

[For09] Lance Fortnow. "The status of the P versus NP problem". In: *Communications of the ACM* 52.9 (Sept. 1, 2009), pp. 78–86. DOI: 10.1145/1562164.1562186.

[GE21] Craig Gidney and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". In: *Quantum* 5 (Apr. 15, 2021). DOI: 10.22331/q-2021-04-15-433.

[GH19] Pierre Dupuy de la Grand'rive and Jean-Francois Hullo. "Knapsack Problem variants of QAOA for battery revenue optimisation". In: arXiv:1908.02210 (Aug. 15, 2019). DOI: 10.48550/arXiv.1908.02210.

[GO21] Andreas Glas and Klaus Ott. "Maskenaffäre in Bayern: 48 Millionen Euro Provision für Andrea Tandler". In: *Sueddeutsche* (Dec. 16, 2021). URL: https://www.sueddeutsche.de/bayern/bayern-lobbyregister-politik-maskenaffaere-andrea-tandler-1.5489722.

[Got97] Daniel Gottesman. "Stabilizer Codes and Quantum Error Correction". Doctoral Thesis. Pasadena, California: California Institute of Technology, May 28, 1997. DOI: 10.48550/arXiv.quant-ph/9705052.

[Gre23] Veronique Greenwood. "Andreas Wagner Pursues the Secrets to Evolutionary Success". In: *Quanta Magazine* (Aug. 15, 2023). In collab. with Andreas Wagner. URL: https://www.quantamagazine.org/andreas-wagner-pursues-the-secrets-to-evolutionary-success-20230815/ (visited on 01/04/2024).

[Gro96] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing*. STOC '96. New York, NY, USA: Association for Computing Machinery, July 1, 1996, pp. 212–219. ISBN: 978-0-89791-785-8. DOI: 10.1145/237814.237866.

[GSŞ21] Shouzhen Gu, Rolando D. Somma, and Burak Şahinoğlu. "Fast-forwarding quantum evolution". In: *Quantum* 5 (Nov. 15, 2021). DOI: 10.22331/q-2021-11-15-577.

[GW16] George Grätzer and Friedrich Wehrung, eds. *Lattice Theory: Special Topics and Applications: Volume 2*. Birkhäuser Cham, 2016. ISBN: 978-3-319-44236-5. DOI: 10.1007/978-3-319-44236-5.

[GY15] Daniel J. Garcia and Fengqi You. "Supply chain design and optimization: Challenges and opportunities". In: *Computers & Chemical Engineering*. Special Issue: Selected papers from the 8th International Symposium on the Foundations of Computer-Aided Process Design, July 13-17, 2014, Cle Elum, Washington, USA 81 (Oct. 4, 2015), pp. 153–170. DOI: 10.1016/j.compchemeng.2015.03.015.

[Had+19] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. "From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz". In: *Algorithms* 12.2 (Feb. 2019). DOI: 10.3390/a12020034.

[HH21] David Harvey and Joris van Der Hoeven. "Integer multiplication in time O(n log n)". In: *Annals of Mathematics* 193.2 (Mar. 1, 2021), pp. 563–617. DOI: 10.4007/annals.2021.193.2.4.

[HHM08] John Harris, Jeffry L. Hirst, and Michael Mossinghoff. *Combinatorics and Graph Theory*. Undergraduate Texts in Mathematics. New York, NY: Springer, 2008. ISBN: 978-0-387-79711-3. DOI: 10.1007/978-0-387-79711-3.

[Hil86] Howard Hiller. "Crystallography and Cohomology of Groups". In: *The American Mathematical Monthly* 93.10 (Dec. 1986), pp. 765–779. DOI: 10.1080/00029890.1986.11971943.

[Hit+03] C. Hitte et al. "Comparison of MultiMap and TSP/CONCORDE for Constructing Radiation Hybrid Maps". In: *Journal of Heredity* 94.1 (Jan. 1, 2003), pp. 9–13. DOI: 10.1093/jhered/esg012.

[HLC23] Connor M. Holland, Yukai Lu, and Lawrence W. Cheuk. "On-demand entanglement of molecules in a reconfigurable optical tweezer array". In: *Science* 382.6675 (Dec. 7, 2023), pp. 1143–1147. DOI: 10.1126/science.adf4272.

[Hro14] Juraj Hromkovič. *Theoretische Informatik: Formale Sprachen, Berechenbarkeit, Komplexitätstheorie, Algorithmik, Kommunikation und Kryptographie*. Wiesbaden: Springer Fachmedien, 2014. ISBN: 978-3-658-06433-4. DOI: 10.1007/978-3-658-06433-4.

[Hug+21] William J. Huggins et al. "Virtual Distillation for Quantum Error Mitigation". In: *Physical Review X* 11.4 (Nov. 19, 2021), p. 041036. DOI: 10.1103/PhysRevX.11.041036.

[Iva00] Lars Ivansson. "Computational Aspects of Radiation Hybrid Mapping". Doctoral Thesis. Stockholm: Royal Institute of Technology, Nov. 2000.

[Kar72] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. The IBM Research Symposia Series. Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9.

[Kay23] Alastair Kay. "Tutorial on the Quantikz Package". In: arXiv:1809.03842 (June 19, 2023). DOI: 10.48550/arXiv.1809.03842.

[Kem05] Julia Kempe. "Discrete Quantum Walks Hit Exponentially Faster". In: *Probability Theory and Related Fields* 133.2 (Feb. 10, 2005), pp. 215–235. DOI: 10.1007/s00440-004-0423-2.

[KL70] B. W. Kernighan and S. Lin. "An Efficient Heuristic Procedure for Partitioning Graphs". In: *The Bell System Technical Journal* 49.2 (Feb. 1970), pp. 291–307. DOI: 10.1002/j.1538-7305.1970.tb01770.x.

[Knu97] Donald Knuth. *The Art of Computer Programming: Volume 1: Fundamental Algorithms*. 3rd ed. Boston, US: Addison Wesley, July 7, 1997. ISBN: 978-0-201-89683-1.

[Knu98] Donald E. Knuth. *The Art of Computer Programming: Volume 3: Sorting and Searching*. 2nd ed. Boston, US: Addison Wesley, May 1, 1998. ISBN: 978-0-201-89685-5.

[Koc21] Bálint Koczor. "Exponential Error Suppression for Near-Term Quantum Devices". In: *Physical Review X* 11.3 (Sept. 15, 2021), p. 031057. DOI: 10.1103/PhysRevX.11.031057.

[Koß+23] Gereon Koßmann, Lennart Binkowski, Christian Tutschku, and René Schwonnek. "Open-Shop Scheduling With Hard Constraints". In: arXiv:2211.05822 (Jan. 23, 2023). DOI: 10.48550/arXiv.2211.05822.

[Koß22] Gereon Koßmann. "A Quantum Algorithm for Job Shop Problems with Group Theory". Master Thesis. Hanover, Germany: Leibniz University Hanover, 2022.

[Kra23] Patrick Krauss. *Künstliche Intelligenz und Hirnforschung: Neuronale Netze, Deep Learning und die Zukunft der Kognition*. Springer, 2023. ISBN: 978-3-662-67179-5. DOI: 10.1007/978-3-662-67179-5.

[KS98] Hans Kurzweil and Bernd Stellmacher. *Theorie der endlichen Gruppen*. Springer-Lehrbuch. Springer, 1998. ISBN: 978-3-642-58816-7. DOI: 10.1007/978-3-642-58816-7.

[LaP21] Ray LaPierre. *Introduction to Quantum Computing*. The Materials Research Society Series. Cham: Springer International Publishing, 2021. ISBN: 978-3-030-69318-3. DOI: 10.1007/978-3-030-69318-3.

[LC17] Guang Hao Low and Isaac L. Chuang. "Optimal Hamiltonian Simulation by Quantum Signal Processing". In: *Physical Review Letters* 118.1 (Jan. 5, 2017), p. 010501. DOI: 10.1103/PhysRevLett.118.010501.

[LC19] Guang Hao Low and Isaac L. Chuang. "Hamiltonian Simulation by Qubitization". In: *Quantum* 3 (July 12, 2019), p. 163. DOI: 10.22331/q-2019-07-12-163.

[Lei79] Frank Thomson Leighton. "A Graph Coloring Algorithm for Large Scheduling Problems". In: *Journal of Research of the National Bureau of Standards* 84.6 (1979), pp. 489–506. DOI: 10.6028/jres.084.024.

[Lev73] Leonid Anatolevich Levin. "Universal Sequential Search Problems". In: *Problemy Peredachi Informatsii* 9.3 (1973), pp. 115–116.

[LK73] S. Lin and B. W. Kernighan. "An Effective Heuristic Algorithm for the Traveling-Salesman Problem". In: *Operations Research* 21.2 (Apr. 1973), pp. 498–516. DOI: 10.1287/opre.21.2.498.

[Llo96] Seth Lloyd. "Universal Quantum Simulators". In: *Science* 273.5278 (Aug. 23, 1996), pp. 1073–1078. DOI: 10.1126/science.273.5278.1073.

[LR13] Richard J. Lipton and Kenneth W. Regan. *People, Problems, and Proofs: Essays from Gödel's Lost Letter: 2010*. Berlin, Heidelberg: Springer, 2013. ISBN: 978-3-642-41422-0. DOI: 10.1007/978-3-642-41422-0.

[Luc14] Andrew Lucas. "Ising formulations of many NP problems". In: *Frontiers in Physics* 2 (2014). URL: https://www.frontiersin.org/articles/10.3389/fphy.2014.00005 (visited on 01/12/2023).

[Mag86] Vangelis F. Magirou. "The Efficient Drilling of Printed Circuit Boards". In: *Interfaces* 16.4 (1986), pp. 13–23. URL: https://www.jstor.org/stable/25060844.

[Mar+21] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. "Grand Unification of Quantum Algorithms". In: *PRX Quantum* 2.4 (Dec. 3, 2021). DOI: 10.1103/PRXQuantum.2.040203.

[MN19a] Andy Matuschak and Michael A. Nielsen. *How does the quantum search algorithm work?* San Francisco. 2019. URL: https://quantum.country/search.

[MN19b] Andy Matuschak and Michael A. Nielsen. *Quantum Computing for the Very Curious*. San Francisco. 2019. URL: https://quantum.country/qcvc.

[MN22] Joaquim R. R. A. Martins and Andrew Ning. *Engineering Design Optimization*. Cambridge, UK: Cambridge University Press, Jan. 2022. ISBN: 978-1-108-83341-7. DOI: 10.1017/9781108980647. URL: https://mdobook.github.io (visited on 02/09/2024).

[Mon16a] Ashley Montanaro. "Quantum algorithms: an overview". In: *npj Quantum Information* 2.1 (Jan. 12, 2016). DOI: 10.1038/npjqi.2015.23.

[Mon16b] Ashley Montanaro. "Quantum walk speedup of backtracking algorithms". In: arXiv:1509.02374 (Jan. 4, 2016). DOI: 10.48550/arXiv.1509.02374.

[Mon20] Ashley Montanaro. "Quantum speedup of branch-and-bound algorithms". In: *Physical Review Research* 2.1 (Jan. 16, 2020), p. 013056. DOI: 10.1103/PhysRevResearch.2.013056.

[Mon23] Zach Montague. "The Race to Save Our Secrets From the Computers of the Future". In: *The New York Times* (Oct. 22, 2023). URL: https://www.nytimes.com/2023/10/22/us/politics/quantum-computing-encryption.html (visited on 02/21/2024).

[Mor24] J. Edward Moreno. "FirstEnergy Ex-C.E.O. and 2 Others Are Indicted in Bribery Scandal". In: *The New York Times* (Feb. 12, 2024). URL: https://www.nytimes.com/2024/02/12/business/firstenergy-indictment-ohio-bribery.html.

[MTZ60] C. E. Miller, A. W. Tucker, and R. A. Zemlin. "Integer Programming Formulation of Traveling Salesman Problems". In: *Journal of the Association for Computing Machinery* 7.4 (Oct. 1, 1960), pp. 326–329. DOI: 10.1145/321043.321046.

[MW03] Samuel A. Mulder and Donald C. Wunsch. "Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks". In: *Neural Networks*. Advances in Neural Networks Research: IJCNN '03 16.5 (June 1, 2003), pp. 827–832. DOI: 10.1016/S0893-6080(03)00130-8.

[Nat00] Melvyn B. Nathanson. *Elementary Methods in Number Theory*. Graduate Texts in Mathematics 195. New York, NY: Springer, 2000. ISBN: 978-0-387-22738-2. DOI: 10.1007/b98870.

[Nau23] John Naughton. "Encryption services are sending the right message to the quantum codebreakers". In: *The Observer* (Oct. 7, 2023). URL: https://www.theguardian.com/commentisfree/2023/oct/07/encryption-services-quantum-computers-cryptography-rsa-signal-whatsapp (visited on 02/21/2024).

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. ISBN: 978-1-107-00217-3.

[Nie15] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/.

[NR16] John Forbes Nash and Michael Th. Rassias, eds. *Open Problems in Mathematics*. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-32162-2. DOI: 10.1007/978-3-319-32162-2.

[NV00] Ashwin Nayak and Ashvin Vishwanath. "Quantum Walk on the Line". In: arXiv:quant-ph/0010117 (Oct. 31, 2000). DOI: 10.48550/arXiv.quant-ph/0010117.

[Pap+23] Andrei Papkou, Lucia Garcia-Pastor, José Antonio Escudero, and Andreas Wagner. "A rugged yet easily navigable fitness landscape". In: *Science* 382.6673 (Nov. 24, 2023), eadh3860. DOI: 10.1126/science.adh3860.

[Pat+24] Yash J. Patel, Sofiene Jerbi, Thomas Bäck, and Vedran Dunjko. "Reinforcement learning assisted recursive QAOA". In: *EPJ Quantum Technology* 11.1 (Dec. 2024), pp. 1–23. DOI: 10.1140/epjqt/s40507-023-00214-w.

[Per+14] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5.1 (July 23, 2014), p. 4213. DOI: 10.1038/ncomms5213.

[Pre18] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2.79 (Aug. 6, 2018). DOI: 10.22331/q-2018-08-06-79.

[Pun22] Abraham P. Punnen, ed. *The Quadratic Unconstrained Binary Optimization Problem: Theory, Algorithms, and Applications*. Cham: Springer International Publishing, 2022. ISBN: 978-3-031-04520-2. DOI: 10.1007/978-3-031-04520-2.

[Rei91] Gerhard Reinelt. "TSPLIB—A Traveling Salesman Problem Library". In: *ORSA Journal on Computing* 3.4 (1991), pp. 376–384.

[Rob96] Derek J. S. Robinson. *A Course in the Theory of Groups*. Vol. 80. Graduate Texts in Mathematics. New York, NY: Springer, 1996. ISBN: 978-1-4419-8594-1. DOI: 10.1007/978-1-4419-8594-1.

[Sag01] Bruce E. Sagan. *The Symmetric Group*. Vol. 203. Graduate Texts in Mathematics. New York, NY: Springer, 2001. ISBN: 978-1-4757-6804-6. DOI: 10.1007/978-1-4757-6804-6.

[Sam17] Benjamin Sambale. *Endliche Permutationsgruppen*. Springer Spektrum Wiesbaden, 2017. ISBN: 978-3-658-17597-9. DOI: 10.1007/978-3-658-17597-9.

[Sat20] Or Sattath. "On the insecurity of quantum Bitcoin mining". In: *International Journal of Information Security* 19.3 (June 1, 2020), pp. 291–302. DOI: 10.1007/s10207-020-00493-9.

[Sch16] Bernd Schröder. *Ordered Sets*. Birkhäuser Cham, 2016. ISBN: 978-3-319-29788-0. DOI: 10.1007/978-3-319-29788-0.

[Sch19] Wolfgang Scherer. *Mathematics of Quantum Computing: An Introduction*. Cham: Springer International Publishing, 2019. ISBN: 978-3-030-12358-1. DOI: 10.1007/978-3-030-12358-1.

[SD08] S.N. Sivanandam and S.N. Deepa. *Introduction to Genetic Algorithms*. Springer, 2008. ISBN: 978-3-540-73189-4. DOI: 10.1007/978-3-540-73190-0.

[Sho94] Peter W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. Nov. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[SMY02] Ruhul Sarker, Masoud Mohammadian, and Xin Yao, eds. *Evolutionary Optimization*. International Series in Operations Research & Management Science 48. Springer US, 2002. ISBN: 978-0-306-48041-6. DOI: 10.1007/b101816.

[SN20] J. J. Sakurai and Jim Napolitano. *Modern Quantum Mechanics*. Third Edition. Cambridge University Press, Oct. 31, 2020. ISBN: 978-1-108-47322-4.

[Spa92] J.C. Spall. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation". In: *IEEE Transactions on Automatic Control* 37.3 (Mar. 1992), pp. 332–341. DOI: 10.1109/9.119632.

[Sti10] John Stillwell. *Mathematics and Its History*. Undergraduate Texts in Mathematics. New York, NY: Springer, 2010. ISBN: 978-1-4419-6053-5. DOI: 10.1007/978-1-4419-6053-5.

[Str13] Norbert Straumann. *General Relativity*. Graduate Texts in Physics. Springer Dordrecht, 2013. ISBN: 978-94-007-5410-2. DOI: 10.1007/978-94-007-5410-2.

[Str15] Norbert Straumann. *Theoretische Mechanik: Ein Grundkurs über klassische Mechanik endlich vieler Freiheitsgrade*. Springer-Lehrbuch. Springer, 2015. ISBN: 978-3-662-43691-2. DOI: 10.1007/978-3-662-43691-2.

[Vaj11] Vincent Vajnovszki. "A new Euler–Mahonian constructive bijection". In: *Discrete Applied Mathematics* 159.14 (Aug. 28, 2011), pp. 1453–1459. DOI: 10.1016/j.dam.2011.05.012.

[Wal14] Stefan Waldmann. *Topology: An Introduction*. Cham: Springer International Publishing, 2014. ISBN: 978-3-319-09680-3. DOI: 10.1007/978-3-319-09680-3.

[Wal21] Stefan Waldmann. *Lineare Algebra 1: Grundlagen für Studierende der Mathematik und Physik*. Berlin, Heidelberg: Springer, 2021. ISBN: 978-3-662-63263-5. DOI: 10.1007/978-3-662-63263-5.

[Wal22] Stefan Waldmann. *Lineare Algebra 2: Anwendungen und Konzepte für Studierende der Mathematik und Physik*. Berlin, Heidelberg: Springer, 2022. ISBN: 978-3-662-63639-8. DOI: 10.1007/978-3-662-63639-8.

[Wil+21] Max Wilson, Rachel Stromswold, Filip Wudarski, Stuart Hadfield, Norm M. Tubman, and Eleanor G. Rieffel. "Optimizing quantum heuristics with meta-learning". In: *Quantum Machine Intelligence* 3.1 (Apr. 13, 2021), p. 13. DOI: 10.1007/s42484-020-00022-w.

[Wil+23] Sören Wilkening, Andreea-Iulia Lefterovici, Lennart Binkowski, Michael Perk, Sándor Fekete, and Tobias J. Osborne. "A quantum algorithm for the solution of the 0-1 Knapsack problem". In: arXiv:2310.06623 (Oct. 10, 2023). DOI: 10.48550/arXiv.2310.06623.

[Wil95] Andrew Wiles. "Modular Elliptic Curves and Fermat's Last Theorem". In: *Annals of Mathematics* 141.3 (1995), pp. 443–551. DOI: 10.2307/2118559.

[Won22] Hiu Yung Wong. *Introduction to Quantum Computing: From a Layperson to a Programmer in 30 Steps*. Cham: Springer International Publishing, 2022. ISBN: 978-3-030-98339-0. DOI: 10.1007/978-3-030-98339-0.

[WZ22] Yanyan Wang and Xiaoxin Zhu. "A Multi-Regional Collaborative Optimization Model of Emergency Medical Materials for Responding to COVID-19". In: *Processes* 10.8 (Aug. 2022), p. 1488. DOI: 10.3390/pr10081488.

[Yac15] Susan Webb Yackee. "Invisible (and Visible) Lobbying: The Case of State Regulatory Policymaking". In: *State Politics & Policy Quarterly* 15.3 (2015), pp. 322–344. URL: https://www.jstor.org/stable/24643837.

[Yan+13] Qingxuan Yang, John Ellis, Khalegh Mamakani, and Frank Ruskey. "Parallel and sequential in-place permuting and perfect shuffling using involutions". In: arXiv: 1204.1958 (Jan. 14, 2013). DOI: 10.48550/arXiv.1204.1958.

[YG10] Xinjie Yu and Mitsuo Gen. *Introduction to Evolutionary Algorithms*. Decision Engineering. Springer, 2010. ISBN: 978-1-84996-129-5. DOI: 10.1007/978-1-84996-129-5.

[YY15] Nengkun Yu and Mingsheng Ying. "Optimal simulation of Deutsch gates and the Fredkin gate". In: *Physical Review A* 91.3 (Mar. 6, 2015), p. 032302. DOI: 10.1103/PhysRevA.91.032302.

[ZCL15] Yu-Jun Zheng, Sheng-Yong Chen, and Hai-Feng Ling. "Evolutionary optimization for disaster relief operations: A survey". In: *Applied Soft Computing* 27 (Feb. 2015), pp. 553–566. DOI: 10.1016/j.asoc.2014.09.041.

[Zyg18] Bernard Zygelman. *A First Introduction to Quantum Computing and Information*. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-91629-3. DOI: 10.1007/978-3-319-91629-3.

Acknowledgements

Within academia, writing a Master's thesis is not very big a deal. Nevertheless, I'd like to take this opportunity to express my gratitude, as I believe myself to be in the fortunate position of being surrounded by people I can be very thankful for.

First, I'd like to thank Tobias Osborne for allowing me to write a thesis in the field of quantum computation. I have learned a lot in the process, and I hope to make good use of this knowledge in the future.

Next, I'd like to thank the people whom I had the pleasure of working with during my time in the group. This includes but is not limited to Lennart Binkowski, Tim Heine, Debora Ramaciotti, Antonio Rotundo, Benjamin Sambale, Sören Wilkening, and Timo Ziegler. In particular, I'd like to thank Lennart Binkowski for collaboration (and supervision) that was fun both intellectually and interpersonally, Benjamin Sambale for insightful comments and important contributions, and Timo Ziegler for valuable advice.

Further, for finding typographical errors, and for encouraging me to split up at least a few of the overly long sentences that I love to write, I'd like to thank Celine von Minden, Maja Scharnagl, and my dad.

Last, and most importantly, I want to express my gratitude to my girlfriend and my family, whose continued support, patience, and love I treasure every day. Specifically, I would like to dedicate this thesis to my grandmother, whose loving and caring ways I can, sadly, from now on only cherish in memory.