GPU-accelerated simulation of guided quantum walks

by

Sebastian Schulz

Master's thesis in Physics

submitted to the

Faculty of Mathematics, Computer Science and Natural Science
at the RWTH Aachen

in October 2022

prepared at the

Jülich Supercomputing Centre
Forschungszentrum Jülich

under supervision of

Prof. Dr. Kristel Michielsen
Prof. Dr. Markus Müller

Contents

List of Figures

Nomenclature

List of abbreviations

`2SAT_N_I`	Label of the 2-SAT problem instance I featuring N qubits.
AQA	Approximate quantum annealing.
AQC	Adiabatic quantum computing.
Cluster space	Closed 2^{N_C} dimensional subspace of the state space in which a gate-cluster is operating.
Constant memory	Read-only global memory (8 KB cached into L1-cache; ≈ 10 cycles latency).
CUDA block	Collection of multiple warps mapped onto one SM.
CUDA thread	Smallest unit of computation on a GPU that is mapped onto one streaming processor.
CUDA warp	Fixed collection of 32 threads that operate in a SIMT fashion.
CUDA	Compute Unified Device Architecture.
Device	Environment of the GPU and its memory (global memory).
`EC_N_I`	Label of the exact cover problem instance I featuring N qubits.
Global memory	Main GPU memory (40 GB; $400 - 800$ cycles latency).
Global qubit	A qubit that is distributed among the memory spaces of two GPUs.
GQW	Guided quantum walk.
HGQW	Heuristic model which is used to apply the GQW to exact cover instances. The model is tuned in an outter classical loop using a set of optimization parameters $\boldsymbol{\lambda}$.
HGQW-A	Heuristic model which is used to apply the GQW to 2-SAT instances. The model is tuned in an outter classical loop using a set of optimization parameters $\boldsymbol{\lambda}$.

Host	Environment of the CPU and its memory (RAM).
JUQCS-G	GPU-accelerated version of the Jülich universal quantum computer simulator.
Local qubit	A qubit that resides in the memory space of one GPU.
MPI	Message Passing Interface.
NISQ device	Noisy Intermediate Scale Quantum device.
NP	Nondeterministic polynomial. This complexity class includes all problems their solution can be verified with polynomial ressources.
NP-complete	Nondeterministic polynomial complete. This complexity class includes all problems of NP that can be used to simulate all other problems in NP with polyonomial ressources.
QAOA	Quantum approximate optimization algorithm.
QFT	Quantum Fourier transformation.
QUBO	Quadratic unconstrained binary optimization problem.
QW	Quantum walk.
SEQCS	Sebastian quantum circuit simulator. A high-performance GPU-accelerated classical simulator of large scale quantum circuits.
Shared memory	Block-private self-managed L1-cache (48 KB per block; ≈ 10 cycles latency).
SM	Streaming Multiprocessor on a GPU.
State space	2^N dimensional complex Hilbert space of N qubits.
Target space	Closed subspace of the state space in which a quantum gate is operating.

List of symbols

$\&$, $\sim$, $\wedge$, $\vert$, $<<_i$, $>>_i$	Bitwise operators: AND, NOT, XOR, OR, left and right shift by i bits.
$A_r = \frac{\langle\Psi\vert\hat{H}_C\vert\Psi\rangle}{\max_{Z_i \epsilon \mathbf{Z}} C(Z_i)}$	Approximation ratio, corresponding to the normalized energy expectation value of the cost Hamiltonian $\hat{H}_C$.
$B = [b_{15} \ \dots \ b_0]_2$	15 bit integer denoting the CUDA blockID, hence the index of a block within a CUDA grid.
$\boldsymbol{\beta} = \{\beta_0, \dots, \beta_{p-1}\}$	Set of spread coefficients that control the time evolution under the mixing Hamiltonian $\hat{H}_M$.

$\lvert\boldsymbol{\beta},\boldsymbol{\gamma}\rangle = \prod_{k=1}^{p} \hat{U}_M(\beta_k)\,\hat{U}_C(\gamma_k)\,\lvert+\rangle$	Variational quantum state after p iterations of phase seperation $\hat{U}_C$ and mixing $\hat{U}_M$ time evolutions.
$\beta(i,\boldsymbol{\lambda})$	Sampling function of the spread coefficients $\boldsymbol{\beta}$ used by the HGQW model and the HGQW-A model. The function uses a constant distribution for $\boldsymbol{\beta}$ tuned by the optimization parameters $\boldsymbol{\lambda}$.
$C = \left[c_{N_C-9}\ \ldots\ c_0\right]_2$	Integer that counts through the amplitudes assigned to a thread. Note that C is traversed using the inverse-gray code.
C_0^{QUBO}, C_0^{Ising}	Constant shifts of the cost function in the QUBO and Ising formulation, respectively.
$\mathrm{C}(Z)$	Cost function of a combinatorial optimization problem, assigning a cost value to a binary string Z.
$\Delta\alpha^i_{B,A} = \alpha_B - \alpha_A$	Initial phase gradient between the complex phases α_A and α_B of the computational state amplitudes ψ_A and ψ_B, respectively, before the application of $\hat{U}_C$ at iteration i of the GQW.
$\Delta\Phi^i_{B,A} = \Delta\alpha^i_{B,A} + \gamma\cdot\Delta E_{B,A}$	Total phase gradient between the complex phases of the computational state amplitudes ψ_A and ψ_B, after the application of $\hat{U}_C$ at iteration i of the GQW, with respect to the initial phase difference $\Delta\alpha^i_{B,A}$. Note that $\Delta E_{B,A} = E_B - E_A$.
$\Delta(\psi_A,\psi_B)$	Hamming distance between the binary representations of the computational basis states ψ_A and ψ_B.
$\Delta\Theta^i_{B,A}$	Phase gradient between the complex phases of the computational state amplitudes ψ_A and ψ_B, caused by the mixing evolution $\hat{U}_M$ at iteration i of the GQW.
$E_\Psi = \langle\Psi\lvert\hat{H}_C\rvert\Psi\rangle$	Energy expectational value when measureing the quantum state $\lvert\Psi\rangle$ in $\hat{H}_C$.
$F_M(t), F_C(t)$	Annealing schedules, controlling the strength of the mixing Hamiltonian $\hat{H}_M$ and problem Hamiltonian $\hat{H}_C$, respectively.
$\boldsymbol{\gamma} = \{\gamma_0,\ldots,\gamma_{p-1}\}$	Set of phase coefficients that control the time evolution under the problem Hamiltonian $\hat{H}_C$.
$\gamma(i,\boldsymbol{\lambda})$	Sampling function of the phase coefficients $\boldsymbol{\gamma}$ used by the HGQW model and the HGQW-A model. The function uses an exponential distribution tuned by the optimization parameters $\boldsymbol{\lambda}$, with coefficient γ_i being sampled at position $x_i = i/(p-1)$.
$\hat{G}_k$	A quantum gate at position k in the gate queue.

$\hat{H}_C$	Cost/Problem Hamiltonian that encodes the cost function of a combinatorial optimization problem in the energy spectrum of an Ising Hamiltonian.
$\hat{H}_M = \sum_{i=0}^{N-1} \hat{\sigma}_i^x$	Driving Hamiltonian that creates maximal mixing in the computational basis.
I_G	2^{N_G} bit binary string that indexes the amplitudes in the subspace of the global qubits.
I_L	2^{N_L} bit binary string that indexes the amplitudes in the subspace of the local qubits.
I_M	64 (diagonal gate) or 32 (non-diagonal gate) bit binary string that denotes the target qubits of a gate/gate-group.
I_S	Bitstring, termed subspace-index, that indexes the target-spaces in the state-space.
I_T	Bitstring, termed target-index, that indexes the amplitudes in a target-space.
$J = [j_{N-1} \ \dots \ j_0]_2$	N bit integer used to index the 2^N computational basis states.
$K_l(\beta) = \cos^{N-l}(\beta) \sin^l(\beta)$	Amplitude of the interaction order l caused by the mixing evolution $\hat{U}_M(\beta)$.
$\boldsymbol{\lambda} = \{\lambda_0, \dots, \lambda_L\}$	Set of optimization parameters used by the HGQW model and the HGQW-A model to tune the sampling functions $\beta(i, \boldsymbol{\lambda})$ and $\gamma(i, \boldsymbol{\lambda})$.
M	Total number of gates in the circuit.
M_C	Number of gates in a gate-cluster.
M_G	Number of gates in a gate-group.
N	Total number of qubits in the system.
N_C	Number of qubits in a gate-cluster.
N_G	Number of global qubits.
N_L	Number of local qubits.
p	Number of iterations performed by the GQW/QAOA/AQA.
$P_1^{\pm} = \pi \pm \arctan\sqrt{\frac{1}{N-1}}$	Location of the extreme/peak values of $K_1(\beta)$.
$P_{gs} = \lvert\langle Z_{gs} \rvert \Psi\rangle\rvert^2$	Success probability, corresponding to the overlap the state vector $\lvert\Psi\rangle$ with the solution state $\lvert Z_{gs}\rangle$.

$\|\Psi\rangle = \sum_{J=0}^{N-1} \psi_J \|J\rangle$	N qubit state vector with complex amplitudes $\psi_J = r_J e^{-i\alpha_J}$ in the computational basis $\{\|0\rangle, \ldots, \|N-1\rangle\}$. Here, $r_J \in \mathbb{R}$ and $\alpha_J \in [0, 2\pi)$ denote the complex amplitude and complex phase, respectively.
R	Rank of an MPI process. It functions as a unique identifier of each GPU.
$S = [s_{N-1} \ \ldots \ s_0]_2$	N dimensional array denoting the configuration of N Ising spins, with $s_i = +1 \mathrel{\widehat{=}} \text{spin}- \uparrow$ and $s_i = -1 \mathrel{\widehat{=}} \text{spin}- \downarrow$. $S_{opt} = S_{gs}$ reffers to the ground state of an Ising Hamiltonian.
$\hat{\sigma^x}, \hat{\sigma^y}, \hat{\sigma^z}$	Single-qubit Pauli operators.
$\sigma(K)$	$2 \times N$ dimensional permutation matrix that mappes a qubit K to its altered bit position in the state vector. σ accounts for changes in the qubit order in the memory due to global-local qubit exchanges.
$T = [t_{15} \ \ldots \ t_0]_2$	15 bit integer denoting the CUDA threadID, hence the index of a thread within a CUDA block.
τ	Annealing time.
T_{Gates}	Gate execution time of a quantum circuit.
T_{MPI}	MPI execution time of a quantum circuit.
$\hat{U}_C(\gamma) = e^{-i\gamma\hat{H}_C}$	Unitary time evolution unter the problem Hamiltonian $\hat{H}_C$. It is used to create complex phase gradients between the computational state amplitudes.
$\hat{U}_M(\beta) = e^{-i\gamma\hat{H}_M}$	Unitary time evolution unter the mixing Hamiltonian $\hat{H}_M$. It is used to create interactions between the computational state amplitudes. Note that $\hat{U}_M$ is π-periodic.
$Z = [z_{N-1} \ \ldots \ z_0]_2$	N bit binary string denoting an element of the domain of the cost function. $Z_{opt} = Z_{gs}$ reffers to the minimum of $\mathrm{C}(Z)$.

Introduction

Can physics be simulated by a universal (classical) computer?

With this question, Richard Feynman began his famous speech at the First Conference on the Physics of Computation in 1982 [1], outlining the challenges in simulating the quantum mechanical nature of the physical world using only classical models on classical hardware. He pointed out that in a system with R degrees of freedom and N states which each degree of freedom can occupy, the number of possible configurations N^R grows exponentially in R, hence quickly exceeding the memory available on any classical computer today. Given this limitation, he imagined a quantum machine, using quantum computing elements to exactly simulate large quantum systems with computational resources that scale only proportional to the space-time volume of the physical system in question. In doing so, he sparked a decade long run on building large-scale quantum computing devices [2–4], including semiconducting, superconducting and trapped ions architectures, as well as developing quantum computing algorithms that can outperform their classical counterparts. The latter exploit quantum phenomena such as superposition of computational basis states, entanglement, and complex phase interference in order to solve computational problems [5]. Especially the discoveries of Grover's search algorithm [6] and Shor's algorithm for integer factorization [7], proven to achieve quadratic and exponential speed-ups compared to their best classical counterparts, respectively, demonstrated the immense computational power associated with quantum computation. Today, promising applications of quantum computation can be found in the fields of combinatorial optimization problems [8, 9], quantum chemistry [10, 11], machine learning [12–14] and cryptography [15].

In practice, however, significant research is still needed in order to achieve reliable quantum computation and eventually quantum advantage. Here, the latter refers to the ability to implement quantum algorithms for solving problems that are inaccessible to classical computers, due to their high computational cost. This is because today's quantum devices are not *ideal quantum computers* in the sense that their computation is susceptible to environmental influence, causing their results to become faulty [16]. These devices are called *Noisy Intermediate Scale Quantum* (NISQ) devices and offer few imperfect quantum bits ($\mathcal{O}(100)$) with limited coherence times and weak error-correction capabilities. Since both Shor's and Grover's algorithm require millions of qubits with error correction techniques [17], recent research has shifted towards noisy and shallow quantum algorithms, in order to achieve useful quantum computation already within the next decade. Especially hybrid quantum-classical algorithms have shown to be well suited to the constraints imposed by NISQ devices, hence being a promising candidate for reaching quantum advantage soon [18–22]. The idea behind these algorithms is to conduct the classically traceable part of some computation on a clas-

sical device, while the classically difficult part is performed on a quantum computer. Given this framework, the following two aspects of quantum computation will be investigated in this thesis.

The first aspect concerns the classical simulation of quantum computation. Caused by the error-proneness of modern NISQ devices, the simulation of quantum algorithms emerged as a necessary discipline for developing and debugging new quantum applications [23–25]. Among others, the study of algorithmic complexity, the benchmarking of existing quantum computing hardware and the design of quantum circuits that are difficult to characterize analytically represent common use cases. Hence, chapter 1 will investigate novel strategies to increase the efficiency of simulating large-scale quantum systems in the context of supercomputers by developing a new universal quantum circuit simulator, termed SEQCS.

The second aspect of quantum computing that will be studied in this thesis concerns its application in the context of combinatorial optimization problems. The latter includes both scientific and industrial use cases, ranging from logistics, supply chain and manufacturing optimizations [26] to the analysis and benchmarking of classical as well as quantum computing hardware [8, 27, 28]. Inspired by hybrid quantum-classical algorithms, chapter 2 will investigate the trotterized evolution of quantum systems in order to reduce the computational complexity and the circuit depths of common quantum optimization strategies. In doing so, a novel quantum algorithm, termed the guided quantum walk, will be developed and analysed.

Chapter 1

Simulation of ideal quantum computing

This chapter concerns the simulation of large scale quantum systems, featuring more than 30 quantum bits, with the goal of achieving a high-performance evaluation of general quantum circuits. A novel quantum circuit simulator termed *SEQCS* using direct quantum state evolutions will hence be developed in the following sections. Its implementation focusses on the *JUWELS Booster supercomputer* [29] and introduces GPU-accelerators spread across multiple compute nodes into the simulation process, in order to significantly reduce the simulation time compared to traditional CPU-based codes. Moreover, novel optimization strategies regarding the memory management, memory access patterns and the load balance will be proposed. This allows to leverage the hardware restrictions of the GPU, which were found to limit the performance of prior simulators [27]. In doing so, SEQCS sets itself among the fastest quantum circuit simulators, managing 17 to 329 times faster single gate executions and 23 to 70 times faster evaluations of certain real-world quantum algorithms compared to the *Jülich universal quantum computer simulator* (JUQCS-G) [30], whose main development efforts are on large-scale simulations using distributed memory. This significant improvement in the simulation speed enables the investigation of quantum optimization algorithms in chapter 2 within parameter regions that were previously impractical to simulate.

This chapter is structured as follows: section 1.1 will give an introduction into quantum computing and provide the reader with the basic concepts involved in the evaluation of quantum circuits. Following that, section 1.2 concerns the design and implementation of SEQCS. First, the simulation technique will be explained, followed by a discussion of the programming framework and its hardware limitations. Finally, the simulation code will be presented by focussing on the three abstraction levels (distributed memory space, GPU caching and gate computation) separately. Concluding this chapter, section 1.3 will offer various benchmark results comparing the performance of SEQCS to JUQCS-G with respect to individual gate executions and real-world quantum algorithms.

1.1. Ideal quantum computing

This section provides an overview of the basic concepts involved in quantum computation, including quantum states, quantum gate operations and quantum measurements. A more detailed description can be found in [31].

1.1.1. Quantum states

Every isolated quantum system of finite dimensions $d \in \mathbb{N}$ is associated to a complex Hilbert space $\mathcal{H}_d \cong \mathbb{C}^d$, called the *state space*. Elements of the state space are l_2-normalized *pure quantum states*, denoted as $|\Psi\rangle \in \mathbb{C}^d$. The dual to $|\Psi\rangle$ is written as $\langle\Psi|$, with $\langle\Phi|\Psi\rangle$ being the inner-product between two pure states $|\Psi\rangle$ and $|\Phi\rangle$. Using the state set $\{|0\rangle, \ldots, |d-1\rangle\}$, constituting an orthonormal basis in $\mathbb{C}^d$ termed the *computational basis*, $|\Psi\rangle$ can be written as:

$$|\Psi\rangle = \sum_{j=0}^{d-1} \psi_j |j\rangle = \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{2^N-1} \end{pmatrix}, \tag{1.1}$$

with $\psi_j = \langle j|\Psi\rangle \in \mathbb{C}$ and $\sum_j |\psi_j|^2 = 1$. A special case in the context of quantum computing are two-level ($d = 2$) quantum systems (e.g. spin-$\frac{1}{2}$ systems), whose state space is $\mathbb{C}^2$. These systems are commonly referred to as *qubits*. Qubits denote the smallest unit of information in quantum computation and represent the quantum mechanical analogue to *classical bits*. While the latter lives in a *discrete state space* $\{0, 1\}$, i.e. at each point in a computational the system can either reside in the 0 or 1 state, a qubit can be in any complex linear combination (*superposition*) of its computational basis states, with $|0\rangle$ ($|1\rangle$) representing the classical bit state 0 (1). Thus, quantum theory provides a much broader *continuous state space* for computation to operate in, determined by the complex *state amplitudes* ψ_0 and ψ_1. A common way to visualize the state and operations performed on a single qubit is to represent it as a unit vector inside a three-dimensional unit sphere. Using $|\psi_0|^2 + |\psi_1|^2 = 1$, implying that $|\psi_0| = \cos(\theta/2)$ and $|\psi_1| = \sin(\theta/2)$ with an angle $\theta \in [0, \pi]$, and the fact that global phases of quantum systems are unobservable, yielding $\phi \in [0, 2\pi)$ as the relative phase between the two state amplitudes, Eq. 1.1 becomes:

$$|\Psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\,. \tag{1.2}$$

Given by the domain of the two angles θ and ϕ, every pure single-qubit state $|\Psi\rangle$ can thus be visualized as a point on the surface of a unit sphere termed the *Bloch sphere* (see Fig. 1.1). In doing so, the Bloch sphere operates in a three-dimensional Cartesian coordinate system, with the unit axes corresponding to the ± 1 eigenstates of the three Pauli matrices:

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \sigma^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.3}$$

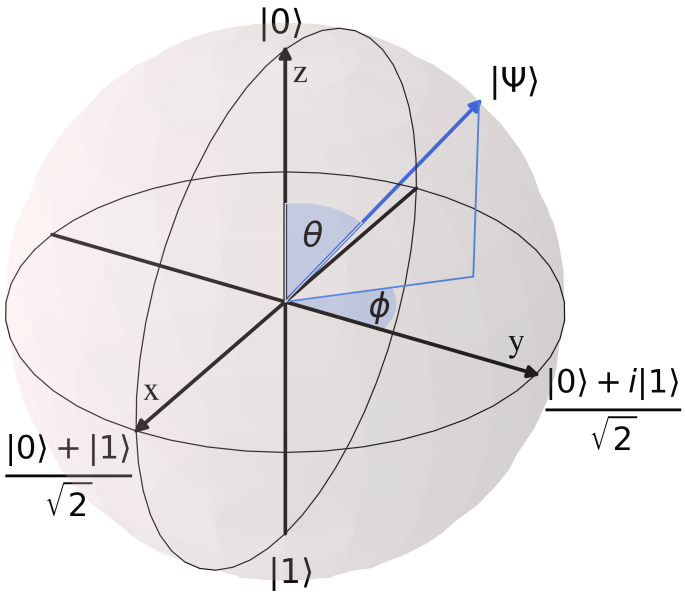


Figure 1.1.: The figure shows a sketch of the Bloch sphere, representing a pure single-qubit state $|\Psi\rangle$. The North and South Pole correspond to the ground state $|0\rangle$ and the excited state $|1\rangle$ of the two-level quantum system, respectively. An arbitrary position on the sphere is parameterized via Eq. 1.2 using the azimuthal angle $\theta \in [0, \pi]$ and the polar angle $\phi \in [0, 2\pi)$.

In particular, the eigenstates of σ^z form the computational basis $\{|0\rangle, |1\rangle\}$ and are located at the North and South Pole of the sphere, respectively. Moreover, the eigenstates $|\pm\rangle = (|0\rangle \pm |1\rangle)$ $(|\pm i\rangle = (|0\rangle \pm i\,|1\rangle))$ of σ^x (σ^y) denote the positive and negative x-axis (y-axis). As a consequence, the three-dimensional *Bloch vector* $\mathbf{r} = (r_x, r_y, r_z)^\top$, representing $|\Psi\rangle$ on the surface of the Bloch sphere, can be derived from the expectation values of $|\Psi\rangle$ with the Pauli matrices:

$$\mathbf{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = \begin{pmatrix} \langle\Psi|\sigma^x|\Psi\rangle \\ \langle\Psi|\sigma^y|\Psi\rangle \\ \langle\Psi|\sigma^z|\Psi\rangle \end{pmatrix} = \begin{pmatrix} r\sin\phi\cos\theta \\ r\sin\phi\sin\theta \\ r\cos\phi \end{pmatrix}. \tag{1.4}$$

Note that in this parametrization, one can think of the Bloch sphere as a distorted half-sphere, since opposite vectors on the sphere represent orthogonal states. So far, only single-qubit systems have been considered. In order to perform meaningful calculations, composite quantum systems of N qubits are needed, where the associated state space $\mathcal{H}_{2^N} = \mathcal{H}_2^{\otimes N} = \text{span}\{|0\rangle, |1\rangle\}^{\otimes N} \cong \mathbb{C}^{2N}$ is given by the N-fold tensor product of the individual state spaces $\mathcal{H}_2$. This means that an N qubit state is a superposition of the 2^N computational basis states generated by all combinations of the single qubit bit states:

$$|\Psi\rangle = \sum_{j_0 \ldots j_{n-1} \in \{0,1\}^N} \psi_{j_0 \ldots j_{n-1}} |j_0 \cdots j_{n-1}\rangle = \sum_{J=0}^{2^N-1} \psi_J |J\rangle = \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{2^N-1} \end{pmatrix}. \tag{1.5}$$

Here, the basis states are indexed by the integer $J = [j_0 \ldots j_{n-1}]_2$, with j_i denoting the computational state of qubit i and $\psi_J = \langle J|\Psi\rangle$. Note that for simplicity of notation, tensor

product states $|\Psi\rangle \otimes |\Phi\rangle$ will be abbreviated as $|\Psi\rangle|\Phi\rangle$ or $|\Psi\Phi\rangle$. An important property of these composite quantum systems is that not all states in the tensor-product space $\mathcal{H}_2^{\otimes N}$ can be expressed as a tensor product themselves. Such states are called *entangled states*, referring to a strong purely quantum-mechanical correlation between the involved qubits, allowing to gain information about a qubit's state by only observing its entangled qubits. As an example, consider the two-qubit entangled Bell state $|\Phi^+\rangle = \frac{1}{2}(|00\rangle + |11\rangle)$. By determining the state of either one of the two qubits, the state of the other one is also inferred. Both quantum entanglement and state superposition are exclusive to quantum systems and are often stated as the main properties of quantum computing that provide an advantage compared to the classical computation model [5].

1.1.2. Quantum gates

In order to perform computations on an N-qubit quantum system, one needs to define a set of operations on the composite quantum state. By analogy with a classical computer that performs elementary gate operations, like *AND* and *NOT*, to alter the state of its classical bits, a quantum computer uses a sequence of *quantum gates* (called a *quantum circuit*) to modify the state amplitudes of one or more qubits. These transformations are governed by the *time-dependent Schrödinger equation* (TDSE) of closed quantum systems:

$$i\hbar \frac{\mathrm{d}|\Psi(t)\rangle}{\mathrm{d}t} = \hat{H}(t)\,|\Psi(t)\rangle\,, \tag{1.6}$$

with $\hat{H} \in \mathbb{C}^{2N\times 2N}$ denoting the Hamiltonian under which the system evolves, and $|\Psi(t)\rangle$ referring to the quantum state at some point in time t. Note that throughout this thesis, units with $\hbar = 1$ will be used. In case $\hat{H}$ is time-independent and the system is initialized in some state $|\Psi_0\rangle$, the time evolution is given by:

$$|\Psi(t)\rangle = e^{-i\,t\,\hat{H}}\,|\Psi_0\rangle\,, \tag{1.7}$$

yielding the unitary evolution operator $\hat{U} = e^{-it\hat{H}}$. Thus, by preparing suitable Hamiltonians $\hat{H}$, arbitrary unitary operations can be applied to the state vector $|\Psi\rangle$. Since the three Pauli matrices (see Eq. 1.3), in combination with the identity operator $\hat{\mathbb{I}}$, form a complete orthogonal basis in $\mathbb{C}^{2\times 2}$, any quantum gate acting on a single qubit can be expressed as a linear combination of time evolutions under the Pauli operators. With respect to the Bloch sphere representation of a single-qubit quantum systems (see Fig. 1.1), these evolutions define rotation operations $R_\alpha(\gamma) = \exp(-i\,\gamma\,\sigma^\alpha/2)$ of the Bloch vector $\mathbf{r}$ around an axis α by some angle γ, with σ^α denoting the respective Pauli matrix. Hence:

$$R_x(\gamma) = e^{-i\frac{\gamma}{2}\sigma^x} = \begin{pmatrix} \cos(\gamma/2) & -i\sin(\gamma/2) \\ -i\sin(\gamma/2) & \cos(\gamma/2) \end{pmatrix}, \tag{1.8}$$

$$R_y(\gamma) = e^{-i\frac{\gamma}{2}\sigma^y} = \begin{pmatrix} \cos(\gamma/2) & -\sin(\gamma/2) \\ \sin(\gamma/2) & \cos(\gamma/2) \end{pmatrix}, \tag{1.9}$$

$$R_z(\gamma) = e^{-i\frac{\gamma}{2}\sigma^z} = \begin{pmatrix} \exp(-i\gamma/2) & 0 \\ 0 & \exp(i\gamma/2) \end{pmatrix}. \tag{1.10}$$

Besides these elementary rotation gates, the *Hadamard gate* H, defined by

$$H = -i\, R_x(\pi)\; R_y(\pi/2) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{1.11}$$

is a commonly used single-qubit gate. Its importance comes from the fact that it allows to create and resolve superpositions between the computational basis states by mapping: $\{|0\rangle, |1\rangle\} \leftrightarrow \{|+\rangle, |-\rangle\}$. However, using single-qubit gates alone does not yield universal quantum computation, as for example entanglement between qubits cannot be created. Consequently, additional two-qubit gates are often considered, including the *controlled-NOT* ($CNOT_{i,j}$) and the *controlled-phase* ($CZ_{i,j}$) gates, which perform conditioned operations on the second qubit j depending on the state of the first qubit i:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{1.12}$$

In doing so, arbitrary qubits can be entangled: e.g. $CNOT_{0,1}\, H_0\, |00\rangle = |\Phi^+\rangle$. Although quantum gates can in general operate on the state spaces of arbitrary numbers of qubits, one usually considers at most two-qubit operations, due to the limited connectivity between qubits on real-world quantum computing hardware, typically allowing only interactions between neighbouring qubits. Fortunately, however, a universal set of quantum gates can already be constructed from the aforementioned single- and two-qubit gates [32, 33]: $\{H,\, T = R_z(\pi/4),\, CNOT\}$. Note that many other sets of gates are also known to be universal, which are all computationally equivalent to each other up to polylogarithmic factors, ensured by the *Solovay-Kitaev Theorem* [34]. A detailed list of common quantum gates can be found in appendix A.

1.1.3. Quantum measurements

After a quantum algorithm, i.e. a series of single- and multi-qubit gates, has been performed on an initial quantum state, e.g. $|0\rangle^{\otimes N}$ or $|+\rangle^{\otimes N}$, one eventually has to extract information from the final state vector $|\Psi\rangle$. In contrast to a classical bit, however, quantum mechanics protects the state amplitudes ψ_J, defining $|\Psi\rangle$, from direct observations. This is because they are associated with measurement probabilities, such that when performing a projective measurement in the computational basis of N qubits, the outcome $J \in \{0,1\}^{\otimes N}$ is obtained with probability $|\psi_J|^2$, yielding the post-measurement state $|J\rangle$ of the quantum system. This collapse of the state vector is unavoidable and necessary for the state to be consistent with the measurement outcome. Often, the more information a measurement extracts from $|\Psi\rangle$, the more disturbed the final state will be. As a consequence, since a single measurement only provides a snapshot of $|\Psi\rangle$, numerous state preparations and measurements are generally required to reconstruct the quantum state with sufficient accuracy.

1.2. Quantum circuit simulation

Quantum computing evolved as an interdisciplinary field between physics and computer science with the goal to accelerate computational bottlenecks using the laws of quantum-mechanics. By exploiting quantum phenomena such as superposition of computational basis states, qubit entanglement and complex phase interferences, quantum computers are able to simultaneously manipulate all bit combinations in a single operation, thus performing calculations/evaluations in a seemingly highly parallel fashion [5]. This sparked the hope of finding quantum algorithms that run exponentially faster compared to their classical counterparts when solving problems of non-polynomial complexity. Promising applications of quantum computing can be found in the fields of combinatorial optimization problems [8, 9], quantum chemistry [10, 11], machine learning [12–14] and cryptography [15]. To the present day, however, access to gate-based quantum computers, despite being partially available as cloud services [35, 36], is still limited, offering only relatively few qubits for operation. Moreover, significant environmental noise on quantum gates and limited connectivity between the qubits will continue to exist on near-term hardware. The former requires error-correction strategies, which significantly reduce the number of logical qubits, for meaningful calculation [37] and the latter increases the total number of gates in a circuit by requiring numerous SWAP gates, in turn introducing additional error sources. As a consequence, classical simulation of quantum computing algorithms emerged as a necessary discipline for developing and debugging new quantum applications [23–25]. Among others, the study of algorithmic complexity, the benchmarking of existing quantum computing hardware and the design of quantum circuits that are difficult to characterize analytically (see e.g. chapter 2), represent common use cases.

The framework that will be considered throughout this thesis treats the ideal quantum computing device as an accelerating hardware to a classical processor [23]. Much like a graphics processing unit (GPU) that is mainly used for rendering, the quantum computer is dedicated to certain problem types and will receive a classical initial quantum state (e.g. $|0\rangle^{\otimes N}$ or $|+\rangle^{\otimes N}$) and a set of instructions (quantum gates) to perform on the state. As the laws of quantum mechanics prohibit a direct readout of the final quantum state, the output of the quantum algorithm will be obtained via measurements, returning a classical discrete computational state according to the measurement probabilities (see section 1.1.1). Using this model of computation, the quantum computer can simply be replaced by a classical simulator, that will be introduced in the following sections.

1.2.1. Simulation techniques

The simulation of ideal universal quantum computation is an inherently non-trivial task on classical digital computers, due to the exponentially growing complexity of the simulated Hilbert space [38]. Consider for example a 45-qubit problem instance. Performing a single gate operation on the quantum state generally requires updating the state amplitudes of 2^{45} basis states, which, in case the full state vector $|\Psi\rangle$ is held in memory, requires $\approx 1/2$ Petabytes of storage. Consequently, the simulation of intermediate quantum circuits ($N \geq 40$) already reaches the capabilities of modern supercomputers. With respect to these massive computational resources required three branches of quantum circuit simulators, each proposing a different dependency of the exponential scaling on the properties of the quantum circuit, have emerged in the literature:

- **Direct state evolution (Schrödinger style):** This ansatz is an instance of the so-called *Schrödinger-style* simulation, a mainstream technique for general-case simulation of quantum algorithms, circuits and physical devices [39]. It represents the full quantum state $|\Psi\rangle$ by a vector of complex-valued amplitudes, which are then modified in place through a series of unitary transformations (gate operations). By keeping a close relation to the theory discussed in section 1.1.2, each state amplitude can be updated using only a few basic operations, e.g. 2 multiplications and 1 addition for a single qubit gate, yielding a small computational overhead and a computational cost that scales linearly in the number of qubits N involved. This, however, comes at the cost of an exponential memory footprint, since the full state vector (2^N complex double precision floats = 2^{N+4} bytes) must be kept in memory throughout the entire simulation. As a consequence, direct state evolution algorithms are currently limited to about 45 qubits on modern supercomputers [24, 30].

- **Tensor network contractions (Feynman style):** The main idea of this approach is based on the representation of a quantum circuit as a tensor network, with rank-2 tensors (tensors of two 2-dimensional indices) denoting single qubit gates, rank-4 tensors (tensors of four 2-dimensional indices) being two-qubit gates, and in general rank-2N tensors representing N-qubit gates [40–42]. A tensor network is then a tensor expression involving the product of several tensors of varying ranks, e.g. $T_{lm} = \sum_{ghijk} A_{gh} B_{hi} C_{ij} D_{gk} E_{jklm}$. In order to apply a gate operation to a qubit in this network, the corresponding gate tensor has to be adjoined to the open indices of the target qubit. By attaching all gates to the network in the order in which they appear in the circuit, the tensor network encodes the output state of the quantum algorithm. A main benefit of this simulation ansatz is its memory efficient representation of the network structure, which grows only linearly in the number of qubits and gates involved, allowing to simulate system sizes of up to 100 qubits on modern supercomputers [43]. This efficiency, however, comes at the cost of computational overhead, as for obtaining the probability amplitudes the full network must be contracted, yielding the multiplication of all tensors. In doing so, the computational and memory cost of the contraction depend heavily on the rank of the largest intermediate tensor and scale (at least) exponentially in the number of open indices, making it impractical for deep circuits [21]. Moreover, finding the optimal contraction order of the network is believed to be NP-complete [44]. Besides that, contracting the network with N open indices again causes an exponential memory footprint, since 2^N amplitudes need to be stored. Consequently, tensor network strategies typically utilize the *Feynman path summation* ansatz of calculating the single state amplitudes separately, thus treating memory cost for computation time [45–47]. As a conclusion, tensor network contraction codes are generally viable in the case of large shallow quantum circuits, where only the calculation of a few amplitudes of the final quantum state is required.

- **Hybrid algorithms (Layered simulation):** Recently, several authors have been exploring hybrid quantum simulation techniques, by combining direct state evolutions with Feynman path summation in order to enable scaling tradeoffs between the circuit depth (number of gates) and size (number of qubits) with respect to the computational resources needed [38, 48, 49]. Generally, these approaches divide the original quantum circuit into several sub-circuits with less qubits, which are then simulated independently using the same method as for the Schrödinger-style simulation. In doing so, the memory requirement for simulating these sub-circuits becomes independent of the total number of qubits and gates in the circuit, but depends only on their config-

urable size. As a result, simulations with up to 128 qubits have been achieved in the literature [27]. However, this ansatz is only viable in case the individual sub-circuits are connected loosely, since each entangling gate doubles the number of independent circuits, as each state configuration of the involved qubits has to be simulated separately and summed up later. This can either be achieved by storing snapshots of the sub-state-vector in memory or computing parts of the circuit multiple times. Consequently, either the memory or computational cost scales exponentially in the number of entangling gates in the cut [21].

The main goal of the quantum circuit simulator, which will be discussed in the following sections, is to provide a time-efficient simulation framework for investigating quantum optimization algorithms in chapter 2. These investigations will focus on variational quantum algorithms in the regime of intermediate circuit depths involving 10 to 30 qubits. The corresponding quantum circuits are structured in layers, with each layer consisting of a set of single-qubit rotation gates and an N-qubit diagonal transformation, with the latter entangling the majority of qubit pairs in the system. An example of such a quantum circuit can be found in appendix B.1. By combining up to 80 gate-layers, the simulator must be able to compute several thousands of individual gate operations efficiently. Moreover, the respective circuits must be evaluated multiple thousands of times, due to the variational character of the studied algorithms. In addition to this, the energy expectation value with respect to a given problem Hamiltonian is needed during the calculations, hence requiring samples of the full state vector at the end of each circuit run.

Although tensor network contraction based algorithms have been applied to variational quantum algorithms in the past [43, 50], the strong requirement of shallow circuit depths makes this technique unsuitable for the targeted investigations. Moreover, the large number of entangling gates spread across all pairs of qubits also prevents an efficient separation into sub-circuits, ruling out hybrid approaches as well. Fortunately, however, the full state vector of up to 30 qubits can be fit into memory, thus a direct state evolution algorithm will henceforth be considered.

1.2.2. Programming framework

The Schrödinger-style simulation of intermediate-sized quantum circuits requires a simulation code that can make efficient use of the parallel architecture of modern supercomputers, utilizing the vast number of compute nodes, hardware accelerators (e.g. GPUs) and inter-network communication bandwidths [38, 51]. Previous work has already demonstrated promising performance improvements by the introduction of GPUs into the simulation process [25, 30, 52], due to the highly parallel nature of the update routines (e.g. applying a q-qubit gate yields 2^{N-q} independent subsets of amplitudes, which can be modified simultaneously). For example, *Willsch et al.* achieved an up to 49 times faster circuit evaluation using the GPU-accelerated version of JUQCS compared to its classical count part. Motivated by these findings, a novel *C++* based quantum circuit simulator, termed SEQCS, operating on GPUs distributed among multiple compute nodes is developed as part of this thesis. Key features of this simulator are (1) an improved memory management by making efficient use of self-managed cache on the GPU, (2) a reduction of gate complexities, by restructuring the quantum circuit and combining multiple gate operations into single update routines, and (3) an advanced internode communication scheme using the *Message Passing Interface* (MPI) [53]. Note, that the following discussions will focus on the JUWELS Booster supercomputer [29], providing access to over 3500 *NVIDIA A100* GPUs [54].

1.2.2.1. CUDA platform

The *Compute Unified Device Architecture* (CUDA) [55] has been developed as both a hardware and software platform that is able to exploit the massive parallel processing capabilities of recent NVIDIA GPUs. Introduced in 2006, it offers a scalable and parallel programming model to bypass the graphics API and thus solve general-purpose problems on the GPU directly using *C++*.

From a hardware perspective, the GPU device is structured into *Graphics Processing Clusters* (GPC), featuring multiple *Texture Processing Clusters* (TPC), that in turn consist of arrays of *Streaming Multiprocessors* (SM). Each SM contains a set of *Streaming Processors* (SP), often separated into single (32-bit) and double (64-bit) precision compute units, load and store memory units, and special function units, e.g. used for calculating trigonomic functions. In Fig. C.6 in appendix C the ampere architecture used by the NVIDIA A100 GPUs is depicted. In doing so, the CUDA execution model is based on a hierarchy of abstraction layers distributed among the SMs: *grids*, *blocks*, *warps* and *threads*. Threads denote the most basic execution units, which are directly mapped onto the individual SPs. A batch of threads cooperating together on one multiprocessor is termed a block. A grid is composed of several blocks, and because there can be more blocks than multiprocessors, different blocks of a grid are scheduled among the array of SMs. The GPU supports the parallel execution of very large numbers of light-weight threads, with minimal overhead for thread creation, synchronization, and termination. This is necessary, as the highly scalable architecture of the GPU demands a *Single Instruction Multiple Threads* (SIMT) execution scheme on the SMs. Thus, each block of compute units within the SM partitions (see Fig. C.7 in appendix C) performs the same instruction on different threads and supposedly on different data elements (Data-Level Parallelism). As a consequence, threads within a block are additionally grouped into warps, i.e. fixed sets of 32 threads, such that in each clock cycle a subset of all warps in a block is assigned for execution to the SM partitions by the warp scheduler. Note that in general it is preferable to have blocks consisting of more warps than SM partitions, as well as multiple blocks under scheduling on the same SM, since it allows the warp scheduler to potentially hide memory access latencies by postponing the execution of one warp in favour of another. The occupancy of the SMs can be calculated by the fraction of *active* threads, limited by the required hardware resources for their execution, to the number of possible threads, determined by the compute capability of the GPU.

A CUDA program is now specified in terms of a *kernel* function, that is executed concurrently by all threads. In order to direct the calculation, the API provides the 16 bit integer variables *threadID* $T = [t_{15} \ \dots \ t_0]_2$ and *blockID* $B = [b_{15} \ \dots \ b_0]_2$, denoting the thread index in a block and the block index in the grid, respectively, which can be used to identify the individual threads. In doing so, CUDA draws a strict barrier between CPU (Host) and GPU (Device) codes, including a separation of their memory spaces. The latter demands explicit memory transfers between the Host and the Device. Regarding Device memory, CUDA grants low-level access to the various memory spaces on the GPU, including *global*, *shared* and *constant memory*: Global memory resembles the main Device memory, offering the largest capacity (40 GB on the A100 GPU) but also the highest access latencies ($400 - 800$ clock cycles). It is available to all threads in the application and will be used to store the entire state vector $|\Psi\rangle$, i.e. the array of double-precision complex state amplitudes, of up to 31 qubits. Shared memory is a form of self-managed L1-cache (latencies ≈ 10 clock cycles), private to each thread block. It has a size of 192 KB per SM (on the A100 GPU) and will store subsets of the state vector of 11 qubits across 4 blocks on each SM. Finally, Constant memory describes

a section of the global Memory. As it is read-only, it is heavily cached into 8 KB of L1-cache on the SMs, yielding similar performances as shared memory in case all threads in a warp access the same memory location. It will be used to store the properties of the quantum circuit (e.g. number of qubits and gates) as well as the series of gate operations.

1.2.2.2. Hardware limitations

In order to achieve high performance in the simulation of quantum circuits using CUDA, the fine-grained parallelism of the GPU has to be exploited by considering the following performance bottlenecks [55]:

- **Warp divergence:** Control flow statements in the kernel function can force threads to follow different execution paths. If this happens between threads of the same warp, the warp must be re-executed for each path taken, each time disabling all threads that do not follow the respective execution path. This process is called *warp divergence* and is caused by the SIMT architecture of the GPU. It can result in performance losses of up to 32 times, in case all threads have to be evaluated separately.
- **Global memory access:** Global memory requests are served via either 32-, 64- or 128-byte memory transactions. When threads of the same warp issue a memory request, their collective access can be satisfied via a single memory transaction, provided that: (1) the memory accesses are sequential; (2) the starting address of the requested memory segment is a multiple of the transaction size; (3) the requested size per thread is 4, 8 or 16 bytes. If all conditions are met, the transactions will be combined, while otherwise the threads will be served separate 32-byte memory transactions, reducing the global memory bandwidth.
- **Bank conflicts:** The high memory bandwidth of the shared memory is achieved by dividing the memory space into 32 parallel accessible banks, where sequential 32-bit words are organized into sequential banks. Memory requests that fall into separate banks can be executed simultaneously. However, if m requests point to the same bank, they must be served sequentially, causing an m-way bank conflict and lowering the achieved bandwidth.
- **Register spilling:** Registers are used by the threads to store variables and intermediate results of calculations. The total number of registers available on each SM is fixed (65536 32-bit registers on the A100 GPU) and is distributed equally across all *active* threads on the SM. In case a block requests more registers than available, register spilling occurs, and register data is temporarily stored in a section of the global memory. This can have a significant impact on performance, as the variable access speeds increase from ≈ 1 cycle to ≥ 400 cycles.

1.2.3. Implementation of SEQCS

The target problem of a quantum circuit simulator based on direct-state evolutions is to compute the complete output state of an N qubit register, $|\Psi^{out}\rangle = \sum_{J=0}^{2^N-1} \psi_J^{out} |J\rangle$, starting from an initial state $|\Psi^{in}\rangle = \sum_{J=0}^{2^N-1} \psi_J^{in} |J\rangle$, given a unitary transformation $\hat{U}$ of dimension $2^N \times 2^N$, i.e. $|\Psi^{out}\rangle = \hat{U} |\Psi^{in}\rangle$. At each point in the simulation, the entire state vector $|\Psi\rangle$ is held in the global memory of the GPU by storing its complex amplitudes ψ_J consecutively in a double-precision array of 2^{N+4} bytes. Note that other approaches to reduce the overall

memory-footprint by using specially designed single-precision encodings have been investigated in the literature before [38, 56, 57]. However, as these typically come at the cost of additional computational overhead and reduced accuracy for deep quantum circuits, they are unsuitable for the targeted simulations in chapter 2 and hence will not be considered here. The unitary transformation $\hat{U}$ is defined by the linear combination of M individual quantum gate operations $\hat{G}_i$ constituting the quantum circuit, i.e. $\hat{U} = \prod_{i=0}^{M-1} \hat{G}_i$. The arrangement of the gate operations $\hat{G}_i$ in the matrix product defines the *gate-queue* $\left\{\hat{G}_0, \ldots, \hat{G}_{M-1}\right\}$, and thus the order in which the transformations are applied to the state vector $|\Psi\rangle$. Note that the initial order of the queue (defined by the user input), will be modified during several preprocessing steps as depicted in sections 1.2.3.1 to 1.2.3.3 in favour of a performant simulation with respect to section 1.2.2.2. This is in contrast to other approaches, which leave the quantum circuit untouched [30]. Without loss of generality, the unitary matrix $\hat{G}$ associated with an application of a quantum gate over n consecutive qubits starting from qubit i, is given by:

$$\hat{G} = \hat{\mathbb{I}}^{N-n-i} \otimes \hat{g} \otimes \hat{\mathbb{I}}^{i}, \tag{1.13}$$

with the sub-matrix $\hat{g} = \sum_{i,k=0}^{2^n-1} u_{i,j} \, |i\rangle \langle j|$ of dimension $2^n \times 2^n$ corresponding to the transformation of the n involved qubits. Applying $\hat{G}$ to a state amplitude ψ_J denoting the N-qubit computational basis state $|J\rangle = |j_{N-1} \ \ldots \ j_1 \ j_0\rangle$, with $[j_{N-1} \ \ldots \ j_1 \ j_0]_2$ being the binary representation of the integer index J, yields:

$$\psi_J^{out} = \hat{G} \psi^{in}_{j_{N-1} \cdots j_1 j_0} \tag{1.14}$$

$$= \sum_{K=k_0 \ldots k_n \in \{0,1\}^n} \begin{cases} u_{0,K} \ \ \psi^{in}_{j_{N-1} \ \ldots \ j_{n+i} \ k_{n-1} \ \ldots \ k_0 \ j_{i-1} \ \ldots \ j_0}, & \text{if } j_{n+i-1} \ldots j_i = 0 \ldots 00 \\ u_{1,K} \ \ \psi^{in}_{j_{N-1} \ \ldots \ j_{n+i} \ k_{n-1} \ \ldots \ k_0 \ j_{i-1} \ \ldots j_0}, & \text{if } j_{n+i-1} \ldots j_i = 0 \ldots 01 \\ \quad \vdots & \\ u_{2^n-1,K} \ \ \psi^{in}_{j_{N-1} \ \ldots \ b_{n+i} \ k_{n-1} \ \ldots \ k_0 \ j_{i-1} \ \ldots \ j_0}, & \text{if } j_{n+i-1} \ldots j_i = 1 \ldots 11 \end{cases} \tag{1.15}$$

$$= \sum_{K=k_0 \ldots k_n \in \{0,1\}^n} u_{A,\,K} \, \psi^{in}_B. \tag{1.16}$$

Here, $A = >>_i (J) \ \& \ (2^n - 1)$ and $B = [J \ \& \ \sim (2^{n+i} - 2^i)] + <<_i (K)$, with & denoting the bit-wise AND, $\sim$ being the bit-wise NOT, and $<<_i$ and $>>_i$ referring to the bit-wise left and right shift operations by i bits, respectively. According to Eq. 1.16, the state transformation is structured into 2^{N-n} disjoint closed *subspaces* of 2^n amplitudes, such that the calculation of each amplitude only involves amplitudes that belong to the same subspace. Consequently, the binary representation $[j_{N-1} \ \ldots \ j_1 \ j_0]_2$ of the amplitude index J can be separated into two kinds of bit strings. The binary values $I_T = [j_{i+n} \ \ldots \ j_i]_2$, denoting the qubit indices which a gate operation is acting on, will henceforth be called *target indices* forming the *target index* I_T of the corresponding amplitude ψ_J in its respective *target-space*. On the other hand, the remaining bits $I_G = [j_{N-1} \ \ldots \ j_{i+M} \ j_{i-1} \ \ldots \ j_0]_2$ describe the *subspace indices*, with the *subspace index* I_S labelling the individual target-spaces. Thus, amplitudes belonging to the same subspace share identical I_S and different I_T. Using this convention, the state update can be computed in parallel across the target-spaces by assigning each subspace to one of

2^{N-n} threads. The details about the implementation and optimization of this process will be discussed in the following three sections, starting with the distribution of the state vector across multiple compute nodes (section 1.2.3.1), continued by the memory management within each GPU (section 1.2.3.2), and completed by the implementations of the various gate types (section 1.2.3.3). Based on these strategies, the execution sequence depicted in Fig. 1.2 is used by SEQCS for the simulation of a quantum circuit. Moreover, an example of a gate-schedule obtained by these strategies can be found in Fig. B.3 in appendix B.1. The program code of SEQCS is given in appendix D.

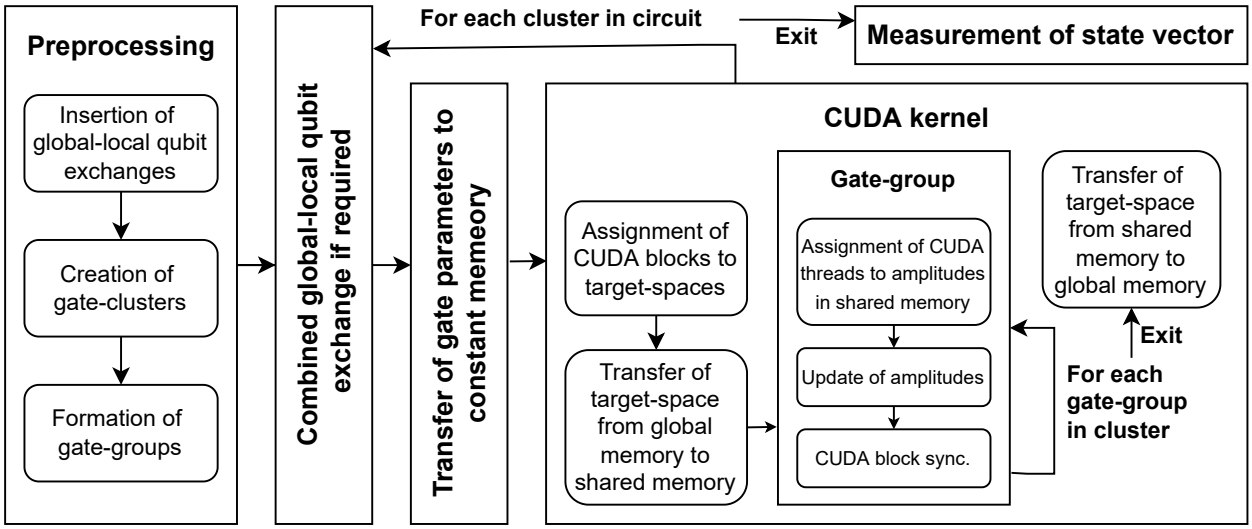


Figure 1.2.: The diagram depicts the execution sequence used by SEQCS for the simulation of an arbitrary quantum circuit. The process starts by analysing the circuit (Preprocessing) and inserting global-local qubit exchanges (see section 1.2.3.1). Afterwards, the gate queue is rearranged in order to form gate-clusters (see section 1.2.3.2) and subsequentially gate-groups (see section 1.2.3.3). Following this preprocessing stage, SEQCS evaluates each gate cluster separately, by first executing global-local qubit swaps and transferring the respective gate parameters into the constant memory. Then, a CUDA kernel is launched, which first transfers the target-space into the shared memory. Next, each gate-group involved in the cluster is executed in-place on the shared memory. Finally, the target-space is transferred back into the global memory and SEQCS continues with the next cluster. In case all clusters have been evaluated, a user-defined measurement is applied to the state vector and the program exits.

1.2.3.1. Multi-node computation

The simulation of large quantum circuits involving numerous qubits using direct state evolution is generally limited by the amount of memory that is available on the simulating system. The NVIDIA A100 GPUs, which are considered here, support state vectors up to $N = 31$ qubits (≈ 32 GB) in their global memory, putting a hard limit on the number of qubits that can be simulated via a single GPU. In order to simulate systems beyond 31 qubits, the highly parallel architecture of the JUWELS Booster system will be utilized, to perform parallel computations on subsets of the state vector $|\Psi\rangle$ distributed among several compute nodes and thus memory spaces. Largely inspired by the work of *De Raedt et al.* [58], the idea is to use an exponentially growing number of GPUs, allowing to store state

vectors of up to 42 qubits (= 2048 A100 GPUs) as well as to compensate the exponentially growing amount of arithmetic operations needed to compute Eq. 1.16. Hence, a close-to ideal weak-scaling of the gate computation time can be achieved, which is only limited by the amount of internode communication during the calculation. The latter is realized using the *CUDA-aware MPI*, which enables direct transfers between the Devices' global memory spaces without expensive exchanges into the Host memory [53]. Note that the investigation of other communication frameworks, such as *NCCL* [59] and *NVSHMEM* [60], both promising significant speed-ups in GPU-GPU-communication compared to MPI [51], is left for future work.

Regarding the implementation of the distributed simulation, a set of 2^{N_G} MPI processes is used, each being individually assigned to a *rank* R and the memory space of one GPU. In doing so, each process is responsible for the computation of a subset of $|\Psi\rangle$ of size 2^{N_L}, with $N_L \leq 31$ and $N = N_L + N_G$. As a consequence, the qubits and thus the bit string $[j_{N-1} \ \dots \ j_1 \ j_0]_2$ of the amplitude index J is separated into two regions:

$$J = [\ \underbrace{j_{N-1} \ \dots \ j_{N_L}}_{\text{global qubits}} \ \underbrace{j_{N_L-1} \ \dots \ j_0}_{\text{local qubits}}\ ]_2. \tag{1.17}$$

The first N_L qubits, i.e. $I_L = [j_{N_L-1} \ \dots \ j_0 \ ]_2$, denote the set of *local qubits*, meaning that for each qubit $i < N_L$ both coefficients $\psi_{j_{N-1} \ \dots \ j_{N_L} \ \dots \ j_{i+1} \ 0/1 \ \dots \ j_0}$ of the state vector $|\Psi\rangle$ are always located at the same local memory of one of the N_G MPI processes for all 2^{N-1} combinations of the computational basis states of the remaining qubits. In contrast to this, the coefficients $\psi_{j_{N-1} \ \dots \ j_{i+1} \ 0/1 \ \dots \ j_{N_L-1} \ \dots \ j_0}$ with $i \geq N_L$ belong to the MPI processes $R_1 = [j_{N-1} \ \dots \ 0 \ \dots \ j_{N_L}]_2$ and $R_2 = [j_{N-1} \ \dots \ 1 \ \dots \ j_{N_L}]_2 = R_1 \wedge 2^{i-N_L}$, respectively, with $\wedge$ denoting the bit-wise XOR. Therefore, they are distributed globally among two separate GPUs and hence termed *global qubits*, with $I_G = j_{N-1} \dots j_{N_L}$ denoting the MPI rank. Note that an arbitrary amplitude ψ_J is located at GPU I_G at memory location I_L.

This distinction of the qubits is necessary with respect to the implementation of the gate operations. Two cases have to be considered here: (1) In case the gate operation (see Eq. 1.13) acts non-trivially only on a subset of local qubits, Eq. 1.16 can be computed independently in parallel on each GPU, as all elements (amplitudes) belonging to each target-space I_S are located in the same local memory of one GPU. Thus, no communication between the MPI processes is necessary. (2) This situation changes once an operation includes global qubits. The problem is that elements of the same target-space are distributed among several GPUs. Since each MPI process has only access to its local memory, transfers of state amplitudes between the GPUs become necessary. As an example, consider a Hadamard gate (see Eq. 1.11) applied to qubit $i \geq N_L$. Its action on the state vector can be computed by temporarily swapping the global qubit i with a local qubit $k < N_L$. Thus, pairs of MPI processes, with R_1 and $R_2 = R_1 \wedge 2^{i-1}$, exchange half of their local state amplitudes, such that $\psi_{j_{N-1} \ \dots \ 0 \ j_{i-1} \ \dots \ j_{N_L-1} \ \dots \ 1 j_{k-1} \ \dots \ j_0}$ ($\psi_{j_{N-1} \ \dots \ 1 \ j_{i-1} \ \dots \ j_{N_L-1} \ \dots \ 0 \ j_{k-1} \ \dots \ j_0}$) gets replaced by the value of $\psi_{j_{N-1} \ \dots \ 1 \ j_{i-1} \ \dots \ j_{N_L-1} \ \dots \ 0 \ j_{k-1} \ \dots \ j_0}$ ($\psi_{j_{N-1} \ \dots \ 0 \ j_{i-1} \ \dots \ j_{N_L-1} \ \dots \ 1 \ j_{k-1} \ \dots \ j_0}$). In doing so, the amplitude sub-spaces I_S are no longer distributed and thus Eq. 1.16 can be evaluated. Finally, the interchanged amplitudes are transferred back to their original memory spaces. Note that only in case of diagonal gates, i.e. $u_{i,k} = 0 \ \forall i \neq k < 2^n$ in $\hat{g}$ (see Eq. 1.13), no data exchange is needed despite acting on global qubits, since each amplitude can be updated exclusively based on its own value.

A clear drawback of this procedure is that it generally involves the transfer of the full state vector over the slow internode network (compare 1555 GB/s = GPU global memory bandwidth [54] and 200 GB/s = InfiniBand bandwidth [29]) each time a non-diagonal gate is acting on global qubits. As demonstrated by *Willsch et al.* [27, 30], these network transfers can account for significant fractions of the average simulation time per quantum gate, since for large circuits the gates are typically distributed equally among the qubits, hence increasing the number of required MPI transfers approximately linearly in N. In order to reduce the amount of communication between the MPI processes, the simulation replaces the final back-transfer of state amplitudes (needed to recover the original distribution across the GPUs) in favour of a bijective permutation operator $\sigma : \{0, \dots, N-1\} \to \{\sigma(0), \dots, \sigma(N-1)\}$. Here, σ is a $2 \times N$ dimensional matrix, mapping the N qubits ordered sequentially in the first row to their respective permutated bit-position. Consequently, Eq. 1.17 becomes: $\sigma^{-1}(J) = \left[j_{\sigma^{-1}(N-1)} \ \cdots \ j_{\sigma^{-1}(0)}\right]_2$. Exchanging a global qubit i at bit-position $\sigma(i) \geq N_L$ with a local qubit k at position $\sigma(k) < N_L$, means to transfer half of the local state vectors between pairs of MPI processes and swap $\sigma(i) \leftrightarrow \sigma(k)$. In doing so, the simulation keeps track of the position of the state amplitudes across the distributed memory spaces by simultaneously reducing the number of transferred bytes per global operation by a factor of two. Note that the permutation matrix σ has to be considered in the application of subsequent gates and measurements, since it might change the bit position these operations are acting on.

A key insight of the strategy depicted above is that the amount of transferred bytes between MPI processes throughout K global-local-qubit exchanges depends on the number of simultaneous swaps. Take for example a non-diagonal two-qubit gate acting on the global qubits i_1 and $i_2 < i_1$. When executing the qubit swaps separately, each MPI process R will first transfer half of its local state vector to the process $R \wedge 2^{i_1}$, swapping $i_1 \leftrightarrow l_1$, followed by a second transfer of half of its altered local amplitudes to the process $R \wedge 2^{i_2}$, exchanging $i_2 \leftrightarrow l_2$. However, when performing both operations simultaneously, each MPI process performs 3 transfers of size 2^{N_L-2} to processes $R \wedge 2^{i_1}$, $R \wedge 2^{i_2}$ and $R \wedge (2^{i_1} + 2^{i_2})$ (see appendix E.2), saving effectively 2^{N_L-2} amplitude transfers. In general, K combined global-local-qubit swaps decrease the number of exchanged amplitudes from $K 2^{N_L-1}$ to $(2^K - 1)\, 2^{N_L-K}$. Thus, in order to further reduce the amount of MPI-communication, the simulation uses the preprocessing algorithm depicted in appendix E.1 to determine optimal sequences of combined global-local-qubit swaps. Note that in general, as K and thus the number of MPI processes communicating in a group grows, *MPI synchronization barriers* can significantly impact the communication time, such that there exists an optimal K_{opt} [38]. The determination of K_{opt} and hence the investigation of the communication performance for various K is, however, left for future work.

The algorithm proposed here focusses on (1) maximizing the number of simultaneous global-local qubit swaps and (2) minimizing the total number of individual swap operations in the circuit. While the former reduces the amount of communicated bytes during the simulation, the latter reduces the number of inter-GPU synchronization barriers, which can impact the simulation time, since GPUs generally obey performance fluctuations, forcing some GPUs to temporarily idle. These fluctuations can only average out if several gate-operations are computed between the synchronization points, hence the algorithm tries to place the swap operations as far apart as possible. As depicted in appendix E.1, the algorithm starts by iterating through all gates in the circuit and investigating their target qubits. Once a gate $\hat{G}_k$ is found that operates on at least one global qubit, the current position $P = k$ in the gate-queue is saved and the required global and local qubits are stored in the arrays A_G and A_L, respectively. Continuing with the subsequent gates $\hat{G}_{j>k}$, their global and local qubits

are henceforth also stored in the respective arrays. This process is continued until adding the target qubits of one gate yields more global qubits in A_G than there are local qubits left in $A_L^{-1} = \{0, \ldots, N-1\} \setminus A_L$. In this case, a swap operation $SWAP_{A_G \leftrightarrow A_L^{-1}}$ is inserted at position P in the circuit, exchanging all global qubits in A_G with the unused local qubits in A_L^{-1}. Then, the state of the permutation operator σ is updated and the target bits of each previously visited gate after $\hat{G}_k$ are altered accordingly. Afterwards, A_G and A_L are cleared, P is set to the current position of the loop, and the process repeats until all gates in the queue have been visited. Considering the gate's local qubits to figure out a set of unused local qubits is important, as swapping the required global qubits with arbitrary local qubits could yield situations, where subsequent gates suddenly operate on global qubits, which would introduce additional SWAP gates into the circuit.

1.2.3.2. Single-Node computation

After having introduced the MPI communication scheme involved in the distributed simulation of the state vector among several compute nodes, this section concerns the update process of the local state vectors, assuming that all applied gates act exclusively on local qubits, i.e. on closed sub-spaces of amplitudes that fully reside in one GPU's global memory. An important property of this process is that the simulation of a gate operation is in general memory-bound [24], which can be seen by considering the application of an arbitrary single-qubit gate: To alter one state amplitude, the process involves two complex multiplications (4 FLOPs) and one complex addition (2 FLOPs) carried out on two entries of the state vector. On average, one complex double-precision float (16 bytes) has to be loaded and written back to memory in that process, yielding a computational intensity of $14/32\ FLOPs/byte < 1\ FLOPs/byte$ according to the Roofline model [61]. This, in combination with the high access latencies of global memory (a global memory transaction takes about 100 times more clock cycles than a double-precision addition), shows that repeated read/write requests during the update routines to the local state vector can become significant performance bottlenecks. As a consequence, optimizations to the memory-management and the memory access patterns will be discussed in the following. Assuming each CUDA thread is responsible for updating one pair of amplitudes in case of a single-qubit gate, then depending on which bit-position i the gate is acting on, the threads in a warp will perform global memory transactions with a stride of 2^i 16 byte words. With respect to section 1.2.2.2, already for $i \geq 3$ the stride exceeds the cache lane size (128 bytes), such that each thread will be served separate 32-byte memory transactions. This significantly reduces the global memory bandwidth by at least one order of magnitude, which cannot be compensated for by the memory latency hiding techniques of the GPU. As a consequence, typically low GPU utilizations of quantum circuit simulators are obtained [27].

With the purpose of obtaining high performance, SEQCS introduces the shared memory, as a form of self-managed on-chip cache, into the simulation process in order to significantly increase the memory access speeds (compare latencies of 10 vs. 800 clock cycles for shared memory and global memory transfers, respectively) [62]. The implemented approach considers a partition of the local coefficient space into closed subsets, termed *cluster-spaces*. This strategy is similar to the distribution of the full state vector among multiple compute nodes (see section 1.2.3.1), with the subsets now acting as computation units at CUDA block-level, such that each cluster-space can be processed independently in parallel in the shared memory. In order to achieve this, the quantum circuit gets structured into *gate clusters* during a preprocessing step. A gate cluster is composed of a closed sequence of consecutive gates,

preserving their original order in the gate-queue, such that each gate only acts on a closed partition of the set of local qubits. Two properties are important here: (1) the size of the joint set of target qubits r (*cardinality*) and (2) the degree of locality c (*coalescing*), meaning that the set of qubits $\{i \mid i < c\}$ is a subset of the joint set of target qubits. The algorithm used for assigning the gates to gate clusters is depicted in appendix E.3.

The algorithm traverses through the circuit (beginning with the first gate in the queue) using two nested loops, aiming to restructure the gate queue such that the number of consecutive gates forming gate clusters is maximized. Here, the outer loop selects a gate $\hat{G}_k$, with $\hat{G}_k$ not being assigned to any cluster formed before, e.g. the first gate in the circuit, and all prior gates $\hat{G}_{j<k}$ already belonging to gate clusters. The inner loop (for a fixed $\hat{G}_k$) then finds gates that cluster with $\hat{G}_k$ and restructures the circuit accordingly. This is achieved by designating each qubit as *unobstructed* (initially) or *obstructed*, with the latter preventing gates $\hat{G}_{j>k}$ acting on it from being moved next to $\hat{G}_k$. In doing so, the inner loop proceeds through the gates $\hat{G}_i$ posterior to $\hat{G}_k$ and classifies them as follows: (1) if all target qubits of $\hat{G}_i$ are unobstructed, then the gate is reordered towards $\hat{G}_k$ to form a larger cluster. (2) In case at least one target qubit of $\hat{G}_i$ is set obstructed, then $\hat{G}_i$ cannot be reordered and all its target qubits are marked as obstructed. The scan from $\hat{G}_k$ continues until all qubits are marked as obstructed or all gates have been scanned. When completed, all reordered gates together with the initial gate $\hat{G}_k$ from a gate cluster. Then the outer loop continues with the next gate not in a cluster and resets all qubits as unobstructed. While this procedure guarantees each gate to be assigned to a cluster, the size of a gate cluster M_C is limited by hardware constraints, potentially causing the inner loop to exit early:

- The amount of shared memory per CUDA block limits the size of the cluster space, i.e. the size of the amplitude subspace the clustered gates can operate on. On the NVIDIA A100 GPU, 164 KB of shared memory are available per SM. Distributed equally among 4 blocks, this allows storing the subspace of up to $N_C = 11$ qubits, yielding $r \leq 11$. In case the inclusion of a gate would yield $r > 11$, the gate cannot be reordered and all of its target qubits must be marked as obstructed. Note that if the algorithm exits and $r < 11$, the simulator fills up the cluster space by the smallest unused $11 - r$ qubits, to ensure that the CUDA kernels are always operating on local copies of 2^{11} amplitudes. This allows for more optimized gate routines in section 1.2.3.3.

- The amount of cached constant memory is set to 8 KB per SM. This memory is mostly used to store the sequence of gates and their auxiliary variables, e.g. target qubits and rotation angles. The algorithm keeps track of these resources required by the gates in a cluster, such that when the addition of a gate would exceed the available memory, the gate is again not reordered and its target qubits are set as obstructed.

- In order to utilize the maximum memory bandwidth when reading/writing the cluster-spaces from/to the global memory, the simulator demands a degree of locality of $c \geq 3$. This means that the first three qubits of the local state vector are always included in the cluster's set of target qubits, even when none of its gates are acting on them. This is because, accessing $2^c = 8$ consecutive complex double-precision floats fully utilizes the cache lane size (128 bytes), such that all global memory requests of the threads in a warp can be processed using only 4 transactions, which are broadcasted among the threads.

By arranging the gates into clusters operating on closed subsets of the amplitude space, each gate cluster is simulated using a separate CUDA kernel call spawning 2^{N_L-r} CUDA blocks (with 4 blocks per SM) for processing the subspaces in parallel. As depicted in Fig. 1.2, in the first stage of the kernel function, the respective cluster-spaces are copied in parallel into the shared memory while keeping the aforementioned coalescing degree of 2^c. This is done, by consecutively assigning the blockID indices $B = [b_{N_L-r-1} \ \dots \ b_0]_2$ to the local non-target bit-positions in Eq. 1.17 and the threadID indices $T = [t_r \ \dots \ t_0]_2$ to the local target bit-positions. For example, in case a gate cluster has a target bit-string $I_M = [0 \ \dots \ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1]_2$ ($r = 11$, $c = 3$), with each 1 denoting a local target qubits of the gates in the cluster, thread T in block B will process amplitude ψ_J, with $J = [b_{N_L-r-1} \ \dots \ b_7\ c_1\ b_6\ c_0\ t_8\ b_5\ t_7\ b_4\ t_6\ t_5\ b_3\ t_4\ b_2\ b_1\ t_3\ b_0\ t_2\ t_1\ t_0]_2$. Note, that the *counting bits* $\{c_i\}$, traversed in a for-loop, have been introduced, as generally the size of the cluster-space 2^{N_C} exceeds the number of threads in a block. The latter is set to $512 = 2^9 \leq 2^{N_C-2}$ for the A100 GPU, limited by the number of available registers and the maximum number of supported threads per SM, yielding 1.0 GPU occupancy. Since each amplitude is given as a complex double-precision float (16 bytes), occupying 4 shared memory banks, the simulator uses a different placement of the data in the shared memory than for the global memory. While in the latter the real and imaginary part (spacing 2 banks each) of the complex numbers are stored sequentially (*array of structs*), allowing to reduce the minimal required degree of coalescing of the gate clusters in turn reducing the number of kernel calls, they are separated in the shared memory into two arrays (*struct of arrays*). In doing so, 4-way bank conflicts are reduced to 2-way bank conflicts, when accessing real and imaginary parts individually during the calculation (see section 1.2.3.3). As a result, the amplitude J from the above example would be stored at the shared memory locations T and $T + 2^{N_C}$, concerning its real and imaginary part, respectively. After the amplitudes have been transferred, the gates of the cluster are computed sequentially on the local copies in the shared memory. Here, each thread is in charge of computing the transformation of 2^2 amplitudes. Note that the updates are carried out in-place and *block-level synchronizations* are required after each gate operation. The latter is necessary, since the gates in a cluster might not act on disjoint qubit sets. Finally, the amplitudes are copied back into the global memory, by reversing the aforementioned process, and the next gate cluster gets simulated using another kernel call. The process is illustrated in Fig. E.8. Note that before executing the kernel function, the properties about the cluster and the circuit (e.g. number of qubits, number of gates), the gate-queue, given as an 8 bit unsigned integer array, and the gate variables, using a double-precision float array, are copied into the constant memory.

A key insight of the above process is that the order of the local qubits, i.e. the arrangement of the local state amplitudes, differs between the shared memory and the global memory. This is because the amplitudes ψ_J, corresponding to the cluster's target-space, are distributed in the shared memory according to the threadIDs. Since the latter are not necessarily assigned to the first 11 bit-positions in J, but can be arbitrarily distributed among the N_L bits, the N_L qubits in the shared memory array must not correspond to the first N_L qubits in the global memory array. In fact, by extracting the cluster-space from the local state-vector, the bit-positions of all target-qubits $\{l_i\}$ are effectively shifted to the 11 right most bits in the amplitude index J, while the remaining non-target qubits $\{p_i\}$ are sifted left to the final $N_L - 11$ bits (see Eq. 1.18), with the former and the latter being indexed by the threadIDs and blockIDs, respectively. Regarding the execution of a gate, this transformation is taken into account during the transfer of the gate variables into the constant memory by altering the respective target bits:

$$\underbrace{J = [\ldots\ p_3\ p_2\ l_2\ p_1\ l_1\ l_0\ p_0]_2}_{\text{Global memory}} \longrightarrow \underbrace{J = [\overbrace{p_{N_L-N_S-1}\ \ldots\ p_0}^{\text{blockID}}\ \overbrace{l_{N_S-1}\ \ldots\ l_0}^{\text{threadID}}]_2}_{\text{Shared memory}}. \tag{1.18}$$

1.2.3.3. Gate implementations

Using the GPU's shared memory to perform the gate operations in-place on local copies of the state vector directly in the fast on-chip cache drastically reduces memory access latencies, hence accelerating the memory-bound simulation. However, this strategy does not change the overall number of memory transactions required for each gate evaluation. Thus, in order to increase the computational intensity further, SEQCS introduces the concept of *gate groups*, referring to the combined execution of multiple single- and two-qubit gates spread across different qubits. Consider for example the application of two Hadamard gates H_0 and H_1 (see Eq. 1.11) on qubits 0 and 1, respectively. Performing the gate operations separately on the N-qubit register, yields 2^N read and write memory request of complex double-precision floats for each gate, as H_0 (H_1) alters the values of 2^{N-1} state amplitude pairs, $\psi_{j_{N-1}\ \ldots\ j_1\ 0/1}$ ($\psi_{j_{N-1}\ \ldots\ j_2\ 0/1\ j_0}$). In doing so, every state amplitude is accessed and modified two times, with a total of 2^{N+1} read/write memory transactions. Combining the gate execution, on the other hand, i.e. considering the tensor-product $H_0 \otimes H_1$, the update is organized into 2^{N-2} tuples of four amplitudes, $\psi_{j_{N-1}\ \ldots\ b_2\ 0/1\ 0/1}$, such that each state amplitude must be accessed only once. This reduces the total number of memory traversals and thus increases the computational intensity by a factor of two, at the expense of an increased register footprint.

Combining the execution of multiple gates is a common technique used in prior quantum circuit simulators, which typically apply one of the following strategies: (1) Clustering adjacent single-qubit gates acting on the same qubit [39]. While delivering good performance on specific kinds of random circuits (e.g. used for demonstrating quantum supremacy [21]), this approach is generally limited by a frequent use of multi-qubit gates, yielding gate-groups of low cardinality in case of real-world quantum circuits (e.g. quantum optimization algorithms). (2) Grouping arbitrary gates operating on a closed subset of up to q qubits [24], by calculating the final tensor-product matrix of size $2^q \times 2^q$ as mentioned in the above example. This ansatz is applicable to real-world quantum circuits, however, generally limited to $q \leq 5$ due to the exponentially growing size of the update matrix, as 2^{2q} matrix elements must be stored in the registers. A key insight of the second approach is that by grouping only gates of the same type, e.g. H gates are grouped separately from T gates, the simulator can exploit the regular structure of the gate matrices, such that the update routines of the amplitudes can be separated into a finite number of instruction branches. Thus, no computationally expensive matrix-matrix multiplications are required and generally arbitrary numbers of gates ($g \leq N$) can be combined. This strategy is motivated by the observation that real-world quantum circuits are often constructed in layers [12], such that the same gate types, e.g. rotation gates, are applied simultaneously to multiple qubits (see e.g. Fig. B.1 in appendix B.1).

With respect to the separation of the quantum circuit into gate clusters, as introduced in section 1.2.3.2, the simulator uses a third preprocessing algorithm to group the execution of gates of the same kind within each cluster. This is done, by utilizing the fact that the order in which diagonal gates as well as non-diagonal gates acting on different qubits are applied does not alter the final quantum state. The algorithm depicted in appendix E.4 is

similar to the one in appendix E.3, featuring two nested loops to iterate through the gates in a cluster. Here, the outer loop selects a gate $\hat{G}_k$ which has not been grouped yet. The inner loop then traverses through the gates $\hat{G}_{j>k}$, trying to reorder all gates of the same type as $\hat{G}_k$, i.e. $type(\hat{G}_k) = type(\hat{G}_i)$, next to it in the queue. This is achieved by designating each qubit as either *unobstructed* (initially), *partially-obstructed* or *obstructed*, with partially-obstructed qubits preventing only non-diagonal gates acting on them to be shifted in the queue, while obstructed qubits prohibit any gate reordering when operating on them. In case the inner loop encounters a gate $\hat{G}_i$, with $type(\hat{G}_k) \neq type(\hat{G}_i)$, all its target qubits are marked as partially-obstructed, if $\hat{G}_i$ is diagonal, or as obstructed, if $\hat{G}_i$ is non-diagonal. On the other hand, if $type(\hat{G}_k) = type(\hat{G}_i)$ and none of the target qubits of the non-diagonal (diagonal) gate $\hat{G}_i$ are obstructed (or partially-obstructed), $\hat{G}_i$ gets reordered to position $k+1$ in the queue and is marked as grouped. Otherwise, its target qubits are also labelled as obstructed (partially-obstructed). The inner loop continues until all qubits have been either scanned or are no longer marked as unobstructed. Once completed, all reordered gates form a new gate-group and the outer loop continues with the next gate not in a group, by resetting all qubits as unobstructed. Applying this algorithm once to the circuit, however, does not produce optimal combined gate-groups. Take for example the circuit: $T_0\ H_1\ T_1\ H_2\ T_2$. Here, the algorithm suggests three groups: $\{T_0\}$, $\{H_1, H_2\}$ and $\{T_1, T_2\}$. Thus, it fails to combine T_0 with T_1 and T_2. To solve this issue, the algorithm is executed twice, first with both nested loops iterating in positive order, followed by a traversal in negative order through the gate-queue. In doing so, the aforementioned circuit becomes: $\{H_1, H_2\}$ and $\{T_0, T_1, T_2\}$. A result of this process is shown in Fig. B.3 in appendix B.1 by the grouping of Hadamard and R_x gates, as well as the combined execution of the diagonal $\hat{U}_C$ transformation.

The process of combining the obtained set of reordered gates into gate-groups of sizes M_G depends largely on the respective gate types and is constrained by the underlying hardware (e.g. number of registers per thread). To achieve maximum performance in the evaluation of every implemented gate (see appendix A), the gates are separated into three classes featuring specialized update kernels each. Moreover, SEQCS has implemented multiple versions of these kernels, which are dedicated to different sizes N_C of the cluster-space. In the following, the general strategies used for the three gate classes are depicted in the context of $N_C = 11$ and 2^9 threads per block.

Single-qubit diagonal gates: This gate-class concerns operations that act diagonally on the state space of a single qubit, performing a rotation of the quantum state around the computational axis in the Bloch sphere (see section 1.1.2). Among the general rotation gate $R_z(\gamma)$, the simulator implements the Z, S and T gates, denoting fixed rotations by π, $\pi/2$ and $\pi/4$, respectively. These gates feature the distinctive property that the state amplitudes ψ_J are altered without mixing or permuting them, allowing to evaluate diagonal gate-groups by traversing the entire state vector only once and processing each amplitude independently. Thus, using any assignment between threads and amplitudes, the contributions of each gate can be aggregated locally, such that diagonal gates acting on any subsets of qubits can be grouped, including global qubits that are spread across multiple GPUs. The simulator uses 64 bit integer-masks I_M to encode the target qubits of gates, with an 1 at bit-position k denoting that the respective gate is acting on qubit k. Regarding diagonal gate-groups, I_M^{Group} is given by combining the individual gate bit masks I_M^i using the bit-wise OR operator |, i.e. $I_M^{Group} = I_M^0 \mid I_M^1 \mid \ldots \mid I_M^{M_G-1}$. Computation wise, I_M is handled as a gate parameter, hence stored in the constant memory, and refers to the qubit arrangement given in the shared memory (see Eq. 1.18). Consequently, the initial 11 bits denote

the cluster-space, the subsequent $N_L - 11$ bits refer to the remaining local qubits distributed among the CUDA blocks, and the final N_G bits correspond to the global qubits distributed among the compute nodes (see Eq. 1.19).

Given an amplitude ψ_J and a gate distribution I_M^{Group}, the update routine of the gate-group can easily be determined using two single-cycle GPU instructions: $l = \_\_\text{popcll}\left(I_M^{Group} \,\&\, J\right)$, with __popcll returning the number of set bits in a 64 bit integer [55]. Using the fact that global phases of quantum states are unobservable, such that each bit in I_M^{Group} contributes either a factor of 1 ($|0\rangle$ state) or a factor of $\exp(i\alpha)$ ($|1\rangle$ state), l denotes the number of gates that act non-trivially on ψ_J, hence $\psi_J \to \psi_J \cdot \exp(i\alpha \cdot l)$. Since $Z^2 = \mathbb{I}$, ($S^4 = \mathbb{I}$, $T^8 = \mathbb{I}$) l can further be taken mod-2 (mod-4, mod-8), i.e. $l \to l \,\&\, 1$ ($l \,\&\, 3$, $l \,\&\, 7$), yielding a closed set of rescaling factors: $\exp(i\alpha \cdot l) \in \{\pm 1\}$ ($\{\pm 1, \pm i\}$, $\left\{\pm 1, \pm i, (\pm 1 \pm i)/\sqrt{2}\right\}$). While for the combined execution of Z gates, ψ_J must only be altered if I_M^{Group} & J has positive parity, 3 and 7 different update routines have to be considered for grouped S and T gates, respectively. In general, this can cause significant warp divergence, as threads of the same warp operate on different control paths. To reduce the impact of diverging instruction trees, all load and store operations are executed unbranched and the calculations are structured, such that expensive float-float multiplications, e.g. $1/\sqrt{2}$ for multiple T gates, are done simultaneously by all concerned threads. Regarding the $R_z(\gamma)$ gate, where the simulation allows different rotation angles γ throughout the group, no closed set of rescaling factors can be determined at compile time, hence the individual rotation angles must be loaded and aggregated from constant memory. Here, the simulation uses the identities $\sin(x+y) = \sin(x)\cos(y) + \cos(x)\sin(y)$ and $\cos(x+y) = \cos(x)\cos(y) - \sin(x)\sin(y)$, in order to shift the calculation of expensive trigonometric functions to the CPU. Hence, by storing $\cos(\gamma/2)$ and $\sin(\gamma/2)$ in the constant memory and using fast-multiply-add GPU instructions [55], the aggregated effect of multiple R_z gates can be evaluated efficiently.

Despite the computation of the altered amplitudes, the challenge in simulating diagonal single-qubit gates lies in the assignment of the threads to the amplitudes as well as in the order in which the latter are computed. Take for example a group of Z gate with:

$$I_M^{Group} = [\ \underbrace{1\ \ldots\ 0}_{\text{Global qubits}}\ \ \underbrace{0\ \ldots\ 1}_{\text{Block qubits}}\ \ \underbrace{0\,0\,1\,0\,0\,1\,0\,1\,1\,0\,1}_{\text{Cluster qubits}}\ ]_2, \tag{1.19}$$

$$J = [r_{N_G-1}\ \ldots\ r_0\ b_{N_L-N_S-1}\ \ldots\ b_0\ c_1\ c_0\ t_8\ t_7\ t_6\ t_5\ t_4\ t_3\ t_2\ t_1\ t_0]_2 \tag{1.20}$$

$$= [r_{N_G-1}\ \ldots\ r_0\ b_{N_L-N_S-1}\ \ldots\ b_0\ t_2\ t_0\ t_8\ t_7\ t_6\ t_5\ t_4\ t_3\ c_1\ t_1\ c_0]_2\,. \tag{1.21}$$

Equation 1.20 depicts a trivial assignment of the threads to the qubits, by distributing the threadID bit string T consecutively among the first 9 bit positions. Here, $R = [r_{N_G-1} \ldots r_0]_2$ denotes the MPI rank, and c_i refer to additional counting bits that are traversed in the *inverse-gray code* order, i.e $[c_0\, c_1]_2 = [00]_2 \to [01]_2 \to [11]_2 \to [10]_2$. A major issue of the above assignment of threads and amplitudes is that it causes half of all threads in a warp to idle during the evaluation of the gate-group, since a non-trivial update requires positive parity of I_M^{Group} & J. For a given configuration $[c_0\, c_1]_2$, the latter is only fulfilled by half of the threads in a warp, hence reducing the effective occupancy by a factor of two. To improve the *load-balance*, the simulator proposes a custom assignment, in order to synchronize the computations:

- As a first step, the threadID indices get reordered, such that both counting bits are assigned to target bits in I_M^{Group}. This is done, by swapping c_0 and c_1 with the threadID indices that occupy the positions of the first two target qubits (here t_0 and t_2), yielding Eq. 1.21.

- The above assignment ensures improved load balance, with all threads performing two updates. However, in case the counting bits are now located within the first 5 bits (denoting the shared memory bank of ψ_J), traversing the counting bits in identical sequences among all threads will yield a 4-way bank conflict, when accessing the state amplitudes in the shared memory. To solve this problem, the threads will start counting from different positions in the inverse-gray code, by initializing c_0 and c_1 with t_0 and t_2, respectively.

- Concerning the Z gate, an additional optimization is used to synchronize the calculations among the threads within a warp. This is motivated by the fact, that for each configuration of counting bits, always one half of the threads in a warp will execute an update, while the other half will idle, with both halves alternating in the parity of $[c_0\, c_1]_2$. To synchronize the execution of the threads, the simulation determines the first target qubit in I_M^{Group} beyond the first five qubits, and sets its bit in J to 1 (0) if c_0 (c_1) is initialized to 0 (1). This effectively corresponds to an exchange of threads between pairs of warps, such that always all threads will either perform an update or not, given a configuration of $[c_0\, c_1]_2$.

Single-qubit non-diagonal gates: Non-diagonal gates operating on a single-qubit Hilbert space denote rotations of the state vector around an axis that is tilted from the computational axis, yielding mixing and permutation of the computational basis states. Among others, the simulation implements the Hadamard H, the $R_x(\gamma)$ and the $R_y(\gamma)$ gates. In contrast to the aforementioned diagonal gates, the combined execution of non-diagonal gates is limited to pairs. On the one hand, this is due to the amount of parallelism in the kernels. Fusing q non-diagonal gates reduces the number of independent target-spaces, distributed among the threads, by a factor of 2^q. Since for the A100 GPU, the shared memory is eight times larger than the proposed block size, combining more than three non-diagonal gates reduces the GPU's occupancy and thus the ability to hide memory latencies, as the number of target-spaces residing on each SM undercuts the number of available threads. On the other hand, the computation of q grouped gates in general requires 2^{2q} matrix elements and thus $2q$ complex amplitudes to be stored in the registers. Since the register count per thread is set to 32, with each complex double-precision amplitude occupying 4 registers, q is limited to 2 considering that registers are also needed to store intermediate results of the calculations. Note that *register spilling* into global memory occurs for $q \geq 3$.

Analogously to the simulation of diagonal gates, bit masks I_M^{Group} are used to encode the target qubits of the pair executions. Here, 32 bit integers are sufficient to store I_M^{Group}, as non-diagonal gates can only be applied to local qubits with $N_L < 32$. Moreover, the threadID indices are distributed similarly to Eq. 1.21 among the qubits, to ensure synchronized calculations and to prevent shared memory bank-conflicts. For example, a pair of Hadamard gates applied to the first two qubits, yields $I_M^{Group} = [\,\ldots\ 0\ 0\ 1\ 1]_2$ and $J = [\,\ldots\ t_1\ t_0\ c_1\ c_0]_2$. As mentioned before, by initializing the counting bits c_0 and c_1 at different positions in the inverse-gray code based on the state of t_0 and t_1, the threads will access the four state amplitudes belonging to the same target-space in a different order. While for diagonal gates

these amplitudes can be processed independently, regarding non-diagonal gates the computation of each amplitude generally requires all 2^{N_G} amplitudes of the target-space. To prevent branch divergences during the expensive memory transactions, SEQCS stores these amplitudes among the variables V_{kl} in the order in which they are accessed. As a consequence, the threads are separated into four groups, with each group featuring a different permutation of the amplitudes ψ_{ij} among the variables V_{kl}. In order to compute the output amplitudes in parallel, a naive strategy could be to reorder the data among the variables. Afterward, each thread must first compute all four altered amplitudes, store them in the registers and finally write them back into the shared memory in the order in which they were initially loaded. Again, the latter is done to prevent bank-conflicts. Unfortunately, however, this approach is prevented by the number of available registers per thread, yielding either register spilling or a sequential processing of the four thread groups, hence significantly increasing the simulation time. To solve this issue, SEQCS proposes an *out-of-order* computation scheme, by utilizing similarities in the calculations of ψ_{ij} and their permutations among the variables V_{kl}. In doing so, carefully placed negations of some variables are sufficient to direct the different operations, causing only neglectable branch divergences, while preventing bank-conflicts. A detailed description of this strategy can be found in appendix E.5. Note that due to the small number of registers, the real and imaginary parts of the amplitudes are processed seperately.

An exception from the above depicted process is given for the X and Y gate, which feature a special symmetry along the off-diagonal axis, such that gate executions acting on up to $N_S - 1$ qubits can be combined. This is because, the involved update routines operate exclusively on two-dimensional subsets of the state space, since both gates perform permutations of the amplitudes in the state vector, e.g. applying two X gates on qubits 0 and 1, swaps the memory locations of $\psi_{i,k} \leftrightarrow \psi_{\sim i,\sim k}$. Similarly to other non-diagonal gates, Eq. 1.21 is used to distribute the threads among the qubits, yielding again an out-of-order computation scheme. While for the X gate, only the respective amplitudes need to be swapped, the Y gate additionally requests a phase shift, i.e. $\psi_J \to e^{i\alpha_J} \cdot \psi_J$. Given the original amplitude index J, α_J can be computed via: $\alpha_J = \pi/2 \cdot [\_\_\_\text{popcll}(M_G)\ \&\ 7 + 2 * (\_\_\_\text{popcll}(M_G\ \&\ J)\ \&\ 1)]$.

Two-qubit control gates: In order to achieve universal quantum computation, the simulator also implements a series of two-qubit control gates, including the CZ and the $CNOT$ gates (see Eq. 1.12) as well as a controlled rotation gate $U(k)$. These gates act on a four dimensional subspace of the state space, with the control qubit conditioning the application of a single-qubit gate on the target qubit. While a combined evaluation of controlled gates acting on disjoint qubit pairs could be realized similarly to the aforementioned single-qubit gates, a key insight in the usage of these gate class in real-world quantum algorithms is that adjacent controlled gates typically operate on non-disjoint sets of qubits. For example, the quantum Fourier transformation applies several $U(k)$ gates to the same qubit conditioned on the computational state of the remaining qubits [31]. As a consequence, SEQCS proposes a new encoding strategy, which allows combining controlled-operations both vertically and horizontally in the circuit, such that the aforementioned $U(k)$-clusters can be simulated via a single gate-group. In contrast to the previous encodings, the size of the gate-groups is not limited, with the only restriction being that the sets of control and target qubits must be disjoint, and the latter are residing in the shared memory in case the conditioned single qubit gates are non-diagonal.

Dealing with large sets of controlled gates, the encoding uses (up to N) 64 bit integer masks I_M^k capturing the effects of multiple gates. The bit masks are assigned to each control qubit k,

such that each set bit in I_M^k denotes a target qubit that is controlled by the computational state of qubit k. Note that in case the controlled gate is diagonal (e.g. CZ gate) a bit mask is assigned to both the target and control qubit, as they are interchangeably. As an example, consider the gate-group $\{CNOT_{0,3}, CNOT_{1,3}, CNOT_{0,4}, CNOT_{2,4}\}$, yielding the three bit masks: $I_M^0 = [1\ 1\ 0\ 0\ 0]_2$, $I_M^1 = [0\ 1\ 0\ 0\ 0]_2$ and $I_M^2 = [1\ 0\ 0\ 0\ 0]_2$. Using Eq. 1.21 to distribute the threads among the qubits, by counting in the first two controlled bits to improve load balance and prevent bank conflicts, the permutation of an amplitude ψ_J can be determined by iterating through all set bits k in J and aggregating the transformations $J' \rightarrow J' \,\&\, I_M^k$ (with $J' = J$ initially), in case qubit k is a control qubit. Finally, ψ_J and $\psi_{J'}$ must be swapped. A main problem of this procedure is that there is not enough register space available to store the full set of bit masks. Thus, each time an amplitude ψ_J is processed, all bit masks must be loaded from constant memory again, causing significant performance penalties in case of numerous control qubits. Fortunately, however, the process can be optimized by exploiting the reflected-gray code used to traverse the counting bits c_0 and c_1, causing only a single bit in J to flip, when going from one computed amplitude to the next. This allows reusing and incrementally updating the state J' from the previous amplitude by loading only a single bit mask each time, hence reducing the complexity of amplitude updates from $\mathcal{O}(N)$ to $\mathcal{O}(1)$ memory accesses after initialization. This becomes especially important in the context of controlled rotation gates, as it significantly decreases the number of memory transactions to the individual rotation angles.

1.3. Benchmarks

In section 1.2 the implementation of the quantum circuit simulator SEQCS has been discussed in detail. With respect to the various performance bottlenecks of GPU-based executions, novel strategies have been proposed to increase the load-balance and improve the memory management of the simulation in order to reduce memory access latencies and increase the overall GPU utilization (both shown to limit the performance of prior quantum circuit simulators). To verify the effectiveness of the proposed ideas, this section presents benchmark results regarding the MPI-communication scheme, the usage of shared memory and the implementation of the individual gate kernels. In doing so, SEQCS will be compared against the *Jülich universal quantum computer simulator* (*JUQCS*), a massively parallel simulator written in *Fortran*, that has been used to study quantum algorithms, such as quantum annealing and the QAOA, to demonstrate Google's quantum supremacy, as well as to benchmark internode communication networks in the past [27, 38, 63]. Recently, a GPU-accelerated version of JUQCS, termed *JUQCS-G* [30], has been presented, using CUDA and CUDA-aware MPI to achieve an up to 49 times faster execution of the quantum circuits investigated by *Willsch et al.*. Note that JUQCS-G does not implement any circuit analysis/optimization strategies, e.g. gate reordering and combined gate execution, nor makes use of the shared memory. Hence, it allows ranking the success of the proposed optimization strategies.

All experiments are conducted on the JUWELS Booster supercomputer [29], using between 1 and 32 NVIDIA A100 GPUs to simulate circuits of up to 36 qubits. In doing so, section 1.3.1 concerns the MPI-communication scheme, section 1.3.2 focusses on the use of shared memory within the kernels, and section 1.3.3 discusses the effects of combined gate executions. These experiments are conducted on *benchmark circuits*, composed of l layers of single qubit gates, with each layer executing one gate on each of the N qubits. Using multiple gate repetitions, i.e. $l \gg 1$, makes potential initialization times of the GPU/CUDA

and MPI negligible. Regarding both simulators, SEQCS and JUQCS-G, the elapsed times of the MPI communication T_{MPI} and the gate execution T_{Gates} are tracked separately. While these investigations only involve artificial circuits, specially tuned to study one aspect of the simulation, in section 1.3.4 both simulators are also applied to real-world quantum algorithms, like the quantum Fourier transformation and the QAOA (see section 2.2.2). Note that throughout all experiments, gate substitution strategies, e.g. replacing pairs of unitary hermitian gates acting on the same qubit by an identity operation (a technique generally used by SEQCS), are deactivated.

1.3.1. MPI-communication scheme

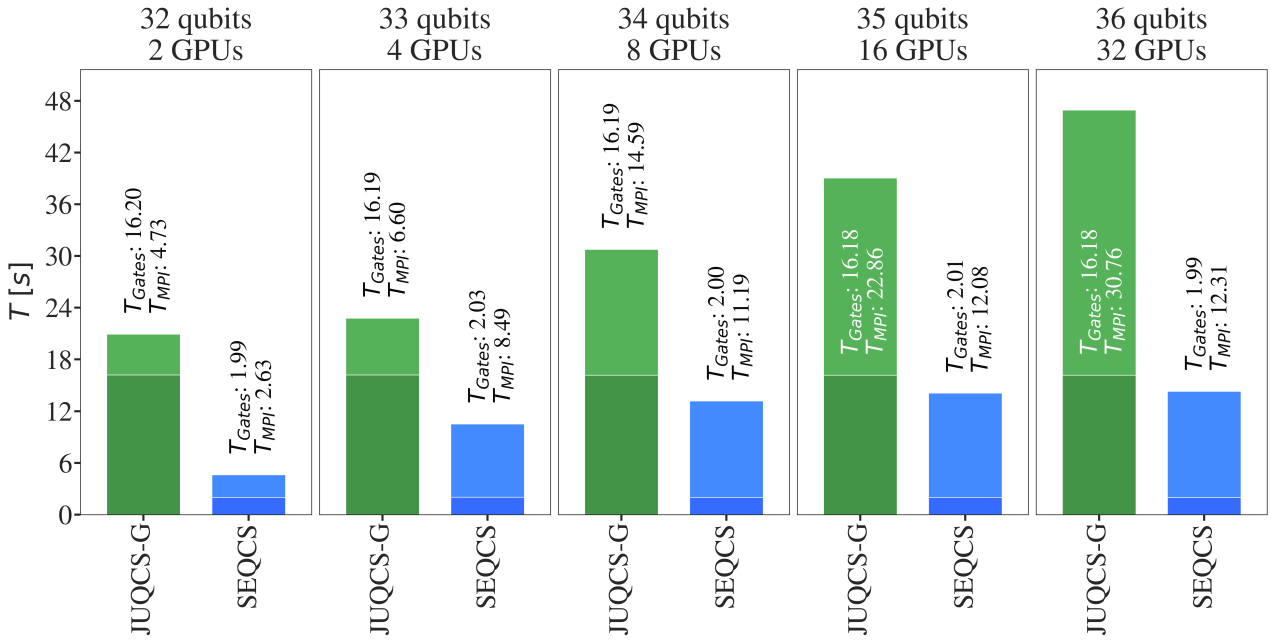


Figure 1.3.: The figure depicts the total MPI communication time T_{MPI} and the total gate execution time T_{Gates} obtained by SEQCS and JUQCS-G for the Hadamard benchmark circuit ($l = 20$, $N \in \{32, 33, 34, 35, 36\}$). The data shown is normalized with respect to the 32 qubit circuit, by multiplying T_{MPI} and T_{Gates} with $32 / N$.

The investigation of the internode communication, i.e. the execution speed of global gate operations, is conducted on Hadamard benchmark circuits using $l = 20$ layers acting on qubit registers of sizes $N \in \{32, 33, 34, 35, 36\}$. The simulations involve $2, 4, 8, 16$ and 32 GPUs, with each A100 GPU operating on a local state vector of 31 qubits, respectively. Since the Hadamard gate is a non-diagonal single qubit gate, both simulators must perform global-local qubit exchanges in order to compute the gate operations on either of the $N - 31$ global qubits. The reason for including multiple layers ($l \gg 1$) and also applying gates to local qubits is to model a realistic situation, where gates can act on both qubits in an exchanged pair. As a consequence, the scheduling order of the gates has a signifiant impact on the overall number of qubit transfers and hence the simulation time. Note that in case the gates are only applied to the initial global qubits, arbitrary numbers of layers can be computed using only $N - 31$ global-local qubit exchanges.

Figure 1.3 depicts the simulation times T_{MPI} and T_{Gates} obtained for the five benchmark circuits. Note that the presented data is normalized with respect to the 32 qubit circuit, taking into account that the number of executed gates increases linearly in N. Regarding both simulators, the figure reveals a dependency of the MPI communication time on the number of involved GPUs, while, on the other hand, the gate execution time remains constant throughout the runs with elapsed times of approximately 2.0 s (SEQCS) and 16.2 s (JUQCS-G). These observations fit to the results obtained by *Willsch et al.* [30] and can be explained by the distribution of the computational work among the processing threads. Thus, as the number of global qubits increases, so does the fraction of gates in the circuit that demand a global-local qubit exchange during their evaluation. Therefore, an increasing number of global gates causes more data to be transferred through the network (despite the normalization), yielding a dependency of T_{MPI} on N_G. Note that T_{MPI} is additionally affected by the increasing number of MPI synchronization barriers required after each qubit exchange, which grows proportional to N_G. In contrast to this, T_{Gates} remains constant for both simulators, since the exponentially increasing number of state amplitudes (2^N), which must be altered during the execution of each gate, is distributed equally among the also exponentially growing number of GPUs. As a consequence, the computational work of each GPU remains independent of the number of global qubits in the circuit, and so does T_{Gates}, yielding an ideal weak scaling.

Comparing the execution times of the two simulators, SEQCS performs both the simulation of the gate operations and the MPI communication significantly faster than JUQCS-G, with speed-ups of ≈ 8.5 for the former and speed-ups ranging between ≈ 1.8 and ≈ 2.5 for the latter. A detailed investigation concerning T_{Gates} will be given in the sections 1.3.2 and 1.3.3. Focussing on T_{MPI} in the following, the faster execution of SEQCS can be traced back to the order in which the gates in the clusters are scheduled for evaluation. While JUQCS-G computes the circuits gate-wise, yielding one global-local qubit exchange for each global gate in each layer, i.e. $(N-31)\cdot l$ exchanges in total, SEQCS uses the preprocessing algorithm depicted in appendix E.1 to rearrange the gate operations and minimize the amount of MPI communication necessary. In doing so, SEQCS processes the Hadamard benchmark qubit-wise, i.e. simulating all l gates acting on one qubit consecutively before continuing with the next qubit. Consequently, global gates are executed only after all local gates have been processed, such that $N-31$ global-local qubit exchanges are sufficient. Although in real-world quantum circuits it is often not possible to process all local and global gates separately as done by SEQCS here, this experiment demonstrates the importance of circuit restructuring regarding the MPI communication time.

In order to exclude the influence of the gate scheduling order from the comparison, Fig. 1.4 depicts T_{MPI} (not normalized) obtained from the first layers of the benchmark circuits only, This allows investigating the differences in the implementations of the individual global-local qubit exchanges. Considering the case of a single qubit exchange, JUQCS-G manages a ≈ 10 times faster transfer of the state amplitudes then SEQCS. As both simulators communicate the same number of bytes (2^{N-1} amplitudes) between pairs of GPUs in this case, this observation reveals that JUQCS-G features a significantly faster MPI implementation then SEQCS. Most likely this is caused by the extensive use of special MPI data types by SEQCS, such as *MPI_Type_vector* and *MPI_Type_indexed*, which seem to be not optimized for the exchange of large memory segments. Moreover, JUQCS-G uses self-managed buffers for the data transfers, while SEQCS leaves this job to the MPI, which might cause expensive Device-Host memory transfers. However, further investigations are needed here. Despite the slower byte-transfers, SEQCS shows a significantly better scaling of T_{MPI} in the number

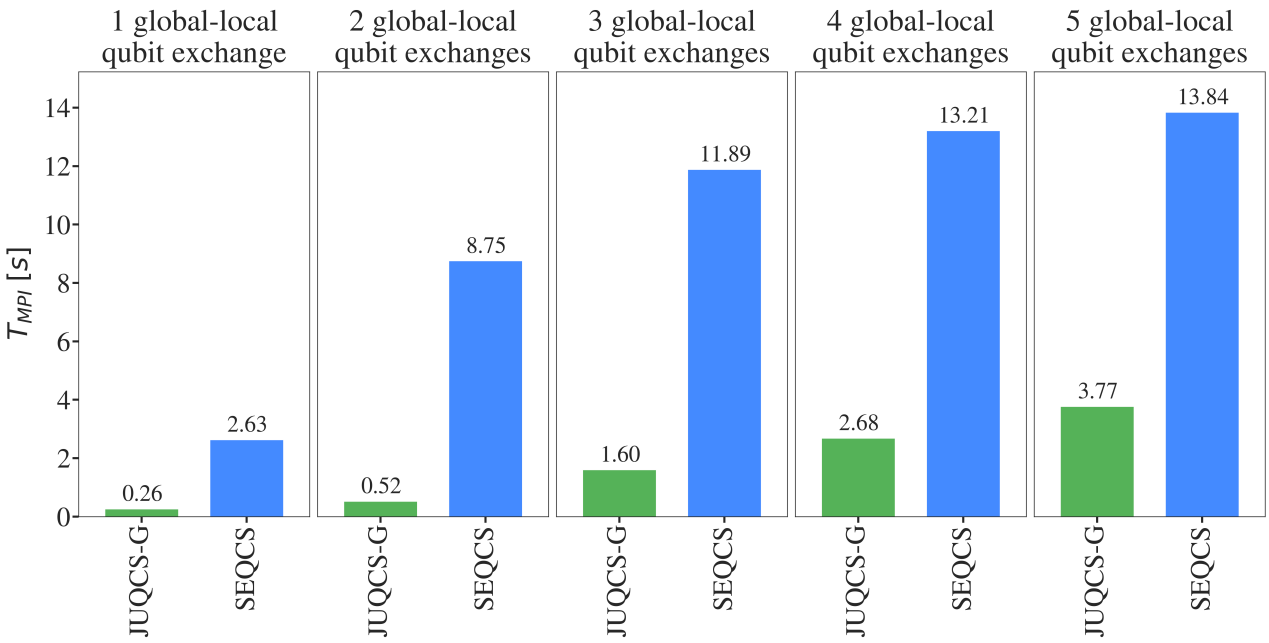


Figure 1.4.: The figure depicts the MPI communication time for 1 to 5 consecutive global-local qubit exchanges obtained by SEQCS and JUQCS-G. The data corresponds to T_{MPI} when executing only a single layer of the Hadamard benchmarks.

of consecutive global-local qubit exchanges than JUQCS-G. Here, the latter scales linearly, taking approximately 260 ms for an intranode exchange, i.e. pairs of communicating GPUs reside on the same compute node (see 1 and 2 qubit exchanges), via CUDA asynchronous copy and 1080 ms for an internode exchange using CUDA-aware MPI, yielding an ideal weak scaling. SEQCS, on the other hand, is able to combine these qubit transfers, hence reducing the amount of memory that must be communicated through the network. As a result, T_{MPI} seems to approach a saturation elapsed time of approximately 14 s, such that the speed-up of JUQCS-G monotonically shrinks as more exchanges can be executed simultaneously. Although, JUQCS-G implements a similar strategy of joining MPI calls, it is only applied to certain measurement tasks, where the communication pattern can be determined at compile-time. The conducted experiments, however, demonstrate that analysing the quantum circuit and rearranging the execution order to combine qubit transfers can significantly improve the simulation times by reducing the amount of MPI communication.

1.3.2. Shared memory

In section 1.2.3.2 the use of shared-memory, i.e. user-managed L1-cache, in the kernel executions was proposed in order to reduce memory access latencies to the individual state amplitudes. The influence of this strategy onto the average execution time per quantum gate $T_{Gate} = T_{Gates} \,/\, (N \cdot l)$ is depicted in Fig. 1.5 by applying both simulators to a $l = 128$ layered Hadamard benchmark circuit. Here, $N = 31$ is chosen, such that the full circuit can be evaluated using a single GPU, hence excluding the effects of the MPI communication seen in section 1.3.1. Regarding SEQCS, the combined execution of H-pairs is deactivated, resulting in similar update routines of the state amplitudes as in JUQCS-G. Moreover, the gate-cluster size, i.e. the number of gates that are computed within a single kernel call in the shared memory, is artificially restricted to $1, 2, 4, 8, 16, 32$ and 128 gates, with the lat-

ter corresponding to the maximum cluster-size limited by the amount of constant memory available on the SMs. Note that the total number of gates ($N \cdot l$) can be evenly divided by all investigated cluster-sizes.

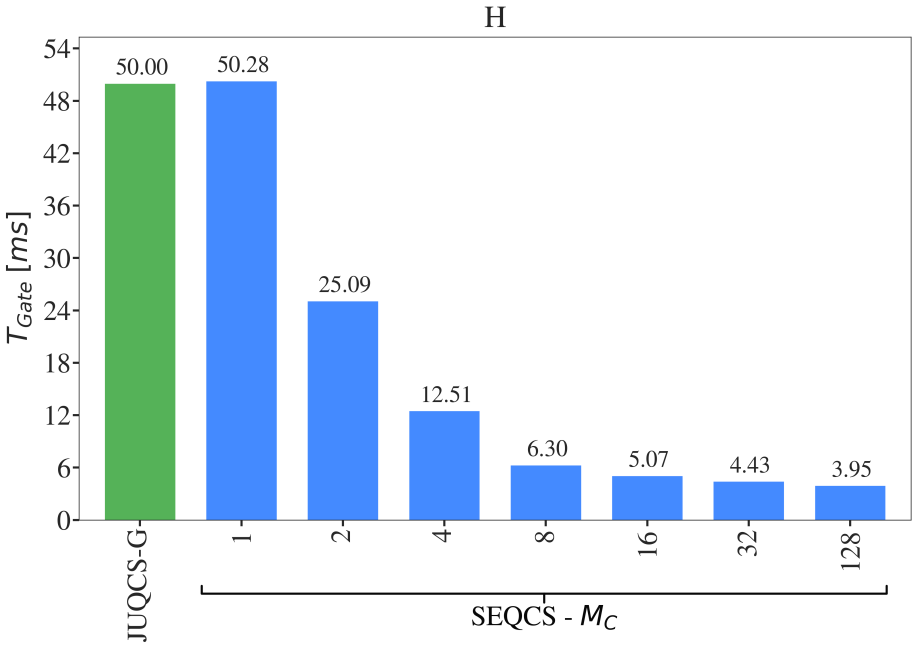


Figure 1.5.: The figure depicts the average gate execution time $T_{Gate} = T_{Gates} \;/\; (N \cdot l)$ obtained by SEQCS and JUQCS-G for the Hadamard benchmark circuit ($l = 128,\ N = 31$). Regarding SEQCS, the gate-group size is artificially set to 1 and the cluster size is limited to $\{1, 2, 4, 8, 16, 32, 128\}$ gates. The standard deviation of T_{Gate} is 0.62 ms (JUQCS-G) and ≤ 0.01 ms (SEQCS).

Using JUQCS-G the Hadamard circuit is simulated with an average execution time per gate of $T_{Gate} = 50.00$ ms. SEQCS achieves a similar time of $T_{Gate} = 50.28$ ms if the gate-cluster size is artificially limited to a single gate. In this case, each Hadamard gate is simulated via a separate kernel call, yielding a similar update strategy as implemented in JUQCS-G. However, this prohibits the reuse of previously loaded state amplitudes residing in the shared memory, since each amplitude is traversed only once in each kernel call. Therefore, extracting the cluster-space from global memory into the shared memory for operation introduces overhead into the computation, as in addition to a single load and store instruction into the global memory, SEQCS requires each thread to additionally perform one read and two write requests into the shared memory. Although this could potentially increase the simulation time, Fig. 1.5 demonstrates that the warp scheduler is able to hide these additional shared memory accesses, which are approximately one order of magnitude faster than global memory transactions, by performing them during the global memory accesses of other threads. This is possible, since the performance of the update kernel is generally restricted by the latter. Moreover, recent *Ampere* GPUs support direct transfers between the global memory and the shared memory without involving registers [54], such that the number of write operations in the shared memory is reduced to one by each thread. As a result, no significant difference in T_{Gate} is found between the simulators, demonstrating that even in the most unfavourable case the overhead associated with the use of shared memory does not cause a significant performance penalty.

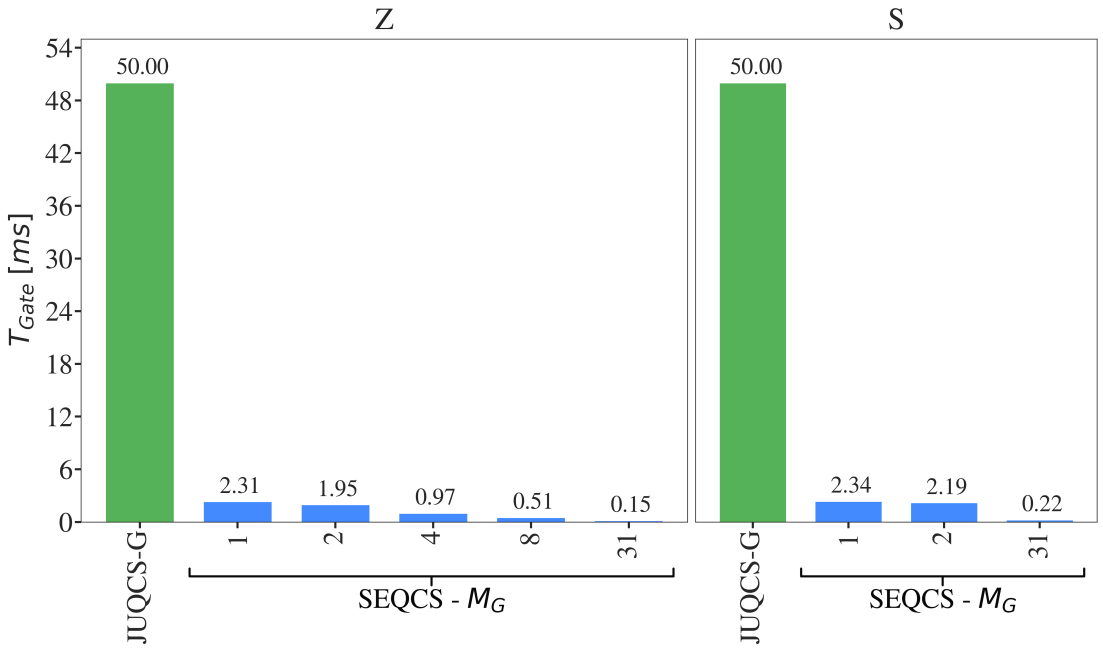
Next, focussing on gate clusters featuring 2, 4 and 8 gates, yielding average execution times of 25.09 ms, 12.51 ms and 6.30 ms, respectively, a close-to ideal strong scaling of T_{Gate} is obtained, such that $\bar{T}_{Gate}$ decreases inversely proportional to the cluster size M_C. This can be explained by state amplitude caching, such that by doubling the number of gates in a cluster, half of the prior global memory transactions are now performed in the shared memory. Since this decrease in the simulation time is entirely based on the differences in the access latencies between the two memory spaces, increasing the cluster-size beyond 8 gates reveals a saturation of T_{Gate} at approximately 4 ms. This is because, the number of global memory transactions per cluster remains constant, while the number of shared memory accesses grows linearly in the number of included gates. Thus, there exists a cluster size, from which on the warp scheduler is no longer able to hide the faster shared memory requests during the slower global memory accesses. If this is the case, the kernel becomes limited by the access speeds of the shared memory, such that an increase of N_C, as long as the total number of kernel calls is not altered, does not improve the simulation performance. This means, that already for 8 gates in each cluster a speed-up compared to the single gate execution of ≈ 8 is achieved, which is close to the maximum speed-up of ≈ 12.7 ($M_C = 128$). Hence, this experiment demonstrates the success of self-managed amplitude caching in the shared memory also for small and heavily connected circuits, allowing only small gate clusters.

1.3.3. Combined gate execution

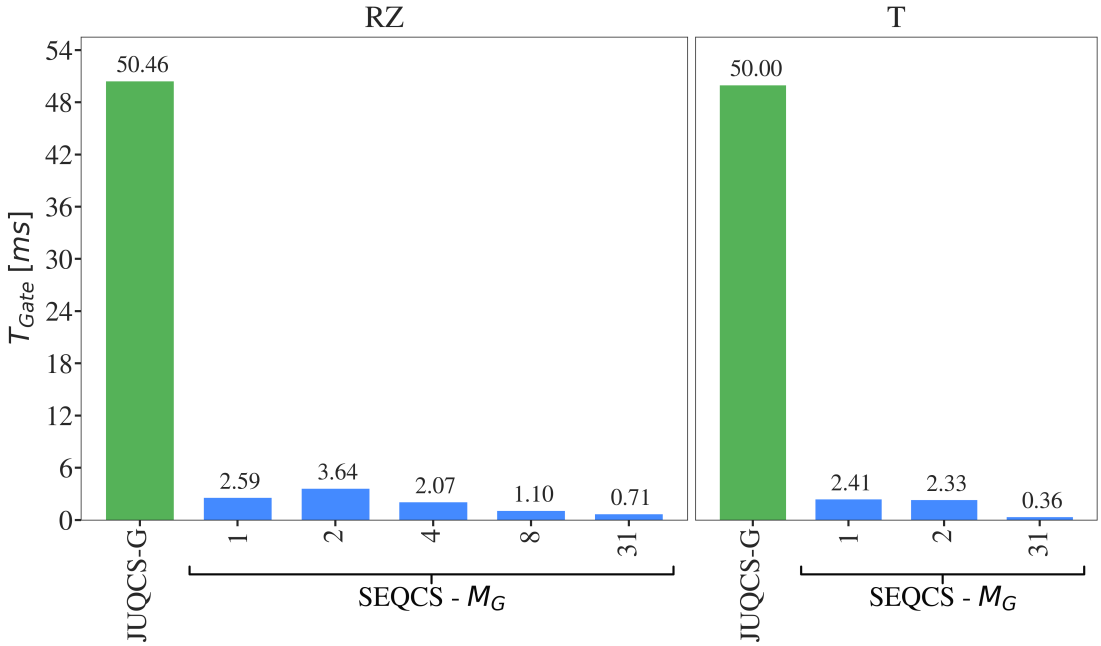
1.3.3.1. Single qubit diagonal gates

Continuing with the analysis of combined gate executions, both simulators are applied to $N = 31$ qubit benchmark circuits composed of $l = 24$ layers of Z, S and T gates. In addition to these fixed rotation gates, the $R_Z(\gamma)$ gate, supporting arbitrary rotations around the computational axis by γ, is also benchmarked in order to study the effects of repeated constant memory accesses on T_{Gate}. Note that $\gamma = \pi/4$ is chosen, hence each R_Z gate simulates the effects of a T gate. In doing so, Fig. 1.6a and Fig. 1.6b depict the average execution times per gate T_{Gate}. Concerning SEQCS, the simulations are conducted with artificial limitations to the gate-group size set to 1, 2, 4, 8 and N gates.

Focussing on the fixed rotation gates first, JUQCS-G takes on average approximately $T_{Gate} = 50.00$ ms for the simulation of each gate, with no significant difference between the three gate types (Z, S and T gate). Relating this to the computation of a Hadamard gate in section 1.3.2, $T_{Gate} \approx 50.00$ ms seems to resemble the average execution time of any quantum gate in JUQCS-G (see also section 1.3.3.2). Since the performance of the respective update kernels is generally limited by the amount of global memory accesses involved, this observation suggests that JUQCS-G fails to exploit the favourable structure of the diagonal update matrices in question here. As quantum mechanics prohibits the observation of global phase factors, the evaluation of the Z, S and T gates can be conducted, such that non-trivial updates must only be performed on computational basis states $|1\rangle$, hence reducing the number of memory transactions by a factor of two. Consequently, one would expect two times faster gate executions, as can be seen for SEQCS in case of $N_G = 1$, i.e. gate-groups of size one. Here, the average execution time lies between 2.31 ms and 2.41 ms, with the Z gate being simulated the fastest and the T taking the longest computation time. Thus, the simulation of these diagonal gates is executed approximately twice as fast as the simulation of the non-diagonal Hadamard gate in section 1.3.2 using $M_C = 31$. Hence, by neglecting global phase factors and caching the state amplitudes, SEQCS performs the single-gate evaluation ≈ 21 times faster than JUQCS-G.



(a) Z and S benchmarks



(b) R_Z and T benchmarks

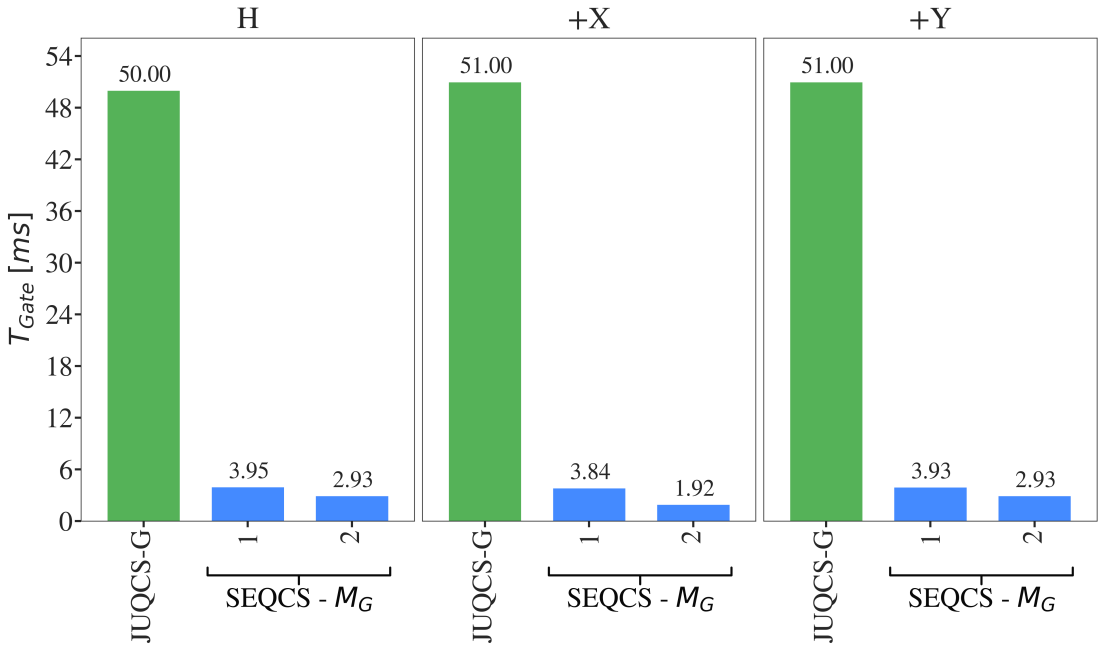
Figure 1.6.: The figures depict the average gate execution time $T_{Gate} = T_{Gates} \;/\; (N \cdot l)$ obtained by SEQCS and JUQCS-G for the Z, S, T and R_Z benchmark circuits ($l = 24$, $N = 31$). Regarding SEQCS, the gate group size is limited to $\{1, 2, 4, 8, N\}$. The standard deviation of T_{Gate} remains consistent for all gates with ≤ 0.01 ms (JUQCS-G) and ≤ 0.01 ms (SEQCS)

Continuing with the combined execution of these gates, SEQCS takes 1.95 ms (Z), 2.19 ms (S) and 2.33 ms (T) for the simulation of gate-groups of size two. The fact that the speed-up with respect to the single-gate execution is significantly below two, indicates that the kernels used for the combined gate executions involve additional computational overhead, e.g. due to the calculation of the thread to amplitude assignment and the aggregation of the effects of multiple gates altering one amplitude. However, since this overhead is independent of the group size, combining the execution of 4 and 8 gates shows again the expected close-to ideal strong scaling, with T_{Gate} decreasing inversely proportional to M_G. Going beyond 8 gates, T_{Gate} saturates at 0.15 ms (Z), 0.22 ms (S) and 0.36 ms (T), yielding maximal speed-ups compared to the single gate executions of SEQCS (JUQCS-G) of approximately 15 (329), 10 (228), and 6 (138), respectively. A key insight into these saturations is that a Z gate can generally be simulated faster than an S gate, which in turn can be computed faster than a T gate. This indicates that, although the respective update kernels are memory-bound, the implemented improvements to the memory management and the memory access patterns significantly increase the computational intensity, such that the amount of arithmetic operations and the degree of branch divergence impact the simulation time. With respect to the S and T gate, the latter plays an important role, as the combined execution of these gates results in 3 and 7 distinct instruction paths, respectively. Since these instruction paths must be executed sequentially within each warp, the obtained performance differences between the gates occur. However, comparing the average execution times of the S and T gate to the Z gate, the decrease in the performance (factors of 1.44 and 2.38, respectively) is significantly lower than the degree of branch divergence. This indicates the success of the proposed strategies to reduce branching, e.g. by executing expensive memory transactions unbranched.

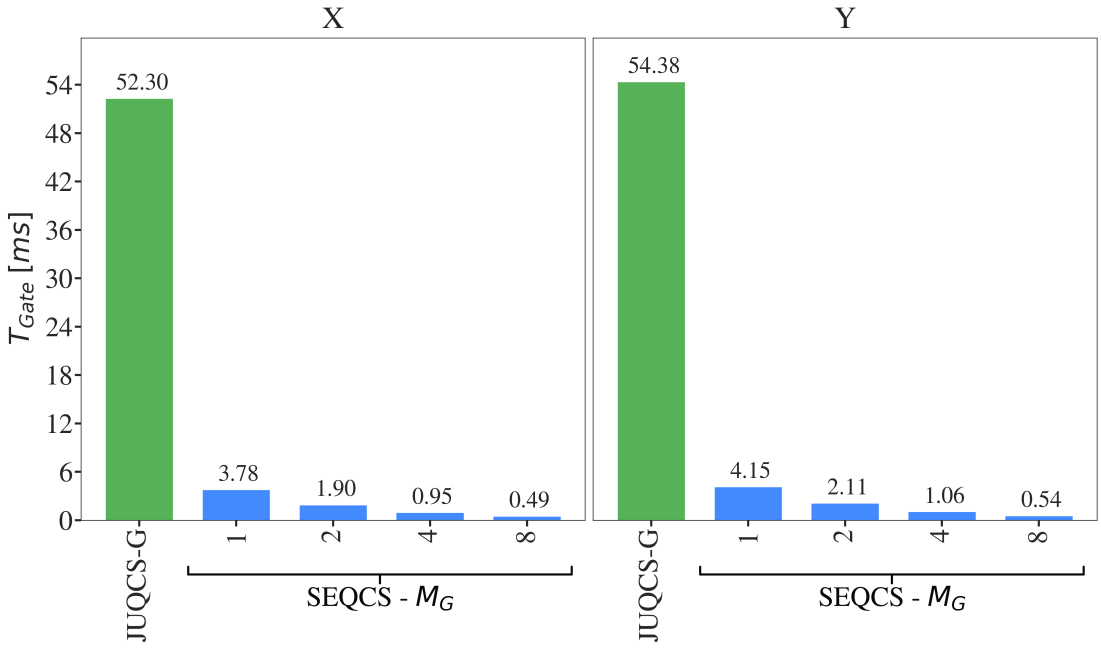
Shifting the focus to the $R_z(\pi/4)$ gate, an average simulation time of 50.46 ms by JUQCS-G is found, fitting to the execution times of the aforementioned gates. With $T_{Gate} = 2.59$ ms (SEQCS), the execution does also take only slightly longer than the simulation of a T gate in the case of $N_G = 1$. The difference can be explained by the two additional read requests to the constant memory in order to obtain $\cos(\pi/4)$ and $\sin(\pi/4)$. Considering combined executions of R_z gates, these memory accesses seem to result in a significant computational overhead, such that when simulating 2 gates simultaneously an increase in the execution time per gate by a factor of ≈ 1.4 is found. Most probably this is caused by the fact that the update kernel used for the grouped gate evaluation demands two constant memory accesses per amplitude, while the single gate execution kernel on average uses only a single constant memory request per amplitude, as it has enough registers available to intermediately store the trigonometric values mentioned above. Consequently, Fig. 1.6b shows that combining R_z gates only becomes beneficial for gate-groups of at least 4 gates, giving a maximum speed-up compared to the single gate execution (JUQCS-G) of approximately 3.6 (70.6). Hence, for intermediate numbers of R_Z gates, it is generally beneficial to replace them by Z, S and T gates or evaluate them separately. Note that a similar behaviour can also be found for the R_x and R_y rotation gates.

1.3.3.2. Single qubit non-diagonal gates

The investigation of single-qubit non-diagonal gates is split into two sets, with the first set considering the $+X$, $+Y$ and Hadamard gate, hence studying the proposed out-of-order calculation scheme of the state amplitudes in a warp, and the second set focussing on the X and Y gate, where SEQCS exploits the off-diagonal symmetry of the gates. All experiments are conducted analogously to section 1.3.3.1, using $l = 24$ and $N = 31$ qubits.



(a) H, $+X$ and $+Y$ benchmarks



(b) X and Y benchmarks

Figure 1.7.: The figure depicts the average gate execution time $T_{Gate} = T_{Gates} \,/\, (N{\cdot}l)$ obtained by SEQCS and JUQCS-G for the H, $+X$, $+Y$, X and Y benchmark circuits ($l = 24$, $N = 31$). Regarding SEQCS, the gate group size is limited to $\{1, 2\}$ (H, $+X$, $+Y$) and $\{1, 2, 4, 8\}$ (X, Y). The standard deviation of T_{Gate} is 0.62 ms (JUQCS-G, H, $+X$, $+Y$), 0.68 ms (JUQCS-G, X), 0.74 ms (JUQCS-G, Y) and ≤ 0.01 ms (SEQCS, H, $+X$, $+Y$, X, Y).

Regarding the first gate set (see Fig. 1.7a), JUQCS-G takes approximately 50.00 ms for the execution of each gate, with no significant differences found between the H, $+X$ and $+Y$ gates, since they generally require the same number of memory transactions during their evaluation. With an average execution time of 3.9 ms, SEQCS thus manages a respective speed-up of ≈ 12.8 compared to JUQCS-G. In addition to this, when combining the application of two gates, SEQCS is able to further decrease T_{Gate} to ≈ 2.93 ms for the H and $+Y$ gate and ≈ 1.92 ms for the $+X$ gate, yielding a 1.34 (17) and 2 (26) times faster simulation with respect to the single gate execution (JUQCS-G), respectively. The fact that only for the $+X$ gate an ideal strong scaling in T_{Gate} is obtained indicates an additional overhead in the out-of-order execution of the $+Y$ and H gates. In fact, only for the $+X$ gate the permutations of the state amplitudes among the internal variables and the permutations of the ± 1 coefficients in the joined matrix, i.e. $+X_i \otimes +X_j$, compensate each other, such that all amplitudes can be processed via the same instruction branch by all threads in a warp (see appendix E.5). In contrast to this, both the update kernels for the H and $+Y$ gate include branch divergences, negatively impacting the simulation speed. In order to decide whether the out-of-order calculation, which was introduced to prevent shared memory bank-conflicts in the expense of small branch divergences, is beneficial for the simulation of the $+Y$ and the H gate, Fig. 1.8 presents a comparison of the simulation times obtained by the out-of-order and an in-order execution scheme. In doing so, the figure reveals that the out-of-order calculation achieves a 17% faster simulation independently of M_G, hence validating the considerations taken in section 1.2.3.3.

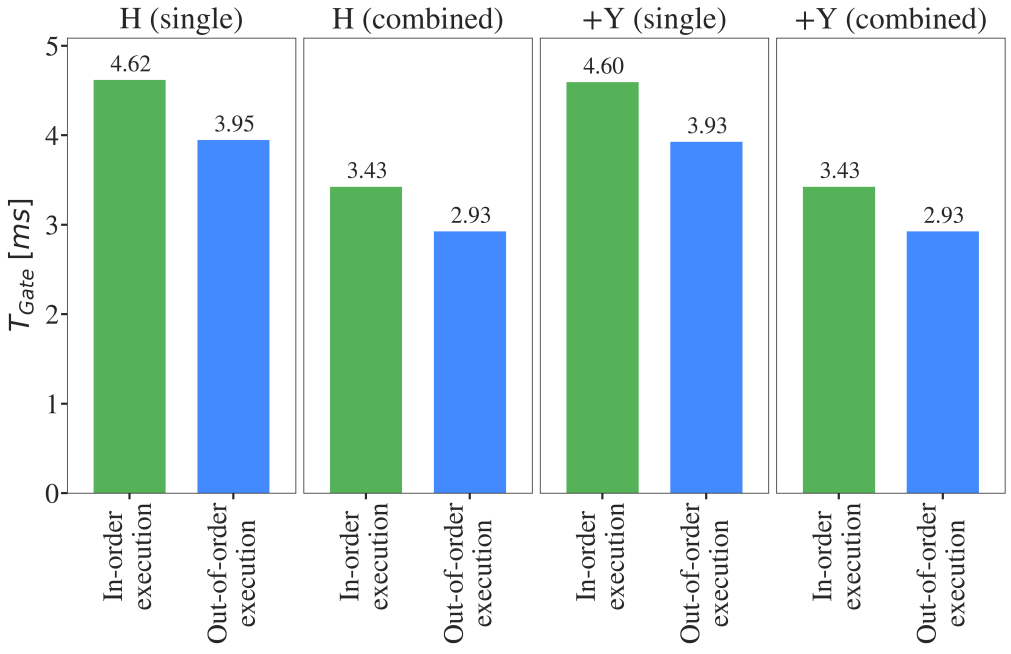


Figure 1.8.: The figure depicts the average gate execution time $T_{Gate} = T_{Gates} \,/\, (N \cdot l)$ obtained by SEQCS for the H and $+Y$ benchmark circuits ($l = 24$, $N = 31$). The simulations are conducted twice, once using the proposed out-of-order execution scheme (see appendix E.5) and once using an in-order calculation causing bank-conflicts.

Continuing with the X and Y gate, Fig. 1.7b shows an increase in the average simulation time per gate compared to the aforementioned non-diagonal gates obtained by JUQCS-G by 4.6% (X) and 8.7% (Y). This is caused by up to 45% longer executions of the gates applied to the first 4 qubits (see also the increase in the standard deviations to 0.68 ms (X) and 0.74 ms (Y), respectively). Since the number of memory transactions is identical to the above-mentioned gates, the reason for this behaviour remains an open question. Regarding SEQCS, simulation times of 3.78 ms and 4.15 ms are observed for the single X and Y gate executions, respectively, yielding speed-ups of 13.8 and 13.1 with respect to JUQCS-G. These elapsed times match approximately to Fig. 1.7a, since for example the X and $+X$ gate feature similar update kernels. Combining multiple gate operations into a single state traversal reveals again an ideal strong scaling of T_{Gate} in the gate-group size, with maximal speed-ups of 7.8 (107) and 7.6 (100) in the case of 8 combined X and Y gates compared to the single gate calculation (JUQCS-G), respectively. Note that the gate-group size is limited by the size of the cluster-space ($N_C = 11$). The fact that the Y gate generally executes approximately 10% slower than the X gate can be explained by additional arithmetic operations required for the complex phase shifts of the amplitudes.

1.3.4. Quantum algorithms

The previous sections (1.3.1 to 1.3.3) demonstrated a generally faster simulation of the benchmark circuits by SEQCS when compared to JUQCS-G. Concerning the internode communication, it was found that SEQCS achieves a significantly improved scaling of T_{MPI} in the number of consecutive global-local qubit exchanges. Using an optimized gate schedule, SEQCS is able to significantly reduce the number of qubit transfers necessary in a circuit evaluation as well as rearrange the exchanges, such that multiple transfers can be combined. As a consequence, the proposed MPI communication scheme is able to significantly reduce the amount of bytes communicated between pairs of GPUs, which was found to limit the simulation of large quantum circuits [27]. However, despite these improvements, it is important to mentioned that the MPI implementation used by SEQCS is not able to fully utilize the network bandwidth. This is most likely caused by the extensive use of unoptimized MPI data structures and an insufficient buffer management. Solving these issues is left for future work. Continuing with the introduction of the shared memory into the simulation process, an up to 12.7 times faster single gate execution was obtained with respect to JUQCS-G. Depending on the gate-cluster size, an ideal strong scaling of T_{Gate} was determined for $M_C \leq 8$, hence the use of shared memory also significantly improves the simulation time of strongly connected quantum circuits with small gate-clusters. Moreover, it was shown that amplitude caching does not cause a significant performance penalty, as the warp scheduler is able to hide the additional memory transfers. Finally, the proposed combined gate execution and the out-of-order computation pattern were found to further reduce the simulation time, by improving the memory access patterns and the load balance. These speed-ups generally depend on the size of the gate-group, yielding up to 17 times and 329 times faster evaluations of non-diagonal and diagonal single qubit gates, respectively.

While these investigations only considered artificial benchmark circuits, specifically tailored to one aspect of the simulation, Fig. 1.9 depicts the elapsed times of both simulators applied to real-world quantum algorithms, in order to see how these improvements translate to the simulation of a quantum adder ($N = 30$, $a = 15345$, $b = 17422$, see appendix B.3), the quantum Fourier transformation ($N = 31$, see appendix B.2) and the QAOA ($N = 31$, $p = 50$, see appendix B.1). Note that all circuits are simulated on a single A100 GPU. Throughout

all algorithms, SEQCS manages a significantly faster evaluation of the respective circuits, achieving speed-ups of 23.8 (Adder circuit), 22.6 (QFT circuit) and 70.4 (QAOA circuit) when compared to JUQCS-G. This shows that the execution times generally depend on the ability of SEQCS to apply the proposed optimization strategies. For example, concerning the quantum adder and the QFT, both exclusively using diagonal multi-qubit $U(K)$ gates, SEQCS is able to evaluate consecutive $U(K)$ gates within a single memory traversal using $2 \cdot \lceil \frac{N}{N_C} \rceil$ and $\lceil \frac{N}{N_C} \rceil$ kernel calls only. In contrast to this, JUQCS-G separately executes 390 and 558 gates, respectively. Important to highlight here is also the execution speed of the QAOA circuit. Its layered structure enables SEQCS to simulate all R_x gates in pairs, evaluate the diagonal transformations $\hat{U}_C$ using single memory traversals and, due to the lack of non-diagonal two-qubit gates, create gate-clusters of more than 8 gates, hence fully utilizing the speed-up of the shared memory. An example gate schedule of the QAOA circuit with $p = 2$ can be found in Fig. B.3 in appendix B.1. This significant improvement, with respect to the simulation time, enables a much more in-depth analysis of quantum optimization algorithms, such as the QAOA, in the following chapter 2. Using SEQCS, it is now possible to study the respective circuits in parameter regions (p and number of circuit evaluations) that were previously too expensive to simulate.

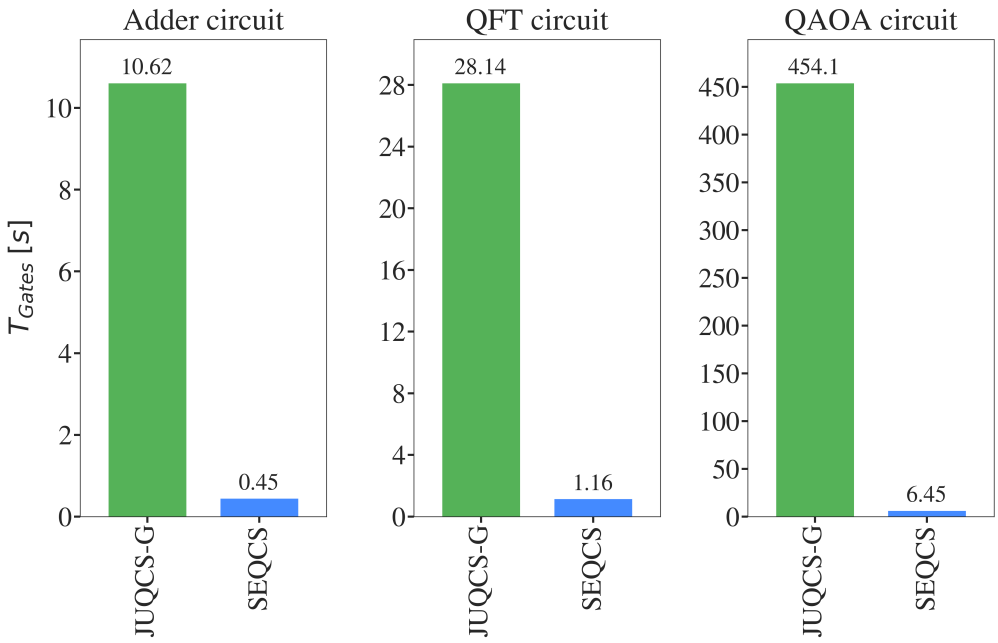


Figure 1.9.: The figure depicts the simulation times of JUQCS-G and SEQCS when applied to a quantum adder ($N = 30$, $a = 15345$, $b = 17422$, see appendix B.3), the quantum Fourier transform ($N = 31$, see appendix B.2) and the QAOA ($N = 31$, $p = 50$, see appendix B.1).

Chapter 2

Quantum optimization algorithms

In chapter 1 a GPU-accelerated simulator of general quantum circuits has been developed. Using its high-performance evaluation of QAOA circuits, this chapter concerns the investigation of quantum optimization algorithms in the context of exact cover and 2-SAT problem instances. A novel hybrid quantum-classical variational algorithm termed *guided quantum walk* (GQW) will hence be discussed in the following sections. The GQW is a heuristic strategy, combining principles of a trotterized continuous quantum walk and the QAOA, in order to deploy a quantum walker on an oriented graph connecting the problem's solution space. Using a series of restrictions governing the choice of variational parameters, which control the probability transfer between connected basis states in the graph, the quantum walker is actively *guided* towards the solution state based on the problem's energy spectrum. Introducing additional *mixing operations* in order to lift degeneracies of the energy levels, the GQW achieves superior success probabilities using intermediate circuit depths compared to the QAOA and the AQA on the investigated problem instances. Moreover, the conducted experiments indicate that the optimization phase of the variational parameters scales only linearly in the number of qubits, making the GQW an interesting candidate for near-term NISQ devices.

This chapter is structured as follows: section 2.1 will introduce the mapping of combinatorial optimization problems onto Ising Hamiltonians, with the latter encoding the problems' solution spaces in the computational bases of N qubit Hilbert spaces. Following that, section 2.2 will give an overview of common optimization algorithms, including the QAOA and the quantum walk. Section 2.3 will concern the derivation of the GQW, discussing the controlled movement of a quantum walker on an oriented graph as well as developing two heuristic models, termed HGQW and HGQW-A, for deploying the GQW on exact cover and 2-SAT problem instances, respectively. Finally, section 2.4 will investigate the scaling of the success probability achieved by the two models as a function of the number of qubits and the circuit depth, and compare the performance of the GQW to the QAOA and the AQA.

2.1. Combinatorial optimization problems

Combinatorial optimization problems occur in numerous contexts in modern society, including both scientific and industrial use cases. Their applications range from logistics, supply chain and manufacturing optimizations [26] to the analysis and benchmarking of classical as well as quantum computing hardware [8, 27, 28]. Max-Cut, travelling salesman, and covering problems among others are important representatives of this class of problems [64]. In general, a combinatorial optimization problem describes the process of finding an element in a vast collection of possible elements that is optimal given some predefined metric. A common property of such problems is that the search space grows at least exponentially in the problem size, making a naive element-wise search quickly intractable in modern frameworks. As a consequence, classical optimization algorithms have been developed in the past exploiting the structure and symmetry of the problem in question and thus reducing the number of inspections needed. With the ever-growing interest into quantum computing since the early 1990s with the discovery of Shor's algorithm [7] and the immense computational capabilities associated with it, a new class of optimization strategies and hardware based around quantum phenomena, such as superposition of computational basis states, qubit entanglement and complex phase interference (see section 1.1), has emerged. Prominent examples are the quantum approximate optimization algorithm (QAOA) [65] and quantum annealers [66], which will be discussed later in this chapter. Before, however, going into more detail on quantum optimization algorithms, the mapping procedure of combinatorial optimization problems into quantum systems is explained in this section.

Combinatorial optimization problems can be formally defined by introducing a classical objective function, commonly referred to as the *cost function*, $\mathrm{C} : Z \to \mathbb{R}$ that assigns a real value, the *cost*, to all elements Z_i in the set of all possible solutions $\mathbf{Z} = \{Z_i\}$. Since the *solution space* of any combinatorial optimization problem can be mapped to a finite discrete set, the elements $Z_i = [z_{N-1} \ \dots \ z_0]_2$ are defined as N-bit binary strings. The combinatorial optimization problem now seeks to find the element $Z_{opt} \in \mathbf{Z}$ with minimal cost, yielding $\mathrm{C}(Z_{opt}) \leq \mathrm{C}(Z_i) \ \ \forall \ Z_i \in \mathbf{Z}$. Note that the minimization problem can also be transformed into a maximization problem by remapping $\mathrm{C}(Z_i) \to -\mathrm{C}(Z_i)$. Two common formulations of the objective function $\mathrm{C}(Z_i)$ used in the literature are the *quadratic unconstrained binary optimization* (QUBO) model and the *Ising* model:

$$\textbf{QUBO:} \qquad \mathrm{C}_{QUBO}(Z = [z_{N-1} \ \dots \ z_0]_2) = \sum_{i \leq j} z_i Q_{ij} z_j + \mathrm{C}_0^{QUBO}, \tag{2.1}$$

$$\textbf{Ising:} \qquad \mathrm{C}_{Ising}(S = [s_{N-1} \ \dots \ s_0]_2) \ = -\sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j + \mathrm{C}_0^{Ising}, \tag{2.2}$$

with $0 \leq i, j < N$. In case of the QUBO formulation (see Eq. 2.1), the problem is encoded via the upper-triangular quadratic matrix $Q \in \mathbb{R}^{N \times N}$ and the binary problem variables $z_i \in \{0, 1\}$. In the Ising model, on the other hand, the description of a physical system consisting of N *Ising spins* arranged on a lattice is exploited for encoding the problem (see Eq. 2.2). These spins can occupy one of two distinct states, spin-↑ and down-↓, corresponding to the problem variables $s_i = +1$ and $s_i = -1$, respectively. Using long-range magnetic interactions, described via the coupling strengths $J_{ij} \in \mathbb{R}$, ferromagnetic and antiferromagnetic coupling is introduced between the pairs of spins, encouraging them to align or anti-align depending on the sign of J_{ij}. In addition to that, an external magnetic field $h_i \in \mathbb{R}$ is applied at each spin site, causing an energy difference between the spin-↑ and spin-↓ state. In

doing so, the combinatorial optimization problem is represented within the properties of the physical system. The parameters $\mathrm{C}_0^{QUBO}, \mathrm{C}_0^{Ising} \in \mathbb{R}$ appearing in Eq. 2.1 and Eq. 2.2, respectively, refer to constant collective shifts of the cost values assigned to the solution strings $\mathbf{Z}$ and $\mathbf{S}$. While affecting neither the problem's solution nor the optimization algorithm, they are used throughout this chapter to map the solution string to zero cost, i.e $\mathrm{C}(Z_{opt}) = 0$, making it convenient to determine whether the optimal solution has been found in the optimization process. Note, however, that the solution cost must be determined beforehand for this procedure, limiting it to research contexts where either Z_{opt} or $\mathrm{C}(Z_{opt})$ are known by construction. In order to transform between the QUBO and Ising formulations (the former being used by SEQCS), the *gate-based quantum computing convention* will be considered:

$$z_i = \frac{1 - s_i}{2}, \tag{2.3}$$

such that $z_i = 0$ ($z_i = 1$) is mapped to $s_i = +1 \mathrel{\widehat{=}} \text{spin}- \uparrow$ ($s_i = -1 \mathrel{\widehat{=}} \text{spin}- \downarrow$) (see appendix F for the full conversion). Note that in the literature on quantum annealing, an alternative conversion $z_i = (1 + s_i) \;/\; 2$ is also common, yielding the remapping $h_i \to -h_i$ [8, 28]. A natural way to transfer the classical objective function C into a quantum system is to construct the Ising formulation of the optimization problem and apply a remapping of the spin variables s_i to the Pauli-z operators $\hat{\sigma}_i^z$ (see Eq. 1.3). In doing so, the *cost Hamiltonian* (formally an Ising-Hamiltonian) $\hat{H}_C$ follows:

$$\hat{H}_C \equiv \hat{H}\left(\hat{\sigma}_0^z, \ldots, \hat{\sigma}_{N-1}^z\right) = \sum_{0 \le i < j \le N-1} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z \; - \sum_{i=0}^{N-1} h_i \hat{\sigma}_i^z \; + \; \mathrm{C}_0^{Ising} \tag{2.4}$$

$$= \sum_{Z \in \{0,1\}^{\otimes N}} \mathrm{C}(Z) \, |Z\rangle \langle Z| \,, \tag{2.5}$$

with $|Z\rangle = |z_0\rangle \otimes \cdots \otimes |z_{N-1}\rangle$ and $\hat{\sigma}_i^z$ referring to the Pauli-z operator acting on the ith qubit. Note that due to the chosen conversion between the binary variables z_i and the spin states s_i (see Eq. F.7), $z_i = 0$ ($z_i = 1$) maps to the $|0\rangle$ ($|1\rangle$) state of the quantum system, yielding $s_i = \langle z_i | \hat{\sigma}_i^z | z_i \rangle$. As a consequence, the spectral decomposition of H_C (Eq. 2.5) encodes the individual solutions to the optimization problem in the computational basis, with the ground state $|Z_{gs}\rangle$ (the lowest eigenvalue state) representing the optimal solution Z_{opt}. For this reason, one introduces the *success probability* P_{gs} and the *approximation ratio* A_r [67], defined as:

Success probability:
$$P_{gs} = |\langle Z_{gs} | \Psi \rangle|^2, \tag{2.6}$$

Approximation ratio:
$$A_r = \frac{\langle \Psi | \hat{H}_C | \Psi \rangle}{\max\limits_{Z_i \in \mathbf{Z}} \mathrm{C}(Z_i)}. \tag{2.7}$$

These metrics are used to gauge the performance of quantum optimization algorithms in tuning the system's state $|\Psi\rangle$ into $|Z_{gs}\rangle$. Note, however, that the success probability is a valid metric only in research contexts, as again the knowledge about the solution string Z_{opt} is required for its calculation. Moreover, even though P_{gs} and $1 - A_r$ feature the same global maximum, their energy landscape can be inherently different [30], making it necessary to test quantum optimization algorithms using these metrics (see e.g. the QAOA in section 2.2.2) also with respect to A_r to verify their performance in real world applications.

Throughout this chapter, two different types of optimization problems, namely *exact cover problems* and *2-SAT problems*, will be investigated and used for benchmarking quantum optimization algorithms. They are chosen, as the considered instances correspond to two extremes in the distribution of the energy levels, which will play an important role in the design of the *guided quantum walk* in section 2.3. All problems investigated feature a unique ground state $|Z_{gs}\rangle$ and are furthermore rescaled by dividing the parameters J_{ij}, h_i and C_{Ising} (see Eq. 2.4) by:

$$\max\left\{\max\left[\frac{\max h_i}{h_{max}}, 0\right], \max\left[\frac{\min h_i}{h_{min}}, 0\right], \max\left[\frac{\max J_{ij}}{J_{max}}, 0\right], \max\left[\frac{\min J_{ij}}{J_{min}}, 0\right]\right\}, \tag{2.8}$$

with $h_{max} = -h_{min} = 2$ and $J_{max} = -J_{min} = 1$. Note that the same rescaling was used by *Willsch et al.* when investigating exact cover problems [30] and is motivated by the normalization done by D-WAVE quantum annealers [66]. This procedure allows bringing the energies of different problem instances to a common scale, while not influencing the problem's solution, which will simplify the application of quantum optimization algorithms later on.

2.1.1. Exact-Cover problems

Exact cover problems belong to the broader class of NP-Complete set covering and partitioning problems [68] and have become an established choice for studying and benchmarking quantum algorithms [9, 28, 69]. The problem can be stated as follows: Consider a set $U = \{x_0, x_1, \ldots, x_{P-1}\}$ of P distinct elements, and N subsets $V_i \subseteq U (i = 0, \ldots, N-1)$, such that $U = \bigcup_i V_i$. The exact cover problem now seeks to find a subset L of the set of sets $\{V_i\}$, such that the elements of L are disjoint sets and the union of the elements of L is U. Exact cover instances are commonly encoded in matrix form using an *exact cover matrix* A, where the matrix columns (enumerated by j) refer to the P individual elements x_j in U and the matrix rows (enumerated by i) correspond to the N subsets V_i. Thus, the matrix coefficients a_{ij} determine whether an element x_i is included in a subset V_j ($a_{ij} = 1$) or not ($a_{ij} = 0$). As a consequence, L is the subset of matrix rows such that in each column of selected rows, the entry 1 appears *exactly once*, yielding the objective function C:

$$\mathrm{C}\left(Z = [z_{N-1} \ \ldots \ z_0]_2\right) = \sum_{j=0}^{P-1}\left(\sum_{i=0}^{N-1} a_{ij}\, z_i - 1\right)^2, \tag{2.9}$$

with z_i encoding the selection of subsets V_i. In order to obtain the Ising formulation of the problem, the binary problem variables $\{z_i\}$ get replaced by spin variables $\{s_i\}$ using Eq. 2.3, yielding the expressions for the coefficients of the Ising model (see Eq. 2.2) after collecting linear, quadratic, and constant terms:

$$h_i = -\frac{1}{2}\sum_j \left(AA^T\right)_{ij} + \left(A\vec{b}\right)_i, \tag{2.10}$$

$$J_{ij} = \frac{1}{2}\left(AA^T\right)_{ij}, \tag{2.11}$$

$$\mathrm{C}_0^{Ising} = \vec{b}^T\vec{b} + \frac{1}{2}\sum_{i<j}\left(AA^T\right)_{ij} + \sum_i\left[\frac{1}{2}\left(AA^T\right)_{ii} - \left(A\vec{b}\right)_i\right]. \tag{2.12}$$

with $\vec{b} = (1, \ldots, 1)^T$ being a P dimensional vector of ones. The exact cover instances used throughout this chapter were generated using the algorithm described in appendix G.1. In total, 48 unique problems have been randomly created with 8 instances for each system size $N \in \{10, 12, 14, ..., 20\}$. The number of elements in U (number of matrix columns) is set to 64 and the number of solution bits (i.e. the number of set bits in the Z_{opt}) is approximately 1/3 of the system size N. The reason for choosing these parameters in the generation process is that the resulting problems feature numerous distinct energy levels ($> 100 \cdot N$) with a relatively low degeneracy of the energies of the first excited eigenstates ($< N$) (see e.g. Fig. 2.1). This is mainly achieved by the large number of elements P (3 to 6 times larger than the number of qubits N), causing numerous interactions between the qubits representing the matrix rows, hence allowing to distinguish a great number of them in energy. Note that in real world applications, such as the tail assignment problem investigated by *Willsch et al.* [28], typically even larger matrices are used with $P > 15 \cdot N$, causing a large energy splitting. The complete set of investigated problems, including their QUBO formulations and energy spectra, is given in appendix G. Moreover, `EC_N_I` will be used for referencing individual problems, with N being the number of qubits and $I \in \{1, 2, \ldots, 8\}$ referring to the problem instance.

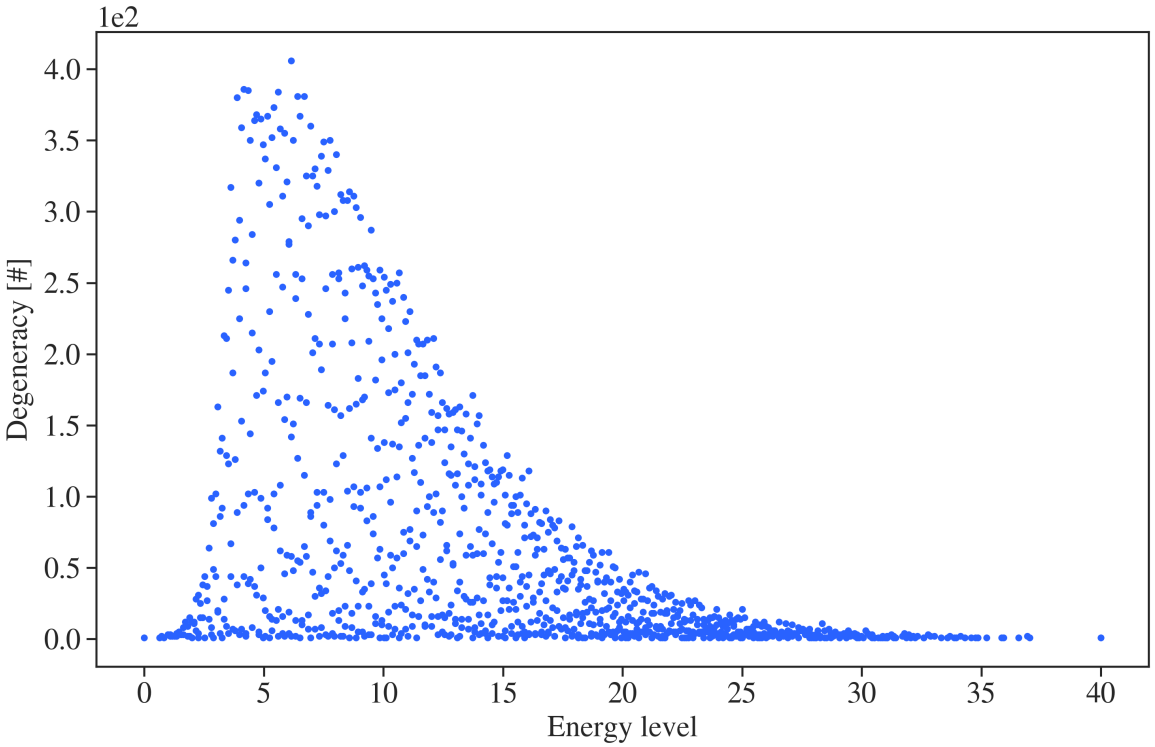


Figure 2.1.: The plot shows the degeneracy of the energy levels of H_C (see Eq. 2.5) for the `EC_16_1` exact cover problem. The problem instance features 1024 unique energies, as a result of the large number of matrix columns $P = 64$ in A, allowing to distinguish a great number of row combinations in energy. Also note the differences in the energy gaps between the energy levels. Similar distributions of the states can be obtained for all investigated problems in the exact cover problem set (see appendix G).

2.1.2. 2-SAT problems

The second type of combinatorial optimization problems that will be considered in this chapter are satisfiability problems, which have been extensively used in the study of quantum annealing in the past [70–72]. Similar to exact cover problems, 2-SAT problems are also NP-Complete and belong to the class of set covering and partitioning problems [68]. A 2-SAT problem seeks to satisfy a binary function F consisting of a conjunction of P binary clauses D_k. Each D_k is given as an or-operation between two binary variables $L_{k,1}$ and $L_{k,2}$, with $L_{k,j} \in \{z_i, \bar{z}_i\}$. Here, $\{z_i\}$ denotes the set of the N binary problem variables. Thus, the goal is to find the bit string $Z = [z_{N-1} \ \ldots \ z_0]_2$, such that every clause D_k is fulfilled:

$$F = D_0 \ \& \ D_1 \ \& \ \ldots \ \& \ D_{M-1}, \tag{2.13}$$

$$D_k = L_{k,1} \mid L_{k,2}. \tag{2.14}$$

2-SAT instances can be mapped to classical objective functions $\mathrm{C}(Z)$ by assigning cost values c to the satisfaction of the individual clauses D_k using $c(D_k, S) = (1 - L_{k,1}) \cdot (1 - L_{k,2}) = [1 - \epsilon_{k,1}\, s(k,1)] \cdot [1 - \epsilon_{k,2}\, s(k,2)]$. Here, $\epsilon_{k,j} = 1$ ($\epsilon_{k,j} = -1$) if z_j appears negated (not negated) in the kth clause and $s(k,j)$ refers to the spin $s_i \in S$ that is mapped to the jth literal of the kth clause. Note that Eq. 2.3 is used to convert between the binary variables Z and the Ising spins S. In doing so, the objective function can be constructed as the sum of the individual penalty terms $c(D_k, S)$:

$$\mathrm{C}(S = [s_{N-1} \ \ldots \ s_0]_2) = \sum_{k=0}^{P-1} c(D_k, S) \tag{2.15}$$

$$= \sum_{k=0}^{P-1} [1 - \epsilon_{k,1}\, s(k,1)] \cdot [1 - \epsilon_{k,2}\, s(k,2)]. \tag{2.16}$$

Since $c(D_k, S) \geq 0$, the solution string S_{opt} fulfils $\mathrm{C}(S_{opt}) = 0$, with all penality terms vanishing. After restructuring Eq. 2.16 into constant, linear and quadratic terms in s_i, the expression for the Ising parameters is obtained:

$$h_{j_i} = \sum_{k=0}^{M-1} \epsilon_{k,j_i}, \tag{2.17}$$

$$J_{j_1,j_2} = \sum_{k=0}^{M-1} \epsilon_{k,j_1} \cdot \epsilon_{k,j_2}, \tag{2.18}$$

$$\mathrm{C}_0^{Ising} = 1. \tag{2.19}$$

The 2-SAT problems which will be investigated in this chapter are taken from the pool of problem used in [73]. In total, 48 problems are considered with 8 instances per system size $N \in \{10, 12, 14, \ldots, 20\}$. In contrast to the exact cover problems described before, the number of energy levels of these problems is small, as the energies that are assigned to the individual states $|Z\rangle$ correspond to the number of violated penalty terms, resulting in $P + 1$ unique energy levels. Since P is in the order of N, while the number of states grows exponentially with 2^N, the problem instances generated by *Mehta et al.* feature high degeneracies in the energy spectrum (see e.g. Fig. 2.2). Therefore, the 2-SAT instances can be thought of as the opposite extreme to the exact cover problems, regarding the energy distribution of $\hat{H}_C$ (see Eq. 2.4). For referencing individual problem instances, an analogue scheme to section 2.1.1 will be applied, using `2SAT_N_I` with the system size N and the problem instance $I \in \{1, 2, \ldots, 8\}$. Moreover, the complete set of 2-SAT problems is given in appendix H.

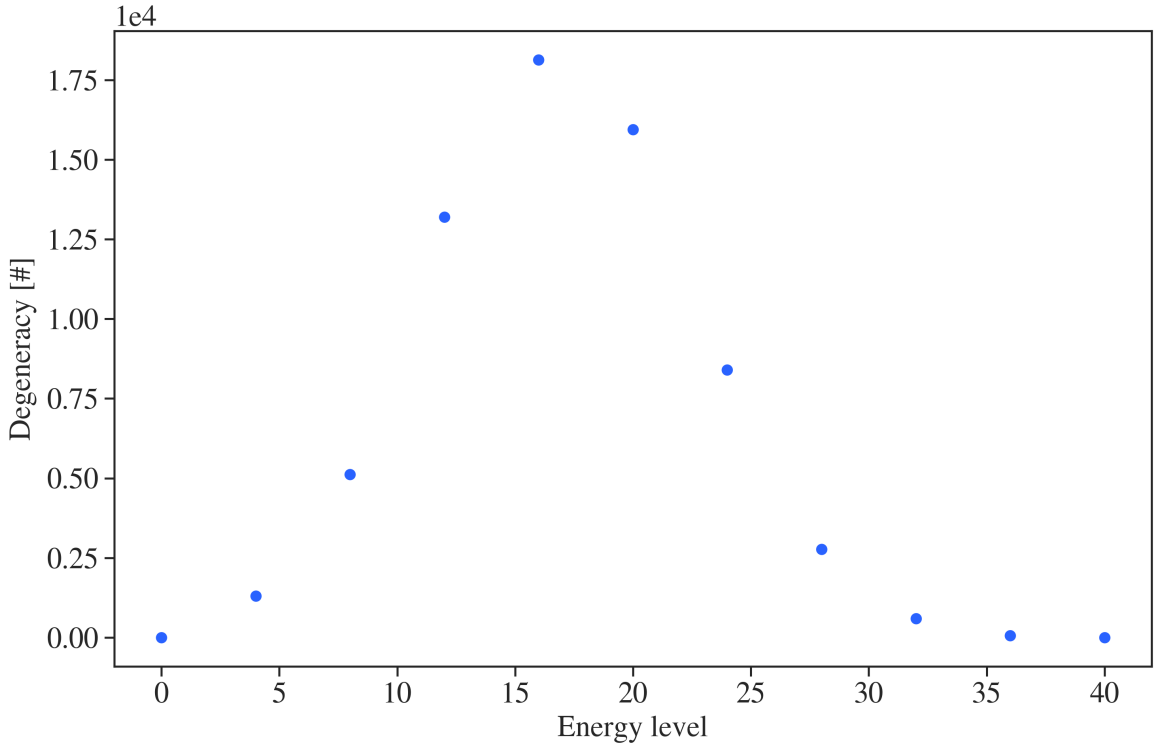


Figure 2.2.: The plot shows the degeneracy of the energy levels of $\hat{H}_C$ (see Eq. 2.5) for the `2SAT_16_1` problem instance. The problem features 11 unique energy levels, referring to the discrete number of violated penalty terms in Eq. 2.16. Also note the equal distribution of the energy levels and their high degeneracies compared to the exact cover problem shown in Fig. 2.1. Similar distributions of the states can be obtained for all problems in the investigated 2-SAT problem set (see appendix H).

2.2. Quantum optimization algorithms

The discovery of *Shor's algorithm* for integer factorization [7] and *Grover's search algorithm* [6], proven to achieve exponential and polynomial speed-ups compared to their best classical counterparts, demonstrated the computational capabilities of quantum computing and initiated a decade long run on building large scale quantum devices [2–4]. Recently, quantum computing has entered the so called *NISQ-era* [37], with the increasing availability of *Noisy Intermediate Scale Quantum* (NISQ) devices, offering few imperfect qubits ($N = \mathcal{O}(100)$) with limited coherence times and weak error-correction capabilities. Since both Shor's and Grover's algorithm require millions of qubits with error correction techniques [17], recent research has shifted towards noisy and shallow quantum algorithms, in order to achieve useful quantum computation already within the next decade [22]. Among others, heuristic algorithms for solving combinatorial optimization problems have emerged as promising candidates for such NISQ devices, with realizations in both the circuit [18, 20, 65, 67] and annealing model [66, 74, 75]. In what follows, the three main branches of quantum optimization algorithms, namely *quantum annealing* (section 2.2.1), the *quantum approximate optimization algorithm* (section 2.2.2) and *quantum walks* (section 2.2.4), will be introduced, providing the basic framework for the development of a novel strategy termed *guided quantum walk* in section 2.3.

2.2.1. Quantum annealing

During the early 2000s, *adiabatic quantum computing* (AQC) emerged as one of the major computational frameworks in quantum computation [74, 76, 77]. Alongside gate-based and measurement-based computing, adiabatic quantum computing is proven to be universal [74] and gained much interest due to its strong connections to both condensed matter physics and complexity theory. The framework is based on the *adiabatic theorem*, stating that a system starting in a non-degenerate ground state of a time-dependent Hamiltonian $\hat{H}(t)$, which is transitioning from some *initial form* $\hat{H}_M$ to some *final form* $\hat{H}_C$, during the *annealing time* τ, will remain in its instantaneous ground state throughout the evolution, provided that the Hamiltonian $\hat{H}(t)$ changes sufficiently *slow* [78, 79]. By using two time dependent annealing function $F_M(t)$ and $F_C(t)$, which control the transition between $\hat{H}_M$ and $\hat{H}_C$, the time evolutions of the initial state $|\Psi_0\rangle$ is given by the time-dependent Schrödinger equation (see Eq. 1.6):

$$\hat{H}(t) = F_M(t)\,\hat{H}_M + F_C(t)\,\hat{H}_C \qquad , \; 0 \leq t \leq \tau, \tag{2.20}$$

$$|\Psi\rangle(t) = \exp\left[-i\int_0^t \hat{H}(t')\,dt'\right]|\Psi_0\rangle\,. \tag{2.21}$$

Note that $F_M(0)/F_C(0) \gg 1$ ($\hat{H}(0) \approx F_M(0)\,\hat{H}_M$) and $F_M(\tau)/F_C(\tau) \ll 1$ ($\hat{H}(\tau) \approx F_C(\tau)\,\hat{H}_C$) are required. Typical annealing schemes used in the literature include linear time evolutions [30, 73], i.e. $F_M(t) = (1 - t/\tau)$ and $F_C(t) = t/\tau$, as well as exponential distributions, which are commonly used by D-Wave quantum annealers [80]. In case the initial Hamiltonian $\hat{H}_M$ is such that it ground state can be constructed easily, and the final Hamiltonian $\hat{H}_C$ encodes the solution to a computational problem in its ground state, the adiabatic evolution of Eq. 2.20 will tune the system into the solution state. According to the adiabatic theorem, the transition between the two Hamiltonians is assured to be adiabatic if the annealing time τ satisfies the condition:

$$\tau_{min} \gg \max_{0 \leq t \leq \tau} \frac{\left|\langle\Psi_{fs}(t)|\,\frac{d\hat{H}}{dt}\,|\Psi_{gs}(t)\rangle\right|}{\Delta E^2(t)}, \tag{2.22}$$

with $|\Psi_{gs}(t)\rangle$ and $|\Psi_{fs}(t)\rangle$ denoting the ground and first exited eigenstate of the instantaneous Hamiltonian $\hat{H}(t)$, and $\Delta E(t)$ being the instantaneous energy gap between them [79]. Eq. 2.22 reveals that the minimal annealing time τ scales inverse quadratically in the minimal energy gap, $\Delta E_{min} = \min_{0 \leq t \leq \tau} \Delta E(t)$, between the ground and first exited state, yielding long annealing times for small ΔE_{min}. Note that this also limits AQC to non-commuting initial and final Hamiltonians, i.e. $[\hat{H}_M, \hat{H}_C] \neq 0$, as otherwise the energy gap closes at some point, causing $\tau \to \infty$. Due to the expected quantum speed-up, noise-induced errors (e.g. thermal fluctuations and control errors) and an unfavourable energy spectrum of $\hat{H}_C$, real world applications often demand adiabatic quantum computation to finish early, hence violating Eq. 2.22 and causing low success probabilities P_{gs} [81]. To solve this problem, optimized annealing schedules [70, 82, 83] and additional *trigger Hamiltonians* [72, 73] have been studied in the literature, trying to increase P_{gs} for too short annealing times ($\tau < \tau_{min}$) and widen the minimum energy gap between $|\Psi_{gs}(t)\rangle$ and $|\Psi_{fs}(t)\rangle$, respectively. Moreover, *diabatic transitions* have been found to improve the success probability [84, 85]. Note, however, that previous findings in the literature also indicate an exponential scaling of τ_{min} in the number of qubits N [86].

Quantum annealing now describes a metaheuristic application of adiabatic quantum computation in the context of combinatorial optimization problems, as introduced in section 2.1. By restricting the final Hamiltonian $\hat{H}_C$ to Ising form (see Eq. 2.4), whose ground state encodes the solution to the optimization problem in question, quantum annealing takes advantage of the adiabatic evolution expressed by Eq. 2.21 to tune the system into the desired solution state. In doing so, quantum annealing is often considered as the quantum analogue of simulated annealing. The latter is a classical strategy for solving optimization problems based on a random walk originating from a random computational basis state by performing Metropolis-Hasting updates with the goal to relax in a minimum of the potential landscape [87]. While simulated annealing relies on thermal fluctuations, by initializing the system at high temperature and gradually cooling it down to overcome energy barriers, quantum annealing operates in the regime of extremely low temperature, where quantum effects are significant. The key idea is, to use quantum fluctuations, controlled by the strength of the initial Hamiltonian, in the eigenbasis of the final Hamiltonian, to achieve tunnelling through potential barriers [87]. Consequently, $\hat{H}_M = \sum_{i=0}^{N-1} \hat{\sigma}_i^x$ is commonly used to initialize the system, since its ground state $|+\rangle^{\otimes N}$ is generally preparable with high probability and provides maximal fluctuations in the computational basis. With the growing availability of commercial quantum annealers from D-Wave, which offer system sizes of more than 5000 qubits [66], and the fact that quantum annealing has been shown to be superior to simulated annealing in case of tall energy barriers [87, 88], quantum annealing has recently developed into a common approach in both research and industry for tackling complex optimization problems.

2.2.2. Quantum approximate optimization algorithm

In 2014, *Farhi et al.* introduced the quantum approximate optimization algorithm (QAOA), a hybrid quantum-classical variational algorithm capable of finding approximate solutions to combinatorial optimization problems by combining a parameterized quantum evolution with a classical optimizer for finding optimal tuning parameters [65]. The QAOA became popular due to its shallow circuit depth and demonstrated success for various kinds of optimization tasks, including Max-Cut and Travelling salesman problems, making it a promising candidate for near-term NISQ devices [9, 63, 67]. Moreover, it was proven by *Farhi et al.* that QAOA circuits cannot be efficiently (i.e. without exponential overhead) simulated on classical computers [89]. In order to construct the QAOA for a specific optimization problem, the following two unitary evolutions are considered:

$$\textbf{Phase separation:} \qquad \hat{U}_C(\gamma) = e^{-i\gamma\hat{H}_C}, \tag{2.23}$$

$$\textbf{Mixing:} \qquad \hat{U}_M(\beta) = e^{-i\beta\hat{H}_M} \qquad \text{with} \qquad \hat{H}_M = \sum_{i=0}^{N-1} \hat{\sigma}_i^x. \tag{2.24}$$

Here, the evolution times β and γ denote variational parameters. Commonly the QAOA begins by preparing the system in the equal superposition state $|+\rangle^{\otimes N}$, i.e. the ground state of $-\hat{H}_M$, providing an unbiased initial distribution of the measurement probabilities. Note that in contrast to quantum annealing, any choice for the initial state is possible, with biased approaches reducing circuit depths for certain problems. The QAOA is composed of p layers, with each layer consisting of a consecutive application of the phase separation evolution $\hat{U}_C(\gamma)$ followed by the mixing evolution $\hat{U}_M(\beta)$. Similar to quantum annealing, the

optimization problem is encoded in the ground state of an Ising Hamiltonian (see Eq. 2.4) termed $\hat{H}_C$. This cost Hamiltonian is used for separating the computational basis states in their complex phases according to their energy (cost) assigned by the optimization problem. Based on these phase differences, the mixing Hamiltonian $\hat{H}_M$ is then applied to cause probability transfers between the basis states and consequently increase the system's overlap with the solution state. As such, $\hat{H}_C$ and $\hat{H}_M$ must not commute, i.e $\left[\hat{H}_C, \hat{H}_M\right] \neq 0$, as otherwise both Hamiltonians share identical eigenbases, yielding only phase changes to the state amplitudes and thus leaving the system physically unchanged. Hence, the sum of Pauli-x operators was proposed for $\hat{H}_M$ by *Farhi et al.*, as it provides maximal mixing in the computational basis. Note, however, that other mixing Hamiltonians have been investigated in the literature [90], including a generalization of the QAOA, called *quantum alternating operator ansatz* [91]. The latter suggests switching mixing operations throughout the QAOA layers. The final state of the QAOA is given by:

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = \prod_{k=1}^{p} \hat{U}_M(\beta_k)\, \hat{U}_C(\gamma_k)\, |+\rangle^{\otimes N} \tag{2.25}$$

$$= \prod_{k=1}^{p} e^{-i\beta_k \hat{H}_M} e^{-i\gamma_k \hat{H}_C} |+\rangle^{\otimes N}. \tag{2.26}$$

Here, $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$ is called the variational state, since it depends on the sets of $2p$ variational parameters $\boldsymbol{\beta} = \{\beta_i\}$ and $\boldsymbol{\gamma} = \{\gamma_i\}$. Note that each β_k lies in the interval between 0 and π, since inserting $\beta_k \to \beta_k + \pi$ into Eq. 2.25 yields only a global phase factor. Likewise, if the eigenvalues of $\hat{H}_C$ are all integers, γ_k lies within 0 and 2π for similar reasons. Moreover, since both $e^{-i\beta_k \hat{H}_M}$ and $e^{-i\gamma_k \hat{H}_C}$ denote rotations in their respective eigenbasis, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are also often referred to as *angles*. The idea of the QAOA is that with optimal angles $\boldsymbol{\beta}_{opt}$ and $\boldsymbol{\gamma}_{opt}$ and sufficient algorithmic depth p, $\left|\boldsymbol{\beta}_{opt}, \boldsymbol{\gamma}_{opt}\right\rangle$ should have a large overlap with computational basis states close (small Hamming distance) to the solution string, yielding sufficient approximate solutions with high probability when measuring the final system in the computational basis. However, determining $\boldsymbol{\beta}_{opt}$ and $\boldsymbol{\gamma}_{opt}$ is generally a non-trivial task, denoting an optimization problem itself [65]. Therefore, *Farhi et al.* proposed a hybrid classical and quantum ansatz, by efficiently preparing Eq. 2.25 via a quantum device and optimizing the sets of variational parameters in an outer classical loop based on the energy expectation of the prepared state (see Fig. 2.3):

$$\left(\boldsymbol{\beta}_{opt}, \boldsymbol{\gamma}_{opt}\right) = \arg\max_{\boldsymbol{\beta}, \boldsymbol{\gamma}} \langle \boldsymbol{\beta}, \boldsymbol{\gamma} | \hat{H}_C | \boldsymbol{\beta}, \boldsymbol{\gamma} \rangle. \tag{2.27}$$

Concerning intermediate sized QAOA circuits, this approach, however, quickly becomes infeasible, due to the increasing complexity of the highly non-convex and exponentially growing parameter search space [30, 67]. Consequently, classical optimizers often get stuck in suboptimal local minima in the parameter space, requiring several optimization runs and thus limiting the QAOA to only shallow instances. Since, however, the optimal success probability of the QAOA increases monotonically in its algorithmic depth p, several strategies have been proposed in the literature for determining good initial guesses for $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, in order to simplify the optimization process in the region of intermediate to large p. These approaches are based on the observation of common patterns in the distribution of the variational parameters occurring across different problem instance [9, 92, 93], yielding machine learning [94, 95] and iterative strategies [67, 96].

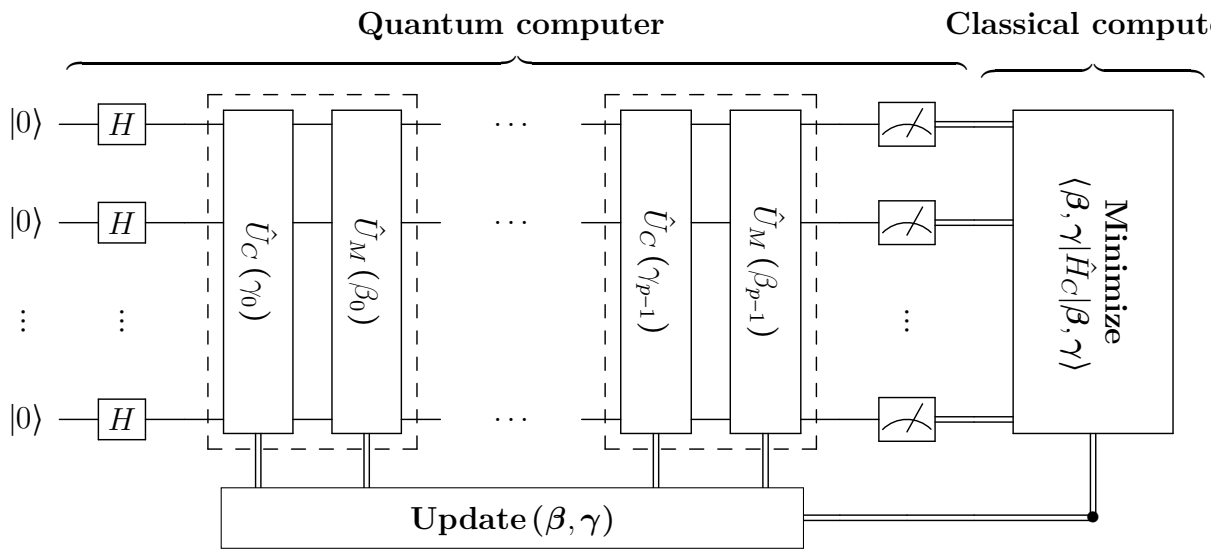


Figure 2.3.: The figure depicts the quantum-classical QAOA procedure for solving combinatorial optimization problems [65]. First, the system is initialized in the equal superposition state $|+\rangle^{\otimes N}$ by applying Hadamard gates H to all qubits. Next, a set of $2p$ angles $(\boldsymbol{\beta},\boldsymbol{\gamma})$ is selected randomly. Based on these angles, p QAOA layers are performed, each consisting of a phase separation evolution $\hat{U}_p(\gamma)$ and a mixing evolution $\hat{U}_M(\gamma)$. Eventually, the system is measured in the computational basis, yielding a classical output bit string Z. The procedure is repeated m times in order to estimate $\bar{E} = \langle\boldsymbol{\beta},\boldsymbol{\gamma}|\hat{H}_C|\boldsymbol{\beta},\boldsymbol{\gamma}\rangle \approx 1/m \sum_{i=1}^{m} C(Z_i)$. Finally, a classical optimizer is queried to update $(\boldsymbol{\beta},\boldsymbol{\gamma})$ based on the energy expectation and the process in repeated, until $\bar{E}$ is sufficiently low.

2.2.3. Approximate quantum annealing

The approximate quantum annealing (AQA) algorithm is a heuristic strategy proposed by *Willsch et al.* for finding approximate solutions to combinatorial optimization problems [18, 30], closing the gap between quantum annealing and the QAOA. Beginning with the simulation of a quantum annealing process, the evolution of the quantum system is described via the time-dependent Schrödinger equation (see Eq. 1.6), for times $0 < t < \tau$, with $\hat{H}(t) = F_M(t)\,\hat{H}_M + F_C(t)\,\hat{H}_C$ denoting the annealing Hamiltonian and $\hat{H}_M$ and $\hat{H}_C$ being the driving and cost Hamiltonian, respectively. Note that the strengths of the latter are controlled via the annealing functions $F_M(t)$ and $F_C(t)$, satisfying $F_M(0)/F_C(0) \gg 1$ ($\hat{H}(0) \approx F_M(0)\,\hat{H}_M$) and $F_M(\tau)/F_C(\tau) \ll 1$ ($\hat{H}(\tau) \approx F_C(\tau)\,\hat{H}_C$) (see section 2.2.1). The AQA can now be derived from the observation, that the time-evolution of the quantum state $|\Psi_0\rangle$ can be simulated using time-stepping and the *second-order Suzuki-Trotter formula* [97]:

$$|\Psi\rangle(\tau) = \hat{U}(\tau)\,|\Psi_0\rangle\,, \tag{2.28}$$

$$\hat{U}(\tau) = \mathcal{T}\exp\left[-i\int_0^{\tau}\hat{H}(t)\,dt\right] \tag{2.29}$$

$$\approx \prod_{k=1}^{p} e^{-i\,\hat{H}(k\Delta t)\,\Delta t} \tag{2.30}$$

$$\approx \prod_{k=1}^{p} e^{i\frac{1}{2}\Delta t\,F_M(k\Delta t)\,\hat{H}_M}\; e^{i\,\Delta t\,F_C(k\Delta t)\,\hat{H}_C}\; e^{i\frac{1}{2}\Delta t\,F_M(k\Delta t)\,\hat{H}_M}. \tag{2.31}$$

Here, $\hat{U}(\tau)$ denotes the time evolution operator, $\mathcal{T}$ is the time-ordering symbol and $\Delta t = \tau/p$ represents the time steps. Choosing $|\Psi_0\rangle = |+\rangle^{\otimes N}$ to initialize the system, $e^{i\frac{1}{2}\Delta t F_M(k\Delta t)\hat{H}_M}\ |+\rangle^{\otimes N}$ can be replaced by $|+\rangle^{\otimes N}$ in Eq. 2.28, as it only introduces an unmeasurable global phase factor. In doing so, Eq. 2.28 has the same structure as the variational state obtained by the QAOA (see Eq. 2.25), giving a bijective mapping between the variational parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ and the annealing functions $F_M(t)$ and $F_C(t)$:

$$\beta_k = -\frac{1}{2}\Delta t\ \left[F_M(\Delta t\cdot(k+1)) + F_M(\Delta t\cdot k)\right] \qquad ,k = 1,\ \ldots,\ p-1, \tag{2.32}$$

$$\beta_p = -\frac{1}{2}\Delta t\ F_M(\Delta t\cdot p)\,, \tag{2.33}$$

$$\gamma_k = \Delta t\ F_C(\Delta t\cdot k) \qquad ,k = 1,\ \ldots,\ p. \tag{2.34}$$

According to *Willsch et al.*, the underlying idea of AQA is to solve the TDSE with time-steps Δt that are too large and too few (i.e. small p) to describe an accurate adiabatic time evolution of a genuine quantum annealing process [30], due to large Trotter errors. Consequently, AQA does not rely on the adiabatic theorem, i.e. in the proposed parameter regime its success is not guaranteed. However, using a classical optimizer for tuning Δt, competitive success probabilities compared to the QAOA as a function of the computational workload can be achieved, due to the significantly reduced parameter search space (1 vs $2p$ dimensional). This can be understood, as previous findings [9, 63, 93] suggest that optimal distributions of variational parameters in the QAOA tend to follow certain curves, which resemble approximate annealing schedules. Thus, using Eq. 2.32, the AQA can also be used for initializing the QAOA parameters, with the hope that sufficient local maxima in the parameter space can be determined more easily compared to random initializations. Note, however, that the QAOA can go beyond quantum annealing and thus AQA, e.g. by utilizing diabatic evolutions, which can reduce the computation times [67]. The AQA will be used throughout this thesis as a reference algorithm for solving optimization problems in the regime of intermediate p.

2.2.4. Quantum walk

First mentioned by *Aharonov et al.* in 1993 [98], quantum walks describe a broad class of continuous-time quantum evolution algorithms, featuring a strong connection to classical random walks. In contrast to their classical counterparts, quantum walks replace the stochastic evolution of a probability vector with the iteration of a unitary matrix on a state vector living in a complex Hilbert space. In doing so, the physical description changes from a walker moving stochastically across vertices on a graph to a quantum walker moving in superposition over the basis states [99, 100]. In recent years, quantum walks gained popularity due to their ability to simulate complex quantum system and derive new quantum algorithms [101]. Two models of quantum walks have been proposed in literature:

- **Discrete quantum walks** typically utilize two Hilbert spaces $\mathcal{H}_C$ (*coin space*) and $\mathcal{H}_P$ (*position space*) for describing the movement of the quantum walker across the vertices (position space) [102]. The coin space decodes the internal state of the walker, which in turn denotes the direction of propagation in the position space. Thus, the discrete-time evolution consists of two operations applied in an alternating order, with an initial coin Hamiltonian driving the walker's internal state and a subsequent shift

Hamiltonian moving the walker according to it. A main requirement of this type of quantum walks is *graph locality*, meaning that the shift operator is only allowed to move the walker to adjacent states according to the proposed graph structure [101]. In doing so, discrete quantum walks share large similarities to classical random walks. Note that also *coinless* discrete quantum walks have been proposed in literature, utilizing graph partitions to derive adjusted shift Hamiltonians for each sub-graph allowing them to operate exclusively in the position space [103].

- **Continuous quantum walks**, on the other hand, are defined in a single Hilbert space using a continuous time evolution of the quantum system described via the time-dependent Schrödinger equation, such that the quantum walker moves at all times [81, 104]. Consequently, this class of quantum walks does not require locality of the evolution operations, as the walker spreads continuously over the vertices, not being limited to discretized steps.

As mentioned above, both types of quantum walks are defined on an undirected graph $G(V,E)$, with $V = \{j\}_{j=0}^{2^N-1}$ denoting the set of 2^N vertices and E describing the set of edges connecting pairs of vertices (j,k). In doing so, the vertices correspond to the positions of the quantum walker across the computational basis states of an N qubit Hilbert space, and the edges indicate the possible transitions between the state. Focussing on the continuous quantum walk in the following, the dynamics of the walker are described via the time-evolution operator $\hat{U}(t) = \exp\left(-i\hat{H}_M t\right)$ of the driving Hamiltonian $\hat{H}_M$. Here, $\hat{H}_M$ is related to G through the adjacency matrix A ($A_{ij} = 1$ for $(j,k) \in E$ and $A_{ij} = 0$ otherwise), the diagonal matrix D ($D_{jj} = deg(j)$, with $deg(j)$ denoting the number of connected edges to vertex j) and the hopping rate $\tilde{\gamma}$, describing the *transfer rate* between connected vertices per unit time: $\langle j|\hat{H}_M|k\rangle = -\tilde{\gamma} L_{jk}$. Note that $L = A - D$ is the Laplacian of G [104]. Common choices of graph layouts include:

$$\textbf{Complete Graph:} \qquad \hat{H}_M = \tilde{\gamma}\left[2^N \cdot \mathbb{I}^{\otimes N} - \sum_{j,k=0}^{2^N-1} |k\rangle\langle j|\right], \tag{2.35}$$

$$\textbf{Hypercube graph:} \qquad \hat{H}_M = \tilde{\gamma}\left[N \cdot \mathbb{I}^{\otimes N} - \sum_{j=0}^{N-1} \hat{\sigma}_j^x\right]. \tag{2.36}$$

Regarding the complete graph, every vertex is connected to every other, while in the hypercube mapping only states with a Hamming distance of one share an edge (see e.g. Fig. 2.4). Since $\hat{H}_M$ is hard to implement in case of the complete graph on both quantum hardware and classical simulators, the hypercube graph will be considered henceforth. To this point, the time evolution of the system will only cause a uniform spread of the walker over all vertices in the graph. In order to use continuous quantum walks to solve combinatorial optimization problems, a second evolution under the cost Hamiltonian $\hat{H}_C$ is added. Similar to quantum annealing and the QAOA, $\hat{H}_C$ uses Eq. 2.4 to encode the solution state in the ground state of an Ising Hamiltonian, yielding: $\hat{H}(\tilde{\gamma}) = \hat{H}_M(\tilde{\gamma}) + \hat{H}_C$.

Using the equal superposition state $|+\rangle^{\otimes N}$ to initialize the system, the optimization is performed by evolving the quantum state under the full Hamiltonian $\hat{H}(\tilde{\gamma})$ for a time τ followed by measuring the system in the computational basis, i.e. the eigenbasis of $\hat{H}_C$. The key idea is, that the fast spreading of the quantum walker via $\hat{H}_M(\tilde{\gamma})$ provides a rapid exploration

of the basis states, while the energy spectrum of $\hat{H}_C$ results in a localization of the walker around low-energy states. Note that the success probability typically oscillates in the total evolution time τ, generally requiring prior knowledge of an optimal τ for the optimization problem in question [104]. Consequently, the protocol is typically repeated multiple times using measurement times τ uniformly distributed at random in an interval $[\tau, \tau + \Delta t]$, in order to prevent only measuring the system where P_{gs} happens to be small. The average single run success probability is then given by: $\bar{P}_{gs} = 1/\Delta t \int_{\tau}^{\tau+\Delta t} P_{gs}(t')\, dt'$ [104]. Moreover, P_{gs} is influenced by the ability of the quantum walker to move between the vertices controlled by the hopping rate $\tilde{\gamma}$, which is usually chosen time-independent. Note that other choices have also been investigated [81]. Comparing $\hat{H}(\tilde{\gamma})$ to Eq. 2.20, the continuous quantum walk can also be viewed as a quantum annealing process using an infinitely fast quench, with $B(0)$ jumping from zero to $A(0)/\tilde{\gamma}$ at $t = 0$ and $A(\tau)$ dropping to zero at the final time $t = \tau$. Based on this observation, adjusted quantum walk protocols, e.g. using pre-annealed quantum states or monotonic quenches where the strength of $\tilde{\gamma}$ constantly decreases, have been proposed, in order to increase the single-shot success probability [81].

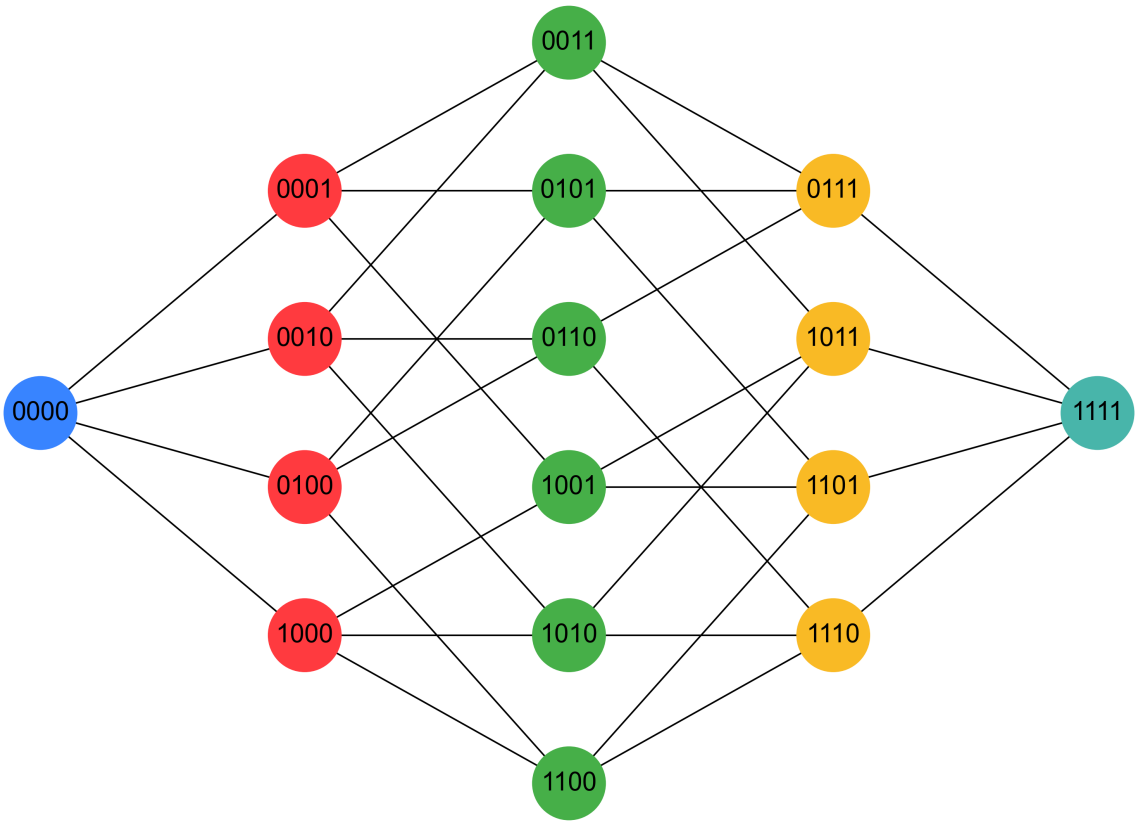


Figure 2.4.: The figure shows the graph structure of a *hypercube* mapping in case of an $N = 4$ qubit Hilbert space. Each node corresponds to one computational basis state. The nodes are grouped by colour depending on their Hamming weight and connected if they have a Hamming distance of one.

2.3. Guided quantum walk

In section 2.2 the three main branches of quantum algorithms aimed at solving combinatorial optimization problems (see section 2.1) have been presented, namely quantum annealing, the QAOA and quantum walks. While not delivering exponential speed-ups in finding the solution to NP-complete problems (e.g. exact cover problems and 2-SAT problems) it is hoped that variants of these algorithms can achieve polynomial speed-ups compared to their classical counterparts. Moreover, *approximate solutions* to combinatorial optimization problems might be efficiently obtainable via these strategies, playing an important role for problem instances where solutions that are *good enough* are still valuable if they can be obtained fast [65, 89]. However, a major drawback of these algorithms are the immense computational resources associated with them in the context of real-world problems [30, 81]. For example, the time required for an optimal quantum annealing scheme scales exponentially in the size of the smallest energy gap in the system, while for the QAOA a complex non-convex parameter space has to be searched. The latter grows exponentially in the number of variational parameters involved, making determining them an NP problem itself [65]. To overcome these drawbacks, a novel quantum optimization strategy, termed the *guided quantum walk* (GQW), will be introduced in this section. The guided quantum walk is based around the concept of a coinless discrete quantum walk (see section 2.2.4) and is inspired by the classical simulated annealing algorithm. The aim of this algorithm is to provide approximate solutions with high success probabilities efficiently, with numbers of operations that scale linearly in the numbers of qubits involved and an optimization phase that is constant in its complexity. In doing so, the guided quantum walk has a direct relationship to the QAOA and increases the understanding of its dynamics for certain sets of variational parameters. A detailed comparison of the GQW to the QAOA and to the AQA will be given in section 2.4.

2.3.1. Movement on directed graphs

The guided quantum walk is an optimization strategy operating on oriented graphs, using the dynamics of the quantum system to accumulate probability near the solution state of the optimization problem in question. A classical analogue of the algorithm is a simplified variant of the simulated annealing strategy [105]. By using a constant low temperature, the process selects an initial guess Z at random from the total set of all possible solutions $\mathbf{Z}$ and then improves it step by step by applying small adjustments (i.e. checking solutions in its neighbourhood) and keeping only those changes that yield overall improvements with respect to some metric. The idea of the guided quantum walk is to achieve a similar process in a quantum system by assigning all 2^N basis states to a graph structure with a quantum walker that moves along directed edges connecting them. Here, the directed edges resemble the evaluation process based on the aforementioned metric in the classical algorithm. The goal is to create an oriented graph in such a way that starting from any node, a directed path towards the solution node is given. Since, however, the internal structure (e.g. the solution state Z_{gs}) is unknown beforehand, designing a problem-specific graph, with all nodes featuring exactly one directed edge pointing towards Z_{gs}, is generally not possible. Instead, the approach taken here relies on a general graph layout, providing the basic framework of the walker's movement. In addition to this, a problem-specific metric is used to *guide* the walker within each step towards the solution state. Moreover, a set of variational parameters allows adapting the walker's dynamics to individual problem instances. While not being restricted to certain kinds of graphs, the *hypercube layout* (see e.g. Fig. 2.4) will be considered in the following due to its ease of implementation.

The guided quantum walk uses a variational wave function ansatz in which the state of the quantum system, initialized in the equal superposition of all basis states $|+\rangle^{\otimes N}$, is transformed using an alternating sequence of discrete time evolutions. The first evolution considered here concerns the movement of the quantum walker on the graph, described via the Hamiltonian $\hat{H}_M$. With respect to a continuous quantum walk, $\hat{H}_M$ is termed the driving Hamiltonian, with $\hat{H}_M = \sum_{i=0}^{N-1} \hat{\sigma}_i^x$ in the case of the *hypercube* mapping (see section 2.2.4). Solving the corresponding Schrödinger equation (see Eq. 1.6), the dynamics of the quantum walker are described via the time evolution operator $\hat{U}_M(t_M) = \exp\left(-i\,\hat{H}_M\,t_M\right)$, with $t_M \in [0, \pi)$ denoting the evolution time. Note that $\hat{U}_M(t_M)$ is π-periodic. In order to understand the effects of $\hat{U}_M(t_M)$ on the state vector, it is useful to first consider a system initialized in one of the computational basis states $|\Psi\rangle$, e.g. $|\Psi\rangle = |0\ 0\ \ldots\ 0\rangle$. Applying one discrete time evolution yields:

$$|\Psi\rangle \xrightarrow{e^{-i\,\hat{H}_M\,t_M}} |\Psi'\rangle \tag{2.37}$$

$$= \prod_{i=0}^{N-1} \left[\cos(t_M)\,\mathbb{I} - i\sin(t_M)\,\hat{\sigma}_i^x\right]|\Psi\rangle \tag{2.38}$$

$$= \begin{pmatrix} \cos(t_M) & -i\sin(t_M) \\ -i\sin(t_M) & \cos(t_M) \end{pmatrix}_0 \otimes \cdots \otimes \begin{pmatrix} \cos(t_M) & -i\sin(t_M) \\ -i\sin(t_M) & \cos(t_M) \end{pmatrix}_{N-1} \begin{pmatrix} \psi_{0\ 0\ \ldots\ 0} = 1 \\ \psi_{0\ 0\ \ldots\ 1} = 0 \\ \vdots \\ \psi_{1\ 1\ \ldots\ 1} = 0 \end{pmatrix} \tag{2.39}$$

$$= \sum_{l=0}^{N} (-i)^l \underbrace{\cos^{N-l}(t_M)\sin^l(t_M)}_{K_l(t_M)} \left[\sum_{S\in\{s_0,\ldots,s_{N-1}\}_{\Delta(S,\Psi)=l}} |S\rangle\right], \tag{2.40}$$

with $\Delta(S, \Psi)$ referring to the Hamming distance of the computational state $|S\rangle$ with respect to the initial state $|\Psi\rangle$, i.e. the number of spin flips required to change $|S\rangle$ into $|\Psi\rangle$. In doing so, Eq. 2.40 reveals that the application of $\hat{U}_M(t_M)$ causes probability transfers between the computational basis states, hence resembling the movement of the quantum walker on the graph in terms of a propagating probability wave. Thus, depending on the Hamming distance of each basis state to the initial state $|\Psi\rangle$, each node stores $K_l(t_D)^2$ amount of probability after one evolution under $\hat{H}_M$. Consequently, the walker's movement can be understood in terms of discrete step sizes l depending on the evolution time t_M. Hence, each coefficient K_l denotes a different order of interaction, allowing only $\binom{N}{l}$ states S of a Hamming distance $\Delta(S, \Psi) = l$ to exchange probability. When considering each coefficient K_l as a function of t_M, the algorithm is able to control the walker's movement by selecting certain ranges of interaction orders. This is possible, since each K_l features unique extrema $t_M = k\,\pi \pm \arctan\left(\sqrt{\frac{l}{N-l}}\right)$ with $k \in \mathbb{N}$, i.e. an evolution time t_M that maximizes the probability transfer $K_l(t_M)$ (see e.g. Fig. I.9 in appendix I). Consider for example Fig. 2.4, featuring an $N = 4$ qubit hypercube graph, and $|\Psi\rangle = |0000\rangle$. Choosing $t_M = \pi$, the system is restricted to zeroth order interactions only, hence the probability (quantum walker) will remain at $|0000\rangle$ ($l = 0$). In contrast to this, evolving the system for $t_M = 4\pi/3$, causes a probability wave to emerge at the *blue node* and spread across the graph's connections. Eventually, the probability wave will have its peak among the group of *yellow nodes* ($l = 3$) with decreasing probabilities towards the groups of increasing Hamming distance $|\Delta(S, \Psi) - 3|$. Here, the

probability distribution depends on the strength of $K_l(t_M)$ and describes the measurement probabilities to obtain the quantum walker at a specific node in the graph. Note that regarding approximate solutions to optimization problems, only the combined probability of a Hamming-weight group and its distance to the solution state is of interest. Motivated by this observation, the evolution time t_M will not be considered as a fixed constant, as for example in a continuous quantum walk (see section 2.2.4), but rather as a variational parameter, called β_i henceforth, allowing to select at every iteration i the range of step-sizes the quantum walker moves. Assuming the chosen graph layout provides sufficient connections between the nodes, the algorithm can adapt to arbitrary problem instances.

To this point, the dynamics presented describe a quantum walker that moves along undirected graphs connecting the basis states depending on the variational parameters β_i. The movement is undirected since no information regarding the optimization problem is included yet, yielding a uniform spread of the probability wave across the states. In order to incorporate the problem's properties into the movement of the quantum walker, the guided quantum walk considers the cost function $\mathtt{C}(Z)$ as a metric to distinguish connected states Z in the system and thus *guide* the walker towards the solution state (state of lowest cost). This approach is inspired by the observation of the immense splitting of energy levels occurring in real world exact cover problems (see e.g. Fig. 2.1), causing only relatively low degeneracies, hence suggesting the cost as a sufficient metric to evaluate the direction of movement of the quantum walker. Consequently, the algorithm introduces a second time evolution under the problem Hamiltonian $\hat{H}_C$ (see Eq. 2.5), i.e. $\hat{U}_C = \exp\left(-i \cdot \hat{H}_C \cdot t_C\right)$ with the evolution time t_C, before the application of each driving evolution $\hat{U}_M$. In doing so, the algorithm is able to *rank* each state depending on its distance to the solution state (given the state energy as a metric) such that the quantum walker *knows* at each step the correct direction to propagate in. The method chosen here relies on the constructive and destructive interference of probability waves propagating from each basis state, such that the initially undirected edges in the graph become directed (see e.g. Fig. 2.5). This is achieved by enforcing a phase gradient, $-\Delta E_{B,A} \cdot \gamma_i$, between pairs (B, A) of connected states with energy difference $\Delta E_{B,A}$. Note that the evolution time t_C of $\hat{H}_C$ is replaced by the variational parameter γ_i, yielding the identical variational state $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle$ after p walker steps (iterations) as for the QAOA (see Eq. 2.25). For further investigation, the general transformation of a state coefficient ψ_J ($|\Psi\rangle = \sum_{J=0}^{2^N} \psi_J |J\rangle$) after one iteration of the guided quantum walk ($\hat{U}_C$ and $\hat{U}_M$ evolution), using $\Delta E_{J_1,J_2} = E_{J_1} - E_{J_2} = \langle J_1|\hat{H}_C|J_1\rangle - \langle J_2|\hat{H}_C|J_2\rangle$, and $\Delta\alpha_{J_1,J_2} = \alpha_{J_1} - \alpha_{J_2}$ denoting the differences in the complex phases, is given by:

$$\psi_J = r_J \, e^{-i\alpha_J} \tag{2.41}$$

$$\xrightarrow{\hat{U}_C(\gamma)} r_J \, e^{-i\,(\alpha_J + \gamma \cdot E_J)} \tag{2.42}$$

$$\xrightarrow{\hat{U}_M(\beta)} \sum_{l=0}^{N} (-i)^l \cdot \underbrace{\cos^{N-l}(\beta) \sin^l(\beta)}_{K_l(\beta)} \cdot \left[\sum_{S \in \{s_0,\ldots,s_{N-1}\}_{\Delta(S,J)=l}} r_S \, e^{-i\,(\alpha_S + \gamma \cdot E_S)} \right] \tag{2.43}$$

$$= r_J \, e^{-i\,(\alpha_J + \gamma \cdot E_J)}$$

$$\cdot \left[K_0 \; - i \, K_1 \left(\frac{r_{J \wedge 1}}{r_J} \, e^{-i\left[\Delta\alpha_{J\wedge 1,J} + \gamma \cdot \Delta E_{J\wedge 1,J}\right]} \; + \; \ldots \right) \right.$$

$$\vdots$$

$$(-i)^{N-1} K_{N-1} \left(\frac{r_{J\wedge(2^N-2)}}{r_J} e^{-i\left[\Delta\alpha_{J\wedge(2^N-2),J} + \gamma\cdot\Delta E_{J\wedge(2^N-2),J}\right]} + \dots \right)$$

$$(-i)^{N} K_{N} \left(\frac{r_{J\wedge(2^N-1)}}{r_J} e^{-i\left[\Delta\alpha_{J\wedge(2^N-1),J} + \gamma\cdot\Delta E_{J\wedge(2^N-1),J}\right]} \right) \Bigg], \tag{2.44}$$

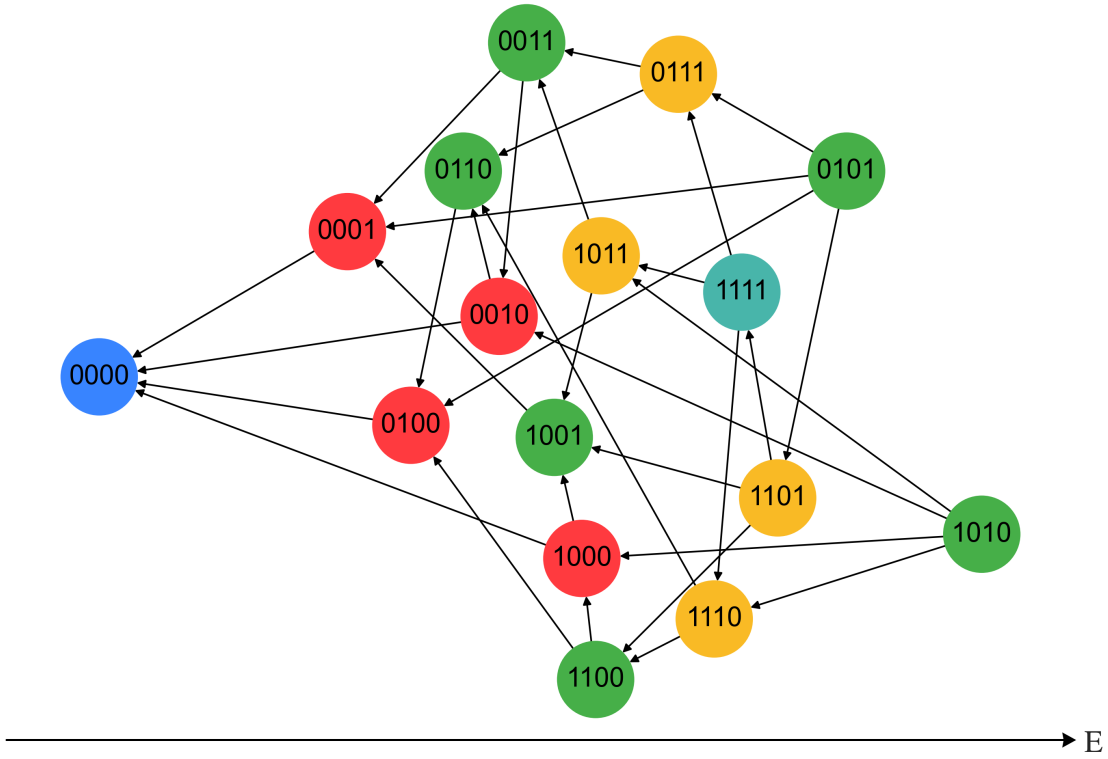


Figure 2.5.: The figure shows the graph structure of a 4 qubit exact cover problem. Each node denotes one computational basis state $|\Psi\rangle$ (possible solution strings to the optimization problem) and different colours are used to group the nodes by their Hamming weight. In contrast to Fig. 2.4, the nodes are horizontally positioned based on their assigned energy value $E = \langle\Psi|\hat{H}_C|\Psi\rangle$. Moreover, the graph is now oriented, such that all edges point in the direction of lower energy. Note that the solution node ($|0000\rangle$ here) has the smallest energy, yielding no outgoing connections.

2.3.2. Controlling the walker's movement

After having introduced the basic framework describing the movement of the quantum walker on a directed graph, the key idea of the GQW is to apply certain restrictions to the way the variational parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are selected, in order to ensure that the probability wave propagates along the directed connection in the graph. As a consequence, the guided quantum walk can be understood as a subspace of the more general QAOA, in which it is hoped that optimal combinations of variational parameters can be obtained more easily, hence significantly reducing the number of circuit evaluations in the optimization phase.

Spread of the quantum walker: The first restriction can be motivated by demanding that the guided quantum walk should induce a probability transfer to the solution state, hence it must accumulate probability into states near $|Z_{gs}\rangle$ (using the state's energy as a distance metric). Defining the *neighbourhood* of the solution state of size ΔE as the subset of all states that feature a maximum energy difference of ΔE to $|Z_{gs}\rangle$, the algorithm must alter the probability distribution in such a way that the majority of probability is located in a neighbourhood whose size shrinks as a function of the number of iterations p. Therefore, the quantum walker, initially distributed equally among the nodes, gets steered step by step towards the solution state, hence improving the approximate solution with every iteration. Consequently, the driving evolution $\hat{U}_M(\beta)$ must provide two properties: (1) the algorithm must provide a mechanism to *accumulate probability* in low energy states, and (2) states of different energies must be able to exchange probability. The former can be achieved via the interaction coefficient $K_0(\beta)$. Regarding the latter, the guided quantum walk considers the coefficient $K_1(\beta)$, enabling probability transfers between basis states of Hamming distance one. Thus, the first restriction requires:

$$\beta_i \in \left[\pi - \arctan\sqrt{\frac{1}{N-1}},\ \pi + \arctan\sqrt{\frac{1}{N-1}}\right], \tag{2.45}$$

with $P_0 = \pi$ and $P_1^{\pm} = \pi \pm \arctan\sqrt{\frac{1}{N-1}}$ being the peak positions of $K_0(\beta)$ and $K_1(\beta)$, respectively (see e.g. Fig. I.9 in appendix I). Note that $K_0(\beta) > 0.5$ is fulfilled for any β in the proposed parameter range, since the peaks of $K_0(\beta)$ and $K_1(\beta)$ overlap. This is a necessary condition in order to provide a probability movement while simultaneously collect probability in low energy states. For example, using $K_N(\beta)$, with a peak located at $\beta = \pi/2$, instead of $K_1(\beta)$ to exchange probability between basis states, the algorithm would prohibit the accumulation of probability, since maximizing K_N causes $K_0 = 0$. Hence, considering a system that already has a large overlap ($P_{gs} > 0.5$) with Z_{gs}, applying the driving evolution would cause a probability exchange between $\psi_{Z_{gs}} \leftrightarrow \psi_{Z_{gs}\wedge(2^N-1)}$, thus reducing the success probability P_{gs}. In this case, the quantum walker would move away from the solution node and the algorithm would fail to guide the walker sufficiently. Although optimal combination of variational parameters using $\beta_i \in [0, \pi)$ exist, as demonstrated by the QAOA, the situation just described, shows the importance of the coefficient $K_0(\beta)$ with respect to the concentration of probability near Z_{gs} (quality of the approximate solution). While the maximization of $K_{l \geq N/2}(\beta)$ generally suppresses $K_0(\beta)$, intermediate interaction orders $K_{1 < l < N/2}(\beta)$, featuring $K_0 > 0$, are also unsuitable here. This is because the maximum amplitude of $K_{l < N/2}$ decreases for increasing l, which can be explained by the increasing number of nodes each basis state is connected to. Therefore, the probability initially located at one basis state is spread across a greater number of nodes within one iteration, as neighbouring interaction orders feature similar amplitudes (see e.q. $t_D = 5/4\pi$ in Fig. I.9 in appendix I), making predictions about the walker's evolution/position more difficult. However, as this will be important in the design of heuristic models based on the GQW in section 2.3.3, $K_1(\beta)$ is chosen here. In doing so, the algorithm is able to essentially tune the spreading of the quantum walker in the directed graph by changing the relative strength of the two coefficients using β_i. Here, $\beta_i = \pi$ keeps the system stationary and $\beta_i = P_1^+$ maximizes interactions within it. Note that this restriction of $\boldsymbol{\beta}$ can be understood as a first order Taylor approximation of the driving evolution $\hat{U}_M(\beta)$ (see Eq. 2.47). Therefore, it ensures graph locality by preventing long range interactions (tunnelling) in the system, yielding a coinless discrete quantum walk (see section 2.2.4) within a continuous time evolution:

$$e^{-i\cdot\hat{H}_M\cdot\beta} = \prod_{i=0}^{N-1}\left[\cos(\beta)\,\mathbb{I} - i\sin(\beta)\,\hat{\sigma}_i^x\right] \tag{2.46}$$

$$\overset{\beta\approx\pi}{=} \cos^N(\beta)\,\mathbb{I} - i\cos^{N-1}(\beta)\sin(\beta)\left[\hat{\sigma}_0^x + \ldots + \hat{\sigma}_{N-1}^x\right] + \mathcal{O}\left(\sin^2(\beta)\right). \tag{2.47}$$

Single directed evolution: After having introduced $K_0(\beta)$ and $K_1(\beta)$ as the two mechanism to control the spread of the quantum walker across the graph, next the directed transfer of probability between connected nodes will be discussed. Since all interactions in the system can be considered as *independent operations*, achieved by the same interference mechanism $K_1(\beta)$, it is sufficient to focus on a single interaction (connection between two nodes in the graph). Therefore, consider a system initialized in an equal superposition of two basis states $|A\rangle$ and $|B\rangle$ with Hamming distance one ($|\Psi\rangle = \frac{1}{\sqrt{2}}\left[|A\rangle + |B\rangle\right]$ using $\psi_A = \psi_B = 1/\sqrt{2}$). Applying one iteration of the guided time evolution yields the transformation of the state coefficients $\psi_A = \langle A|\Psi\rangle$ and $\psi_B = \langle B|\Psi\rangle$ according to:

$$\psi_A = \frac{1}{\sqrt{2}} \to \frac{1}{\sqrt{2}}\, e^{-i\,\gamma\cdot E_A}\left[\cos^N(\beta) - i\cos^{N-1}(\beta)\sin(\beta)\, e^{-i\gamma\cdot\Delta E_{B,A}}\right] \tag{2.48}$$

$$= \frac{1}{\sqrt{2}}\, e^{-i\,\gamma\cdot E_A}\Big[\cos^N(\beta) - \cos^{N-1}(\beta)\sin(\beta)\sin(\gamma\cdot\Delta E_{B,A}) - i\cos^{N-1}(\beta)\sin(\beta)\cos(\gamma\cdot\Delta E_{B,A})\Big], \tag{2.49}$$

$$\psi_B = \frac{1}{\sqrt{2}} \to \frac{1}{\sqrt{2}}\, e^{-i\,\gamma\cdot E_B}\left[\cos^N(\beta) - i\cos^{N-1}(\beta)\sin(\beta)\, e^{+i\gamma\cdot\Delta E_{B,A}}\right] \tag{2.50}$$

$$= \frac{1}{\sqrt{2}}\, e^{-i\,\gamma\cdot E_B}\Big[\cos^N(\beta) + \cos^{N-1}(\beta)\sin(\beta)\sin(\gamma\cdot\Delta E_{B,A}) - i\cos^{N-1}(\beta)\sin(\beta)\cos(\gamma\cdot\Delta E_{B,A})\Big], \tag{2.51}$$

with $E_A = \langle A|\hat{H}_C|A\rangle$ and $E_B = \langle B|\hat{H}_C|B\rangle > E_A$ denoting the state energies and $\Delta E_{B,A} = E_B - E_A > 0$. Note that in the following discussion the *real* parts of the complex numbers in the square brackets of Eq. 2.49 and Eq. 2.51 will be called *in-phase amplitudes* and the *imaginary* parts are termed *out-of-phase amplitudes*. The goal of the guided quantum walk is to cause a probability transfer from ψ_B to ψ_A, since $|A\rangle$ features a smaller energy then $|B\rangle$, bringing it closer to the solution state Z_{gs}. This is achieved by increasing (decreasing) the complex amplitude of ψ_A (ψ_B) during the evolution. Since Eq. 2.49 and Eq. 2.51 differ only in their in-phase amplitudes, the probability exchange operates on the real parts of the two state coefficients. This approach is valid, since, on the one hand, $|K_0(\beta)| > |K_1(\beta)|$ for any $\beta \in [P_1^-, P_1^+]$ (see Eq. 2.45), and on the other hand, both states feature the same initial complex value (i.e. $\psi_A = \psi_B$). Without loss of generality, $\gamma_i > 0$ will be considered in the following, yielding $\sin(\gamma\cdot\Delta E_{B,A}) > 0$ for $\gamma\cdot\Delta E_{B,A} \in (0,\pi)$. As a consequence, $sgn(K_0(\beta)) = -sgn(K_1(\beta))$ is required to increase (decrease) the complex amplitude of ψ_A (ψ_B), yielding $\cos(\beta) < 0$ for both cases of even and uneven N (see e.g. Fig. I.9 in appendix I). Here, $sgn(A)$ denotes the sign-function. Hence, choosing $\beta_i \in [P_1^-, \pi]$, a net transfer of probability towards $|A\rangle$ is achieved, assigning a direction to the interaction between the two states. Note that $\beta \in [\pi, P_1^+]$ in combination with $\gamma < 0$ is also a valid choice. Consequently, the driving parameters $\boldsymbol{\beta}$ must further be restricted to:

If $\gamma_i > 0$**:**
$$\beta_i \in \left[\pi - \arctan\sqrt{\frac{1}{N-1}},\ \pi\right], \tag{2.52}$$

If $\gamma_i < 0$**:**
$$\beta_i \in \left[\pi,\ \pi + \arctan\sqrt{\frac{1}{N-1}}\right]. \tag{2.53}$$

Multiple directed evolutions: After having introduced the main mechanism for probability transport in the system, the next step is to consider multiple iterations of the guided walk and their influence on the state amplitudes. In contrast to the aforementioned case in Eq. 2.49 and Eq. 2.51, multiple time evolutions will cause initial phase and amplitude differences, that can counteract the transfer process into lower energy states. This can be accounted for by introducing the phase and amplitude parameters $\alpha_{\{A,B\}}$ and $r_{\{A,B\}}$, respectively, which can be understood as the *history* of the state amplitudes:

$$\psi_A = r_A\, e^{-i\,\alpha_A} \tag{2.54}$$
$$\begin{aligned}\to r_A\, e^{-i\,(\alpha_A + \gamma\cdot E_A)}\Big[&\cos^N(\beta)\\ &- \frac{r_B}{r_A}\cos^{N-1}(\beta)\sin(\beta)\sin(\Delta\alpha_{B,A} + \gamma\cdot\Delta E_{B,A})\\ &- i\,\frac{r_B}{r_A}\cos^{N-1}(\beta)\sin(\beta)\cos(\Delta\alpha_{B,A} + \gamma\cdot\Delta E_{B,A})\Big],\end{aligned} \tag{2.55}$$

$$\psi_B = r_B\, e^{-i\,\alpha_B} \tag{2.56}$$
$$\begin{aligned}\to r_B\, e^{-i\,(\alpha_B + \gamma\cdot E_B)}\Big[&\cos^N(\beta)\\ &+ \frac{r_A}{r_B}\cos^{N-1}(\beta)\sin(\beta)\sin(\Delta\alpha_{B,A} + \gamma\cdot\Delta E_{B,A})\\ &- i\,\frac{r_A}{r_B}\cos^{N-1}(\beta)\sin(\beta)\cos(\Delta\alpha_{B,A} + \gamma\cdot\Delta E_{B,A})\Big].\end{aligned} \tag{2.57}$$

First, consider the effect of the initial amplitude difference ($r_A \neq r_B$) between the two basis states. In the case of $E_A < E_B$, $r_A > r_B$ is assumed, since the guided quantum walk (at least initially) causes a probability flow towards the lower energy state $|A\rangle$. A main assumption of the interference mechanism introduced above is that $|K_0(\beta)| > |K_1(\beta)|$ is fulfilled for any parameter combination $\boldsymbol{\beta}$ within the proposed bounds. Thus, by achieving $sgn\,(K_0(\beta)) = -sgn\,(K_1(\beta))$, the algorithm decreases the probability of $|B\rangle$. However, due to the introduction of amplitude differences, a rescaling of $K_1(\beta)$ based on the quotient $r_{A,B} = \frac{r_A}{r_B}$ of the two complex amplitudes occurs. As a consequence, there exists some $r > 1$, such that for any $r_{A,B} \leq r$ an upper-bound $\beta \in [P_1^-, \pi]$ can be found, such that for any $\beta_i \leq \beta$, $|K_0(\beta_i)| < r_{B,A} \cdot |K_1(\beta_i)|$ is given. If this is the case, then the application of the time evolution $\hat{U}_M(\beta)$ will yield a negative in-phase amplitude for the higher energy state $|B\rangle$. Since both the in-phase and the out-of-phase amplitudes go quadratically into the calculation of the state's probability, an unintentional increase in the probability of $|B\rangle$ can be the consequence. In order to prevent this counteracting process, $\beta_i > \beta$ is required throughout all iterations $i < p$ of the guided walk. Due to this, the optimal transfer rates $\boldsymbol{\beta}$ depend on the number of total iterations p, since $r_{A,B}$ increases as more successful probability exchanges are

performed on the system. Thus, for large p, it is to be expected that β_i will approach π (see e.g section 2.4.1). This behaviour can be interpreted, as the system reducing its dynamics ($K_1(\beta)$) and shifting its focus more towards the exchange of smaller probability fractions. In that sense, the system operates slower but more carefully, by keeping larger fractons of probability stationary in lower energy states ($K_0(\beta)$). Moreover, the derived restriction for $\boldsymbol{\beta}$ also limits the out-of-phase amplitude, keeping the controllable in-phase amplitude as the major contribution to the state's probability, hence validating the interference mechanism also in the case of multiple iterations.

Secondly, the effect of the initial phase difference $\Delta\alpha_{B,A} = \alpha_B - \alpha_A$ has to be investigated. The main problem introduced here is that the total phase gradient $\Delta\Phi_{B,A} = \Delta\alpha_{B,A} + \gamma \cdot \Delta E_{B,A}$, which is used to guide the probability flow (direction of the edges in the graph) towards lower energy states, now depends on the history of the states. Thus, in case the initial phases α^i cause a total phase gradient beyond $(0, \pi)$, counteracting processes emerge, inverting the direction of the probability flow into higher energy states. An example of this process can be seen in Fig. 2.8 in section 2.3.4. Since $\Delta\alpha_{B,A}$ is the result of several iterations of guided walks, it is useful to look at the development of the initial phase gradient (i.e. $\Delta\alpha^i_{B,A} \rightarrow \Delta\alpha^{i+1}_{B,A}$) caused by one iteration of the $\hat{U}_C$ and $\hat{U}_M$ evolutions:

$$\Delta\alpha^{i+1}_{B,A} = -(\underbrace{\Delta\alpha^i_{B,A} + \gamma \cdot \Delta E_{B,A}}_{\Delta\Phi^i_{B,A}}) + \Delta\Theta^i_{B,A}, \tag{2.58}$$

$$\begin{aligned}\Delta\Theta^i_{B,A} = \quad & \arg\left[\frac{\cos(\beta)}{\sin(\beta)} \cdot \frac{r_B}{r_A} + \sin\left(\Delta\Phi^i_{B,A}\right) - i\cos\left(\Delta\Phi^i_{B,A}\right)\right] \\ - \ & \arg\left[\frac{\cos(\beta)}{\sin(\beta)} \cdot \frac{r_A}{r_B} - \sin\left(\Delta\Phi^i_{B,A}\right) - i\cos\left(\Delta\Phi^i_{B,A}\right)\right],\end{aligned} \tag{2.59}$$

with arg denoting the complex phase. A three-dimensional plot of Eq. 2.58 is given in Fig. 2.6 using $\beta = 3.0$, with $\Delta\Phi^i_{B,A}$ and $r^i_{B,A} = r^i_B \,/\, r^i_A$ denoting the function variables at iteration i. In order to investigate the evolution of the total phase gradient $\Delta\Phi^i_{B,A}$ throughout multiple iterations, it is useful to first consider the case of $\gamma_i = 0$ to understand how the system behaves without external influences. Assuming an initial phase difference $\Delta\alpha^0_{B,A} \in (-\pi, 0)$, a probability transfer according to the aforementioned interference mechanism will occur. Since $\Delta\Phi^0_{B,A} = \Delta\alpha^0_{B,A}$, the direction of the probability flow is from the higher energy state $|B\rangle$ towards the lower energy state $|A\rangle$. As the system is initialized in the equal superposition of both states, $r^0_{B,A} = 1$ is given. This results in a linear dependency between $\Delta\alpha^1_{B,A}$ and $\Delta\Phi^0_{B,A}$, yielding $\Delta\alpha^1_{B,A} = \Delta\Phi^0_{B,A}$. Continuing with the second iteration, another probability transfer towards $|A\rangle$ is performed, as the total phase difference between the two states did not change. In doing so, the quotient $r_{B,A}$ between the complex amplitudes ψ_A and ψ_B decreases, causing the system to move along a path on the surface of Fig. 2.6. The shape of this path depends on the initial starting point $\Delta\alpha^0_{B,A}$. As a consequence, the relationship between $\Delta\alpha^{i+1}_{B,A}$ and $\Delta\Phi^i_{B,A}$ deviates from the initial linear dependency, causing a change in the total phase gradient $\Delta\Phi^i_{B,A}$ as more iterations are applied (i.e. the value of $r^i_{B,A}$ shrinks). Hence, $\Delta\Phi^i_{B,A} = \Delta\alpha^i_{B,A} \neq \Delta\Phi^{i-1}_{B,A}$ in general. By investigating the shape of the surface in Fig. 2.6, one can see, that the change in the relationship causes initial phase differences and consequently the total phase gradients to occupy values outside $(-\pi, 0)$ (*blue* area in Fig. 2.6). Since, $\Delta\Phi^i_{B,A} \in (-\pi, 0)$ is required by the aforementioned interference mechanism to direct the

probability flow into lower energy states, counteracting processes will occur that transport probability back into higher energy states. This, however, causes an increase in the amplitude quotient $r^i_{B,A}$, allowing the system to eventually recover $\Delta\Phi^i_{B,A}$ and move back into the *blue* area. As a consequence, an oscillation of the system's probability between the two states will occur (see e.g. Fig. 2.8 in section 2.3.4). Therefore, the variational parameters $\boldsymbol{\gamma}$ are necessary, on the one hand, to cause an initial phase gradient $\Delta\Phi^0_{B,A}$ in the first place, and on the other hand, to compensate for the changing relationship between $\Delta\alpha^{i+1}_{B,A}$ and $\Delta\Phi^i_{B,A}$. The latter is needed to ensure a sufficient guidance of the walker throughout the iterations by always shifting the total phase difference $\Delta\Phi^i_{B,A}$ between $(-\pi, 0)$. In doing so, two observations can be made: First, consider a single connection between two states in the graph. Since $\Delta E_{B,A}$ is fixed in this case, $\Delta\Phi^i_{B,A}$ can be precisely controlled via γ_i. Thus, as the deviations of $\Delta\alpha^{i+1}_{B,A}$ from $\Delta\Phi^i_{B,A}$ increase, an increase in γ_i is required in order to assure a correct guidance of the walker. Secondly, the phase shift achieved by γ_i depends linearly on the energy difference $\Delta E_{B,A}$. Therefore, when considering all connections present in the system, the phase shifts can only be adjusted to subsets of all connections, which feature similar energy gradients. Interactions, on the other hand, with lower or higher energy differences will gain significantly smaller or larger phase shifts, respectively, potentially driving counteracting probability transfers into higher energy states. Due to this, rather than achieving directed probability exchanges between all pairs of connected states, the guided quantum walk can only focus on transfers within certain energy ranges at a time.

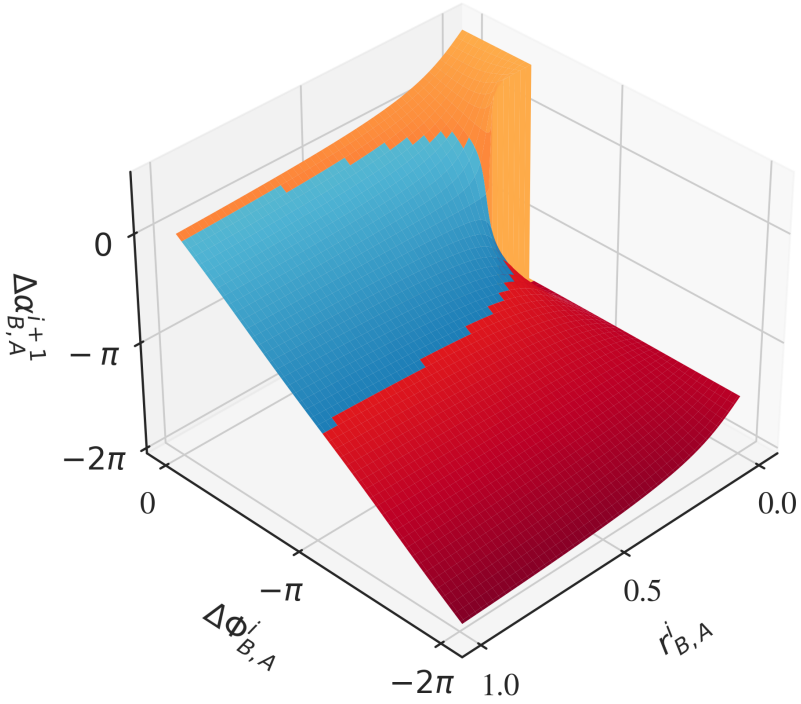


Figure 2.6.: The plot shows the initial phase gradient $\Delta\alpha^{i+1}_{B,A}$ between two interacting states after one iteration as a function of the total phase difference $\Delta\Phi^i_{B,A}$ and the complex amplitude quotient $r^i_{B,A}$ (see Eq. 2.58). Note that $\beta = 3.0$ and $\Delta E > 0$ are used. Moreover, the parameter space corresponding to $\Delta\alpha^{i+1}_{B,A} \in [-\pi, 0]$ is marked *blue*.

2.3.3. Heuristic model

In the previous two sections, the basic framework of the guided quantum was introduced, yielding a novel strategy for obtaining approximate solutions to combinatorial optimization problems using the concept of a coinless discrete quantum walker propagating on an oriented graph. In doing so, the GQW can be understood as an instance of the general QAOA, subject to several restrictions that govern the choice of the variational parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. Based on these restrictions, a heuristic model termed *HGQW* will be developed in this section, allowing to select parameter sets according to the walker's movement proposed in section 2.3.2. Its derivation process will be structured into two steps, by first considering the unrealistic situation where all information about a problem instance is available. This includes especially the solution state as well as the distribution of the computational basis states across the individual energy levels, for which the problem has to be solved beforehand. While not being applicable to the real world, the goal of this investigation is to gain further insights into the distribution of optimal sets of variational parameters following the principles of a guided quantum walk. In doing so, the task of finding $2p$ parameters will be mapped to certain functions based on the problem's properties, which will then be approximated, yielding the HGQW model. Note that the following considerations will focus on exact cover problem instances as introduced in section 2.1.1. This is done as a first step, because the interference mechanism used by the guided quantum walk to direct the probability flow in the graph is entirely based on the state's energy as a metric to distinguish connected states. Since, the exact cover problems feature numerous distinct energy levels with low degeneracies, it was found that these instances are particularly well suited for the GQW. 2-SAT problem instances, involving a high degeneracy of only a few unique energy levels (see section 2.1.2), will be treated in section 2.3.5.

The HGQW model is composed of two functions $\beta(i, \boldsymbol{\lambda})$ and $\gamma(i, \boldsymbol{\lambda})$, with $i < p$ denoting the index of iteration and $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_M\}$ being a fixed set of M additional parameters. Their goal is to determine sets of variational parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ which follow closely an optimal route through the graph according to the dynamics discussed in section 2.3.2. This approach is inspired by the observations of [9, 67, 83, 92, 93, 106] who found that general patterns in the optimal distribution of variational parameters across different problem instances can occur in the QAOA. However, till today, there is not much understanding in why these patterns exist and how they can be obtained efficiently [92]. Most approaches presented in previous work focussing on the derivation of optimal parameter sets either rely on the application of machine learning algorithms [70, 94, 95, 107, 108] or iterative procedures [67, 96, 106, 109]. Here, the latter use pre optimized sets of parameters (usually obtained via brute-force search) for small $p \approx 4$ and then iteratively extrapolate them to larger p. While these attempts work well on small problem instances that require only a few QAOA iterations, they quickly become infeasible as the number of qubits N or the number of iterations p increase. Regarding iterative procedures, this can be seen as the number of optimization runs increases linearly in p, while the search space grows exponentially with each run. Neural networks, on the other hand, need an initial training phase, requiring numerous optimized parameter sets for various problem instances. As p and N increase, it becomes exponentially harder to obtain such training sets, since for example the solution states for the NP-complete problems must be determined. Moreover, long access times to quantum computers (or simulators) are necessary for training. The main problem when optimizing variational parameters in the QAOA is that the observed patterns can in principle follow any function, as the exponentially growing search space is complex and non-convex, featuring numerous local minima. As a consequence, localizing the global minimum or at least *good enough* local minima, providing a sufficient

success probability, is in general a difficult task [30]. The advantage of the HGQW model is now that instead of performing a *blindfolded* search in the parameter space, the knowledge about the dynamics of the guided quantum walk are incorporated into the optimization process. This allows reducing the search to certain patterns of the variational parameters, which are then adjusted to the individual problem instances by tuning the set of additional parameters $\boldsymbol{\lambda}$. The latter is done using the Nelder-Mead optimizer [110] in an outer-classical optimization loop based on the minimization of $\langle\Psi|\hat{H}_C|\Psi\rangle$ (the procedure is identical to Fig. 2.3). Since the number M of parameters will be fixed, the complexity of the optimization process, for obtaining approximate solutions, becomes *constant*, compared to *exponentially* growing in p. Note that the AQA (see section 2.2.3) resembles a similar approach of reducing the optimization work required by the QAOA by exploiting its connection to continuous quantum annealing. A comparison between the QAOA, the AQA and the HGQW will be given in section 2.4.3.

Spread coefficients $\boldsymbol{\beta}$: The distribution of the variational parameters $\boldsymbol{\beta}$, controlling the overall spread of the quantum walker among the nodes, is restricted to $[P_1^-, \pi]$ using $\boldsymbol{\gamma} > 0$. Given the discussion in section 2.3.2, this parameter range is necessary to ensure graph locality and hence to allow for a sufficient accumulation of probability in low energy states. The HGQW model considers here a constant distribution of $\boldsymbol{\beta} = \lambda_1$, providing a plain movement of the walker across the graph that is independent of the problem's properties. In doing so, the propagation of the probability wave is entirely altered by the phase coefficients $\boldsymbol{\gamma}$, hence reducing the among of interplay between the two transformations ($\hat{U}_M(\beta_i)$ and $\hat{U}_C(\gamma_i)$) and hopefully simplifying the classical optimization process. This ansatz is motivated by two observations: (1) Tuning the optimization parameter λ_1 separately for each of the 48 exact cover instances reveals almost identical distributions of the spread coefficients $\boldsymbol{\beta}$ that are independent of the problem's size or structure (see Fig. 2.12b in section 2.4.1). Although the set of investigated problem instances is limited, this suggests that (at least in the GQW) the variational parameters $\boldsymbol{\beta}$ resemble a universal property of the quantum walk, which seem to only depend on the total number of iterations p (see section 2.4.1). (2) Moreover, the system is initialized in the equal superposition state. Thus, all connections in the graph will contribute to the walker's movement at every iteration i. Hence, it is not possible to follow a classical path through the graph, as all paths must be considered simultaneously. Therefore, it is impractical to control the probability transfer rate between two nodes, but rather use the phase coefficients $\boldsymbol{\gamma}$ to alter the interference processes individually. Consequently, a fixed value is proposed for $\boldsymbol{\beta}$:

Spread coefficients:

$$\beta\,(i, \boldsymbol{\lambda}) = \lambda_1, \tag{2.60}$$

$$\lambda_1 \in \left[\pi - \arctan\sqrt{\frac{1}{N-1}},\ \pi\right]. \tag{2.61}$$

Phase coefficients $\boldsymbol{\gamma}$: The variational parameters $\boldsymbol{\gamma}$ are used as a tool to guide the spread of the quantum walker across the graph. By introducing complex phase differences $-\gamma_i \cdot \Delta E_{B,A}$ between connected nodes (A, B), the algorithm is able to alter the interference mechanism and thus control the amount of probability that is transferred between pairs of basis states. As mentioned in section 2.3.2, a main property of this mechanism is its *locality*, meaning that the

alignment (direction of probability transfer) in the graph can only be controlled for subsets of all edges simultaneously. This is because the phase shift induced by γ_i is proportional to the energy difference $\Delta E_{B,A}$. Hence, $\gamma_i \cdot \Delta E_{B,A}$ must be within $(0, 2\pi)$, such that $\Delta\Phi^i_{B,A}$ can be set between 0 and π, yielding a probability transfer into lower energy states. As a consequence, edges governing small energy gradients $\Delta E_{B,A}$, i.e. $\gamma_i \cdot \Delta E_{B,A} << 1$, will effectively gain no additional phase shift by the cost evolution, instead they are entirely influenced by the initial phase differences $\Delta\alpha_{B,A}$. On the other hand, edges featuring $\gamma_i \cdot \Delta E_{B,A} >> 1$ will experience significant phase shifts. However, since the total phase $\Delta\Phi^i_{B,A}$ is taken modulo 2π, the resulting phase gradient can be anywhere within 0 and 2π. In both cases, the algorithm has no direct control over the dynamics of the interactions, potentially causing contradicting probability transfers and unintended changes to $\Delta\alpha_{B,A}$. As a result, the guided walk can quickly become uncontrollable, once the total phase gradients $\Delta\Phi_{B,A;i}$ in the system diverge. In order to prevent this, the HGQW model proposes a sequential *activation* of the edges based on their energy gradients ΔE, by monotonically increasing the strength of the phase coefficients γ_i. This ansatz is motivated by the observation that compensating probability transfers (into higher energy states) are based on the strength of the initial phase differences $\Delta\alpha_{B,A}$. Hence, beginning with a small γ_0, only edges of large energy gradients will gain sufficient phase differences, thus *enabling* directed probability exchanges, while all other edges will remain unaltered, leaving their initial phase gradients $\Delta\alpha_{B,A}$ intact. Next, $\gamma_1 > \gamma_0$ is chosen for the second iteration. On the one hand, this allows compensating the effects of the changing relationship between $\Delta\alpha^{i+1}_{B,A}$ and $\Delta\Phi^i_{B,A}$ as the probability gradient $r^i_{B,A}$ decreases (see section 2.3.2). On the other hand, it enables additional edges to participate in the probability transfer, as sufficient phase gradients can be achieved with smaller energy gradients. In doing so, the algorithm iteratively enables the probability exchange between states in decreasing order of their energy gradient, while keeping control over the initial phases of still disabled interactions. Thus, a *barrier*, corresponding to the aforementioned neighbourhood of the solution node, forms, which prevents high energy states from extracting probability from low energy states. Here, the underlying assumption is that the average energy gradient decreases monotonically for decreasing node energies and that the HGQW is able to transfer the probability towards Z_{gs} faster than the occurrence of compensating probability exchanges.

Based on the above considerations, the HGQW model proposes a sampling of γ_i inspired by the distribution of the energy gradients within the graph. The latter can be determined by first calculating the energy spectrum $E = \langle\Psi|\hat{H}_C|\Psi\rangle$ of all computational basis states $|\Psi\rangle$. Next, according to the hypercube mapping (see e.g. Fig. 2.5), the energy gradient $\Delta E_{B,A}$ of each edge in the graph is derived and stored in an array L. The latter is sorted based on the largest node energy involved in each connection (see *blue dots* in Fig. 2.7). As a result, choosing any element $\Delta E \in L$, its corresponding probability transfer can be enabled (assuming a neglectable initial phase gradient $\Delta\alpha$) via the phase coefficient $\gamma = \frac{\lambda_2}{\Delta E}$. Here, the idea is that the phase gradient $\gamma \cdot \Delta E = \lambda_2$ necessary to enable an edge does not depend on the node's energy (position in the graph) nor on the state of the quantum walk, but is identical for all connections. Note that this phase gradient is tuned via the optimization parameter λ_2. In order to sample the set of p variational parameters γ_i from L, the HGQW model uses a conservative ansatz by only considering the largest energy gradients (smallest phase coefficients) at each energy level. Moreover, to ensure a monotonically increasing distribution, each element at position i in L is further replaced by the largest element in the subset $\{L[j] \mid j \leq i\}$. This is done in order to postpone the appearance of compensating probability transfers, as otherwise $\gamma_i < \gamma_{i-1}$ might occur when sampling L in inverse

order. Note that L is sorted based on the largest energy level connected to an edge instead of its energy gradients, because the idea is that the guided quantum walk will focus on the extraction of probability from high energy states first, until continuing with intermediate and low energy states. In doing so, it serves the picture of a shrinking neighbourhood containing the majority of probability, such that the approximate solution should improve with every additional iteration. As a final step, a polynomial $f(E)$ of degree 6 is fitted to L (see *red* curve in Fig. 2.7) and the set of variational parameters is then sampled exponentially from $f(E_i) \cdot \lambda_2$ using $E_{max} \cdot (e^{-x_i} - 1) / (e^{-\lambda_3} - 1)$ with $x_i = i/(p-1)$ (see *red triangles* in Fig. 2.7). Here, λ_3 denotes an additional optimization parameter, which adjusts the exponential sampling speed. This non-uniform sampling of $f(E)$ is chosen, as the step size of the quantum walker in the energy domain decreases for decreasing energy gradients. Hence, the walker initially moves faster towards states of smaller energy than it does near the end of the algorithm, where the majority of probability is already located close to the solution state (see monotonically decreasing $f(E_{max} - E)$ in Fig. 2.7). An example of a set of phase coefficients obtained this way is depicted in Fig. 2.8 for the `EC_16_1` exact cover problem.

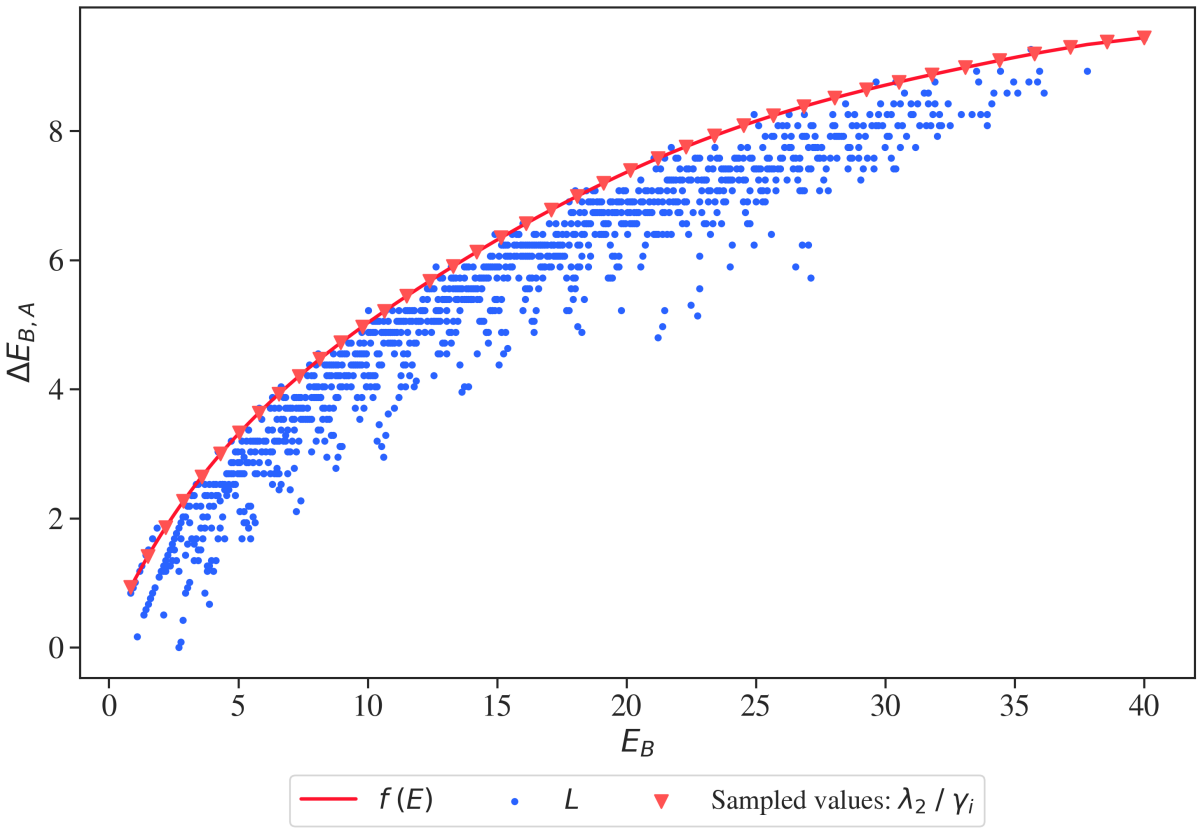


Figure 2.7.: The plot shows the distribution of the largest energy gradients $\Delta E_{B,A}$ as a function of the energy levels E_B (*blue* points). The *red* curve denotes a polynomial $f(E)$ of degree 6 that is fitted to L. $f(E)$ is used for sampling the variational parameters λ_2 / γ_i. An example for $p = 40$ is presented by the *red* triangles. The data shown is obtained for the `EC_16_1` exact cover problem.

A main observation from the strategy depicted above is that the function $f(E)$ has a similar shape across all investigated exact cover instances. As a consequence, the HGQW model tries to approximate $f(E)$ in order to make the guided quantum walk accessible to real-world problem, where the energy spectrum is generally not available. Due to the logarithmic shape of $f(E_i)$ and the exponential sampling method of E_i, $\gamma_i = \lambda_2 \,/\, (1 - x_i)$ using $x_i = i \,/\, (p - 1)$ is considered in the following. In addition to this, three optimization parameters, λ_3, λ_4 and λ_5, are also introduced into the model, in order to separately adjust the exponential scaling of $\gamma(i, \boldsymbol{\lambda})$ for small, intermediate, and high x-ranges, respectively, yielding:

Phase coefficients:

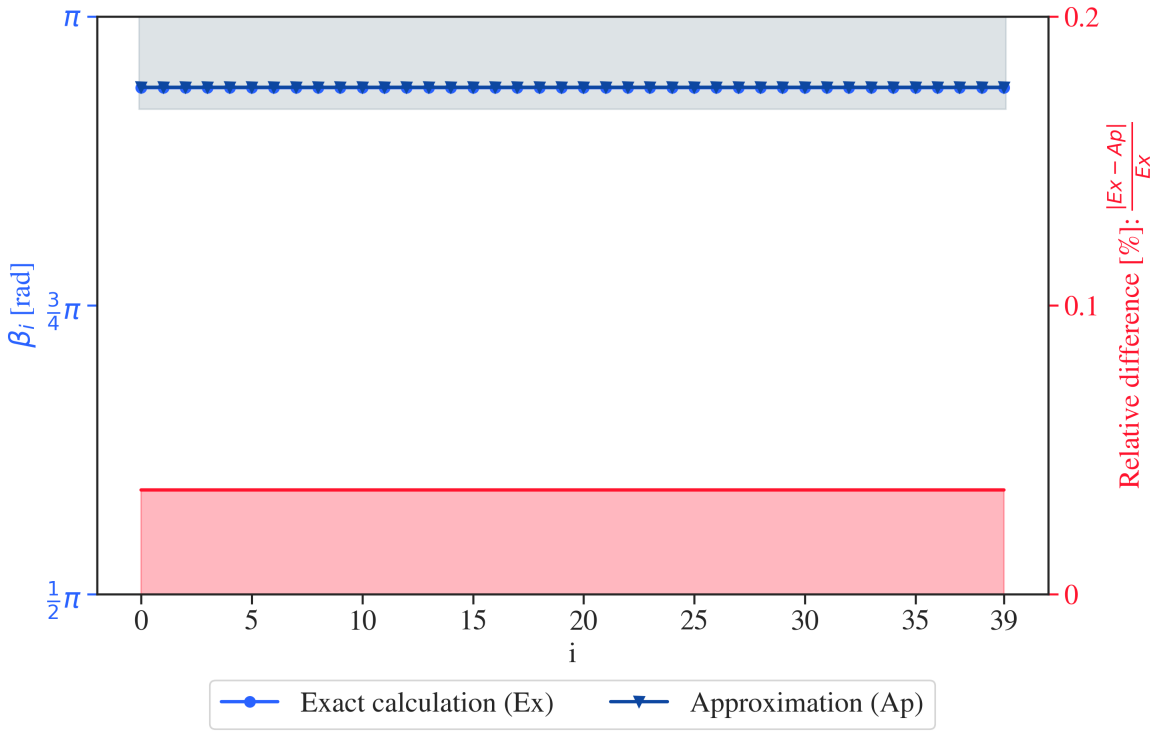
$$\gamma(i, \boldsymbol{\lambda}) = \frac{\lambda_2}{\frac{1}{\lambda_3 \cdot x_i + \lambda_4} - \frac{\lambda_5}{\lambda_3 + \lambda_4}} \qquad \text{with} \qquad x_i = \frac{i}{p-1}, \tag{2.62}$$

$$\lambda_1 \in \left[\pi - \arctan\sqrt{\frac{1}{N-1}},\ \pi\right], \qquad \lambda_2 > 0, \qquad \lambda_3 > 0, \tag{2.63}$$

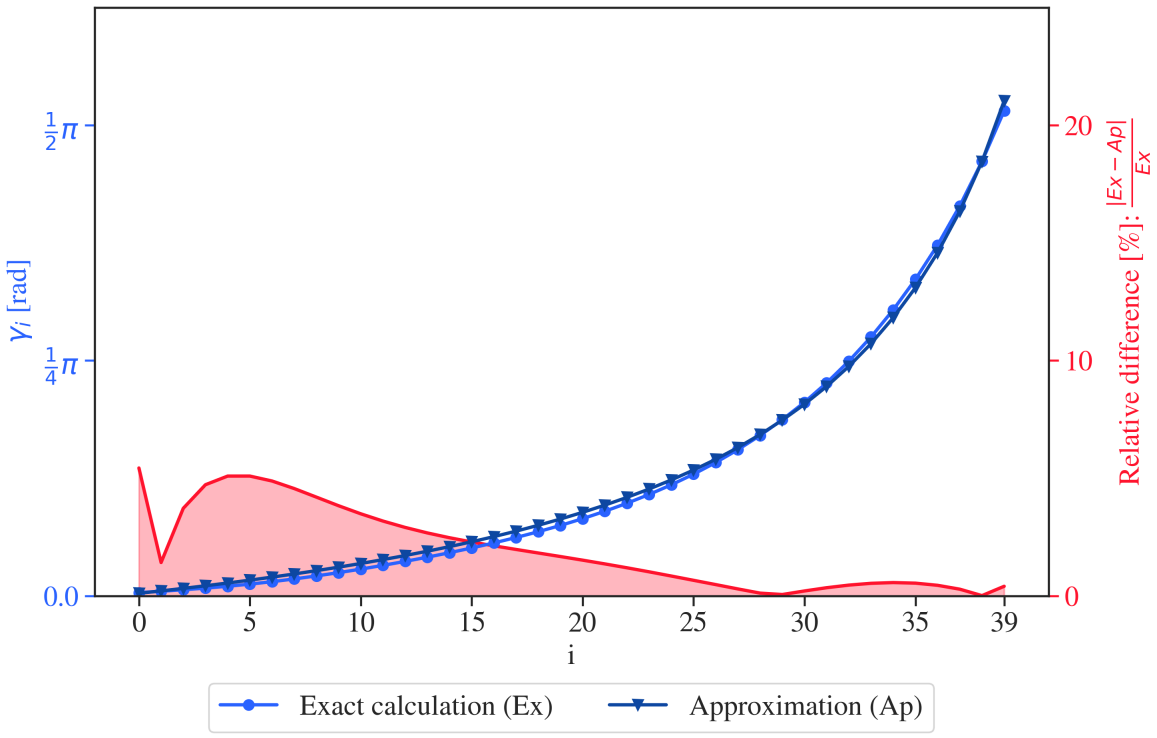
$$\lambda_4 \in (0, 1], \qquad \lambda_5 \in [0, 1). \tag{2.64}$$

Note that a naive strategy of approximating $f(E)$ via a polynomial of degree 6 is generally impractical, due to the way the individual parameter sets $\boldsymbol{\lambda}$ get evaluated. Here, the quality of the polynomial can only be rated by the overall success probability/energy expectation at the end of the algorithm. Since a polynomial of degree 6 can describe functions of various different shapes and the parameter search space is generally highly non-convex, the optimization process is likely to tune $\boldsymbol{\lambda}$ into a local minimum that is not corresponding to a guided quantum walk. Hence, numerous optimization runs are typically necessary. Using Eq. 2.62, on the other hand, ensures a distribution based on the principles of a GQW, thus hopefully accelerating the classical optimization process.

A comparison between Eq. 2.62 and the exact distribution obtained via $f(E)$ for the `EC_16_1` exact cover problem is given in Fig. 2.8. Note that both models have been tuned using at most $1000 \cdot M$ optimization steps within 20 separate optimization runs, with the best (highest success probability) runs shown in the figure. In doing so, both model produce almost identical $\beta(i, \boldsymbol{\lambda})$ distributions, with relative differences of 0.036%, while the $\gamma(i, \boldsymbol{\lambda})$ coefficients agree within 13.58%. As a consequence, it can be concluded that the proposed approximation is able to sufficiently reproduce $f(E)$, thus realizing guided quantum walks. Also note that slight differences between the models do not necessarily go along with a decrease in the success probability. In fact, slightly higher success probabilities are usually found using the approximate model, which is probably due to the increased number of degrees of freedom compared to the exact approach.



(a) Spread coefficients $\beta\,(i, \boldsymbol{\lambda})$



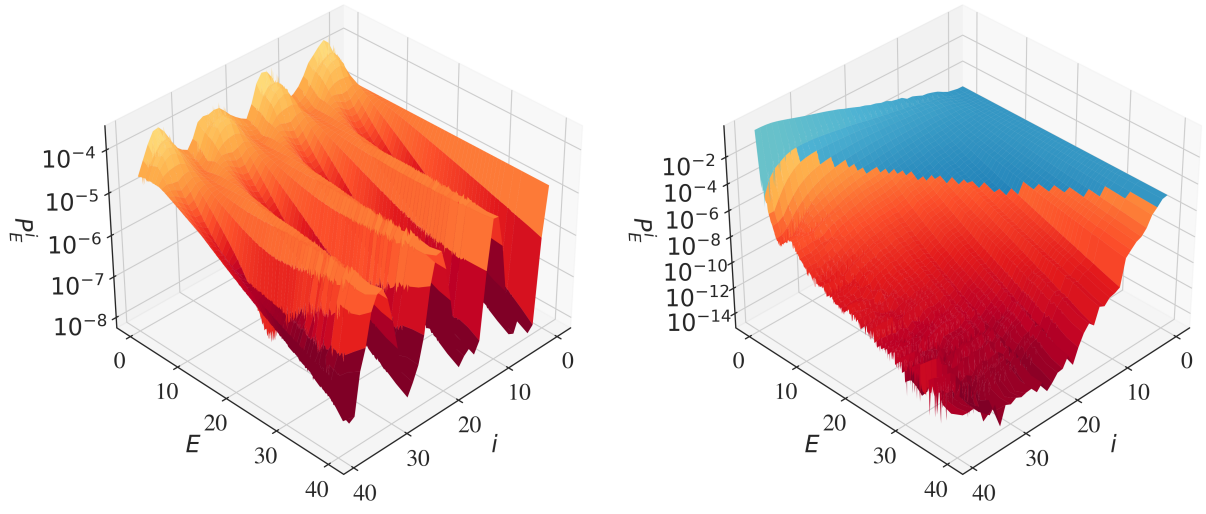
(b) Phase coefficients $\gamma\,(i, \boldsymbol{\lambda})$

Figure 2.8.: The plots show the sets of variational parameters $\beta\,(i, \boldsymbol{\lambda})$ and $\gamma\,(i, \boldsymbol{\lambda})$ determined by the HGQW model for the `EC_16_1` exact cover problem ($p = 40$). The model has been tuned twice using at most $1000 \cdot M$ optimization steps within 20 separate optimization runs (the best ones shown), considering the exact calculation (Ex, *light blue*) of $f\,(E)$ as well as its approximation (Ap, *dark blue*). The *red* curve indicates the relative differences ($|Ex - Ap| \,/\, Ex$) between both approaches, with maximum differences of 0.036% for $\beta\,(i, \boldsymbol{\lambda})$ and 13.58% for $\gamma\,(i, \boldsymbol{\lambda})$.

2.3.4. Dynamics of the HGQW model

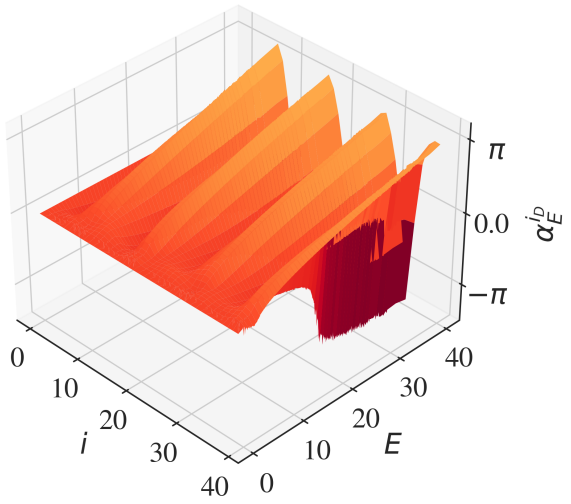
In the previous section, the HGQW model has been introduced as a heuristic approach for sampling sets of variational parameters $\beta(i, \boldsymbol{\lambda})$ and $\gamma(i, \boldsymbol{\lambda})$ based on common properties in the distributions of the energy gradients between connected states in the graph throughout various exact cover instances. Since several assumptions regarding the behaviour of a quantum system under certain parameter sets have been proposed during its derivation, the gradual evolution of a system based on the HGQW model will be studied in this section. This is done, by analysing the transformation of the full state vector $|\Psi^i\rangle = \sum_J \psi_J^i |J\rangle$ throughout the iterations $i < p$. Hence, snapshots of $|\Psi\rangle$ are taken after each time evolution, with i_M indexing state configurations after the driving evolution $\hat{U}_M(\beta_{i_M})$, and i_C referring to snapshots obtained after the cost evolution, $\hat{U}_C(\gamma_{i_C})$. In doing so, the measurement probabilities ($P_J^i = |\Psi_J^i|^2$), as well as the initial phases $\alpha_J^{i_M}$ and total phases $\Phi_J^{i_C}$ are captured within each iteration. Sufficiently controlling the latter is important, as it determines the ability of the HGQW model to guide the walker through the graph. Regarding the visualization, a projection of the aforementioned metrics onto the energy levels of $\hat{H}_C$ will be considered in the following. Thus, the states are first grouped by their respective energy level E, followed by calculating the mean value of each metric within each group. Choosing the energy domain for investigation is motivated by the fact that the guided quantum walk operates on energy levels not on individual basis states, hence causing global probability transfers based on the energy differences. Therefore, it is to be expected that states of the same energy govern similar dynamics, which is also encouraged by the regular distributions obtained for the three metrics (see e.g. Fig. 2.8). Note that the `EC_16_1` exact cover instance will be considered in this section.

Since the guided quantum walk shares a great connection to conventional quantum walks (see section 2.2.4), the dynamics of the HGQW model will be compared to the evolution of a quantum system exposed to an instance of a trotterized continuous quantum walk (CQW model). The CQW model uses the same sequence of unitary time evolutions, with an identical set of spread coefficients $\beta(i) = 2.948$ as the HGQW model and a set of constant phase coefficients $\gamma(i) = 0.05$. The latter corresponds to $\gamma(0, \boldsymbol{\lambda})$ in the HGQW model, hence allowing to investigate the influence of an exponential increase in the phase shifts on the evolution of a quantum walker.

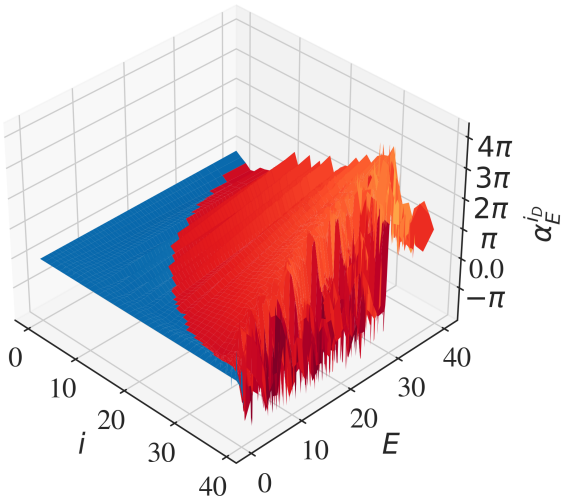


(a) CQW model: P_E^i

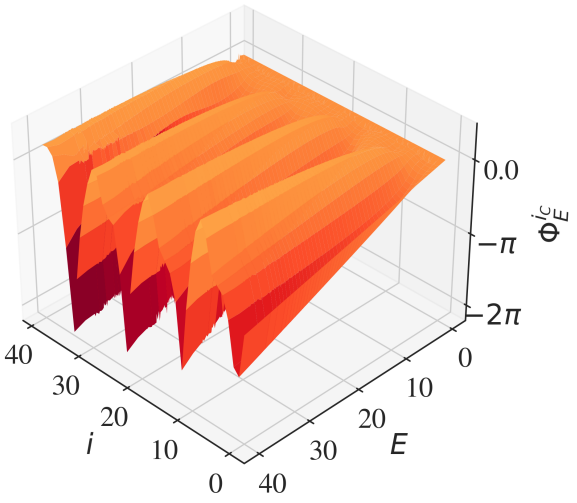
(b) HGQW model: P_E^i



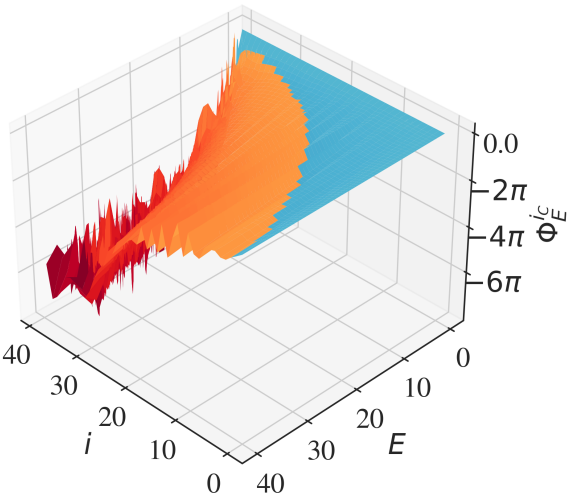
(c) CQW model: α_E^{iD}



(d) HGQW model: α_E^{iD}



(e) CQW model: Φ_E^{iC}



(f) HGQW model: Φ_E^{iC}

Figure 2.8.: The plots show the evolution of a quantum system under the influence of the CQW model (a, c, e) and the HGQW model (b, d, f) as a function of the iteration index i. The evolution is obtained by sampling the full state vector $|\Psi\rangle$ after each time evolution and grouping the computational basis states according to their energy E. In doing so, (a) and (b) present the measurement probabilities P_E^i, (c) and (d) show the initial phases α_E^{iD}, and (e) and (f) portray the total phases Φ_E^{iC}. Note that each metric is averaged over all states governed by an energy level. Moreover, data points (E, i) that feature $P_E^i/P_0^i \geq 10^{-2}$ are marked *blue* in (b), (d) and (f). The data is obtained for the `EC_16_1` exact cover problem using the parameter set shown in Fig. 2.8 for the HGQW model. The CQW model uses $\beta(i) = 2.948$ and $\gamma(i) = 0.05$, corresponding to $\beta(0, \boldsymbol{\lambda})$ and $\gamma(0, \boldsymbol{\lambda})$ of the HGQW model.

CQW model: The evolutions of the mean measurement probabilities P_E^i, the mean initial phases $\alpha_E^{i_M}$ and the mean total phases $\Phi_E^{i_C}$ as a function of the energy level E and the iteration index i are depicted in Fig. 2.9a, Fig. 2.8c and Fig. 2.8e, respectively. Note that P_{gs} does not reach unit probability, as the transfer rate $\tilde{\gamma} \propto \beta(i)/\gamma(i)$ is not optimized for the problem instance in question. However, the characteristic oscillation of the success probability in the iteration index i (i.e. the evolution time) is still present in Fig. 2.9a. The oscillation has an approximate period of $\Delta i = 10$ and can also be observed in the distribution of $\alpha_E^{i_M}$ and $\Phi_E^{i_C}$. The latter demonstrates the strong connection between the probability transport through the graph and the complex phase gradients across the basis states, as discussed in section 2.3.2. In general, Δi depends on the strength of the transfer rate, such that an increase in $\tilde{\gamma}$ causes a faster oscillation, since the quantum walker propagates faster through the graph. This behaviour can be understood as follows: Beginning at $i = 0$, the system is initialized in the equal superposition state $|+\rangle^{\otimes N}$, causing identical measurement probabilities P_E^0 and initial phases $\alpha_E^{0_M}$ at all energy levels E. The subsequent cost evolution $\hat{U}_C$ decreases the complex phase $\Phi_E^{0_C}$ at each energy level proportional to its energy E (see Fig. 2.8e). Hence, $\Phi_E^{0_C}$ becomes monotonically decreasing in E, such that the phase gradients $\Delta\Phi_{A,B}^{0_C} = \Phi_B^{0_C} - \Phi_A^{0_C}$ between arbitrary connected states (A, B) $(E_A < E_B)$ in the graph lay within $(-\pi, 0)$, yielding global probability transfers into lower energy states (see interference mechanism in section 2.3.2). This can be seen in Fig. 2.9a, as the average measurement probability of low energy states increases, while it decreases for high energy states. However, in doing so, the probability quotient $r_{B,A}^1 = \psi_B^1/\psi_A^1$ decreases, in turn causing phase shifts $\Delta\Theta_A^1 < \Delta\Theta_B^1$ via the driving evolution $\hat{U}_M$ (see Eq. 2.58). Since these phase shifts increase the relative complex phase of ψ_B with respect to ψ_A, the probability transfer counteracts the phase shifts given by the cost evolution $\hat{U}_C$ (see Fig. 2.6). As a consequence, at $i \approx 2$ both phase shifts become equal, yielding a minimum in Fig. 2.8c and Fig. 2.8e. For $2 < i < 5$, $\alpha_E^{i_M}$ starts to increase, causing a decrease in the phase gradients $\left|\Delta\Phi_{B,A}^{i_C}\right|$. Hence, the probability transfer through the graph slows down, as the interference interaction makes greater use of the *out-of-phase* amplitudes (see discussion in section 2.3.2). At $i \approx 5$, the control evolution is no longer able to compensate the initial phase gradients, causing $\Delta\Phi_{B,A}^{i_C} > 0$ and thus probability exchanges into higher energy states (see maximum of P_{gs} in Fig. 2.9a). However, this forces $r_{B,A}^i$ to increase again, reducing the phase gradients caused by the driving evolution and thus decelerating the increase in the initial phases (see Fig. 2.8c). At $i \approx 8$, $0.05 \cdot \Delta E_{B,A} \geq \Delta\Theta_{B,A}^i$, achieving a decrease in the initial phase (see maximum in Fig. 2.8c and Fig. 2.8e). At $i \approx 10$, $\Delta\Phi_{B,A}^{i_C}$ becomes negative again, guiding the probability flow back into lower energy states. Hence, the above process repeats itself.

HGQW model: The CQW model demonstrates, that the interplay between the complex phase shifts caused by the driving and cost evolutions is the limiting factor for guiding the probability flow towards the solution state. This is because, once the probability quotients $r_{B,A}^i$ of connected states become too small, $\gamma_i \cdot \Delta E_{B,A} = const$ cannot compensate $\Delta\Theta_{B,A}^i$ any more, hence inverting the interference mechanism. To prevent this, the HGQW model proposes an exponentially increasing parameter set $\gamma(i, \boldsymbol{\lambda})$, which is hoped to compensate the increase in $\alpha_E^{i_M}$, such that $\Delta\Phi_{B,A}^{i_C} < 0 \ \forall \ i_C$. The evolution of a quantum system under the influence of the HGQW model is depicted in Fig. 2.8 (b, d, f), regarding the mean measurement probability P_E^i, the mean initial phases $\alpha_E^{i_M}$ and the mean total phases $\Phi_E^{i_C}$, respectively.

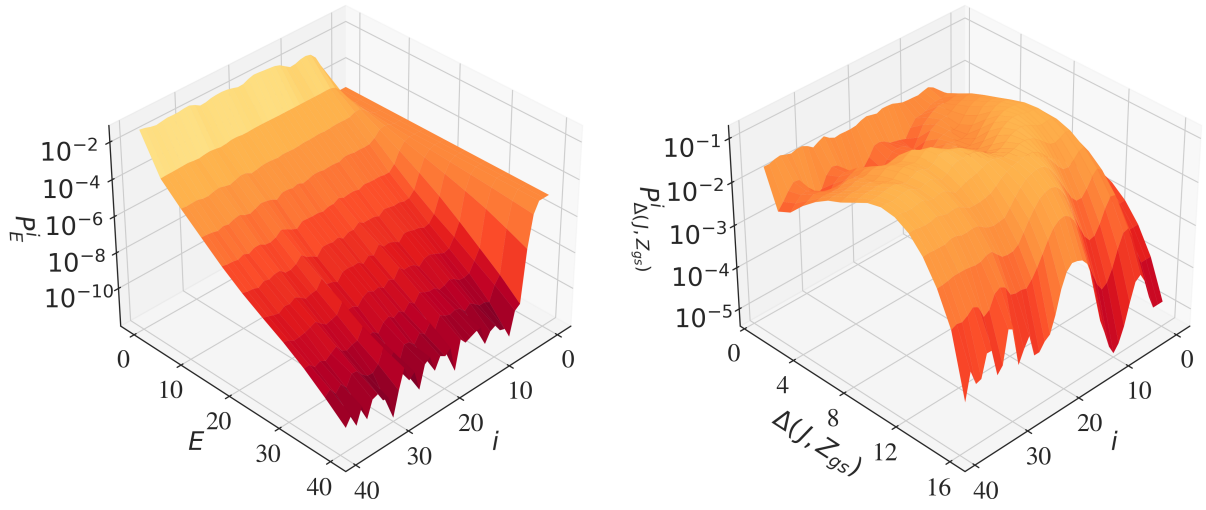
These figures reveal two distinct areas in the dynamics of the quantum walk: The first area can be located between $i = 0$ and $i \approx 25$. Here, the system's evolution is governed by the same processes as described above for the CQW model. Hence, the phase gradients of connected states are controlled via the cost evolution $\hat{U}_C$, causing a global probability transfer towards lower energy states. This in turn decreases $r^i_{B,A}$, yielding a gradual increase in the initial phases $\alpha^{i_M}_E$, which counteracts the intended phase gradients $\Delta\Phi^{i_C}_{B,A}$. Consequently, a wave pattern occurs in Fig. 2.8d and Fig. 2.8e, with maxima located at $i \approx 10$ and $i \approx 20$. In contrast to the CQW model, however, the HGQW model manages to keep full control over the phase gradients, hence achieving $\Delta\Phi^{i_C}_{B,A} < 0 \ \forall \ i_C \leq 25$. In doing so, the algorithm is able to prevent probability transfers into high energy states by effectively adjusting the strength of the phase shifts $\gamma(i, \boldsymbol{\lambda}) \cdot \Delta E_{B,A}$ to the increase in $\Delta\Theta^i_{B,A}$ caused by the driving evolution $\hat{U}_M$. This can be understood as the introduction of anharmonicity into the previously (approximately) harmonic oscillation. Therefore, the HGQW model can suppress compensating effects and maintain a guided probability transport for a longer period than the CQW model. Note that the occurrence of *stairs* in the three figures suggest that small adjustments to the proposed variational parameter set might further improve the model's performance (see section 2.4.4).

Continuing with the second area, i.e. $i > 25$, the dynamics of the system begin to change. Starting at high energy states first, this can be explained by the continuous decrease of the phase gradient $\Delta\Phi^{i_C}_{B,A}$ with every iteration i_C. Since, $\Delta\Phi^{i_C}_{B,A}$ is taken modulo 2π by the interference mechanism, with $\Delta\Phi^{i_C}_{B,A} \ mod \ 2\pi \in (\pi, 2\pi)$ ($\Delta\Phi^{i_C}_{B,A} \ mod \ 2\pi \in (0, \pi)$) transferring probability into the lower (higher) energy state $|A\rangle$ ($|B\rangle$), the HGQW model cannot guide the probability flow once the complex phases of the two basis states diverge too far. This phenomenon happens locally for each basis state, depending on the energy gradients to its connected states in the graph. As a result, a *valley* of rapidly changing initial and total phases emerges in Fig. 2.8d and Fig. 2.8f, respectively. Consequently, unpredictable probability transfers occur in Fig. 2.9b. Although these processes could potentially compensate the probability transport induced in the first area, the HGQW model uses the observation that the aforementioned valley occurs in decreasing order of the energy levels. This can be understood as high energy states typically feature large energy gradients ΔE to their connected states, while ΔE tends to decrease as the states get closer to the solution node (see e.g. Fig. 2.7). Based on this observation, the HGQW model assumes that it can transport the majority of probability fast enough towards Z_{opt}, such that intermediate energy levels can effectively form a barrier to protect it from unpredictable transfers. To verify this, all data points (E, i) with $P^i_E / P^i_0 \geq 10^{-2}$ have been marked *blue* in the three figures. The resulting subset of states can be viewed as the neighbourhood of the solution state introduced in section 2.3.1. Since the *blue* area does not overlap with the aforementioned valley, the assumption is confirmed. Moreover, the *blue* area fits to the exponential sampling strategy proposed in section 2.3.3. This shows, that the heuristic model is able to sufficiently guide the quantum walker through the graph.

2.3.5. Adjustments to the HGQW model

The investigation of the dynamics of a quantum walk under the HGQW model in the previous section demonstrated that the guided quantum walk is able to sufficiently direct a probability flow from high energy states towards the ground states. As such, it can be understood as a continuous quantum walk that is actively *pushed* in the direction of low energy in order to suppress compensating probability transfers in the originally harmonic oscillation of P_{gs}. However, so far only exact cover instances according to section 2.1.1 have been

studied. As this problem class is characterized by a high number of distinct energy levels with low degeneracies (see e.g. Fig. 2.1), it is particularly well suited for a guided walk which determines the direction of propagation based on the energy gradient between connected states in the graph. In contrast to this, 2-SAT problems as introduced in section 2.1.2 feature only a few equally spaced energy levels (see e.g. Fig. 2.2). This type of optimization problems is generally considered as *hard to solve* on quantum computing devices, due to high degeneracies and small energy gaps, yielding long optimal annealing times [71, 73]. In the following, the application of the HGQW model on 2-SAT instances will be investigated, by first studying the dynamics of the probability flow given the `2SAT_16_1` problem followed by the proposal of an adjusted heuristic model, termed HGQW-A, adapting to the differences in the problem structure. In doing so, success probabilities similar to the application on exact cover problems become achievable (see section 2.4.2).



(a) P^i_E (b) $P^i_{\Delta(J,Z_{gs})}$

Figure 2.9.: The plots show the evolution of the probability distribution obtained by the HGQW model for the `2SAT_16_1` problem as a function of the iteration index i (see section 2.3.3). Plot (a) presents the mean probability P^i_E of the total system as a function of the energy level E. Plot (b) focusses on the mean probabilities $P^i_{\Delta(J,Z_{gs})}$ of the first excited energy level with respect to the Hamming distance $\Delta(J, Z_{gs})$ of a state $|J\rangle$ to the solution state $|Z_{gs}\rangle$.

Figure 2.9a depicts the evolution of the mean probability distribution P^i_E throughout the iterations i as a function of the energy level E. The data is obtained for the `2SAT_16_1` problem instance using the HGQW model. The figure reveals a similar initial dynamic of the global probability flow in the graph as seen in section 2.3.4, yielding a steady probability transport in the direction of low energy states. However, for $i \geq 5$ the probability flow stops and a saturation of the success probability $P_{gs} \approx 4.81\%$ can be observed. Interestingly, the probability distribution throughout the whole system remains approximately constant, hence becoming independent of the number of iterations i. This phenomenon suggests the existence of internal *barriers*, which prohibit the probability exchange across the energy levels at a certain point in the algorithm. In order to understand this behaviour, Fig. 2.9b shows the evolution of the measurement probability $P^i_{\Delta(J,Z_{gs})}$ within the first energy level ($E = 4$) as a function of iteration index i and the Hamming distance $\Delta(J, Z_{gs})$ between

the states $|J\rangle$ and the ground state $|Z_{gs}\rangle$. According to section 2.3.2 and the hypercube mapping (see e.g. Fig. 2.5), the proposed parameter range for the spreading coefficients $\boldsymbol{\beta}$ restrict the walker's movement, such that only transitions between basis states of Hamming distance one are allowed. Simultaneously, the guided quantum walk demands that probability transfers are always directed towards lower energy states, which in case of the first energy levels means in the direction of the ground state. However, regarding the considered 2-SAT instances, the high degeneracy of the energy levels means that states of various different Hamming weights will feature the same energy E. Since, in case of the first energy level, the algorithm can only guide the probability flow into $|Z_{gs}\rangle$ for basis states with $\Delta(J, Z_{gs}) = 1$, states that feature $\Delta(J, Z_{gs}) \geq 2$ must first interact with states of smaller Hamming distance $\Delta(J, Z_{gs})$. However, these states are only connected to other states of the same or higher energy. Hence, they already resemble the lowest energy state in their connection path, causing a local accumulation of probability in the graph. This process can be seen in Fig. 2.9b, as a significant fraction of probability is held in $6 \leq \Delta(\Psi, \Psi_{gs}) \leq 12$ for $i \geq 20$. Consequently, this probability is not exchanged towards $\Delta(\Psi, \Psi_{gs}) = 1$, hence limiting the maximum success probability. In order to estimate the latter, the fraction of all 2^N states, which can be reached from the ground state using a path of strictly increasing state energies in the graph can be used:

$$P_{gs}^{est} = \frac{\left|\{|J\rangle\}_{\exists \text{ path between } |J\rangle \text{ and } |Z_{gs}\rangle \text{ of strictly increasing energies}}\right|}{2^N} \tag{2.65}$$

Given Eq. 2.65, $P_{gs}^{est} \approx 5.71\%$ is found, agreeing with the actually obtained success probability of $P_{gs} \approx 4.81\%$ for the `2SAT_16_1` problem. Note that similar estimations were found for all investigated 2-SAT instances, while $P_{gs}^{est} > 95\%$ was determined regarding the exact cover problems. The latter shows that the large number of energy levels sufficiently allows distinguishing the individual basis states.

The above considerations, although neglecting interactions with other energy levels (unwanted probability transfers into higher energy states can in general also provide a path for probability to reach the ground state), suggest that adjustments to the proposed heuristic model in section 2.3.3 are necessary to achieve a sufficient guidance of the quantum walker. As demonstrated above, the main challenge lies in the distribution of connected states across the energy levels, with numerous of them featuring zero energy gradient. Since the states' energies are determined by the number of violated clauses (see section 2.1.2), the number of distinct energy levels is a fixed entity, ruling out naive approaches like recasting the 2-SAT problems into exact cover instances or choosing a different driving Hamiltonian $\hat{H}_M$ to change the graph layout. Instead, the heuristic model must adapt throughout the iterations to the probability distributions within the energy levels, in order to allow probability exchanges between states of the same energies. Two possible approaches are presented below:

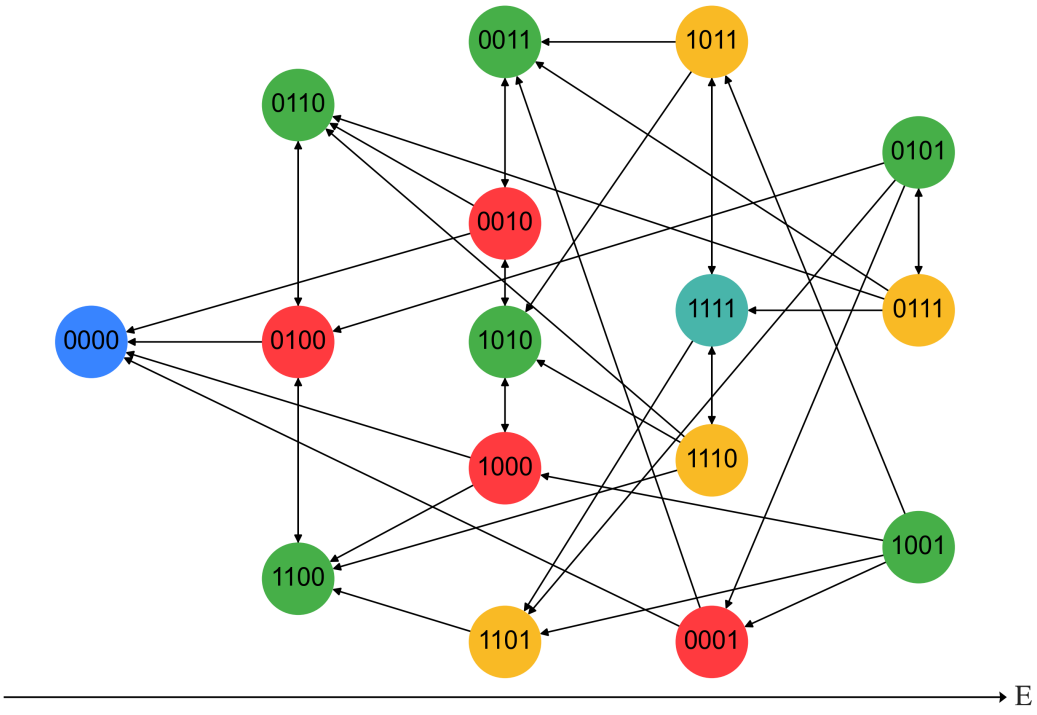
- The first ansatz is based around the observation, that the hypercube mapping prohibits direct transfers of states of different energies in case they have a Hamming distance greater than 1. As a consequence, probability exchanges within an energy level are required until the Hamming distance is sufficiently reduced. Using a different graph layout, on the other hand, can in principle allow interactions between states of great Hamming distance. Thus, one idea could be to use an alternating sequence, that switches between multiple driving evolutions and thus graph layouts, in order to

provide sufficient dynamics at all states and prevent immovable probabilities. In doing so, the guided quantum walk is extended into a subspace of the more general quantum alternating operator ansatz (see section 2.2.2). Although this approach provides high degrees of freedom, generally allowing to reach high success probabilities ($P_{gs} \approx 1$), finding an optimal set of driving evolutions is not a trivial task, as the state distribution across the energy levels is unknown. Thus, without a technique for sampling these driving Hamiltonians, this ansatz is impractical.

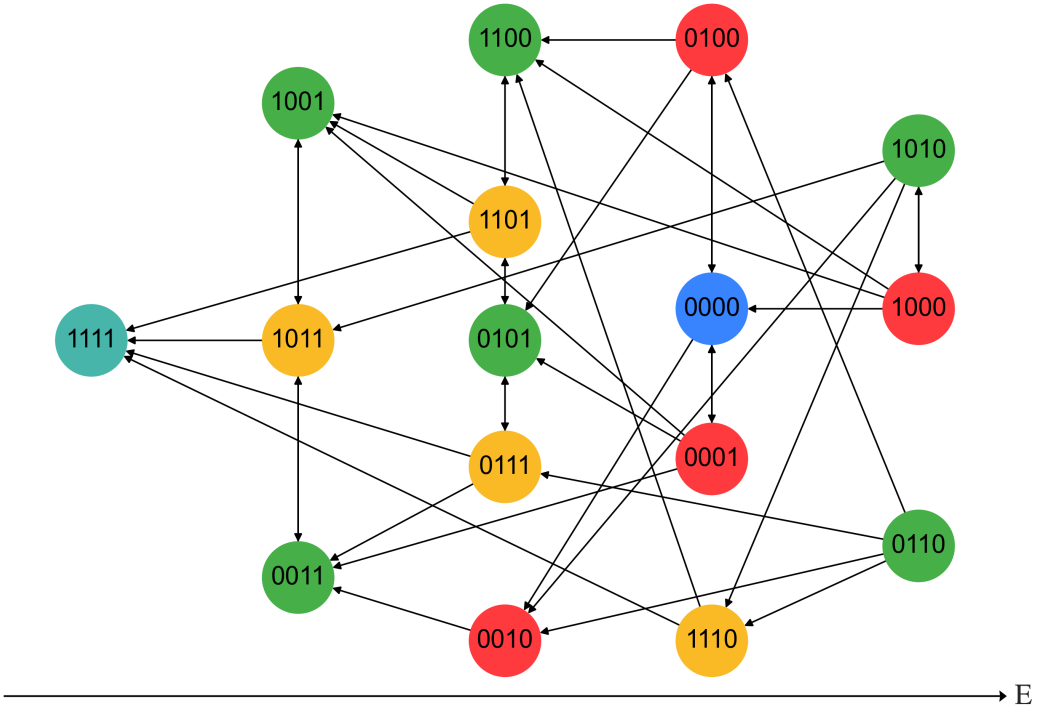
- The second approach focusses on the dynamics of the hypercube mapping, providing several sub graphs for the states' interactions based on the coefficients K_i (see section 2.3.2). So far, the system is restricted to zero and first order interactions, due to the proposed parameter bounds, $\beta_i \in [P_1^-, \pi]$. Extending this range to $\beta_i \in [\pi/2, \pi]$ would allow the system to perform tunnelling and thus exchange probability across greater Hamming distances, $\Delta(J, Z_{gs}) \geq 2$. Thus, combining multiple sub graphs in an alternating sequence could again provide sufficient dynamics to prevent stationary probabilities. A major problem of this ansatz, however, is that the guided quantum walk relies on the ability to accumulate probability at low energy states, which is controlled via the coefficient K_0. Choosing β_i close to $\frac{\pi}{2}$, which is required for transactions of large Hamming distance, K_0 is suppressed, making the distribution of the probabilities and phases uncontrollable. Consequently, the quantum walk becomes *unguided*.

Inspired by the above strategies, an adjusted parameter space for the spread coefficients $\boldsymbol{\beta}$ will be explored in the following. Still relying on the hypercube layout, this ansatz introduces additional energy gradients between connected states while keeping track of the evolution of the probability and phase distributions. The approach is inspired by a bijective mapping between the computational basis states that can be achieved via the driving evolution $\hat{U}_M$: Considering an arbitrary state $|J\rangle = |j_0, \ldots, j_{N-1}\rangle \in \mathcal{H}_{2^N}$. Flipping the computational state of every qubit j_i, $|J\rangle$ is transferred into its *inverse state* $|\bar{J}\rangle \in \mathcal{H}_{2^N}$. This transition can be achieved by applying Pauli-x operators, $\hat{\sigma}_i^x$, to each qubit i, corresponding to an $\beta = \pi$ evolution under $\hat{U}_M$. The graph of the *inverse states* will henceforth be called the *inverse graph* and its energy distribution the *inverse energy spectrum*. By transforming the state vector between both pictures, the connections between the initial state amplitudes do not change, e.g. two interacting amplitudes ψ_A and ψ_B still interact in the inverse graph ($\psi_{\bar{A}}$ and $\psi_{\bar{B}}$), since their Hamming distance remains the same. However, the energy distribution of the states can change, thus $\Delta E_{A,B} \neq \Delta E_{\bar{A},\bar{B}}$ in general (see e.g. Fig. 2.10). Consequently, it is hoped that by switching between both pictures, the degeneracy of the energy levels can be lifted by artificially introducing directed edges in the inverse graph between nodes of identical energies in the regular graph. Hence, this procedure can be understood as a mixing operation that tries to make originally immovable probability accessible to the ground state.

In section 2.3.1 Eq. 2.44 has been derived describing the evolution of the full state vector $|\Psi\rangle = \sum_J \psi_J |J\rangle$ under the transformations $\hat{U}_C$ and $\hat{U}_M$. The proposed parameter range for the spread coefficients $\boldsymbol{\beta}$, yields a selection of the interaction orders K_0 and K_1 while suppressing $K_{i\geq 2}$. By introducing the inverse graph and the transition operation, $\exp\left(-i\pi\hat{H}_M/2\right)$, the selection of the coefficients K_i changes, such that whenever the system transforms between both pictures, K_{N-1} and K_N are maximized. Although this mean $K_0 \approx 0$ (see e.g. Fig. I.9 in appendix I), in contrast to the aforementioned second approach, the system keeps control over the accumulation of probabilities and phases, since it can track the exchanges between both graphs. Thus, it is not important whether the probability is concentrated in the ground state or the inverse ground state, since a bijective mapping between them exists. In doing



(a) Regular graph



(b) Inverse graph

Figure 2.10.: The figures show the regular (a) and inverse (b) graph structure of a hypercube mapping for an $N = 4$ qubit 2-SAT problem. Each node denotes a computational basis state $|J\rangle$ and different colours indicate its Hamming distance $\Delta(J, Z_{gs})$ to the solution state $|Z_{gs}\rangle = |0000\rangle$. The states are ordered along the x-axis according to their energy E. Note that the solution state has only in the regular graph the lowest energy.

so, K_N and K_{N-1} provide the same accumulation and interaction mechanisms as K_0 and K_1, respectively, with the only difference being a swap between the two graphs. Concluding these observations, the parameter range is extended to $\beta_i \in R_L \cup R_H = [P_1^- - \pi/2,\ P_1^+ - \pi/2] \cup [P_1^-,\ \pi]$. Equations 2.67 and 2.68 summarize the evolution under both parameter spaces for the interaction of two basis states $|A\rangle$ and $|B\rangle$ with $E_A = \langle A|\hat{H}_C|A\rangle$ and $E_B = \langle B|\hat{H}_C|B\rangle$, respectively:

$$\Psi_A = r_A\ e^{-i\,\alpha_A} \tag{2.66}$$

$$\begin{aligned}\xrightarrow{\beta \in R_H}\ r_A\ e^{-i\,(\alpha_A + \gamma \cdot E_A)}\Big[&\cos^N(\beta)\\ &- \frac{r_B}{r_A}\cos^{N-1}(\beta)\sin(\beta)\sin(\Delta\alpha_{B,A} + \gamma \cdot \Delta E_{B,A})\\ &- i\ \frac{r_B}{r_A}\cos^{N-1}(\beta)\sin(\beta)\cos(\Delta\alpha_{B,A} + \gamma \cdot \Delta E_{B,A})\Big],\end{aligned} \tag{2.67}$$

$$\begin{aligned}\xrightarrow{\beta \in R_L}\ r_{\bar{A}}\ e^{-i\,(\alpha_{\bar{A}} + \gamma \cdot E_{\bar{A}})}\Big[&\sin^N(\beta)\\ &- \frac{r_{\bar{B}}}{r_{\bar{A}}}\cos(\beta)\sin^{N-1}(\beta)\sin\left(\Delta\alpha_{\bar{B},\bar{A}} + \gamma \cdot \Delta E_{\bar{B},\bar{A}}\right)\\ &- i\ \frac{r_{\bar{B}}}{r_{\bar{A}}}\cos(\beta)\sin^{N-1}(\beta)\cos\left(\Delta\alpha_{\bar{B},\bar{A}} + \gamma \cdot \Delta E_{\bar{B},\bar{A}}\right)\Big].\end{aligned} \tag{2.68}$$

It is important to note here that, in contrast to the regular graph, the solution node in the inverse graph does in general not correspond to either the largest or smallest energy state, but can often be found at intermediate energy levels (see e.g. Fig. 2.10). Thus, it is not possible to sufficiently guide the quantum walker towards Z_{gs} while using the inverse picture, as depending on the basis state the direction of optimal probability transfer varies. Instead, the inverse graph will be used exclusively to provide additional dynamics to the system. Consequently, the proposed parameter range includes both transfers into higher ($\beta_i \in [\pi/2,\ P_1^+ - \pi/2]$) and lower ($\beta_i \in [P_1^- - \pi/2,\ \pi/2]$) energy states within the inverse picture. Note that in the regular graph, R_L is restricted to $[P_1^- - \pi/2,\ \pi/2]$.

Based on the above considerations, an adjusted heuristic model, termed HGQW-A, for guided quantum walks on 2-SAT problem instances will be introduced in the following. This model uses an alternating sequence of the regular and inverse graph, controlled via two sets of spread coefficients $\beta^R(i, \boldsymbol{\lambda})$ and $\beta^I(i, \boldsymbol{\lambda})$. Similar to the HGQW model, a constant distribution is used for both parameter sets, determined by the optimization parameters λ_1 and λ_2, respectively. In doing so, $\beta^R(i, \boldsymbol{\lambda})$ is applied at even iterations ($i\ mod\ 2 = 0$) and $\beta^I(i, \boldsymbol{\lambda})$ is used during odd iterations ($i\ mod\ 2 = 1$). Hence, after every guided probability transfer, the system switches into the inverse graph and performs a mixing operation. This process continues for $p - k$ (even) iterations, followed by k iterations using $\beta_i = \beta^R(i, \boldsymbol{\lambda}) + \pi/2$. Here, the latter (leaving the system in the regular graph) is motivated by the observation that near the end of the algorithm, the majority of probability is typically located within the first energy levels at states $|J\rangle$ with $\Delta(J, Z_{gs}) = 1$, such that no mixing operation is required. In fact, pursuing the alternation typically results in a reduced success probability, as the solution state leaks probability in the inverse graph. Note that $k = 2$ was found optimal in the experiments conducted in section 2.4.2. Regarding the phase coefficients $\boldsymbol{\gamma}$, the HGQW-A

model also alternates between two parameter sets depending on the active picture. In the regular graph, the same distribution $\gamma(i, \boldsymbol{\lambda})$ as used by the HGQW model is applied (see Eq. 2.62). Within the inverse picture a parabola shape, hence providing small initial and final mixing, was found optimal. The HGQW-A model is summarized below, with an example distribution obtained for the `2SAT_16_1` instance shown in Fig. 2.10.

Spread coefficients:
$$\beta(i, \boldsymbol{\lambda}) = \begin{cases} \lambda_1 & , i \text{ even or } i \geq p-k \\ \lambda_2 & , \text{otherwise} \end{cases} \tag{2.69}$$

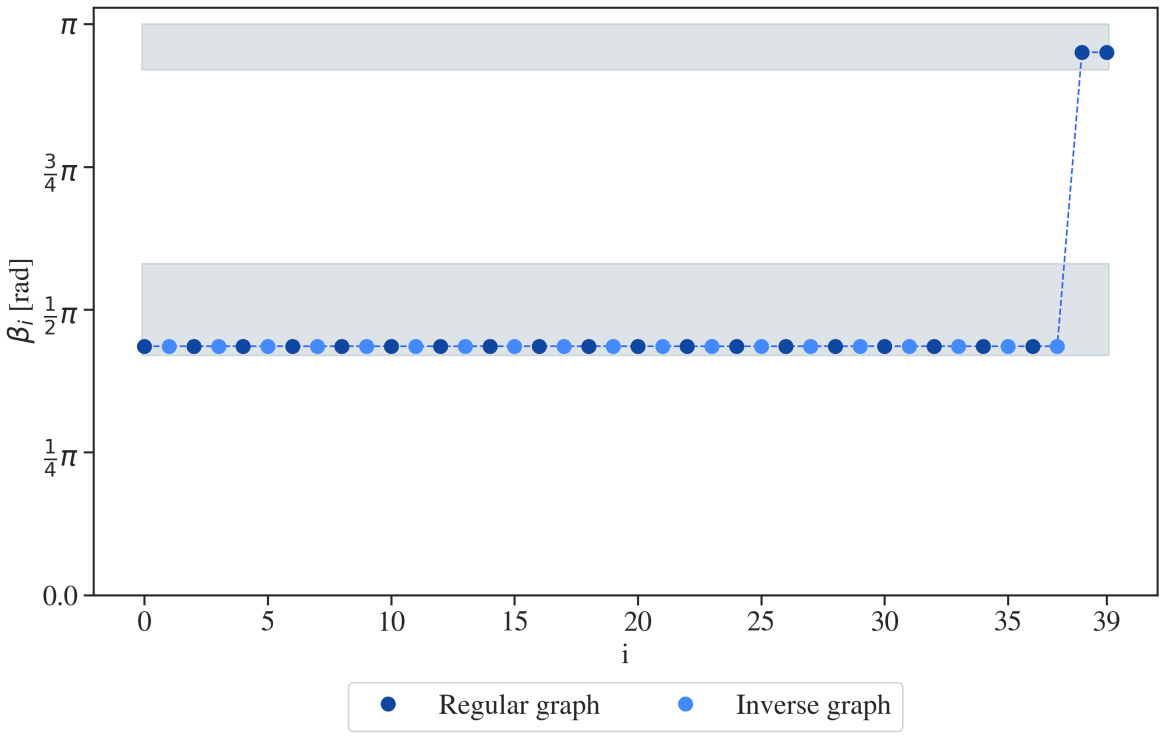
Phase coefficients:
$$\gamma(i, \boldsymbol{\lambda}) = \begin{cases} \dfrac{\lambda_3}{\frac{1}{\lambda_4 \cdot x_i + \lambda_5} - \frac{\lambda_6}{\lambda_4 + \lambda_5}} & , i \text{ even or } i \geq p-k \\ \lambda_7 \cdot \left[\lambda_8 - (x - \lambda_9)^2\right] & , \text{otherwise} \end{cases} \tag{2.70}$$

$$\lambda_1 \in \left[P_1^- - \frac{\pi}{2}, \frac{\pi}{2}\right], \qquad \lambda_2 \in \left[P_1^- - \frac{\pi}{2}, P_1^+ - \frac{\pi}{2}\right], \qquad \lambda_3 > 0, \tag{2.71}$$

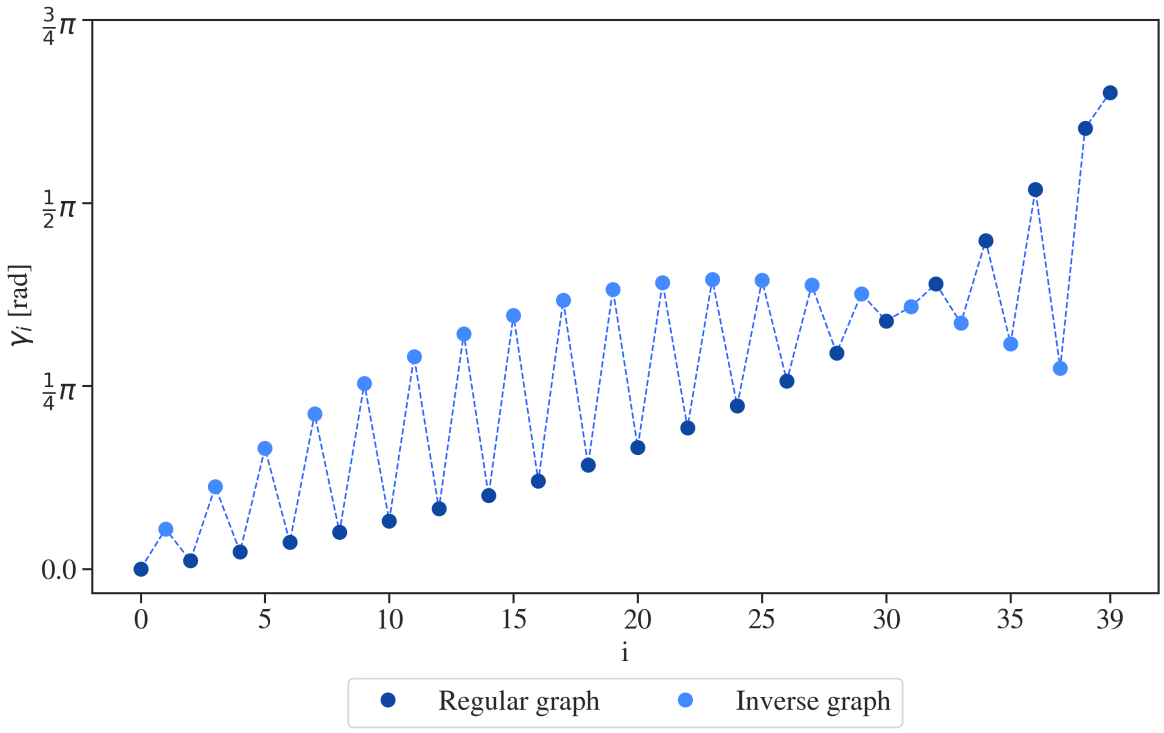
$$\lambda_4 > 0, \qquad \lambda_5 \in (0, 1], \qquad \lambda_6 \in [0, 1), \tag{2.72}$$

$$\lambda_7 > 0, \qquad \lambda_8 > 0, \qquad \lambda_9 \in [0, 1), \tag{2.73}$$

$$x_i = \frac{i}{p-1}. \tag{2.74}$$



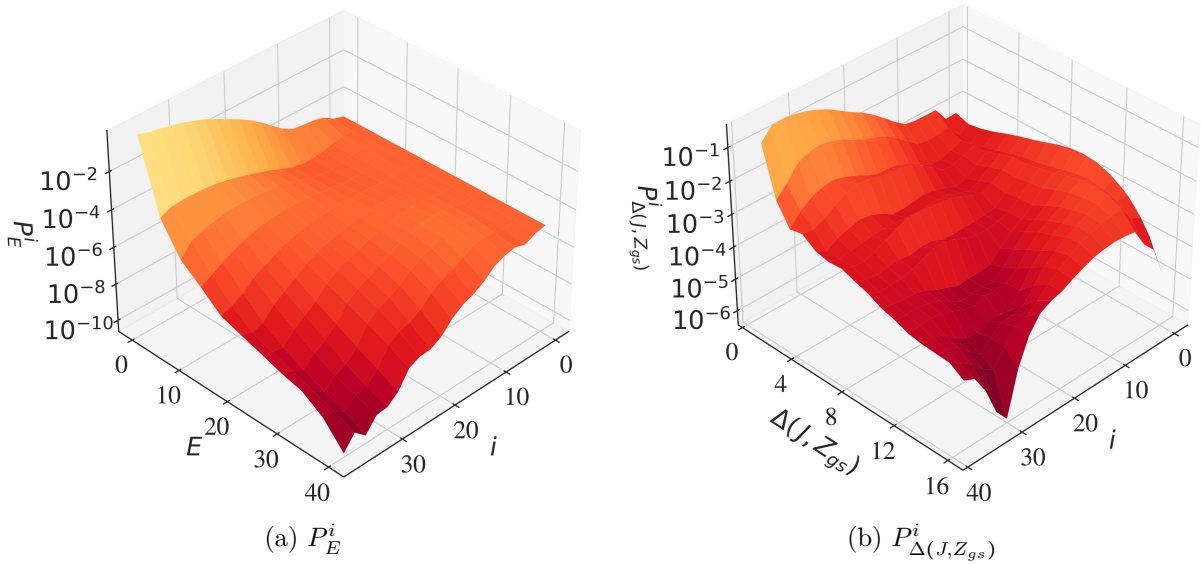
(a) Spread coefficients $\beta(i, \boldsymbol{\lambda})$



(b) Phase coefficients $\gamma\,(i, \boldsymbol{\lambda})$

Figure 2.10.: The plots show the sets of variational parameters $\beta\,(i, \boldsymbol{\lambda})$ and $\gamma\,(i, \boldsymbol{\lambda})$ determined by the HGQW-A model for the `2SAT_16_1` problem instance (p = 40). The model has been tuned using at most $1000 \cdot M$ optimization steps within 20 separate optimization runs (best one shown).

Given the distribution of variational parameters in Fig. 2.10, the evolution of the mean measurement probabilities P_E^i $\left(P^i_{\Delta(J,Z_{gs})}\right)$ of the whole system (the first energy level) as a function of the iteration index i and the energy level (Hamming distance $\Delta(J, Z_{gs})$ to the solution state Z_{gs}) for the aforementioned `2SAT_16_1` problem is depicted in Fig. 2.11a (Fig. 2.11b). Note that since the model alternates between the regular graph (even i) and the inverse graph (odd i), only the data obtained at even i is shown. Comparing the evolution of P_E^i to Fig. 2.9a, the HGQW-A model achieves significantly improved dynamics of the probability transport, as throughout all iterations a probability flow towards the ground state is established. Hence, a significantly higher success probability $P_{gs} \approx 95\%$ compared to $P_{gs} \approx 4.81\%$ via the HGQW model is observed at the end of the algorithm. This can be understood with respect to the evolution of $P^i_{\Delta(J,Z_{gs})}$ in Fig. 2.11b. While the HGQW model causes probability to remain stationary between $6 \leq \Delta\,(\Psi, \Psi_{gs}) \leq 12$ (see Fig. 2.9b), as no sufficient phase gradient could be established between states of the same energy level, the transformation into the inverse graph allows the HGQW-A model to artificially create an energy gradient and thus a phase difference between the states. In doing so, despite an initial increase in probability at high Hamming distances ($10 \leq \Delta\,(\Psi, \Psi_{gs}) \leq 16$ for $i < 10$), Fig. 2.11b reveals that the HGQW-A model is able to iteratively transport probability towards smaller hamming distances, hence making it accessible to the solution state. Note, however, that compared to the HGQW model, the probability transport seems to require more iterations, as odd iteration numbers do not contribute to the guided transfer (see steeper initial increase of P_{gs} in Fig. 2.9a compared to Fig. 2.11a).



(a) P_E^i (b) $P^i_{\Delta(J,Z_{gs})}$

Figure 2.11.: The plots show the evolution of the probability distribution obtained by the HGQW-A model for the `2SAT_16_1` problem as a function of the iteration index i. Plot **(a)** presents the mean probability P_E^i of the total system as a function of the energy level E. Plot **(b)** focusses on the mean probabilities $P^i_{\Delta(J,Z_{gs})}$ of the first excited energy level with respect to the Hamming distance $\Delta(J, Z_{gs})$ of a state $|J\rangle$ to the solution state $|Z_{gs}\rangle$.

The above observations verify the use of the inverse graph as a tool to artificially and temporarily increase the dynamics in the system. With respect to problem instances that feature intermediate degeneracies of their energy levels (lying between the exact cover and 2-SAT problems investigated in this thesis), these findings also suggest that an adjusted alternation sequence between the regular and inverse graph might allow solving other types of combinatorial optimization problems as well. Moreover, an improved sampling strategy of the phase coefficients in the inverse graph could further accelerate the probability transfer process. Hence, additional research is needed.

2.4. Performance analysis

In the previous sections, two heuristic strategies, namely the HGQW model and the HGQW-A model, have been developed for deploying guided quantum walks on both exact cover and 2-SAT problem instances. By studying the evolution of a quantum system under these models, i.e. the development of the complex phases and measurement probabilities of the computational basis states, it was verified that both strategy are able to create and guide a probability flow in the direction of the solution state. Using a hypercube graph layout and the state's energy as an evaluation metric, the models provide two sampling functions $\beta(i, \boldsymbol{\lambda})$ and $\gamma(i, \boldsymbol{\lambda})$ for the sets of spread and phase coefficients, respectively. In doing so, the guided quantum walk is executed analogously to the QAOA (see appendix B.1 for the quantum circuit) by deploying a hybrid quantum-classical procedure of tuning the sets of variational parameters (see Fig. 2.3). However, in contrast to the QAOA, the heuristic models focus on a set of 5 to 9 optimization parameters $\boldsymbol{\lambda}$, instead of adjusting $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ individually. Motivated

by patterns commonly occurring in optimized distributions of the variational parameters, the hope is that by refining the shape of the sampling functions and thus collectively tuning the individual parameters, the optimization process can be significantly simplified, yielding a more convex search space and a reduced convergence time. It is important to note here that the functions proposed for the two models are based on common characteristics in the internal structure of their respective problem type (e.g. numerous distinct energy levels with low degeneracy in case of the studied exact cover problems). Consequently, the heuristic models are restricted to a subspace of the more general guided quantum walk, which only limits the system to zero and first order interactions and introduces a directed interference mechanism based on the chosen driving evolution (see section 2.3.2). In order to verify the success of the HGQW model and the HGQW-A model, the scaling of the success probability P_{gs} both as a function of the number of iterations p and the number of qubits N in the system will be studied in the following sections. Moreover, the guided quantum walk will be compared to the QAOA (see section 2.2.2) and the AQA (see section 2.2.3).

To ensure comparability between the investigated optimization algorithms, all data presented in the following sections is obtained via SEQCS, using similar program codes for simulating sets of exact cover and 2-SAT problems. Regarding each problem class, 48 individual optimization instances will be considered, which are separated into smaller *sample groups* of size 8 according to the number of involved qubits $N \in \{10, 12, 14, 16, 18, 20\}$ (see appendix G and appendix H). The exact cover problems are generated randomly using the algorithm described in appendix G.1 and the 2-SAT instances are taken from the pool of problems provided by *Mehta et al.* [73]. Note that all investigated problems feature a unique ground state, and the success probability P_{gs} will be averaged over the highest probabilities obtained within each sample group. Moreover, exact cover and 2-SAT problems will be treated separately due to the significant differences in their energy spectra (compare e.g. Fig. 2.1 and Fig. 2.2 in section 2.1).

Regarding the classical optimization phase of the three algorithms, the Nelder-Mead optimizer [110] will be used, which assesses the performance of the variational parameter sets based on a reduction of the approximation ratio A_r (see Eq. 2.7). In doing so, the algorithms are compared in a realistic context, where no additional information about the energy spectrum and the solution state of the optimization problem in question is available. Note, however, that using A_r instead of P_{gs} during the parameter optimization typically causes smaller success probabilities, due to the differences in the search spaces of the two metrics: Considering for example two different sets of variational parameters, which result in the final states $|\Psi_1\rangle = \frac{1}{2}(|A\rangle + |C\rangle)$ and $|\Psi_2\rangle = |B\rangle$, respectively, with $|A\rangle$ denoting the solution state and $E_A \approx E_B \ll E_C$. With respect to the approximation ratio, the classical optimizer would prefer the state $|\Psi_2\rangle$, as it gives an overall reduction of the energy expectation $\langle\Psi_i|\hat{H}_C|\Psi_i\rangle$ compared to $|\Psi_1\rangle$. In fact, $|\Psi_2\rangle$ might even correspond to a local maximum in the parameter space of $1 - A_r$, while $|\Psi_1\rangle$ lies close to a local minimum. In contrast to this, an evaluation based on P_{gs} would tune the system towards $|\Psi_1\rangle$, since $P_{\Psi_1} \gg P_{\Psi_2}$. In this picture, $|\Psi_2\rangle$ resembles a global minimum, as it yields zero success probability, and $|\Psi_1\rangle$ is in the neighbourhood of the global maximum. These contradicting ratings of the two parameter sets demonstrate that for both metrics, even though they share the same global maximum (the solution state), the distribution of their local maxima differs in general [30].

These differences in the search spaces could harm the performance of the optimization algorithm in case the variational parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are tuned unconstrained, as done by the QAOA. This is because an optimization process based on A_r could propose parameter sets

that feature low energies, but also small success probabilities (see e.g. $|\Psi_2\rangle$). Moreover, as demonstrated by *Willsch et al.* [30], the success of the optimization process also greatly depends on the classical optimization algorithm (e.g. Nelder-Mead, BFGS or SLSQP [111]) as well as the number of optimization parameters used, with the latter influencing the size and complexity of the search space and the former determining the way the algorithm traverses through it. This can be understood, as classical optimizers often get stuck at suboptimal local maxima, due to the non-convexity of the search space, hence requiring multiple optimization runs to determine the solution state with sufficient accuracy. Interestingly, concerning the guided quantum walk, only a weak dependence of the scaling of P_{gs} on both the optimization algorithm and the evaluation metric was observed during the simulations. This can be explained, on the one hand, by the relatively small search space (5 (9) dimensional for HGQW (HGQW-A) model), which, in combination with the specially designed sample functions ($\beta(i, \boldsymbol{\lambda})$ and $\gamma(i, \boldsymbol{\lambda})$), seems to be sufficiently convex. On the other hand, the increase in the success probability throughout the guided walk is highly coupled to the decrease in the overall energy of the system. This is because, the guided quantum walk achieves a directed probability flow from high energy states towards low energy states, gradually increasing the probability located in the shrinking neighbourhood around the solution state. Thus, the GQW prohibits final state configurations, which (with high probability) include basis states of greatly different energies (see e.g. $|\Psi_1\rangle$). Consequently, the two search spaces seem to mostly align. For comparison, the results of the GQW simulations using the P_{gs} evaluation are provided in Fig J.12 in appendix J.1.

2.4.1. Performance of the HGQW model

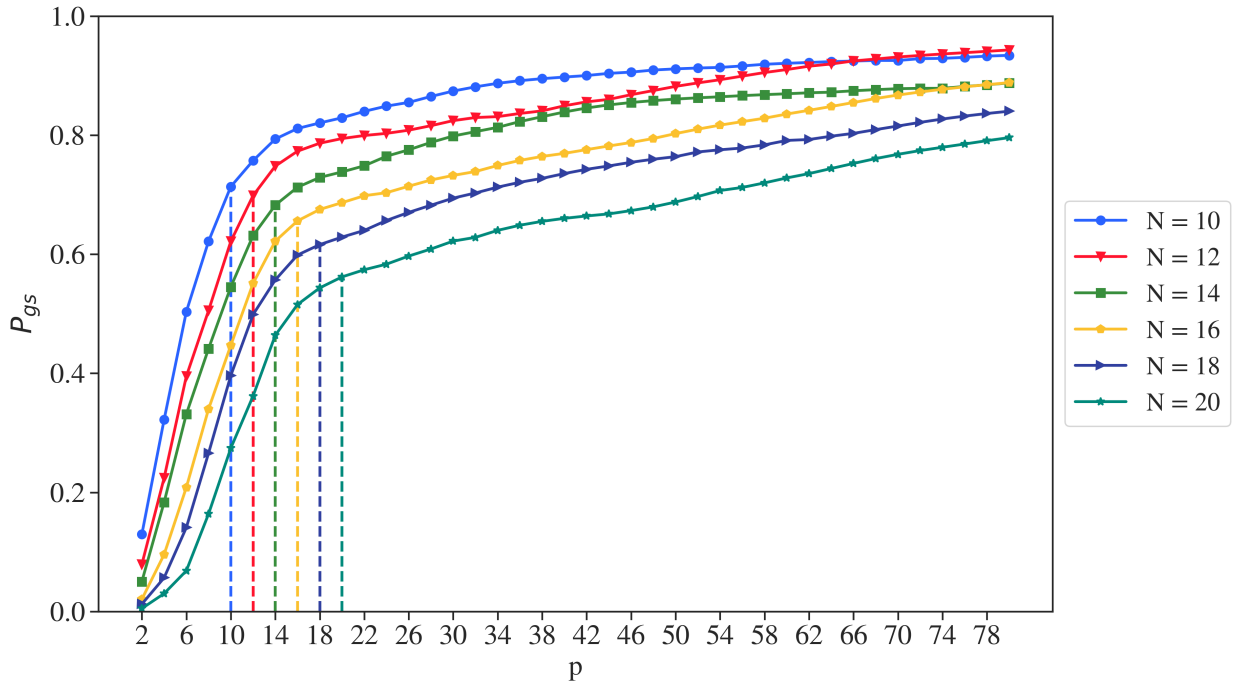
The distribution of the success probabilities P_{gs} obtained by the HGQW model on the set of exact cover problems (see appendix G) is depicted in Fig. 2.12a as a function of the number of iterations p and system size N. The presented data corresponds to the average probabilities of the best (highest P_{gs}) parameter sets obtained in each sample group within 200 optimization runs using a maximum of 4000 queries to the classical optimizer. Note that a three-dimensional plot of the data is presented in Fig. J.10 in appendix J.1. In doing so, Fig. 2.12a reveals common characteristics in the distributions of P_{gs} as functions of p across all investigated system sizes N. With a rapid increase of the success probability for $p \leq N$, followed by a beginning saturation of P_{gs} for $p \gg N$, the behaviour of the guided quantum walk can be separated into two regions of operation.

The first region, spanning from $p = 0$ to $p \approx N$, concerns the situation where the number of iterations p is less than the number of qubits N in the system. This provides a challenge to the HGQW model, since the GQW restricts the quantum walker to zero and first order interactions only, while the largest Hamming distance to the solution state in the system is N. Thus, limiting probability transfers to pairs of basis states of Hamming distance one, the algorithm is not able to establish a probability flow from states $|J\rangle$ with $\Delta(J, Z_{gs}) > p$ into the solution state $|Z_{gs}\rangle$. This limits the maximally achievable success probability, which can be approximated by the fraction of reachable state within p iterations (given by the sum of binomial coefficients $\sum_{k=0}^{p \leq N} \binom{N}{k}$) with respect to the total number of states in the graph 2^N. Consequently, the rescaled shape of this fraction as a function of p can approximately be found in $P_{gs}(p)$ within the first region. For example, the $N = 18$ and $N = 20$ sample groups feature a greatest increase of P_{gs} at $p \approx N/2$ (centre of the region), with decreasing slopes towards the region's boarders. However, for small system sizes (e.g. $N = 10$ and $N = 12$) a steep increase in the success probability is also obtained at small $p < 6$, which can be

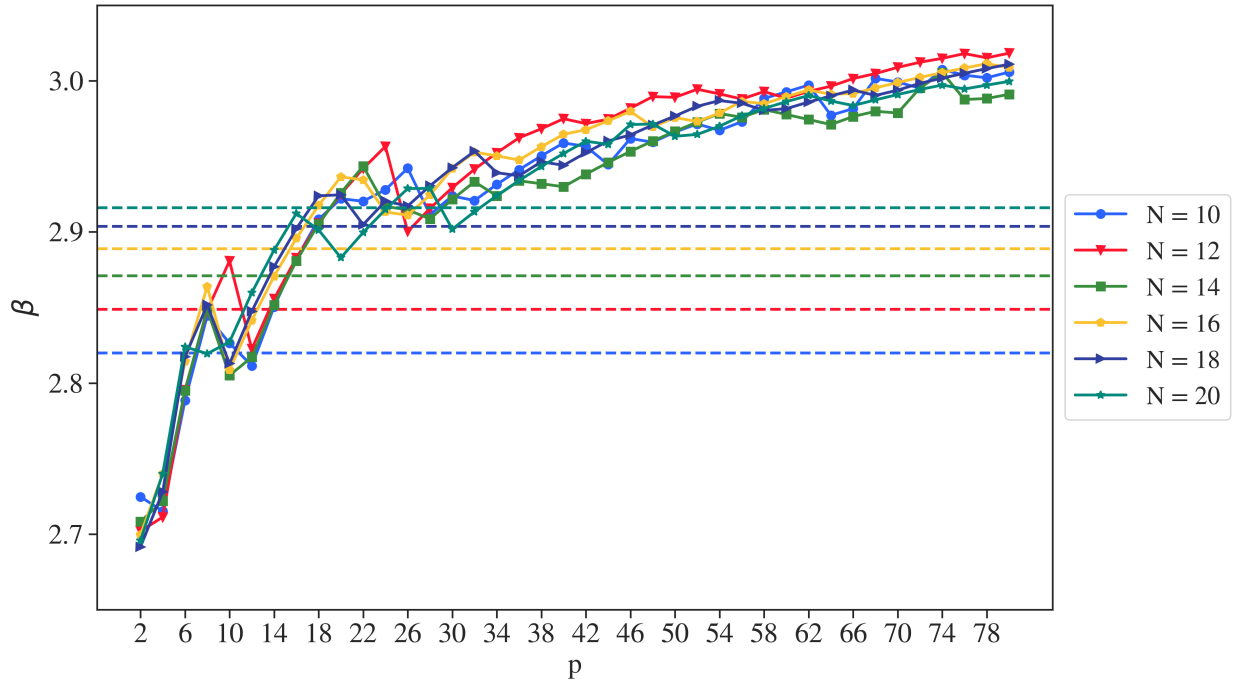
understood as follows: In contrast to the original definition of the guided quantum walk, the HGQW model applied within the first region uses an extended parameter range for the spread coefficients $\boldsymbol{\beta}$ set to $[\pi/2, \pi]$. This was motivated by the observation that for small p (i.e. $p \ll N$), the classical optimizer repeatedly sets $\beta_i = P_1^-$ (lower bound of the parameter region). By allowing the optimizer to go below P_1^-, the HGQW model is able to obtain significantly higher success probabilities at small p and small N. The mean distributions of the chosen spread coefficients are depicted in Fig. 2.12b with respect to the parameter sets shown in Fig. 2.12a. Note that P_1^- is indicated by horizontal *dashed* lines for each system size. The plot reveals, that for each sample group, values below the proposed lower bound P_1^- are occupied at $p \in [0, \approx 3/4N]$. Although this generally causes a reduction of the accumulation coefficient K_0 (see e.g. Fig. I.9), this behaviour allows the HGQW model to access states of Hamming distance $\Delta(J, Z_{gs}) > p$ by performing large range interactions, hence increasing the amount of probability that can be transferred into $|Z_{gs}\rangle$. Note that this effect is most noticeable at small systems sizes, as the relative increase in the number of reachable states from the solution state with respect to the total number of basis states decreases in N (see e.g. $\left[\binom{N}{2} - \binom{N}{1}\right]/2^N$), yielding only minor improvements for large N.

Besides that, Fig. 2.12a also shows that for none of the investigated problem instances, unit success probability is obtained at $p = N$. Since in this case, the quantum walk is no longer restricted by the states' Hamming distance to the solution state, this observation is most likely caused by the algorithm's ability to transfer probability between connected states in the graph. This is because only probability fractions (determined by $\boldsymbol{\beta}$ and the phase gradient $\Delta\Phi$) can be exchanged between basis states during one iteration. Moreover, the number of iterations required for a probability P_J (initially located at a state $|J\rangle$) to reach the solution state is equal to $\Delta(J, Z_{gs})$. Hence, states that have a great Hamming distance to $|Z_{gs}\rangle$ in the graph will generally transport less probability into the solution state during the same number of iterations. As a result, $P_{gs}(p = N)$ decreases approximately exponentially as a function of the system size N, due to the linear increase of the mean Hamming distance of all basis states (see dashed lines in Fig. 2.12a).

Regarding the second region of operation, i.e. $p \in [N, 80]$, the slope of P_{gs} decreases significantly, causing an approximately linear scaling that seems to saturate for large p below unit probability ($P_{gs} \in [79\%, 95\%]$ at $p = 80$). With respect to Fig. 2.12b, this behaviour can be understood by the monotonic increase of $\boldsymbol{\beta}$ as a function of p, which causes an increase of the quotient $\cos(\beta)/\sin(\beta)$. In doing so, the GQW is able to maintain a guided probability transfer for larger probability gradients $r^i_{B,A}$ across connected states by postponing the appearance of the valley of unpredictable probability transfers (see section 2.3.4). However, note that an increase of $\boldsymbol{\beta}$ coincides with a decrease in the interaction coefficient K_1, thus limiting the amount of probability that gets exchanged during one iteration between two states. Interestingly, almost identical distributions of the spread coefficients are observed across and within the sample groups. This suggests, that $\boldsymbol{\beta}$ is a problem independent parameter, which is only influenced by the number of iterations p and the underlying graph structure. This could allow sampling the spread coefficients from a predefined function, reducing the number of optimization parameters of the heuristic models by one. However, this is left for future work. Besides that, $P_{gs}(p)$ seems to saturate at a value below one. Most likely, this is caused by immovable probability, due to insufficient energy gradients at some nodes in the graph (see discussion in section 2.3.5). This suggests that mixing operations (i.e. transforming the system into the inverse graph) as well as small adjustments to the variational parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ might significantly improve the performance of the HGQW model (see section 2.4.4).



(a) Success probability P_{gs}



(b) Spread coeffiencts $\boldsymbol{\beta}$

Figure 2.12.: The plots show the distribution of the success probability P_{gs} (a) and the spread coefficients $\boldsymbol{\beta}$ (b) obtained by the HGQW model as a function of the total number of iterations p for the exact cover problems given in appendix G. The data is derived using a maximum of 4000 evaluations within 200 runs of the Nelder-Mead optimizer, focussing on a reduction of the approximation ratio A_r (see Eq. 2.7). For each problem instance and each circuit depth p, the best (highest P_{gs}) parameter set is chosen, and the data is averaged within the sample groups (i.e. across problems of the same size N). The individual sample groups are distinguished by colour, and *dashed* lines indicate $p = N$ and $\boldsymbol{\beta} = P_1^-$, respectively.

Concluding this section on the HGQW model, a weak exponential scaling of the success probability P_{gs} with respect to the system size N is found, yielding e.g. $P_{gs}(N) \propto 2^{-0.13N}$ at $p = 10$ and $P_{gs}(N) \propto 2^{-0.06N}$ at $p = 20$. Hence, the scaling improves for larger p, causing a close-to linear distribution once the algorithm enters the second region of operation for all system sizes. This can be explained by the differences in the slopes of P_{gs} within the two regions. Since the transitioning point scales linearly in N and P_{gs} increases significantly faster in the first region than in the second region, the probability gradient decreases between systems operating in different regions. Hence, the guided quantum walk seems to be able to achieves high success probabilities within numbers of iterations p that scale only linearly in the system size N. This property makes it a promising candidate for near-term NISQ devices, where the circuit depth is limited due to noise and decoherence [37]. Moreover, with respect to the P_{gs} to p ratio, the HGQW model seems to be most effective for $p \in [N, 2N]$.

2.4.2. Performance of the HGQW-A model

The performance of the HGQW-A model is analysed analogously to the HGQW model, by studying the distribution of the success probability as a function of the number of iterations p and the system size N with respect to the set of 2-SAT problems given in appendix H. The data presented in Fig. 2.13a corresponds to the highest success probabilities averaged over each sample group using 200 optimization runs with up to 4000 evaluations for each problem instance. A three-dimensional plot of Fig. 2.13a is given Fig. J.11 in appendix J.1. Note that Fig. 2.13b and Fig. 2.12c show the distribution of the spread coefficients β_i in the regular and inverse graph, respectively. In doing so, two regions of operation can be distinguished in the scaling of $P_{gs}(p)$, yielding a similar shape in the probability distributions as previously obtained for the HGQW model (see section 2.4.1).

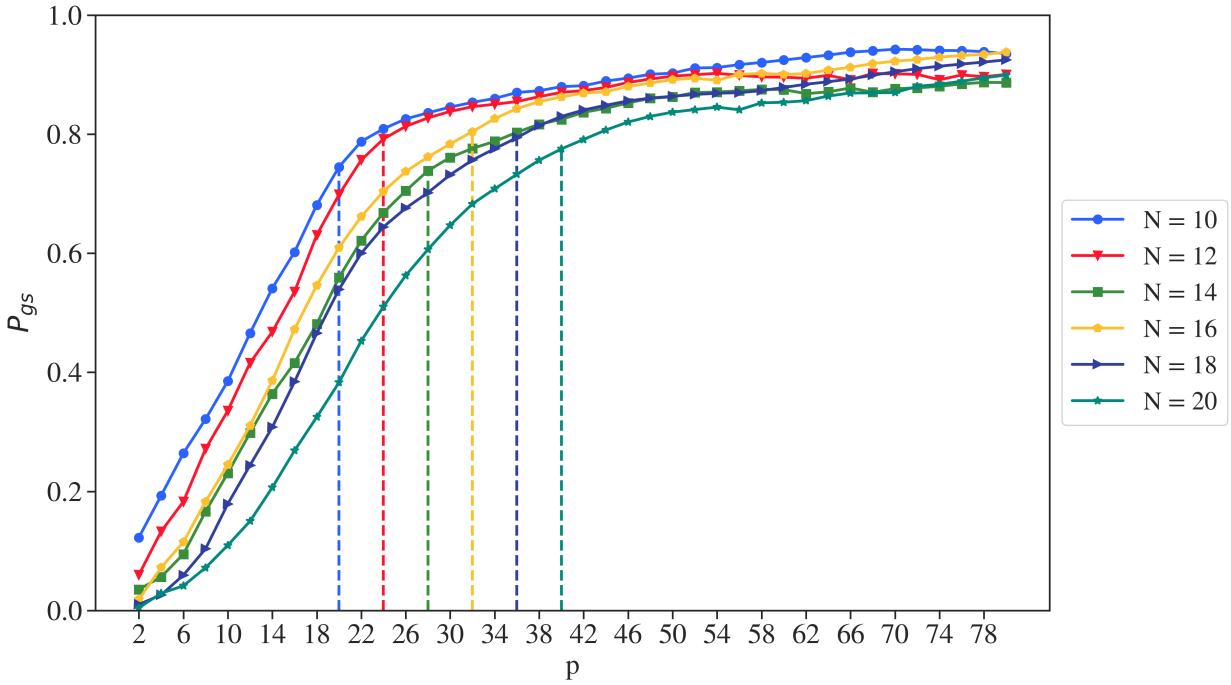
The first region of operation, located between $p = 0$ and $p \approx 2N$, corresponds to the situation where the success of the GQW is limited by the number of iterations p with respect to the Hamming distance of the basis states to the solution state. Thus, as p increases, P_{gs} scales approximately like a rescaled sum of binomial coefficients (see section 2.4.1), yielding the steepest increase in success probability at the region's centre with decreasing first derivatives towards the region's boarders. Note that the model was again allowed to occupy spread coefficients $\boldsymbol{\beta}$ below the proposed parameter bound $P_1^- - \pi/2$ within the first region (see Fig. 2.13b). Similar to Fig. 2.12b, the classical optimizer hence uses high interaction orders (i.e. $\boldsymbol{\beta} < P_1^- - \pi/2$) to artificially increase the number of accessible states in the graph in case of $p < 10$. As this increase with respect to the total number of states decreases as a function of N, the effect is most noticeable at small system sizes, causing a significant increase in P_{gs} for small p (see e.g. 10 and 12 qubit case in Fig. 2.13a for $p \leq 6$). Continuing with the second region (i.e. $p \in [2N + 1,\ 80]$), the success probabilities on all investigated system sizes seem to saturate between 88% and 94% at $p = 80$. This behaviour can again be explained by the distribution of the spread coefficients $\boldsymbol{\beta}$. Here, both Fig. 2.13b and Fig. 2.12c show an average increase of $\boldsymbol{\beta}$ as a function of p towards $\pi/2$, hence maintaining a guided quantum walk for larger probability gradients $r^i_{B,A}$. Concerning the regular graph, similar distributions to figure 2.12b are observed, with a rapid increase of $\boldsymbol{\beta}$ for $p \leq 10$, followed by a relatively fast saturation around 1.45. Again, as only minor differences are observed between the system sizes of 12 to 20 qubits, $\beta(i, \boldsymbol{\lambda})$ might be replaceable by a problem independent distribution, decreasing the number of optimization parameters by one. Moreover, regarding the inverse graph, only values within the lower half of the proposed parameter spectrum are used by the optimizer (i.e. $\boldsymbol{\beta} \leq \pi/2$), suggesting a reformulation of the parameter range in order to

further simply the optimization process. However, compared to Fig. 2.12b the distribution of $\boldsymbol{\beta}$ includes significant fluctuations, especially within the first region of operation. This indicates that the proposed sampling function $\gamma(i, \boldsymbol{\lambda})$ (see Eq. 2.70) is not optimal and additional research is required regarding the system's behaviour in the inverse graph in order to derive an improved heuristic sampling function.

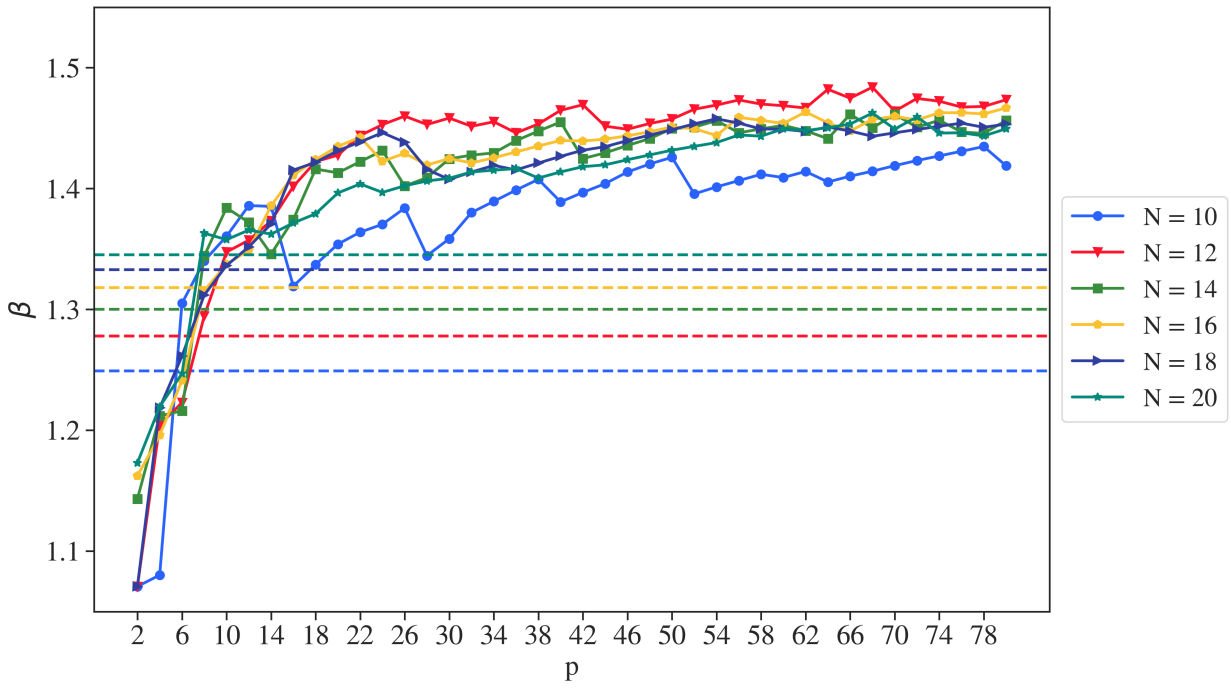
Although both the HGQW model and the HGQW-A model obtain similar shapes of the P_{gs} distributions within the two regions of operation, both models differ significantly regarding the sizes of these regions. Here, the HGQW-A model requires approximately twice as many iterations to access the probability of all states in the graph compared to the HGQW model, causing a significantly slower probability transfer. The latter was also observed in section 2.3.5 and can be explained by the small number of directed edges in the regular 2-SAT graphs. Thus, going from p to $p+1$ does not necessarily increase the number of reachable states by $\binom{N}{p+1}$, as probability transfers are likely to be blocked by insufficient phase gradients, due to the high degeneracies of the energy levels. Hence, the HGQW-A model proposes an alternation between the regular and inverse graph layout, with the latter functioning as a mixing operation. The success of this ansatz can be seen by the high success probabilities obtained at large p ($P_{gs}(p = 80) > 88\%$), which are significantly higher than the success probabilities obtained using only the regular graphs (see $P_{gs} \approx 5\%$ in section 2.3.5). However, the probability flow through the graph can only be directed within the regular graph, as in the inverse graph the interference mechanism cannot distinguish between transfers into states of lower and higher Hamming distance to the solution state. Therefore, only every second iteration applies a guided probability transfer, hence increasing the number of iterations necessary to access all states by approximately a factor of two.

Besides that, the derived distributions also seem to be less dependent on the number of qubits N. While an approximately exponential decrease in the success probability as a function of N was found in Fig. 2.12a, the distributions shown in Fig. 2.13a lie closer together and are less ordered in the system size. For example, similar probabilities are obtained for the $N = 14$ and $N = 18$ qubit problems, with the $N = 16$ qubit probabilities surpassing both. This observation suggests that the performance of the guided quantum walk on the 2-SAT instances is less dependent on the problem size, but on the ability of the mixing operation to overcome the system's degeneracies. This can also be seen by the fact that for larger system sizes (i.e. $N = 16$ to $N = 20$ qubit problems) higher success probabilities are derived at $p \geq 60$ than for smaller problem instances (i.e. $N = 10$ to $N = 14$ qubit problems). Most likely, this is related to the only linearly in N growing number of energy levels of the investigated 2-SAT problems (compare to exponentially growing in N for the exact cover instances). However, additional research is needed here.

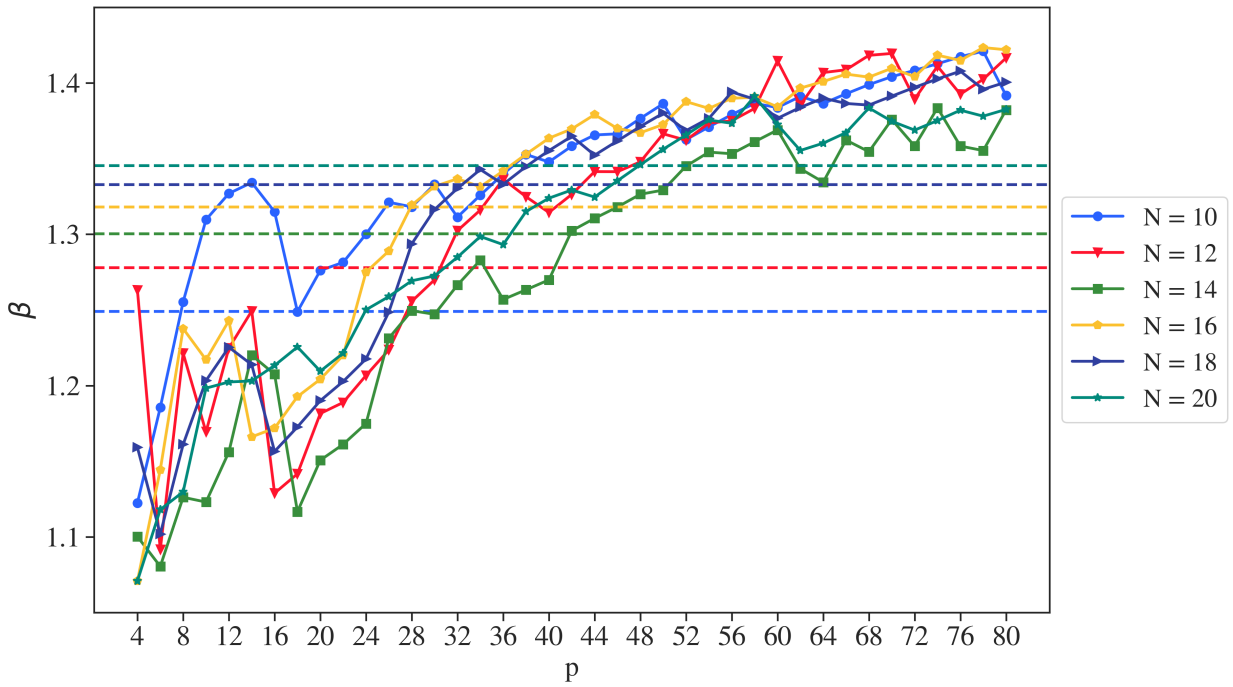
Concluding this section on the HGQW-A model, a weak exponential scaling in N is found, with e.g. $P_{gs}(N) \propto 2^{-0.16N}$ at $p = 10$ and $P_{gs}(N) \propto 2^{-0.01N}$ at $p = 40$, yielding a close-to constant distribution within the second region of operation. This demonstrates that the proposed alternating sequence between the regular and inverse picture allows the guided quantum walk to achieve similar success probabilities and scalings as seen for the HGQW model in section 2.4.1. Although requiring approximately twice as many iterations, the probability distributions still feature a region of rapidly increasing P_{gs} that grows seemingly linearly in N, making the GQW an interesting application on near-term NISQ devices also in the context of 2-SAT problems.



(a) Success probability P_{gs}



(b) Spread coeffiencts β (regular graph)



(c) Spread coeffiencts $\boldsymbol{\beta}$ (inverse graph)

Figure 2.12.: The plots show the distribution of the success probability P_{gs} (a) and the spread coefficients $\boldsymbol{\beta}$ (b, c) obtained by the HGQW-A model as a function of the total number of iterations p for the 2-SAT problems given in appendix H. The data is derived using a maximum of 4000 evaluations within 200 runs of the Nelder-Mead optimizer, focussing on a reduction of the approximation ratio A_r (see Eq. 2.7). For each problem instance and each circuit depth p, the best (highest P_{gs}) parameter set is chosen, and the data is averaged within the sample groups (i.e. across problems of the same size N). The individual sample groups are distinguished by colour, and dashed lines indicate $p = 2N$, $\boldsymbol{\beta} = P_1^- - \pi/2$ and $\boldsymbol{\beta} = P_1^- - \pi/2$, respectively.

2.4.3. Comparison of the GQW, the AQA and the QAOA

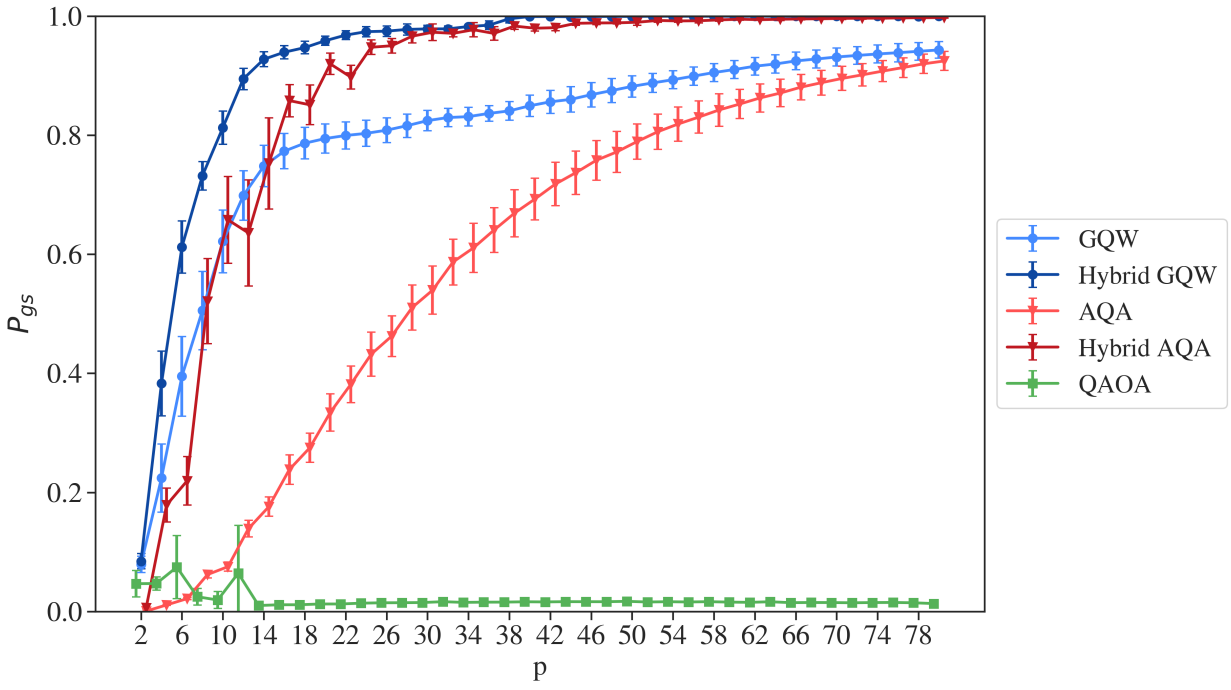
Both the investigations of the HGQW model and the HGQW-A model revealed promising properties of the guided quantum walk for solving exact cover and 2-SAT problems on near-term NISQ devices. Among others, a great initial increase of P_{gs} for $p \leq aN$ ($a = 1$ for HGQW and $a = 2$ for HGQW-A) was discovered, resulting in intermediate success probabilities ($P_{gs} \geq 0.5$) after already a few iterations (i.e. $p = aN\,/\,2$). For $p > aN$, where the algorithm is no longer limited by the accessibility of the basis states, the success probability was found to saturate, as a general decrease of the spread coefficients causes a rapid drop in the first derivative of P_{gs}. Since the scaling of P_{gs} in the number of qubits N flattens for increasing p, an almost constant distribution for $p > 4N$ is obtained. Even though the performance of the proposed heuristic models seems to be limited by some maximal success probability, both observations (strong scaling in p and weak scaling in N) suggest a competitive performance of the GQW at intermediate numbers of iterations (i.e. $p \approx 2N = \mathcal{O}(10)$). In order to verify this, the performance of the HGQW model and the HGQW-A model will be compared to two quantum optimization strategies previously proposed in the literature: On the one hand, the approximate quantum annealing (AQA) model developed by *Willsch et al.* will be considered here [30]. Explained in detail in section 2.2.3, AQA is a heuristic approach for solving

combinatorial optimization problems originating from the trotterized simulation of quantum annealing. It was motivated by the observation that competitive success probabilities are obtainable in case too large trotter errors are used in the numerical calculation. Consequently, the AQA focusses on the case of large time steps and thus low to intermediate p, similar to the GQW. The seconds ansatz that will be considered in the following is the standard QAOA as developed by *Farhi et al.* (see section 2.2.2) [65]. Even though both the GQW and the AQA live in a subspace of the QAOA, the goal of this comparison is to investigate the efficiency of the algorithms in tuning the system into the solution state after the same number of circuit runs (evaluations of the classical optimizer). This is important, as for the QAOA the parameter search space grows exponentially in p, while for the AQA and the GQW it remains constant, with complexities corresponding to $p = 0.5$ (AQA), $p = 2.5$ (HGQW model) and $p = 4$ (HGQW-A model).

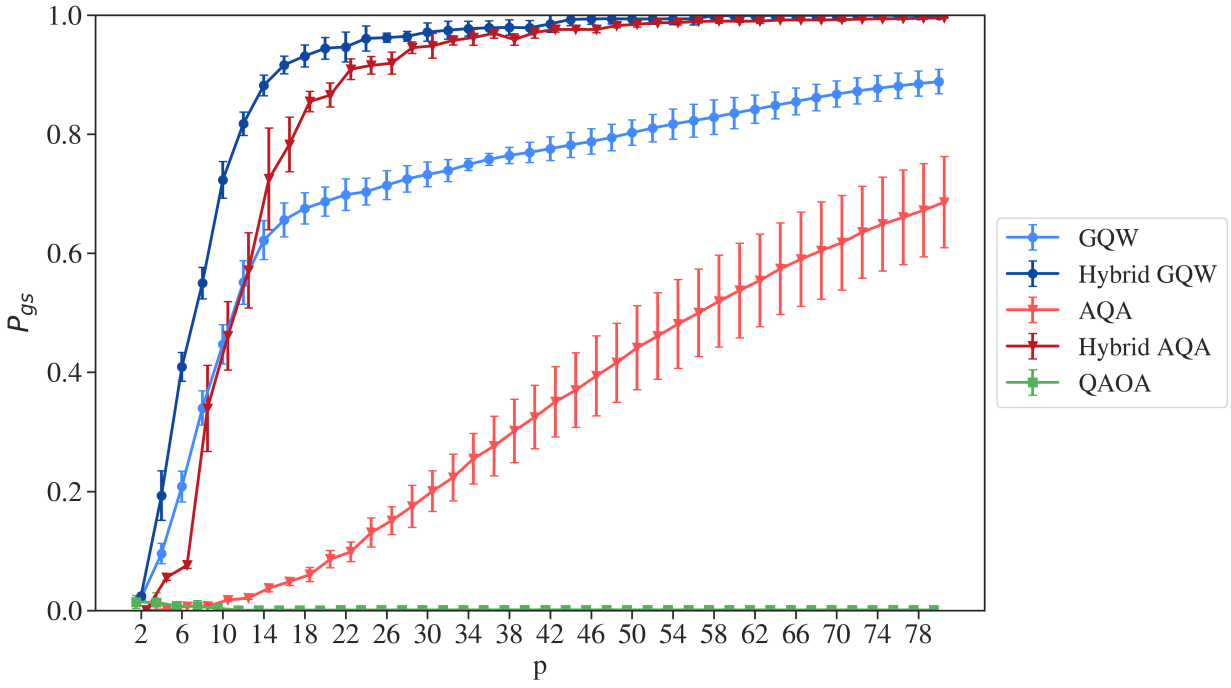
The data for both the AQA and the QAOA is obtained in the same manner as for the heuristic models in sections 2.4.1 and 2.4.2 by performing 200 optimization runs for each problem instance and each p, with randomly initialized parameter sets and a maximum of 4000 evaluations using the Nelder-Mead algorithm [110]. Note that the *best* (highest P_{gs}) optimization run is selected and averaged over the sample groups N and that the D-Wave annealing scheme [80] is used for the AQA as proposed by *Willsch et al.* [30]. The complete data set can be found in appendix J.2 and appendix J.3.

2.4.3.1. Performance on exact cover problems

The scaling of the success probabilities obtained by the three algorithm as a function of the circuit depth p (and thus the number of variational parameters) on the set of exact cover problems (see appendix G) is presented in the Fig. 2.13. Here, Fig. 2.13a (Fig. 2.13b) considers problem instances of $N = 12$ ($N = 16$) qubits. Beginning with the AQA (*red* curve), a monotonic increase of $P_{gs}(p)$ across the investigated problem instance is found. Here, the success probability approximately occupies the shape of a stretched sigmoid distribution, featuring an increasing first derivative for $p < 24$ ($p < 48$) and an approaching saturation for $p \geq 28$ ($p \geq 56$) and $N = 12$ ($N = 16$). Interestingly, the position of the turning point p_t (i.e. $P_{gs}(p_t) = 0.5$) increases as a function of N. By comparing the distribution of p_t across the simulated system sizes ($N = \{10, 12, 14, 16, 18\} \rightarrow p_t = \{24, 28, 40, 56, 72\}$) an approximately exponential scaling is found, suggesting that the AQA requires exponentially increasing circuit depths p in order to achieve similar success probabilities for growing problem size N. A possible explanation for this observation can be found in the structure of the energy spectra of the generated exact cover instances. As mentioned before, these problems feature numerous distinct energy levels, with quantities that increase approximately exponentially in the number of qubits N (see section 2.1.1). After rescaling each problem Hamiltonian to the same energy range, the smallest energy gab between the ground and the first excited state in a quantum annealing process generally decreases in N. Since, the AQA resembles trotterized QA for large p, and p is proportional to the annealing time τ, p_t must increase according to the adiabatic theorem (see Eq. 2.22). This could also explain the significantly larger standard deviations obtained for the AQA compared to the GQW (see e.g. Fig. 2.13b), as the performance of the AQA strongly depends on the size of the energy gaps, which vary across the investigated problem instances. Regarding the guided quantum walk, on the other hand, the standard deviations are within 5% and independent of the number of iterations p. This is because the GQW does not depend on the sizes of the energy gaps, but rather on their distribution, such that sufficient phase gradients can be established during the algorithm. Therefore, the GQW seems to be less dependent on the



(a) $N = 12$ qubits



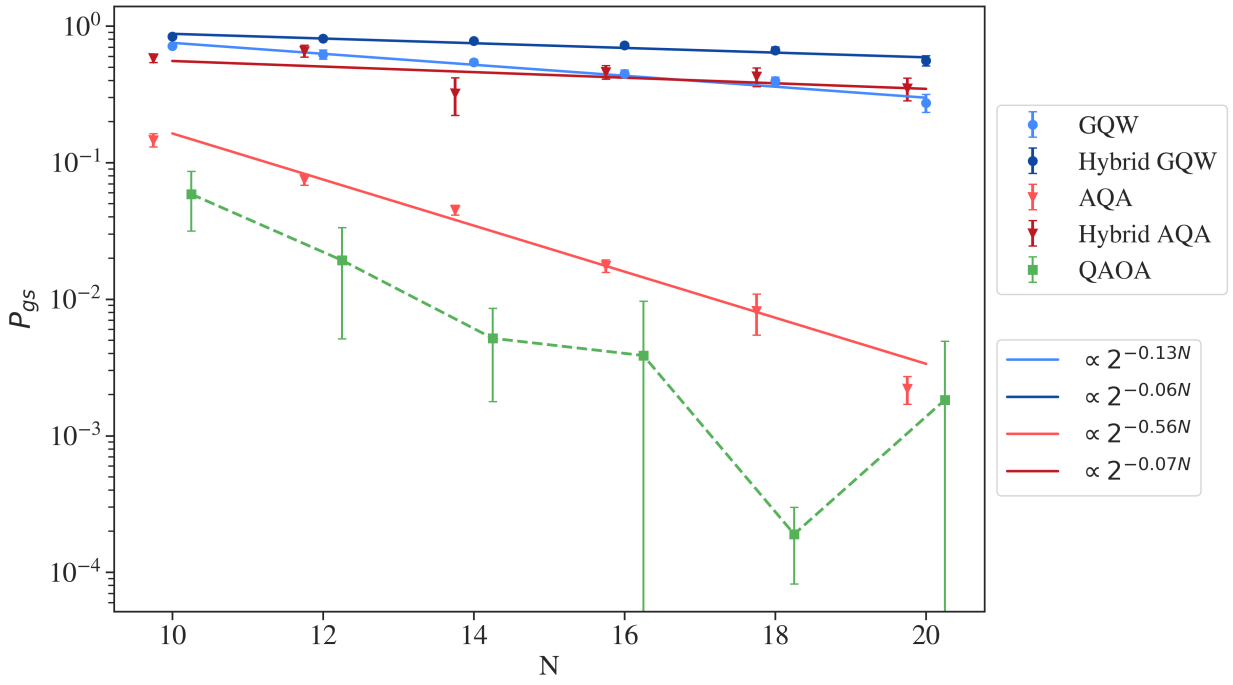
(b) $N = 16$ qubits

Figure 2.13.: The plots show the *best* success probabilities obtained by the GQW (blue points), the AQA (*red* triangles) and the QAOA (*green* squares) for the sets of $N = 12$ and $N = 16$ qubit exact cover problems (see appendix G) as a function of the number of iterations p. The data is averaged within the sample groups, with the standard deviations included in the plots. The *hybrid GQW* and the *hybrid AQA* curves represent the probability distributions obtained by initializing the QAOA with variational parameter sets derived by the GQW (*dark blue* curve) and the AQA (*dark red* curve), respectively. Note that the Nelder-Mead optimizer with 200 runs and a maximum of 4000 evaluations was used.

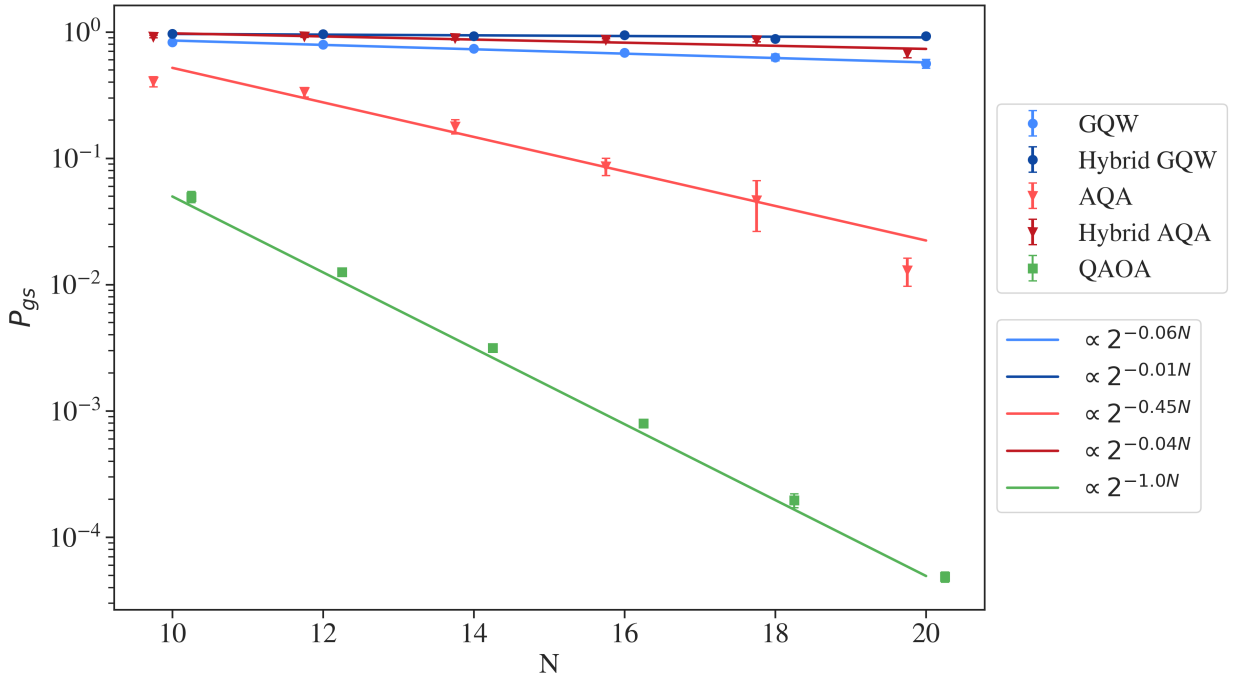
specific problem instance than the AQA, as long as the energy spectrum provides enough guidance. However, note that due to the small number (8) of simulated exact cover problems per system size N, additional research is needed to verify these assumptions. Besides that, across all problem instances, the AQA is not able to achieve higher success probabilities than the GQW for $p \leq 80$. Although $P_{gs} = 1$ is guaranteed by the adiabatic theorem for $p \to \infty$ using the AQA, concerning the regime of intermediate $p = \mathcal{O}(N)$, the guided quantum walk delivers superior success probabilities. In addition to that, the probability ratio $r_{AQA}^{GQW} = P_{gs}^{GQW}(p = N) \,/\, P_{gs}^{AQA}(p = N)$ seems to monotonically increase in N, indicating that the GQW will also outperform the AQA on real world problem instances with $N \gg 1$.

Continuing with the QAOA, which represents the third and most general optimization ansatz, where the $2p$ variational parameters are tuned individually, this observation is further supported. For $p \in [0, 6]$ the QAOA achieves success probabilities larger than the AQA but below the guided quantum walk. Interestingly, even for $p = 2$, where the QAOA searches a smaller parameter space than the GQW, this observation holds true, suggesting that the proposed heuristic functions $\beta(i, \boldsymbol{\lambda})$ and $\gamma(i, \boldsymbol{\lambda})$, are able to significantly reduce the complexity of the parameter search (i.e. increase its convexity), hence allowing the classical optimizer to find *good* sets of variational parameter more easily. In addition to that, as the parameter search space grows exponentially in p for the QAOA, the probability ratio $r_{QAOA}^{GQW} = P_{gs}^{GQW} / P_{gs}^{QAOA}$ monotonically increases as a function of p. Since the success probability decreases approximately exponentially, the AQA is able to surpass the QAOA at $p = 8$. This observation can be explained by the fixed maximal number of circuit evaluations, limiting the classical optimizer in locating sufficient local optima as the search space becomes more complex. Concerning the GQW and the AQA, on the other hand, the parameter search space has a constant dimensionality, such that the process of parameter tuning is not negatively effected by an increase of p.

Next, the scaling of P_{gs} as a function of N is shown in the Fig. 2.14a and Fig. 2.14b, with the former ($p = 10$) corresponding to a sample within the first region of operation ($p \leq N$), and the latter ($p = 20$) showing the scaling within the second region of operation ($p \geq N$). In order to compare the scaling of the three algorithms, exponential fits using $P(N) = a \cdot 2^{-bN}$ are also included in the plots (see *solid* curves). Regarding these fits, the QAOA achieves the poorest scaling out of the three algorithms, with a scaling of P_{gs} that matches random guessing (2^{-N}) for $p = 20$. This can be understood by the growing complexity of the search space, such that the classical optimizer generally uses all 4000 circuit evaluations. Interestingly, slightly better success probabilities are obtained at $N = 16$ and $N = 20$ for $p = 10$ than expected for the 2^{-N} scaling, indicating that $p = 10$ might be near the maximum number of iterations, where the QAOA can deliver sufficient success probabilities given 4000 evaluations. Consequently, this also demonstrates the efficient use of computational resources by both the GQW and the AQA, which deliver significantly better scalings with $b \leq 0.56$. Interestingly, both algorithms improve in their scalings with increasing p, due to the decreasing slopes of $P_{gs}(p)$ within the second regions of operation. Note that since the turning point for AQA shifts exponentially towards larger p, a curvature in the distribution of the success probabilities emerges (see e.g. Fig. 2.14b). Concerning the guided quantum walk, even within the first region of operation a significantly better scaling ($b = 0.13$) is found compared to both the AQA and the QAOA ($b \geq 0.45$), while in the second region of operation the scaling becomes approximately linear ($b = 0.06$). This verifies the aforementioned observations that r_{AQA}^{GQW} and r_{QAOA}^{GQW} increase in N, yielding superior success probabilities of the GQW for $p \in [N, 2N]$ on the set of exact cover problems.



(a) $p = 10$ iterations



(b) $p = 20$ iterations

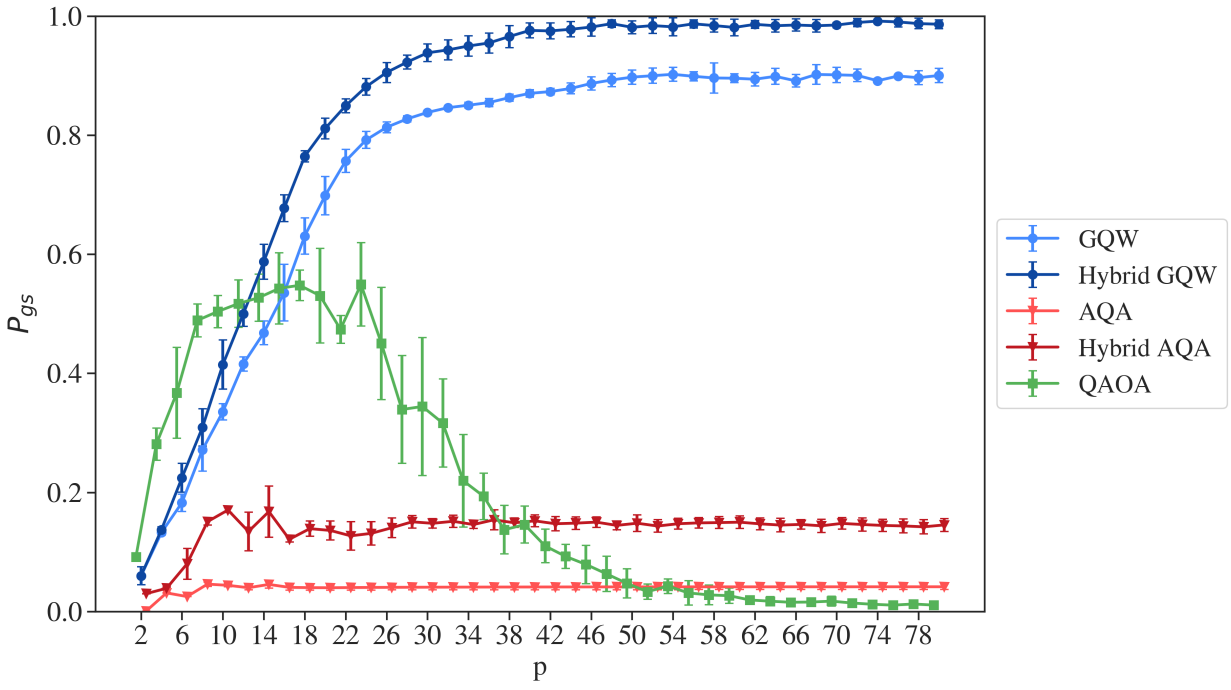
Figure 2.14.: The plots show the *best* success probabilities obtained by the GQW (blue points), the AQA (*red* triangles) and the QAOA (*green* squares) for $p = 10$ and $p = 20$ iterations as a function of the number of qubits N on the set of exact cover problems (see appendix G). The data is averaged within the sample groups, with the standard deviations included in the plots. The *hybrid GQW* and the *hybrid AQA* represent the probability distributions obtained by initializing the QAOA with variational parameter sets derived by the GQW (*dark blue* points) and the AQA (*dark red* triangles), respectively. Note that the Nelder-Mead optimizer with 200 runs and a maximum of 4000 evaluations is used. Moreover, the *solid* curves denote exponential fits using $a2^{-bN}$.

2.4.3.2. Performance on 2-SAT problems

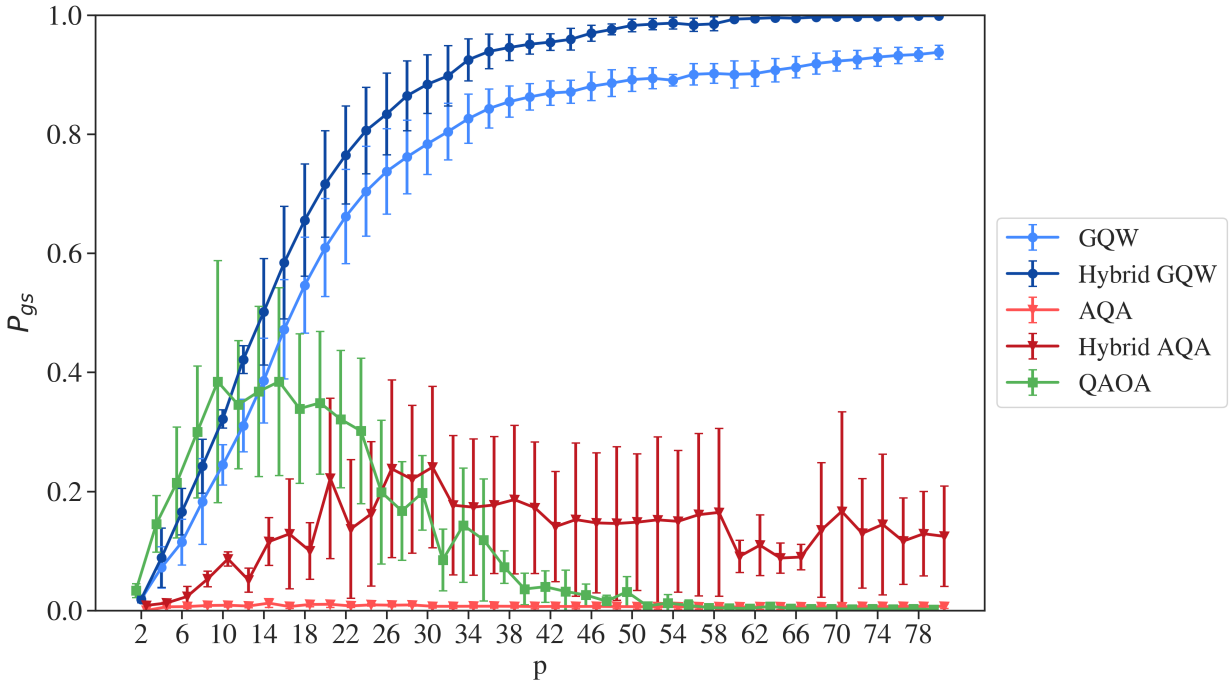
Continuing with the set of 2-SAT problems (see appendix H), the scaling of the success probabilities as a function of the circuit depth p is presented in Fig. 2.15a ($N = 12$) and Fig. 2.15b ($N = 16$) for the three algorithms. The plots reveal similar distributions of P_{gs} in case of the GQW as seen in Fig. 2.13. As discussed in section 2.4.2, the size of the first region of operation is approximately scaled by a factor of two due to the reduced number of directed connections in the graph. In contrast to this, both the AQA and the QAOA show a completely different scaling behaviour compared to section 2.4.3.1.

Regarding the AQA, a constant distribution is obtained for both system sizes of $N = 12$ and $N = 16$ qubits, with success probabilities of $P_{gs} = 4\%$ and $P_{gs} = 0.6\%$, respectively. Although the investigated 2-SAT instances were designed to be hard to solve on quantum annealing devices, the lack of a continues increase of P_{gs} as seen for the exact cover problems remains an open question. Since the adiabatic theorem assures unit success for $p \to \infty$, the AQA might require significantly more iterations, in order to increase the annealing time τ and decrease trotterization errors, such that the system can remain in its ground state. Concerning the QAOA ansatz, a significant improvement of the success probability is found compared to its application on the set of exact cover problems (see Fig. 2.13). For $p \in [0, 12]$, P_{gs} increases monotonically, reaching peak success probabilities of $P_{gs}(14) \approx 0.52$ and $P_{gs}(14) \approx 0.36$ at $p \approx 14$ for $N = 12$ and $N = 16$, respectively. For $p \geq 14$, however, the success probabilities start to decrease exponentially, driven by the increasing size of the parameter search space, until they reach saturation probabilities at $p \geq 66$ of $P_{gs}(80) \approx 1\%$ and $P_{gs}(80) \approx 0.1\%$, respectively. In doing so, the QAOA is able to achieve higher success probabilities between $p = 0$ and $p = 14$ than both the guided quantum walk and the AQA. This observation suggests, that the parameter search space is significantly easier to sample for this problem type than for the exact cover instances. A possible explanation could be that due to the high degeneracies of the energy levels and the resulting small number of unique energy gradients between the basis states, the system is less sensitive to large jumps in the variational parameter distributions, which is something that typically occurs when initializing $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ randomly. On the other hand, the exact cover instances feature numerous distinct energy gaps which also tend to be ordered in size (see e.g. Fig. 2.7). As a consequence, the variational parameters have to be chosen much more carefully, in order to tackle the correct exchanges and prevent counteracting effects. However, additional research is required.

Finally, Fig. 2.16a and Fig. 2.16b depict the scaling of P_{gs} as a function of the system size at $p = 20$ (first region of operation) and $p = 40$ (second region of operation), respectively. Since the distribution of the AQA shows large fluctuations, no exponential fitting is included for it. Compared to section 2.4.3.1 both the scaling of the GQW and the QAOA are significantly improved, with $b \leq 0.36$ and $b \leq 0.08$, respectively. As to be expected, the scaling of the QAOA worsens with increasing p, due to the growing search space, such that for large p a similar scaling as seen for the exact cover problems (i.e. $P_{gs}(N) \propto 2^{-N}$) is to be expected. Regarding the GQW, an almost constant scaling is found for the investigated problem sizes up to 20 qubits. This supports the approach of using an alternating sequence between the regular and inverse spectrum to increase the dynamics of the quantum walker. Thus, the guided quantum walk, is able to deliver superior success probabilities compared to both the QAOA and the AQA given a fixed number of circuit evaluations on the set of 2-SAT problems.

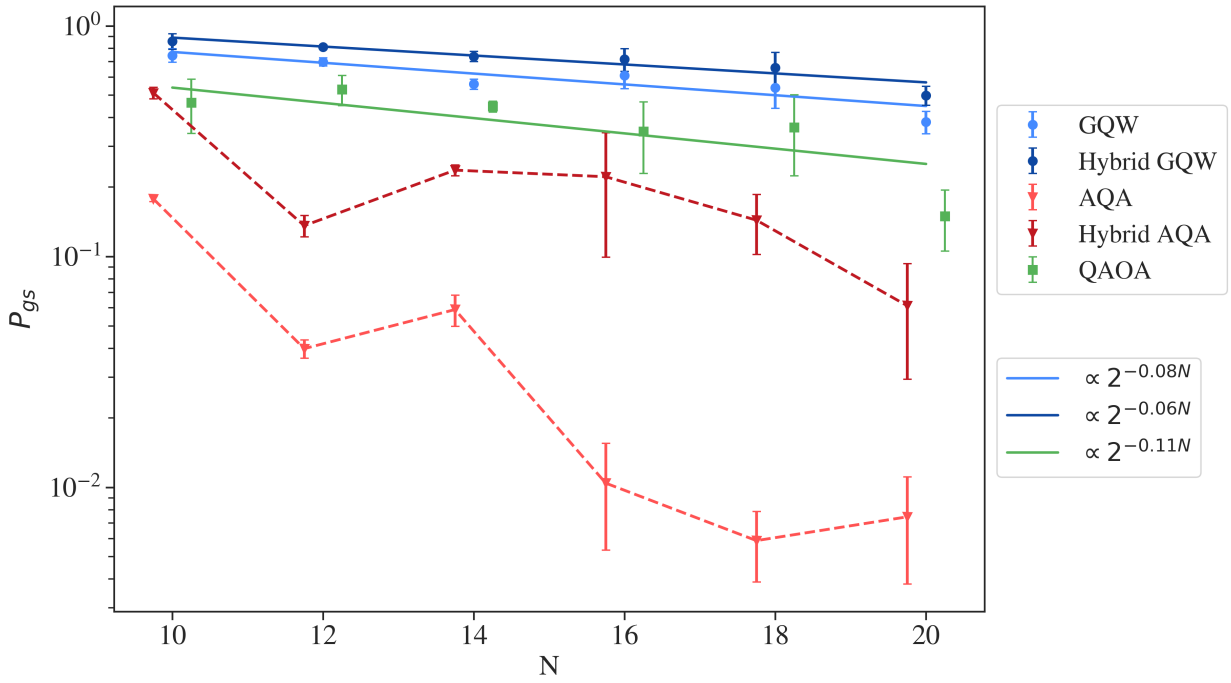


(a) $N = 12$ qubits

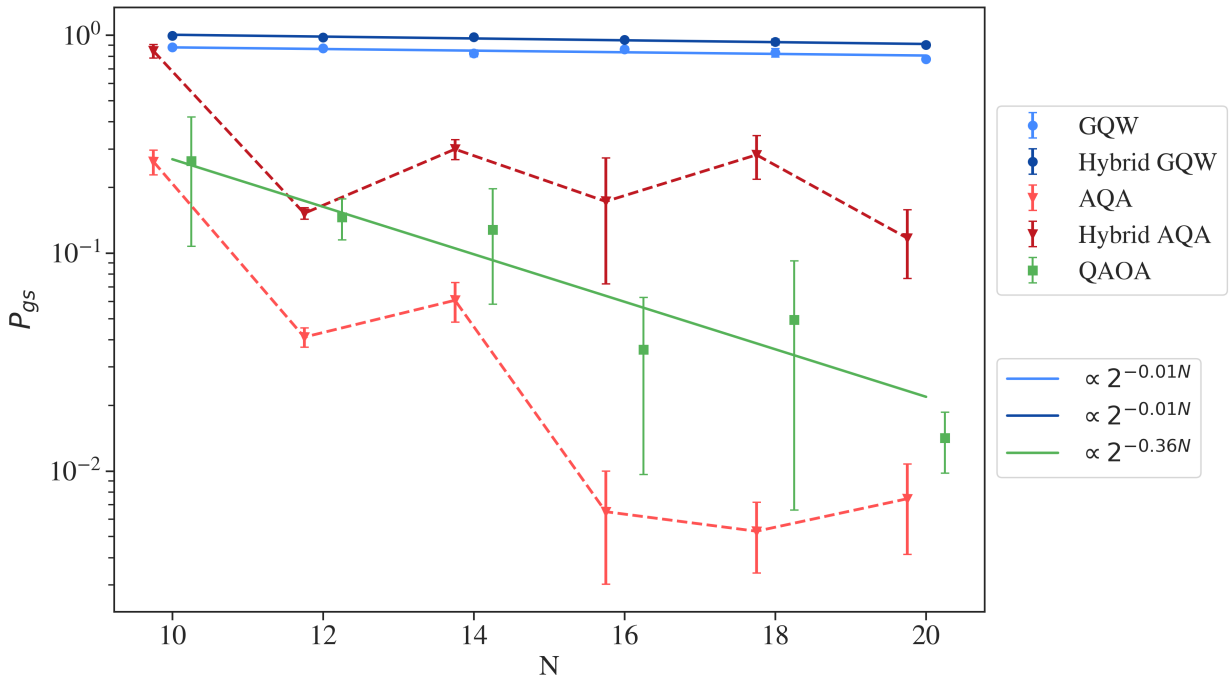


(b) $N = 16$ qubits

Figure 2.15.: The plots show the *best* success probabilities obtained by the GQW (blue points), the AQA (*red* triangles) and the QAOA (*green* squares) for the sets of $N = 12$ and $N = 16$ qubit 2-SAT problems (see appendix H) as a function of the number of iterations p. The data is averaged within the sample groups, with the standard deviations included in the plots. The *hybrid GQW* and the *hybrid AQA* curves represent the probability distributions obtained by initializing the QAOA with variational parameter sets derived by the GQW (*dark blue* curve) and the AQA (*dark red* curve), respectively. Note that the Nelder-Mead optimizer with 200 runs and a maximum of 4000 evaluations was used.



(a) $p = 20$ iterations



(b) $p = 40$ iterations

Figure 2.16.: The plots show the *best* success probabilities obtained by the GQW (blue points), the AQA (*red* triangles) and the QAOA (*green* squares) for $p = 10$ and $p = 20$ iterations as a function of the number of qubits N on the set of 2-SAT problems (see appendix H). The data is averaged within the sample groups, with the standard deviations included in the plots. The *hybrid GQW* and the *hybrid AQA* represent the probability distributions obtained by initializing the QAOA with variational parameter sets derived by the GQW (*dark blue* points) and the AQA (*dark red* triangles), respectively. Note that the Nelder-Mead optimizer with 200 runs and a maximum of 4000 evaluations is used. Moreover, the *solid* curves denote exponential fits using $a2^{-bN}$.

2.4.4. Hybrid algorithms

With the comparison of the success probabilities between the guided quantum walk, the AQA and the standard QAOA, section 2.4.3 demonstrated that the GQW can achieve superior success probabilities in the regime of intermediate circuit depths ($p = \mathcal{O}(N)$) compared to other common quantum optimization algorithms. By analysing the distribution of P_{gs} as a function of p, the GQW revealed two regimes of operation, with a great increase of P_{gs} within the first N iterations, due to the exponential increase in the number of accessible states, and a saturation of the success probability below one for $p \geq N$. In order to improve the performance of the heuristic models in especially the second region, a hybrid algorithm will be investigated in this section, utilizing the strong connection between the GQW and the QAOA. The algorithm is structured in two stages: First, a guided quantum walk is performed on the system, yielding a set of *already good* variational parameters. However, since approximations are considered in the derivation of the heuristic sampling functions $\beta(i, \boldsymbol{\lambda})$ and $\gamma(i, \boldsymbol{\lambda})$, in order to keep the parameter search space feasible, small adjustments to the individual values might significantly improve P_{gs}. Thus, as a second step, the parameter set proposed by the GQW is used to initialize the QAOA. In other words, the GQW is used to determine the general distributions of the variational parameters, which are then fine-tuned individually via the QAOA. The hope is that with this pre-tuned parameter set, the QAOA should already start in the neighbourhood of a good local optimum, hence significantly reducing the number of evaluations required by the classical optimizer. This strategy is motivated by the fact that the QAOA alone struggles to find *good* parameter sets for $p \geq 20$ for both exact cover and 2-SAT problems, due to the high complexity of the search space, when initialized randomly (see sections 2.4.3.1 and 2.4.3.2).

This hybrid ansatz is applied to the aforementioned problem instances using the previously derived parameter sets and is shown alongside the original success probabilities in Fig. 2.13 to Fig. 2.16 (*dark blue* dashed curves). For comparison, an analogue hybrid strategy using the variational parameters obtained by the AQA as initial values (*dark red* dashed curves) is also shown. Considering the hybrid GQW approach, the obtained mean success probabilities as a function of p approximately follow a rescaled distribution of P_{gs} as found for the original GQW on both the sets of exact cover and 2-SAT problems (see Fig. 2.13 and Fig. 2.15). Consequently, the hybrid strategy features again two regions of operation, with a great increase of P_{gs} for $p \leq N$ ($p \leq 2N$) and a saturation for $p > N$ ($p > 2N$) on the exact cover (2-SAT) problems. This suggests, that the fine-tuned sets of variational parameters are still located in the subspace of the guided quantum walk, verifying the discussions in section 2.3.2 on restricting the movement of the quantum walker. With respect to Fig. 2.18 and Fig. 2.19 this assumption is confirmed in the case of $N = 16$ and $p = 40$, as fine-tuned parameters (*dark blue* points) are located close to the initial parameters sets (*blue* points), hence following a similar distribution on average. In doing so, $p_{gs} \geq 0.9$ is achieved within the second regions, reaching almost unit success probability for $p = 80$. This is especially interesting in the case of the $N = 12$ qubit 2-SAT problems, where the GQW saturates at $P_{gs} \approx 0.9$, indicating the limited complexity of the heuristic sampling functions. With respect to the computational complexity (i.e. number of circit evaluations), this supports the hybrid ansatz of combining the GQW (providing simplified initial parameter distributions) and the QAOA (increasing the complexity of the parameter distributions). In doing so, the final success probabilities seem to be less dependent on the system size N, yielding an approximately constant scaling within the second region as a function of N (see Fig. 2.14 and Fig. 2.16). Moreover, the standard deviations within the sample groups are similar to the ones obtained for the original GQW, suggesting only a weak dependency on the individual problem instances.

Concerning the hybrid AQA ansatz, significant differences in the scaling behaviour of the success probability on both exact cover and 2-SAT problems compared to the original AQA ansatz are found. Regarding exact cover problems, the hybrid AQA strategy now achieves a similar distribution of $P_{gs}(p)$ as the hybrid GQW, shifted right by approximately $\Delta p = 4$. Hence, a strong increase in the success probability is obtained for $p \in [0, 22]$, yielding $P_{gs} \geq 0.9$ for $p \geq 22$. This change in the distribution of P_{gs} compared to the original AQA suggests that the fine-tuned parameter set does not necessarily correspond to a trotterized annealing scheme any more. This assumption is supported by Fig. 2.18 and Fig. 2.19 in the case of $N = 16$ and $p = 40$, as the final variational parameters lie far away from their initial values. Regarding the 2-SAT problems, the success probabilities obtained by the hybrid AQA strategy lie significantly below the ones of the hybrid GQW ansatz. Here, P_{gs} increases for $p < 10$ ($p < 28$) until it saturates at $\approx 15\%$ (fluctuates between $\approx 8\%$ and $\approx 18\%$) on the $N = 12$ ($N = 16$) qubit problems. This can be understood by the aforementioned observation that the AQA struggles to find good parameter sets, such that the initial position of the QAOA in the parameter search space supposedly has a great distance to any sufficient local optimum. This also explains the large standard deviations seen in Fig. 2.15b, since the classical optimizer (executed only once) has to move a long distance in the highly complex search space.

Comparing both hybrid approaches, the hybrid GQW ansatz seems to provide superior success probabilities when solving 2-SAT problems compared to the hybrid AQA approach. Regarding exact cover problems, on the other hand, both hybrid strategies achieve similar success probabilities for $p \geq 2N$, making the hybrid AQA strategy a competitive approach at intermediate p. However, while performing the simulations, it was found that the hybrid ansatz initialized via the AQA required significantly more evaluations by the classical optimizer than the one using the GQW parameter sets. As this is caused by differences in the distances between the initial and final parameter sets in the search space, Fig 2.17 depicts the distance metric $d_{\beta,\gamma}$ [106] defined as:

$$d_{\beta,\gamma} = \sum_{i=0}^{p-1} \left[\left| \beta_i^{init} - \beta_i^{opt} \right|_{\frac{\pi}{2}} + \left| \gamma_i^{init} - \gamma_i^{opt} \right| \right]. \tag{2.75}$$

Here, $|\ldots|_{\frac{\pi}{2}}$ denotes the absolute value modulo $\pi/2$ which takes into account symmetries, and *init* and *opt* label the initial and optimized parameter sets obtained by the GQW (AQA) and the hybrid GQW (AQA) ansatz, respectively. In doing so, Fig 2.17 reveals an approximately linear increase of $d_{\beta,\gamma}$ as a function of p in the context of the hybrid AQA. In contrast to this, $d_{\beta,\gamma}$ stays approximately constant at 5 in case of the hybrid GQW. Interestingly, no significant differences are found between the investigated system sizes N. As a result, the computational work needed for the second stage of the hybrid AQA ansatz increases in p and consequently in N, as more iterations p are typically required for increased system sizes N. However, regarding the hybrid guided quantum walk, the computational work of the second stage seems to be constant, which, in combination with the proposed linear scaling of p in N, suggests that the guided quantum walk could be able to reach unit success probability using only computational resources that scale linearly in the number of qubits N.

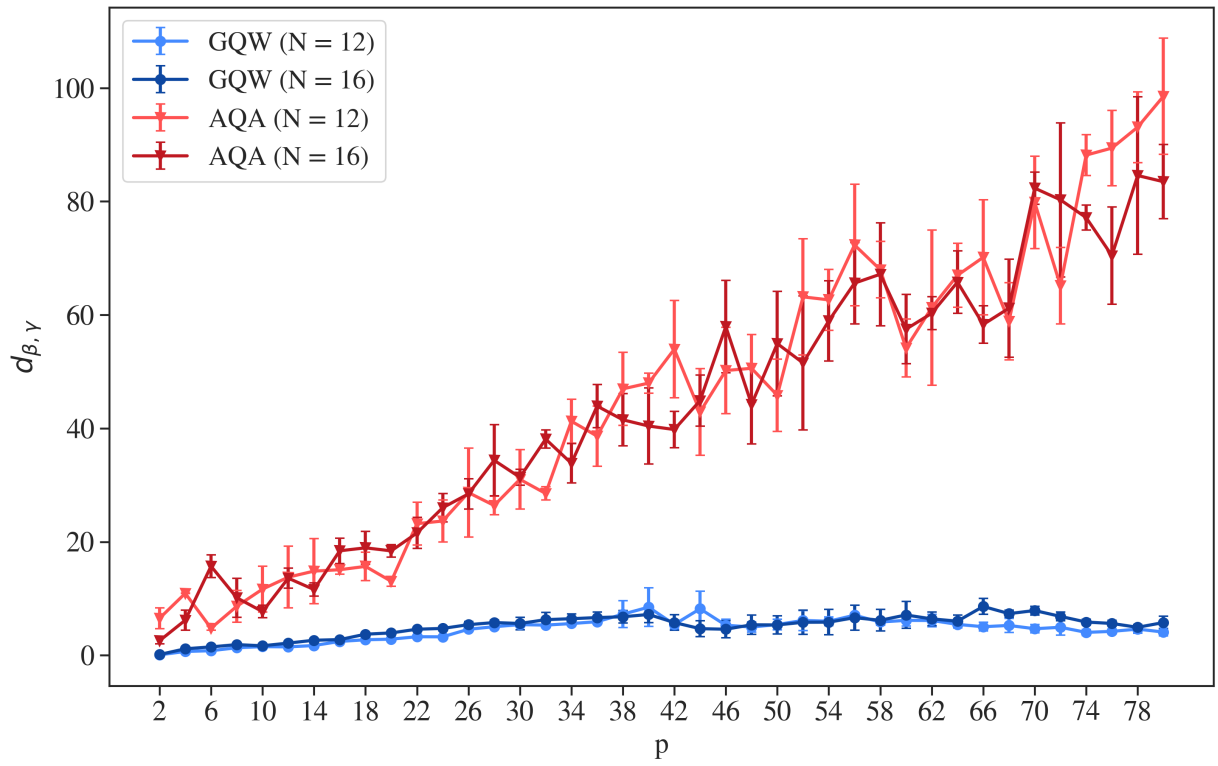
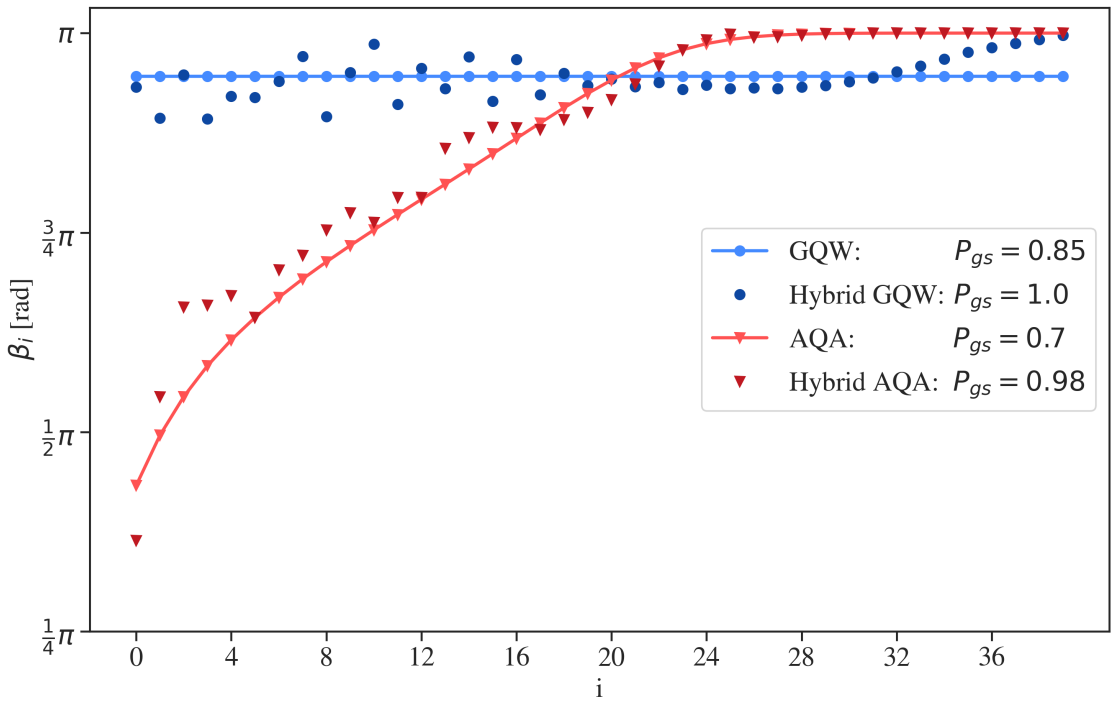
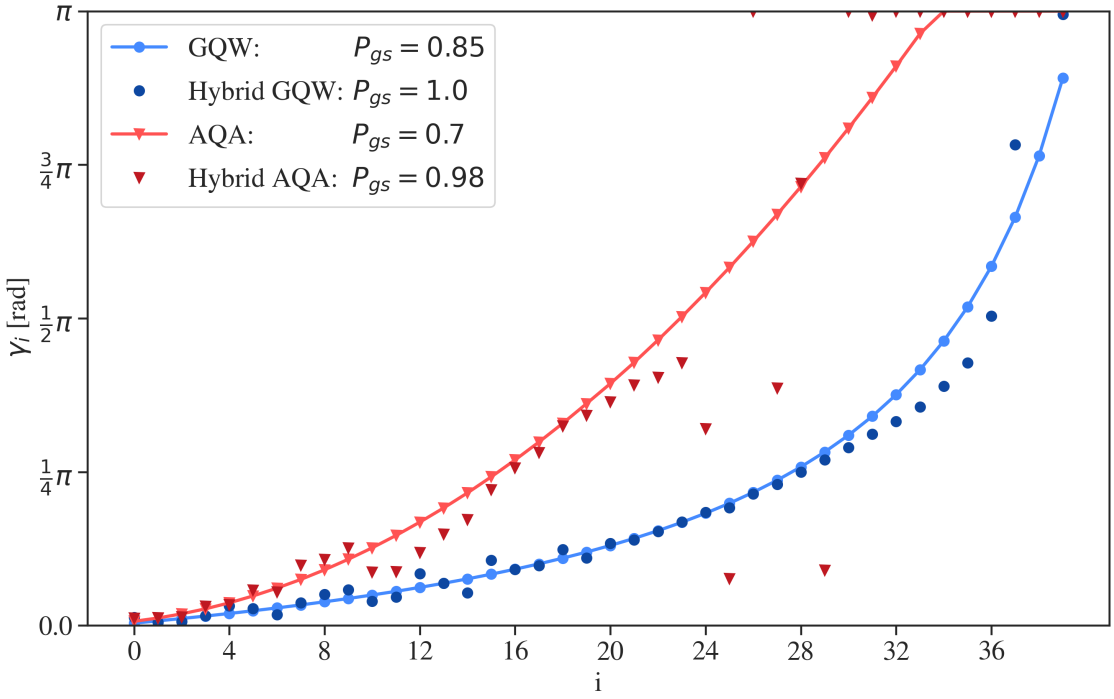


Figure 2.17.: The figure shows the parameter distances $d_{\beta,\gamma}$ (see Eq. 2.75) as a function of the number of iterations p between the initial and final parameter sets of the hybrid GQW (*blue* curve) and hybrid AQA (*red* curve) strategies. The data is averaged within the sample groups and shown for the $N = 12$ (*blue* and *red* curves) qubit and $N = 16$ (*dark blue* and *dark red* curves) qubit problems. The initial parameter sets are determined using 200 optimization runs with a maximum of 4000 evaluations of the Nelder-Mead routine. The subsequent QAOA optimization was limited to 4000 evaluations.

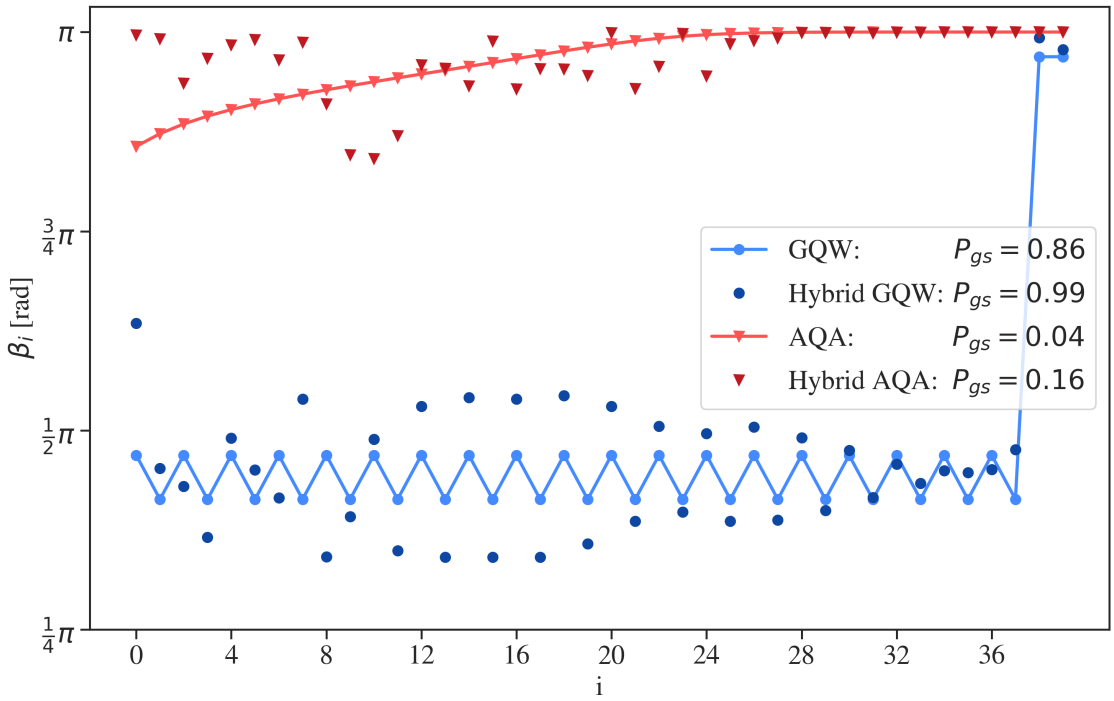


(a) Spread coeffients $\boldsymbol{\beta}$

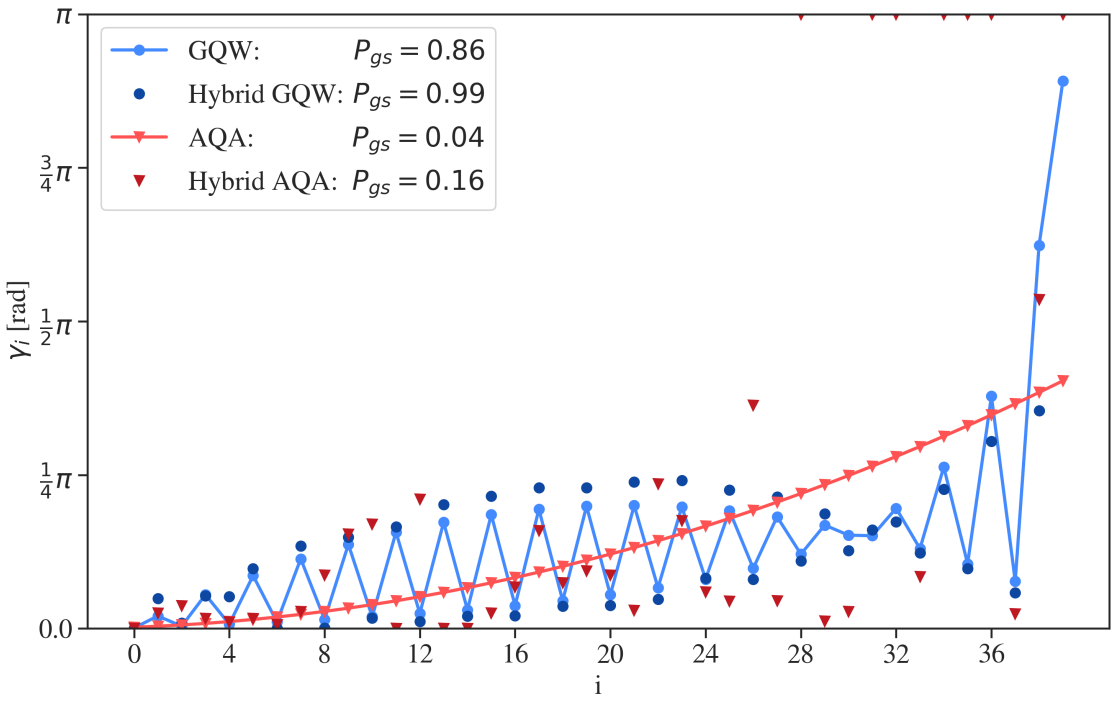


(b) Phase coeffients $\boldsymbol{\gamma}$

Figure 2.18.: The plots show the initial (*blue* and *red* points) and fine-tuned (*dark blue* and *dark red* points) sets of the spread coefficients $\boldsymbol{\beta}$ (a) and phase coefficients $\boldsymbol{\gamma}$ (b) determined by the hybrid GQW (*blue*) and the hybrid AQA (*red*) strategies for the `EC_16_1` exact cover problem. The initial parameter sets are determined using 200 optimization runs with a maximum of 4000 evaluations of the Nelder-Mead routine. The subsequent QAOA optimization was limited to 4000 evaluations.



(a) Spread coeffients $\boldsymbol{\beta}$



(b) Phase coeffients $\boldsymbol{\gamma}$

Figure 2.19.: The plots show the initial (*blue* and *red* points) and fine-tuned (*dark blue* and *dark red* points) sets of the spread coefficients $\boldsymbol{\beta}$ (a) and phase coefficients $\boldsymbol{\gamma}$ (b) determined by the hybrid GQW (*blue*) and the hybrid AQA (*red*) strategies for the `2SAT_16_1` 2-SAT problem. The initial parameter sets are determined using 200 optimization runs with a maximum of 4000 evaluations of the Nelder-Mead routine. The subsequent QAOA optimization was limited to 4000 evaluations.

Conclusion

SEQCS The first chapter of this thesis concerned the high-performance simulation of large scale quantum systems ($N \geq 30$). Motivated by the high parallelism involved in the computation of general quantum gates, SEQCS, a novel quantum circuit simulator utilizing GPU-accelerators spread across multiple compute nodes, has been developed. In its implementation, novel optimization strategies regarding the memory management, memory access patterns and the load balance have been proposed in order to increase the overall utilization of the GPU running the memory bound code.

The first area of improvement concerned the distribution of a state vector across multiple GPUs, allowing SEQCS to simulate quantum circuits of more than 30 qubits. Here, each GPU operates on an exclusive subset of the state space, separating the qubits into local (located at a single GPU) and global (distributed among two GPUs) qubits. Since, pairs of GPUs must exchange half of their memory each time a non-diagonal gate is executed on a global qubit, SEQCS proposes a reordering algorithm in order to alter the execution sequence of the gate operations in the circuit. This was motivated by the observation that combining multiple qubit exchanges into a single MPI transfer significantly reduces the amount of bytes communicated through the internode network. Moreover, SEQCS introduces a permutation operator to track the locations of the qubits among the GPUs, allowing it to skip the back transfer of the state amplitudes after the execution of a global gate operation by swapping the memory locations of a global and a local qubit. In doing so, SEQCS achieved an up to 2.5 times faster MPI communication than JUQCS-G on the 36 qubit Hadamard benchmark circuit.

The second area of improvement focussed on the caching of the state amplitudes during the execution of local quantum gates. Here, SEQCS introduces the shared memory into the simulation process in order to reduce the number of global memory transactions. This was motivated by the fact that accesses to the shared memory can be executed up to 100 times faster than request to the global memory. Hence, SEQCS proposes a second preprocessing algorithm to rearrange the gate operations into gate clusters. Each gate cluster operates on an 11 qubit subspace of the local state vector, such that it can be simulated in parallel among the streaming multiprocessors. In doing so, each update routine is executed on local copies of the state amplitudes in the shared memory, thus significantly increasing the memory access speeds and the computational intensity. As a result, an ideal strong scaling of the gate execution time in the gate cluster size for up to 8 clustered gates is found, with a maximum speed-up of 12.7 for 128 clustered Hadamard gates.

The third area of improvement dealt with the combined execution of multiple gate operations. Here, SEQCS proposes a preprocessing algorithm to group quantum gates acting on disjoint qubit sets into a single update routine. By grouping only gates of the same type,

SEQCS is able to accumulate the effects of the individual gate matrices, such that each state amplitude must be accessed only once during the execution of a gate group. In doing so, the computational intensity of the simulation is further increased, yielding a close-to ideal strong scaling of the gate execution time in the number of grouped single qubit diagonal gates. For example, SEQCS can simulate the Z gate up to 329 times faster than JUQCS-G. Note that the speed-ups depend on the amount of arithmetic operations involved in the update kernels, indicating that the simulation is no longer memory-bound. This demonstrates the success of the proposed optimizations to the memory management. Regarding non-diagonal gates, the gate group size is limited to two gates, yielding an up to 17 times faster execution of the Hadamard gate compared to JUQCS-G. This restriction is due to the small number of registers available to each CUDA thread. Besides that, SEQCS uses an out-of-order execution scheme for this gate type in order to prevent shared memory bank conflicts and increase the load balance within the CUDA warps. It was shown that this execution scheme reduces the execution time by 17% compared to an in-order gate execution.

As a result of these optimizations, SEQCS is able to simulate the 31 qubit quantum Fourier transformation 22.6 times faster and the 31 qubit QAOA circuit ($p = 50$) 70.4 times faster than JUQCS-G. Hence, SEQCS provides a valuable tool for investigating and simulating large-scale quantum circuits on classical hardware. Potential further improvements to the simulator include the following aspects:

- During the distributed simulation of the Hadamard benchmark circuit, it was found that the implementation of a single qubit exchange between two GPUs is approximately 10 times slower than the implementation used in JUQCS-G. This is most likely caused by the extensive use of unoptimized MPI data structures (e.g. *MPI_Type_vector* and *MPI_Type_indexed*) and an insufficient buffer management by MPI, potentially causing expensive Device-Host memory transfers. Hence, an improved implementation of the GPU-GPU communication is required in order to fully utilize the speed-up provided by the combined qubit exchanges.

- Different communication frameworks, such as *NCCL* [59] and *NVSHMEM* [60] could further reduce the GPU-GPU communication time.

- The investigation of the various benchmark circuits showed that the update kernel used for combined gate executions introduces additional overhead into the simulation process. This becomes especially important in the case of arbitrary rotation gates that require frequent accesses to the constant memory. Hence, accelerating the algorithm used to map the state amplitudes to the CUDA threads and storing frequently used data in the registers, could increase the simulation speed.

- Regarding certain quantum algorithms (e.g. the QAOA and Shor's integer factorization), the simulation speed could be significantly increased by designing specialized update routines that replace large sections of the respective quantum circuits. This strategy is called quantum circuit emulation.

- In order to allow the investigation of quantum algorithms running on NISQ devices, the simulator could be extended to include error sources during the application of quantum gates and qubit measurements.

Guided quantum walk The second chapter of this thesis concerned the investigation of quantum optimization algorithms. Inspired by the operation of continuous quantum walks and the QAOA, the guided quantum walk, a novel hybrid quantum-classical variational algorithm for deriving approximate solutions to combinatorial optimization problems, has been developed. The algorithm deploys a quantum walker on an oriented graph connecting the problem's solution space in order to direct a probability flow towards the solution state. Considering a hypercube mapping between the computational basis states, the GQW uses a set of spread coefficients to select specific interaction orders between the computational basis states based on their Hamming distance. Here, the GQW restricts the walker's movement to nearest neighbour interactions only in order to accumulate probability in low energy states. In addition to this, the direction of probability transfer is determined by a set of phase coefficients, controlling the complex phase gradients between connected basis states in the graph.

Based on the above concepts, the HGQW model for executing guided quantum walks on exact cover problems has been introduced. This model is inspired by the immense splitting of the energy levels found across all investigated exact cover instances, allowing to sufficiently distinguish neighbouring states in the graph based on their energy. Hence, two heuristic sampling functions focussing on the distribution of the energy gradients in the problem Hamiltonian have been proposed, yielding a constant and an exponential distribution of the spread and phase coefficients, respectively. In doing so, the HGQW model tunes a fixed set of optimization parameters, which are used to adjust the shape of the sampling functions to each specific problem instance, instead of optimizing each variational parameter separately. This strategy was motivated by the observation of common patterns in optimized parameter sets and reduces the exponential in the number of parameters growing complexity of the optimization process to a constant factor. By applying the HGQW model to an exact cover instance, it was shown that the GQW is able to introduce anharmonicity into the otherwise harmonic oscillation of the success probability obtained by a conventional quantum walk. As such, the GQW can be understood as a continuous quantum walk that is actively pushed in the direction of the solution state, hence iteratively concentrating probability in states of low energy. However, regarding 2-SAT problems, it was demonstrated that the probability flow can be restricted by internal barriers caused by insufficient energy gradients between neighbouring states, due to the high degeneracy of the energy levels. To conquer this problem, an adjusted heuristic model, termed HGQW-A, was developed. It proposes a mixing operation that alternates the state amplitudes between the regular and the inverse graph layout, in order to introduce artificial phase gradients and thus lift the degeneracy. In doing so, similar dynamics to the HGQW model were achieved.

The final investigation of the models' performances on sets of exact cover and 2-SAT problems revealed two regions of operation, with a great initial increase of P_{gs} for $p \leq aN$ ($a = 1$ for HGQW and $a = 2$ for HGQW-A) resulting in intermediate success probabilities ($P_{gs} \geq 0.5$) after already a few iterations (i.e. $p = aN/2$). For $p > aN$, where the algorithms are no longer limited by the accessibility of the basis states, the success probability was found to saturate between 80% and 94%. Important to mention here is that the HGQW-A model requires approximately twice as many iterations as the HGQW model to reach similar success probabilities, since the quantum walker can only be guided within the regular graph. Comparing the performance of the two models to the AQA and the QAOA, the experiments showed that the GQW delivers superior success probabilities in the region of low to intermediate p. Here, the GQW seems to be most efficient for $p \in [aN, 2aN]$, hence requiring computational resources that scale only linearly in N. Moreover, with $P_{gs}(N) \propto 2^{-0.13N}$ ($P_{gs}(N) \propto 2^{-0.08N}$) in

the first region and $P_{gs}(N) \propto 2^{-0.06N}$ ($P_{gs}(N) \propto 2^{-0.01N}$) in the second region on exact cover (2-SAT) problems, significantly better scaling behaviours of the success probability in the number of qubits are found. This observation suggests that instances of the GQW are able to outperform the QAOA and the AQA also on real world optimization problems featuring numerous qubits. In addition to this, a hybrid approach of combining the GQW with the QAOA achieved $P_{gs} \geq 95\%$ at $p = 2aN$, suggesting that the GQW can also be used as a pre optimization stage within other optimization algorithms in order to simplify the process of parameter tuning.

The two heuristic model studied in this thesis can be understood as a first demonstration of the guided quantum walk as a competitive strategy to other common quantum optimization algorithms, delivering high success probabilities in the region of intermediate computational resources. Especially its property of an optimization phase of constant complexity, a steep increase of the success probability for intermediate circuit depths and an accumulation of probability at low energies in the graph make it a promising candidate for near term NISQ devices. In order to develop the GQW into a general optimization algorithm, addition research has to focus on the following aspects:

- The mixing operation was introduced in the HGQW-A model to partially lift the degeneracy of the energy levels by transforming the system between the regular and the inverse graph layout. It is left to find a heuristic sequence for applying these operations, for example governed by a set of optimization parameters. This would allow combining the HGQW model and the HGQW-A model into a single strategy, as well as reliably solving other types of combinatorial optimization problems.

- The strong fluctuations of the spread coefficients in the inverse graph within the first region of operation for the HGQW-A model indicate that the sampling function of the phase coefficients in the inverse graph is not optimal. Hence, improvements to this function (potentially including additional optimization parameters) could be investigated.

- The distribution of the spread coefficients for both the HGQW model and the HGQW-A model showed no significant dependence on the problem size or the energy spectrum. This suggests that the corresponding optimization parameter could be replaced by a predetermined distribution.

- With respect to the hybrid GQW ansatz, the guided quantum walk could be reformulated as an iterative process, where every iteration increases the complexity (number of optimization parameters) of the optimization process. In doing so, it could be possible to reach almost unit success probability within reasonable circuit depths.

Bibliography

[1] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6-7 (1982), pp. 467–488. ISSN: 1572-9575. DOI: 10.1007/BF02650179. URL: https://link.springer.com/article/10.1007/BF02650179.

[2] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (2019), pp. 505–510. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1666-5. URL: https://www.nature.com/articles/s41586-019-1666-5.

[3] Davide Castelvecchi. "Quantum computers ready to leap out of the lab in 2017". In: *Nature* 541.7635 (2017), pp. 9–10. ISSN: 1476-4687. DOI: 10.1038/541009a. URL: https://www.nature.com/articles/541009a.

[4] Nathalie P. de Leon et al. "Materials challenges and opportunities for quantum computing hardware". In: *Science* 372.6539 (2021). ISSN: 1095-9203. DOI: 10.1126/science.abb2823.

[5] S. Marsh and J. B. Wang. "A quantum walk-assisted approximate algorithm for bounded NP optimisation problems". In: *Quantum Information Processing* 18.3 (2019). ISSN: 1573-1332. DOI: 10.1007/s11128-019-2171-3. URL: https://arxiv.org/pdf/1804.08227.

[6] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. URL: https://arxiv.org/pdf/quant-ph/9605043.

[7] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Review* 41.2 (1999), pp. 303–332. ISSN: 0036-1445. DOI: 10.1137/S0036144598347011.

[8] Carlos D. Gonzalez Calaza, Dennis Willsch, and Kristel Michielsen. "Garden optimization problems for benchmarking quantum annealers". In: *Quantum Information Processing* 20.9 (2021). ISSN: 1573-1332. DOI: 10.1007/s11128-021-03226-6. URL: https://arxiv.org/pdf/2101.10827.

[9] Pontus Vikstål et al. "Applying the Quantum Approximate Optimization Algorithm to the Tail-Assignment Problem". In: *Physical Review Applied* 14.3 (2020). ISSN: 2331-7019. DOI: 10.1103/PhysRevApplied.14.034009. URL: https://arxiv.org/pdf/1912.10499.

[10] Yudong Cao et al. *Quantum Chemistry in the Age of Quantum Computing*. 2019. DOI: 10.1021/acs.chemrev.8b00803. URL: https://arxiv.org/pdf/1812.09976.

[11] Sam McArdle et al. *Quantum computational chemistry*. 2020. DOI: 10.1103/RevModPhys.92.015003. URL: https://arxiv.org/pdf/1808.10402.

[12] Edward Farhi and Hartmut Neven. "Classification with Quantum Neural Networks on Near Term Processors". In: *MIT-CTP* (2018). URL: https://arxiv.org/pdf/1802.06002.

[13] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. *Quantum algorithms for supervised and unsupervised machine learning*. 2013. URL: https://arxiv.org/pdf/1307.0411.

[14] D. Willsch et al. "Support vector machines on the D-Wave quantum annealer". In: *Computer Physics Communications* 248 (2020), p. 107006. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2019.107006. URL: https://arxiv.org/pdf/1906.06283.

[15] S. Pirandola et al. "Advances in quantum cryptography". In: *Advances in Optics and Photonics* 12.4 (2020), p. 1012. DOI: 10.1364/AOP.361502. URL: https://arxiv.org/pdf/1906.01645.

[16] Simon J. Devitt, William J. Munro, and Kae Nemoto. "Quantum error correction for beginners". In: *Reports on Progress in Physics* 76.7 (2013), p. 076001. ISSN: 1361-6633. DOI: 10.1088/0034-4885/76/7/076001. URL: https://arxiv.org/pdf/0905.2794.

[17] Suguru Endo et al. "Hybrid Quantum-Classical Algorithms and Quantum Error Mitigation". In: *Journal of the Physical Society of Japan* 90.3 (2021), p. 032001. ISSN: 1347-4073. DOI: 10.7566/JPSJ.90.032001. URL: https://arxiv.org/pdf/2011.01382.

[18] Dennis Willsch et al. "Hybrid Quantum Classical Simulations". In: *NIC Symposium 2022* (2022). URL: https://arxiv.org/pdf/2210.02811.

[19] Sergey Bravyi, Graeme Smith, and John A. Smolin. "Trading Classical and Quantum Computational Resources". In: *Physical Review X* 6.2 (2016). ISSN: 2160-3308. DOI: 10.1103/PhysRevX.6.021043. URL: https://arxiv.org/pdf/1506.01396.

[20] Nikolaj Moll et al. "Quantum optimization using variational algorithms on near-term quantum devices". In: *Quantum Science and Technology* 3.3 (2018), p. 030503. ISSN: 2058-9565. DOI: 10.1088/2058-9565/aab822. URL: https://arxiv.org/pdf/1710.01022.

[21] Benjamin Villalonga et al. "Establishing the quantum supremacy frontier with a 281 Pflop/s simulation". In: *Quantum Science and Technology* 5.3 (2020), p. 034003. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab7eeb. URL: https://arxiv.org/pdf/1905.00444.

[22] Sergio Boixo et al. "Characterizing quantum supremacy in near-term devices". In: *Nature Physics* 14.6 (2018), pp. 595–600. ISSN: 1745-2473. DOI: 10.1038/s41567-018-0124-x. URL: https://arxiv.org/pdf/1608.00263.

[23] Eladio Gutiérrez et al. "Quantum computer simulation using the CUDA programming model". In: *Computer Physics Communications* 181.2 (2010), pp. 283–300. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2009.09.021.

[24] Thomas Häner and Damian S. Steiger. "0.5 petabyte simulation of a 45-qubit quantum circuit". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC* (2017), pp. 1–10. DOI: 10.1145/3126908.3126947. URL: https://arxiv.org/pdf/1704.01127.

[25] Tyson Jones et al. "QuEST and High Performance Simulation of Quantum Computers". In: *Scientific Reports* 9.1 (2019), p. 10736. ISSN: 2045-2322. DOI: 10.1038/s41598-019-47174-9. URL: https://arxiv.org/pdf/1802.08032.

[26] Gang Yu. *Industrial Applications of Combinatorial Optimization*. Springer New York, NY, 1998. DOI: 10.1007/978-1-4757-2876-7.

[27] Dennis Willsch et al. "Benchmarking Supercomputers with the Jülich Universal Quantum Computer Simulator". In: *NIC Symposium 2020* (2020). URL: https://arxiv.org/pdf/1912.03243.

[28] Dennis Willsch et al. "Benchmarking Advantage and D-Wave 2000Q quantum annealers with exact cover problems". In: *Quantum Information Processing* 21.4 (2022). ISSN: 1573-1332. DOI: 10.1007/s11128-022-03476-y. URL: https://arxiv.org/pdf/2105.02208.

[29] Supercomputing Support. "JUWELS: Modular Tier-0/1 Supercomputer at Jülich Supercomputing Centre". In: *Journal of large-scale research facilities JLSRF* 5 (2019). ISSN: 2364-091X. DOI: 10.17815/jlsrf-5-171. URL: https://apps.fz-juelich.de/jsc/hps/juwels/index.html.

[30] Dennis Willsch et al. "GPU-accelerated simulations of quantum annealing and the quantum approximate optimization algorithm". In: *Computer Physics Communications* 278 (2022), p. 108411. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2022.108411. URL: https://arxiv.org/pdf/2104.03293.

[31] Michael A. Nielsen, Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[32] D. Deutsch, A. Barenco, and A. Ekert. *Universality in quantum computation*. 1995. DOI: 10.1098/rspa.1995.0065. URL: https://arxiv.org/pdf/quant-ph/9505018.

[33] DiVincenzo. "Two-bit gates are universal for quantum computation". In: *Physical review. A, Atomic, molecular, and optical physics* 51.2 (1995), pp. 1015–1022. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.51.1015. URL: https://arxiv.org/pdf/cond-mat/9407022.

[34] Christopher M. Dawson and Michael A. Nielsen. *The Solovay-Kitaev algorithm*. 2005. URL: https://arxiv.org/pdf/quant-ph/0505030.

[35] JUNIQ. 2022. URL: https://juniq.fz-juelich.de/.

[36] IBM Quantum. 2021. URL: https://quantum-computing.ibm.com/.

[37] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://arxiv.org/pdf/1801.00862.

[38] Hans de Raedt et al. *Massively parallel quantum computer simulator, eleven years later*. 2019. DOI: 10.1016/j.cpc.2018.11.005. URL: https://arxiv.org/pdf/1805.04708.

[39] Aneeqa Fatima and Igor L. Markov. "Faster Schrödinger-style simulation of quantum circuits". In: *HPCA* (2020). URL: https://arxiv.org/pdf/2008.00216.

[40] Sergio Boixo et al. *Simulation of low-depth quantum circuits as complex undirected graphical models*. 2017. URL: https://arxiv.org/pdf/1712.05384.

[41] Danylo Lykov et al. *Tensor Network Quantum Simulator With Step-Dependent Parallelization*. 2020. URL: https://arxiv.org/pdf/2012.02430.

[42] Edwin Pednault et al. *Leveraging Secondary Storage to Simulate Deep 54-qubit Sycamore Circuits*. 2019. URL: https://arxiv.org/pdf/1910.09534.

[43] Ya-Qian Zhao et al. "Simulation of quantum computing on classical supercomputers with tensor-network edge cutting". In: *Physical Review A* 104.3 (2021). ISSN: 2469-9934. DOI: 10.1103/PhysRevA.104.032603. URL: https://arxiv.org/pdf/2010.14962.

[44] John Brennan et al. *Tensor Network Circuit Simulation at Exascale*. 2021. URL: https://arxiv.org/pdf/2110.09894.

[45] Edwin Pednault et al. *Pareto-Efficient Quantum Circuit Simulation Using Tensor Contraction Deferral*. 2017. URL: https://arxiv.org/pdf/1710.05867.

[46] Xiao Yuan et al. "Quantum Simulation with Hybrid Tensor Networks". In: *Physical Review Letters* 127.4 (2021), p. 040501. ISSN: 1079-7114. DOI: 10.1103/PhysRevLett.127.040501. URL: https://arxiv.org/pdf/2007.00958.

[47] Alwin Zulehner and Robert Wille. *Advanced Simulation of Quantum Computations*. 2017. URL: https://arxiv.org/pdf/1707.00865.

[48] Zhimin Wang et al. "A quantum circuit simulator and its applications on Sunway TaihuLight supercomputer". In: *Scientific Reports* 11.1 (2021), p. 355. ISSN: 2045-2322. DOI: 10.1038/s41598-020-79777-y. URL: https://arxiv.org/pdf/2008.07140.

[49] Zhao-Yun Chen et al. "64-qubit quantum circuit simulation". In: *Science Bulletin* 63.15 (2018), pp. 964–971. ISSN: 2095-9273. DOI: 10.1016/j.scib.2018.06.007. URL: https://www.sciencedirect.com/science/article/pii/S2095927318302809.

[50] Cupjin Huang et al. "Efficient parallelization of tensor network contraction for simulating quantum computation". In: *Nature Computational Science* 1.9 (2021), pp. 578–587. ISSN: 2662-8457. DOI: 10.1038/s43588-021-00119-7. URL: https://www.nature.com/articles/s43588-021-00119-7.

[51] Li, Ang and Fang, Bo and Granade, Christopher and Prawiroatmodjo, Guen and Heim, Bettina and Roetteler, Martin and Krishnamoorthy, Sriram. "SV-Sim: Scalable PGAS-Based State Vector Simulation of Quantum Circuits". In: (2021). DOI: 10.1145/3458817.3476169.

[52] Jun Doi et al. "Quantum computing simulator on a heterogenous HPC system". In: ACM, 2019. DOI: 10.1145/3310273.3323053.

[53] Message Passing Interface. 2022. URL: https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf.

[54] NVIDIA. "nvidia-ampere-architecture-whitepaper". In: (). URL: https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf (visited on 10/04/2022).

[55] NVIDIA. *Compute Unified Device Architecture (CUDA)*. URL: https://docs.nvidia.com/cuda/.

[56] Santiago I. Betelu. *The limits of quantum circuit simulation with low precision arithmetic*. 2020. URL: https://arxiv.org/pdf/2005.13392.

[57] Xin-Chuan Wu et al. *Full-state quantum circuit simulation by using data compression*. 2019. DOI: 10.1145/3295500.3356155. URL: https://arxiv.org/pdf/1911.04034.

[58] K. de Raedt et al. *Massively parallel quantum computer simulator*. 2007. DOI: 10.1016/j.cpc.2006.08.007. URL: https://arxiv.org/pdf/quant-ph/0608239.

[59] NVIDIA. *NVIDIA Collective Communications Library (NCCL)*. URL: https://docs.nvidia.com/deeplearning/nccl/.

[60] NVIDIA. *NVSHMEM*. URL: https://docs.nvidia.com/nvshmem/index.html.

[61] Samuel Williams, Andrew Waterman, and David Patterson. "Roofline". In: *Communications of the ACM* 52.4 (2009), pp. 65–76. ISSN: 0001-0782. DOI: 10.1145/1498765.1498785.

[62] A. Amariutei and S. Caraiman. "Parallel quantum computer simulation on the GPU". In: *15th International Conference on System Theory, Control and Computing*. 2011, pp. 1–6.

[63] Madita Willsch et al. "Benchmarking the quantum approximate optimization algorithm". In: *Quantum Information Processing* 19.7 (2020). ISSN: 1573-1332. DOI: 10.1007/s11128-020-02692-8. URL: https://arxiv.org/pdf/1907.02359.

[64] B. Korte and J. Vygen. "Combinatorial Optimization: Theory and Algorithms". In: *undefined* (2007). URL: https://www.semanticscholar.org/paper/Combinatorial-Optimization%3A-Theory-and-Algorithms-Korte-Vygen/bd8563e97ca1278db2bda9cce53abadc912f7a18.

[65] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A Quantum Approximate Optimization Algorithm". In: *MIT-CTP* (2014). URL: https://arxiv.org/pdf/1411.4028.

[66] D-Wave Systems Inc. "The D-Wave Advantage System: An Overview". In: (). URL: https://www.dwavesys.com/media/s3qbjp3s/14-1049a-a_the_d-wave_advantage_system_an_overview.pdf (visited on 10/04/2022).

[67] Leo Zhou et al. "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices". In: *Physical Review X* 10.2 (2020). ISSN: 2160-3308. DOI: 10.1103/PhysRevX.10.021067. URL: https://arxiv.org/pdf/1812.01041.

[68] Andrew Lucas. "Ising formulations of many NP problems". In: *Frontiers in Physics* 2 (2014). ISSN: 2296-424X. DOI: 10.3389/fphy.2014.00005. URL: https://arxiv.org/pdf/1302.5843.

[69] Andreas Bengtsson et al. "Improved Success Probability with Greater Circuit Depth for the Quantum Approximate Optimization Algorithm". In: *Physical Review Applied* 14.3 (2020). ISSN: 2331-7019. DOI: 10.1103/PhysRevApplied.14.034010.

[70] Yu-Qin Chen et al. *Optimizing Quantum Annealing Schedules with Monte Carlo Tree Search enhanced with neural networks*. 2020. URL: https://arxiv.org/pdf/2004.02836.

[71] Ting-Jui Hsu et al. *Quantum annealing with anneal path control: application to 2-SAT problems with known energy landscapes*. 2018. URL: https://arxiv.org/pdf/1810.00194.

[72] Vrinda Mehta et al. *Quantum Annealing with Trigger Hamiltonians: Application to 2-SAT and Nonstoquastic Problems*. 2021. DOI: 10.1103/PhysRevA.104.032421. URL: https://arxiv.org/pdf/2106.04864.

[73] Vrinda Mehta et al. "Quantum annealing for hard 2-satisfiability problems: Distribution and scaling of minimum energy gap and success probability". In: *Physical Review A* 105.6 (2022). ISSN: 2469-9934. DOI: 10.1103/PhysRevA.105.062406. URL: https://arxiv.org/pdf/2202.00118.

[74] Tameem Albash and Daniel A. Lidar. "Adiabatic quantum computation". In: *Reviews of Modern Physics* 90.1 (2018). ISSN: 1539-0756. DOI: 10.1103/RevModPhys.90.015002.

[75] Philipp Hauke et al. *Perspectives of quantum annealing: methods and implementations*. 2020. DOI: 10.1088/1361-6633/ab85b8. URL: https://arxiv.org/pdf/1903.06559.

[76] E. Farhi et al. "A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem". In: *Science* 292.5516 (2001), pp. 472–475. ISSN: 1095-9203. DOI: 10.1126/science.1057726. URL: https://arxiv.org/pdf/quant-ph/0104129.

[77] A. B. Finnila et al. "Quantum annealing: A new method for minimizing multidimensional functions". In: *Chemical Physics Letters* 219.5-6 (1994), pp. 343–348. ISSN: 0009-2614. DOI: 10.1016/0009-2614(94)00117-0. URL: https://arxiv.org/pdf/chem-ph/9404003.

[78] M. Born and V. Fock. "Beweis des Adiabatensatzes". In: *Zeitschrift fr Physik* 51.3-4 (1928), pp. 165–180. ISSN: 1434-601X. DOI: 10.1007/BF01343193.

[79] M. H. S. Amin. "Consistency of the adiabatic theorem". In: *Physical Review Letters* 102.22 (2009), p. 220401. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.102.220401. URL: https://arxiv.org/pdf/0810.4335.

[80] D-Wave Systems Inc. *Anneal Schedules*. URL: https://docs.dwavesys.com/docs/latest/doc_physical_properties.html.

[81] Adam Callison et al. "Energetic Perspective on Rapid Quenches in Quantum Annealing". In: *PRX Quantum* 2.1 (2021). ISSN: 2691-3399. DOI: 10.1103/PRXQuantum.2.010338. URL: https://arxiv.org/pdf/2007.11599.

[82] Lishan Zeng, Jun Zhang, and Mohan Sarovar. "Schedule path optimization for adiabatic quantum computing and optimization". In: *Journal of Physics A: Mathematical and Theoretical* 49.16 (2016), p. 165305. ISSN: 1751-8121. DOI: 10.1088/1751-8113/49/16/165305. URL: https://arxiv.org/pdf/1505.00209.

[83] Lucas T. Brady et al. "Optimal Protocols in Quantum Annealing and Quantum Approximate Optimization Algorithm Problems". In: *Physical Review Letters* 126.7 (2021), p. 070505. ISSN: 1079-7114. DOI: 10.1103/PhysRevLett.126.070505. URL: https://arxiv.org/pdf/2003.08952.

[84] E. J. Crosson and D. A. Lidar. "Prospects for quantum enhancement with diabatic quantum annealing". In: *Nature Reviews Physics* 3.7 (2021), pp. 466–489. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00313-6. URL: https://arxiv.org/pdf/2008.09913.

[85] Siddharth Muthukrishnan, Tameem Albash, and Daniel A. Lidar. "Tunneling and Speedup in Quantum Optimization for Permutation-Symmetric Problems". In: *Physical Review X* 6.3 (2016). ISSN: 2160-3308. DOI: 10.1103/PhysRevX.6.031010. URL: https://arxiv.org/pdf/1511.03910.

[86] Itay Hen and A. P. Young. "Exponential complexity of the quantum adiabatic algorithm for certain satisfiability problems". In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 84.6 Pt 1 (2011), p. 061152. ISSN: 1539-3755. DOI: 10.1103/PhysRevE.84.061152. URL: https://arxiv.org/pdf/1109.6872.

[87] Tameem Albash and Daniel A. Lidar. "Demonstration of a Scaling Advantage for a Quantum Annealer over Simulated Annealing". In: *Physical Review X* 8.3 (2018). ISSN: 2160-3308. DOI: 10.1103/PhysRevX.8.031016. URL: https://arxiv.org/pdf/1705.07452.

[88] Michael Streif and Martin Leib. *Comparison of QAOA with Quantum and Simulated Annealing*. 2019. URL: https://arxiv.org/pdf/1901.01903.

[89] Edward Farhi and Aram W. Harrow. "Quantum Supremacy through the Quantum Approximate Optimization Algorithm". In: *MIT/CTP-* (2019). URL: https://arxiv.org/pdf/1602.07674.

[90] Andreas Bartschi and Stephan Eidenbenz. "Grover Mixers for QAOA: Shifting Complexity from Mixer Design to State Preparation". In: *LA-UR-20-* (2020), pp. 72–82. DOI: 10.1109/QCE49297.2020.00020. URL: https://arxiv.org/pdf/2006.00354.

[91] Stuart Hadfield et al. "From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz". In: *Algorithms* 12.2 (2019), p. 34. ISSN: 1999-4893. DOI: 10.3390/a12020034. URL: https://arxiv.org/pdf/1709.03489.

[92] V. Akshay et al. "Parameter concentrations in quantum approximate optimization". In: *Physical Review A* 104.1 (2021). ISSN: 2469-9934. DOI: 10.1103/PhysRevA.104.L010401. URL: https://arxiv.org/pdf/2103.11976.

[93] Brandao, Fernando G. S. L. et al. *For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances*. 2018. URL: https://arxiv.org/pdf/1812.04170.

[94] Sami Khairy et al. "Learning to Optimize Variational Quantum Circuits to Solve Combinatorial Problems". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.03 (2020), pp. 2367–2375. ISSN: 2159-5399. DOI: 10.1609/aaai.v34i03.5616. URL: https://arxiv.org/pdf/1911.11071.

[95] Michael Streif and Martin Leib. *Training the Quantum Approximate Optimization Algorithm without access to a Quantum Processing Unit*. URL: https://arxiv.org/pdf/1908.08862.

[96] Ruslan Shaydulin, Ilya Safro, and Jeffrey Larson. "Multistart Methods for Quantum Approximate optimization". In: *2019 IEEE High Performance Extreme Computing Conference (HPEC)* (2019), pp. 1–8. DOI: 10.1109/HPEC.2019.8916288. URL: https://arxiv.org/pdf/1905.08768.

[97] Hans de Raedt and Bart de Raedt. "Applications of the generalized Trotter formula". In: *Physical Review A* 28.6 (1983), pp. 3575–3580. ISSN: 2469-9926. DOI: 10.1103/PhysRevA.28.3575.

[98] Dorit Aharonov et al. "Quantum Walks On Graphs". In: *Proceedings of ACM Symposium on Theory of Computation (STOC'01), July* (2002). URL: https://arxiv.org/pdf/quant-ph/0012090.

[99] Salvador Elías Venegas-Andraca. "Quantum walks: a comprehensive review". In: *Quantum Information Processing* 11.5 (2012), pp. 1015–1106. ISSN: 1573-1332. DOI: 10.1007/s11128-012-0432-5. URL: https://arxiv.org/pdf/1201.4780.

[100] Edward Farhi and Sam Gutmann. "Quantum computation and decision trees". In: *Physical review. A, Atomic, molecular, and optical physics* 58.2 (1998), pp. 915–928. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.58.915. URL: https://arxiv.org/pdf/quant-ph/9706062.

[101] Bruno Chagas and Renato Portugal. "Discrete-Time Quantum Walks on Oriented Graphs". In: *Electronic Proceedings in Theoretical Computer Science* 315 (2020), pp. 26–37. DOI: 10.4204/EPTCS.315.3. URL: https://arxiv.org/pdf/2001.04814.

[102] Mahesh N. Jayakody, Chandrakala Meena, and Priodyuti Pradhan. *One-dimensional discrete-time quantum walks with general coin*. 2021. URL: https://arxiv.org/pdf/2102.07207.

[103] Renato Portugal, Stefan Boettcher, and Stefan Falkner. "One-dimensional coinless quantum walks". In: *Physical review. A, Atomic, molecular, and optical physics* 91.5 (2015). ISSN: 1050-2947. DOI: 10.1103/PhysRevA.91.052319. URL: https://arxiv.org/pdf/1408.5166.

[104] Adam Callison et al. "Finding spin glass ground states using quantum walks". In: *New Journal of Physics* 21.12 (2019), p. 123022. DOI: 10.1088/1367-2630/ab5ca2. URL: https://arxiv.org/pdf/1903.05003.

[105] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing". In: *Science (New York, N.Y.)* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671.

[106] Stefan H. Sack and Maksym Serbyn. "Quantum annealing initialization of the quantum approximate optimization algorithm". In: *Quantum* 5 (2021), p. 491. ISSN: 2521-327X. DOI: 10.22331/q-2021-07-01-491. URL: https://arxiv.org/pdf/2101.05742.

[107] Charles Moussa et al. *Unsupervised strategies for identifying optimal parameters in Quantum Approximate Optimization Algorithm*. 2022. DOI: 10.1140/epjqt/s40507-022-00131-4. URL: https://arxiv.org/pdf/2202.09408.

[108] Matteo M. Wauters et al. "Reinforcement-learning-assisted quantum optimization". In: *Physical Review Research* 2.3 (2020). DOI: 10.1103/PhysRevResearch.2.033446. URL: https://arxiv.org/pdf/2004.12323.

[109] Ruslan Shaydulin et al. *Parameter Transfer for Quantum Approximate Optimization of Weighted MaxCut*. 2022. URL: https://arxiv.org/pdf/2201.11785.

[110] J. A. Nelder and R. Mead. "A Simplex Method for Function Minimization". In: *The Computer Journal* 7.4 (1965), pp. 308–313. ISSN: 1460-2067. DOI: 10.1093/comjnl/7.4.308.

[111] Mario Fernández-Pendás et al. "A study of the performance of classical minimizers in the Quantum Approximate Optimization Algorithm". In: *Journal of Computational and Applied Mathematics* 404 (2022), p. 113388. ISSN: 0377-0427. DOI: 10.1016/j.cam.2021.113388. URL: https://www.sciencedirect.com/science/article/pii/S0377042721000078.

[112] Thomas G. Draper. *Addition on a Quantum Computer*. 2000. URL: https://arxiv.org/pdf/quant-ph/0008033.

[113] Chandra, Rohit and Dagum, Leo and Kohr, David and Menon, Ramesh and Maydan, Dror and McDonald, Jeff. *Parallel programming in OpenMP*. 2001. URL: http://www.openmp.org/resources/openmp-compilers/.

Appendix

A. Implemented gate operations

The following list gives an overview of the quantum gates implemented by SEQCS. Note that $\{H,\, T = R_z(\pi/4),\, CNOT\}$ represents a universal gate set, allowing to approximate any quantum circuit to arbitrary precision [32, 33].

Single-qubit diagonal gates:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad S^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & (1+i)/\sqrt{2} \end{pmatrix} \qquad T^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & (1-i)/\sqrt{2} \end{pmatrix} \qquad R_z(\gamma) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\gamma} \end{pmatrix}$$

$$R(k) = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} \qquad R^\dagger(k) = \begin{pmatrix} 1 & 0 \\ 0 & e^{-2\pi i/2^k} \end{pmatrix}$$

Single-qubit non-diagonal gates:

$$H = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$\pm X = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & \pm i \\ \pm i & 1 \end{pmatrix} \qquad \pm Y = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & \pm 1 \\ \mp 1 & 1 \end{pmatrix}$$

$$R_x(\gamma) = \begin{pmatrix} \cos\left(\frac{\gamma}{2}\right) & -i\sin\left(\frac{\gamma}{2}\right) \\ -i\sin\left(\frac{\gamma}{2}\right) & \cos\left(\frac{\gamma}{2}\right) \end{pmatrix} \qquad R_y(\gamma) = \begin{pmatrix} \cos\left(\frac{\gamma}{2}\right) & -\sin\left(\frac{\gamma}{2}\right) \\ \sin\left(\frac{\gamma}{2}\right) & \cos\left(\frac{\gamma}{2}\right) \end{pmatrix}$$

$$U2(\phi,\lambda) = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i(\phi+\lambda)} \end{pmatrix} \qquad U3(\theta,\phi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

Multi-qubit gates:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

$$U(k) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^k} \end{pmatrix} \qquad U^{\dagger}(k) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-2\pi i/2^k} \end{pmatrix}$$

$$ZZ(\gamma) = \begin{pmatrix} e^{-i\frac{\gamma}{2}} & 0 & 0 & 0 \\ 0 & e^{i\frac{\gamma}{2}} & 0 & 0 \\ 0 & 0 & e^{i\frac{\gamma}{2}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\gamma}{2}} \end{pmatrix} \qquad TOFFOLI = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

B. Quantum circuits

B.1. Quantum approximate optimization algorithm

Figures B.1 and B.2 depict the circuit representation of the Quantum approximate optimization algorithm (QAOA) [65]. A detailed description of the algorithm can be found in section 2.2.2. Figure B.3 presents a typical gate schedule used by SEQCS to evaluate the QAOA.

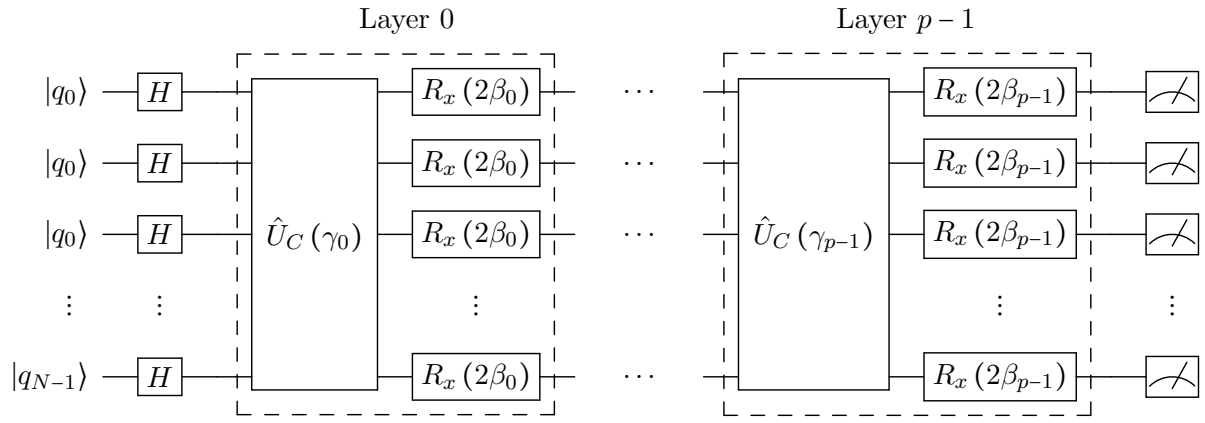


Figure B.1.: The figure depicts the QAOA circuit (see section 2.2.2) applied to an N qubit register. The circuit is composed of p layers (see dashed-boxes), each consisting of a fully diagonal transformation $\hat{U}_C(\gamma_p)$ and a set of $R_x(2\beta_p)$ rotations applied to every qubit. The respective operations are controlled via the sets of variational parameters $\{\beta_{k<p}\}$ and $\{\gamma_{k<p}\}$. $\hat{U}_C$ is determined by the problem Hamiltonian (see Eq. 2.2) in question. Figure B.2 presents a possible circuit realization of $\hat{U}_C$.

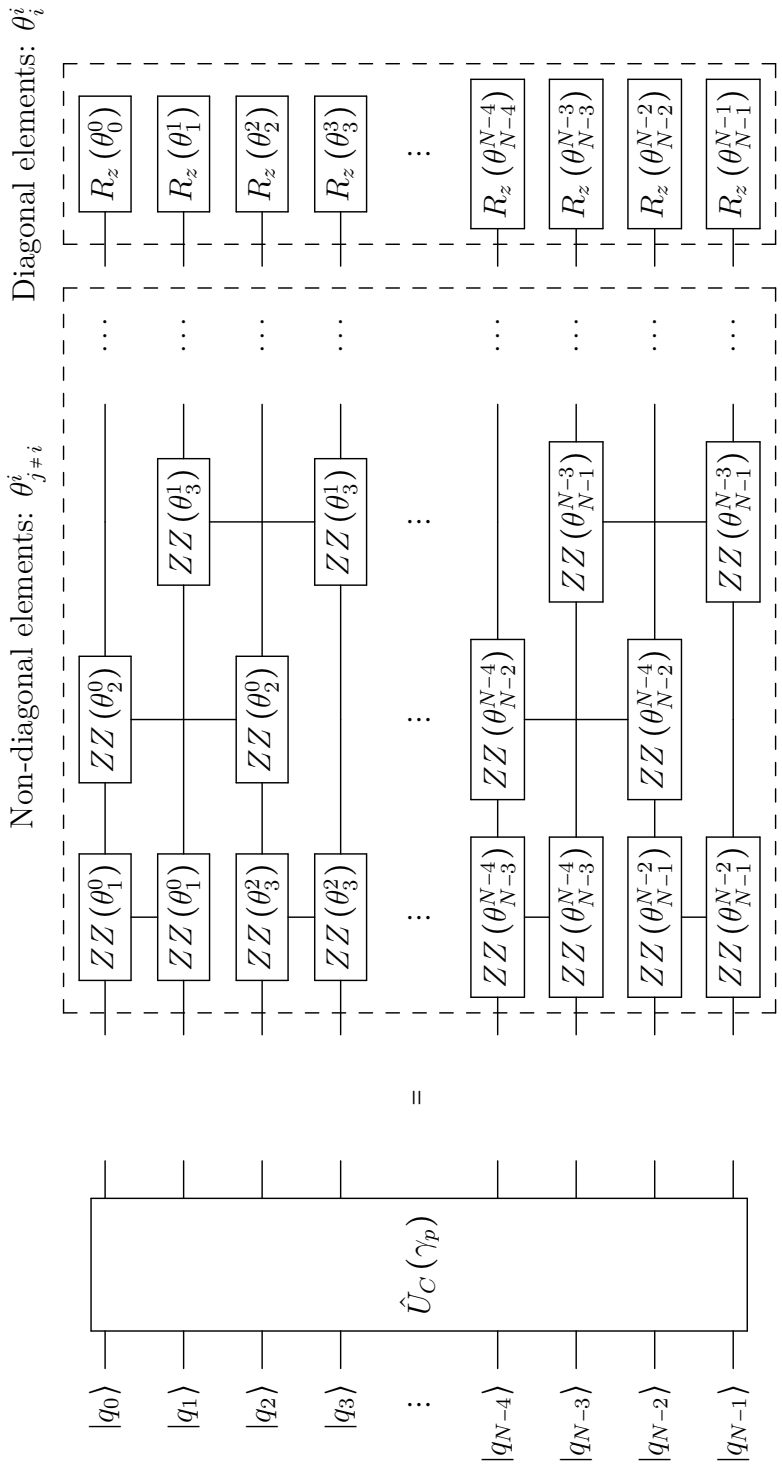


Figure B.2.: The figure depicts a circuit realization of the $\hat{U}_C(\gamma_p)$ transformation on an N qubit register used within one QAOA layer (see section 2.2.2). The rotation angles $\theta_j^i = 0.125 \cdot \gamma_p \cdot Q_{i,j}$ are determined using the variational parameter γ_p of the *pth* QAOA layer and the matrix element $Q_{i,j}$ of the problem's QUBO formulation (see Eq. 2.1). Since the circuit is constructed using only diagonal gates (ZZ and R_z gates), SEQCS is able to evaluate $\hat{U}_C(\gamma_p)$ within a single memory traversal with no MPI communication involved.

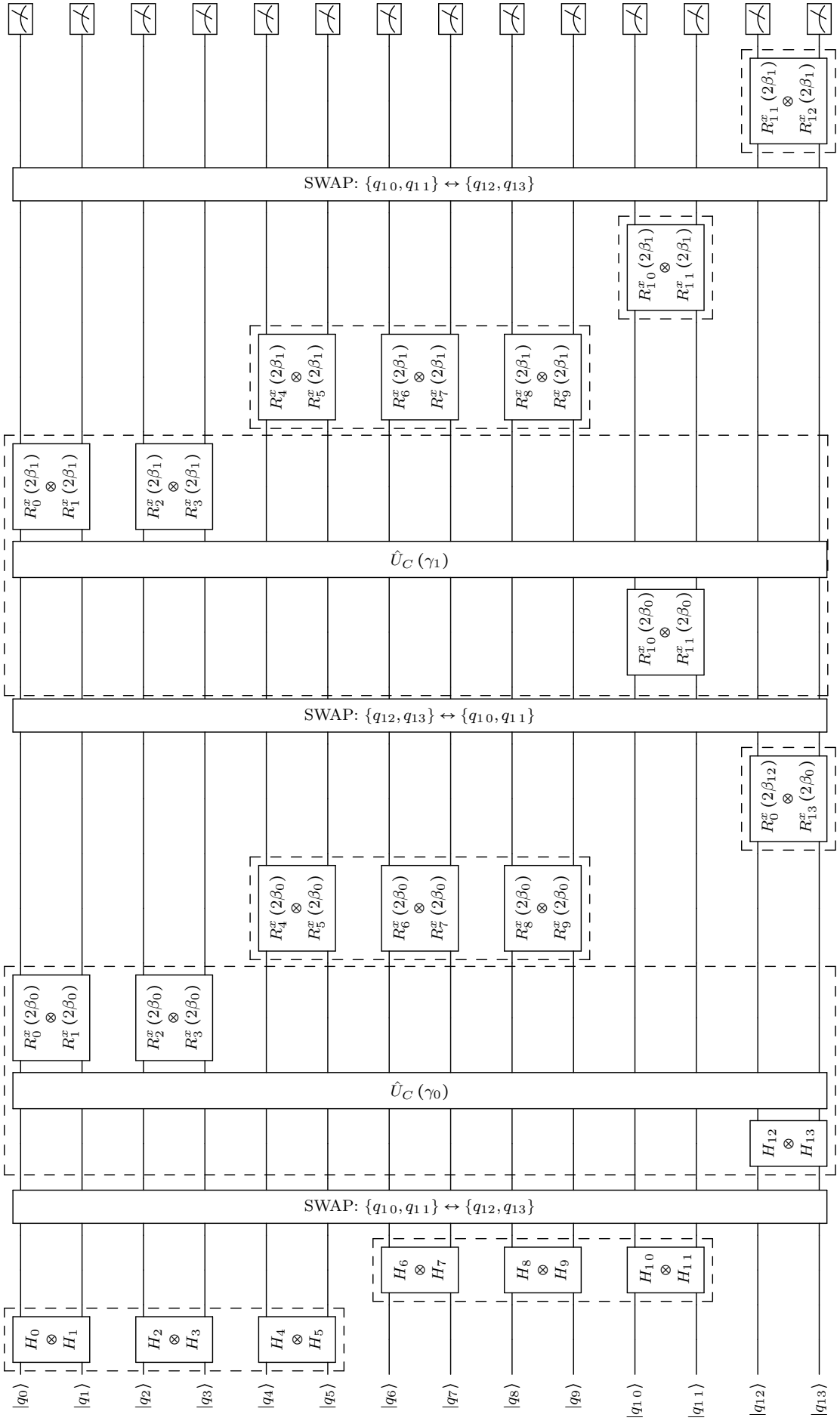

$|q_0\rangle$
$|q_1\rangle$
$|q_2\rangle$
$|q_3\rangle$
$|q_4\rangle$
$|q_5\rangle$
$|q_6\rangle$
$|q_7\rangle$
$|q_8\rangle$
$|q_9\rangle$
$|q_{10}\rangle$
$|q_{11}\rangle$
$|q_{12}\rangle$
$|q_{13}\rangle$
$H_0 \otimes H_1$
$H_2 \otimes H_3$
$H_4 \otimes H_5$
$H_6 \otimes H_7$
$H_8 \otimes H_9$
$H_{10} \otimes H_{11}$
$H_{12} \otimes H_{13}$
SWAP: $\{q_{10}, q_{11}\} \leftrightarrow \{q_{12}, q_{13}\}$
$\hat{U}_C(\gamma_0)$
$R_0^x(2\beta_0) \otimes R_1^x(2\beta_0)$
$R_2^x(2\beta_0) \otimes R_3^x(2\beta_0)$
$R_4^x(2\beta_0) \otimes R_5^x(2\beta_0)$
$R_6^x(2\beta_0) \otimes R_7^x(2\beta_0)$
$R_8^x(2\beta_0) \otimes R_9^x(2\beta_0)$
$R_{10}^x(2\beta_0) \otimes R_{11}^x(2\beta_0)$
$R_0^x(2\beta_{12}) \otimes R_{13}^x(2\beta_0)$
SWAP: $\{q_{12}, q_{13}\} \leftrightarrow \{q_{10}, q_{11}\}$
$\hat{U}_C(\gamma_1)$
$R_0^x(2\beta_1) \otimes R_1^x(2\beta_1)$
$R_2^x(2\beta_1) \otimes R_3^x(2\beta_1)$
$R_4^x(2\beta_1) \otimes R_5^x(2\beta_1)$
$R_6^x(2\beta_1) \otimes R_7^x(2\beta_1)$
$R_8^x(2\beta_1) \otimes R_9^x(2\beta_1)$
$R_{10}^x(2\beta_1) \otimes R_{11}^x(2\beta_1)$
$R_{11}^x(2\beta_1) \otimes R_{12}^x(2\beta_1)$
SWAP: $\{q_{10}, q_{11}\} \leftrightarrow \{q_{12}, q_{13}\}$

Figure B.3.: (Previous page.) The figure illustrates a typical gate-schedule used by SEQCS to simulate an $N = 14$ qubit QAOA circuit with $p = 2$. For illustration purposes, the number of local qubits is set to $N_L = 10$, yielding $N_G = 2$ global qubits (= 4 GPUs), and the gate-cluster size is limited to 6 qubits. Each gate-cluster is visualized by a dashed-box. Moreover, the diagonal transformation $\hat{U}_C(\gamma_p)$ is evaluated using a single memory traversal. Note that SEQCS generally removes the initial set of Hadamard gates, by initializing the system in the equal superposition state $|+\rangle^{\otimes N}$. Consequentially, one combined global-local qubit exchange is required per QAOA layer.

B.2. Quantum Fourier transformation

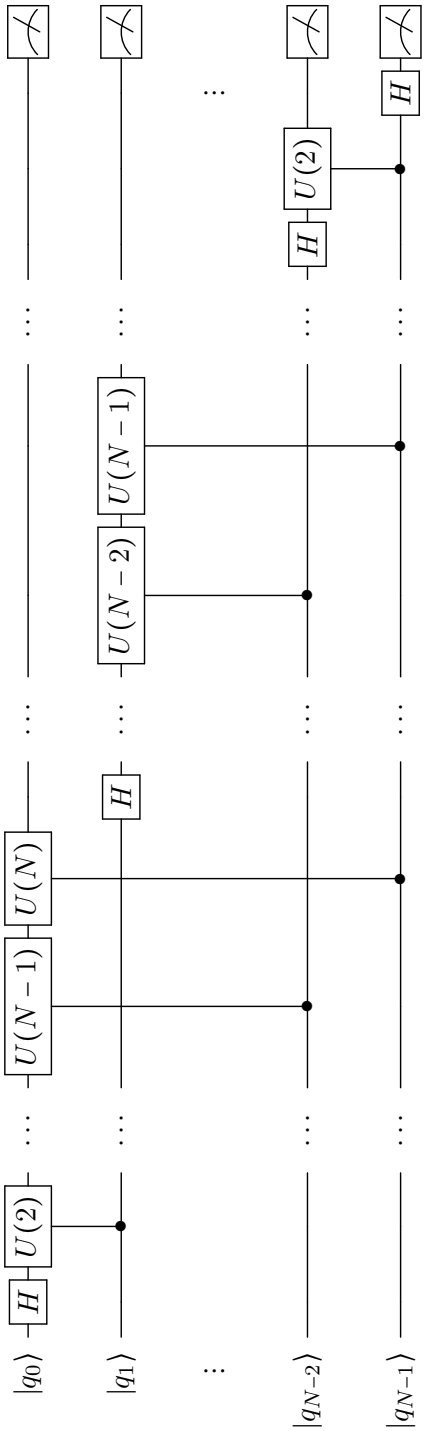


Figure B.4.: The figure depicts the circuit of the quantum Fourier transformation (QFT) [31] applied to an N qubit register. Given some initial state $|J\rangle = |q_0\rangle \otimes \cdots \otimes |q_{N-1}\rangle$ the circuit produces the transformed state $\frac{1}{\sqrt{2^N}} \left[|0\rangle + e^{2^{N-1}\pi i x}|1\rangle\right] \otimes \cdots \otimes \left[|0\rangle + e^{2^0 \pi i x}|1\rangle\right]$. Hence, measuring the qubits in the computational basis afterwards, one obtains the QFT of J with the bits in reversed order. Since the circuit only involves diagonal multi-qubit gates, SEQCS is able to simulate the QFT using $\lceil \frac{N}{N_C} \rceil$ gate-clusters. Moreover, consecutive $U(k)$ gates can be combined into a single memory traversal.

B.3. Quantum adder

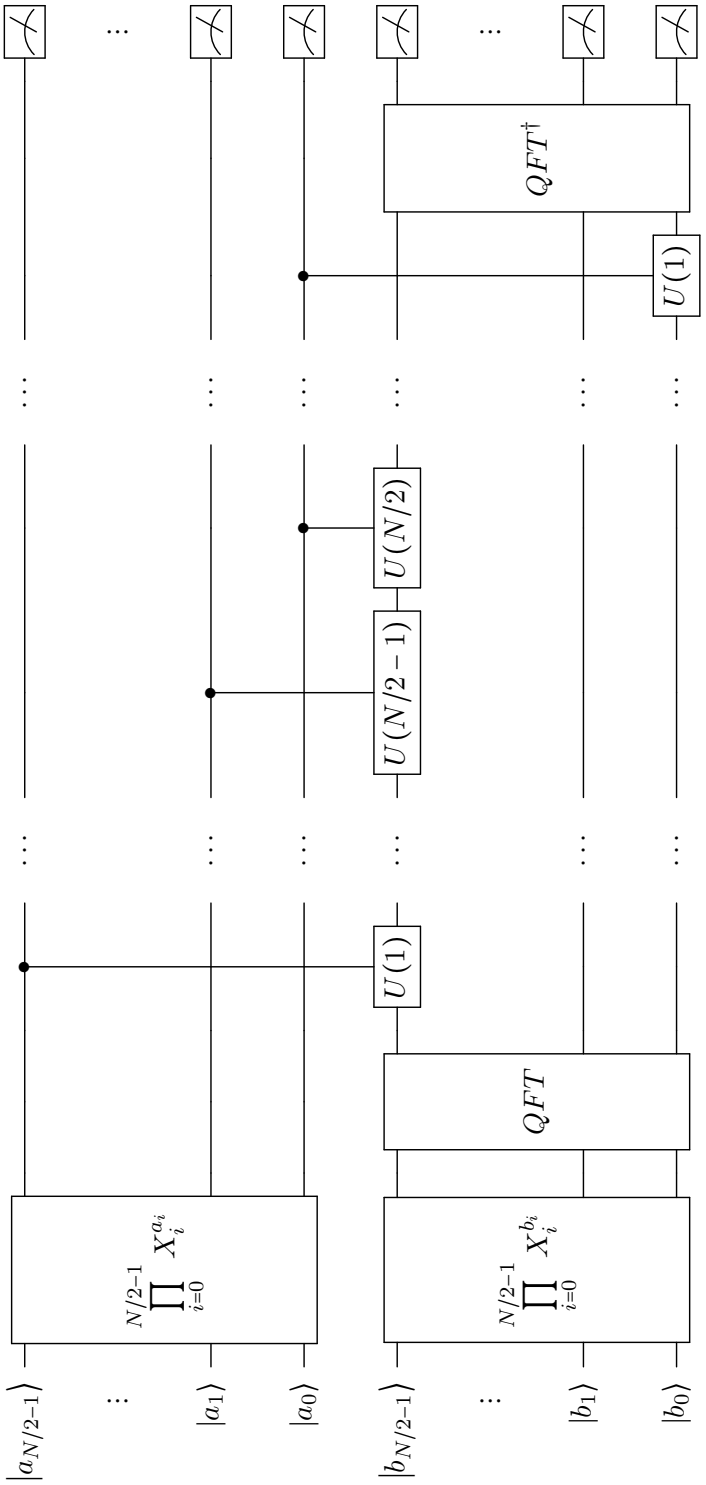


Figure B.5.: The figure depicts the circuit of a quantum adder [112] using the quantum Fourier transformation to calculate the sum of two $N/2$ bit integers $a = \left[a_{N/2-1} \ \ldots \ a_0\right]_2$ and $b = \left[b_{N/2-1} \ \ldots \ b_0\right]_2$. Both registers are initialized in the ground state $|0\rangle^{\otimes N}$ and set to a and b using a sequence of X gates based on the binary representation of the two integers. Note that SEQCS is able to simulate consecutive $U(k)$ gates within a single memory traversal.

C. GPU architecture

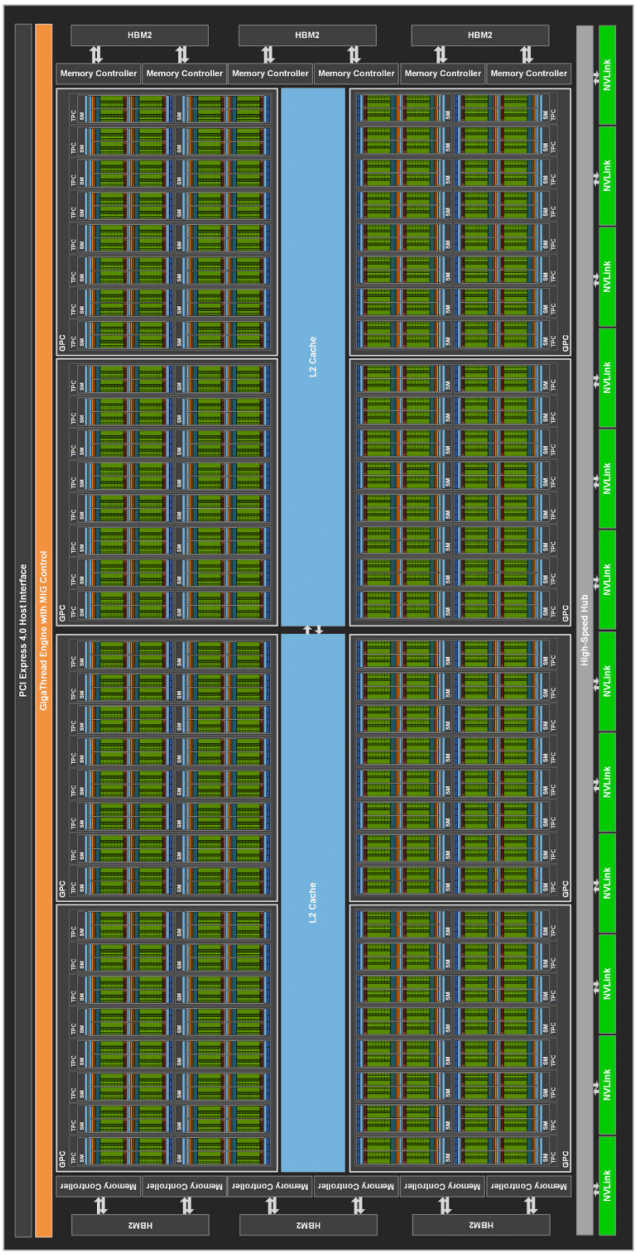


Figure C.6.: The figure depicts the Ampere architecture of a NVIDIA GA100 GPU featuring 128 SMs (the A100 GPU offers the same architecture with 108 SMs) [54].

Figure C.7.: The figure depicts the design of a Streaming Multiprocessor (SM) in the Ampere architecture used by the A100 GPU [54].

D. SEQCS program code

SEQCS is a high-performance GPU-accelerated classical quantum circuit simulator written in C++ using CUDA and CUDA-aware MPI. Its program files can be found at `https://jugit.fz-juelich.de/qip/seqcs/-/tree/main/src`.

E. Algorithms

E.1. Insertion of qubit swaps

The algorithm depicted in E.1 is used during the circuit preprocessing of SEQCS (see Fig. 1.2) to insert combined global-local qubits exchanges into the gate queue. The SWAP operations are optimized, such that multiple consecutive exchanges are combined and the total number of qubit transfers is minimized.

```
for each gate Ĝ_k in the queue {

    if Ĝ_k operates on global qubits of σ_k {

        A_G = [T_G of Ĝ_k];
        A_L = [T_L of Ĝ_k];

        for each gate Ĝ_{j>k} in the queue {

            if size(A_G) > N − size(A_L ∪ T_L)
                break;

            A_G = A_G ∪ T_G;
            A_L = A_L ∪ T_L;

        }

        Insert SWAP_{A_G ↔ A_L^{-1}} at position k in the circuit;
        Update σ_{j>k};

    }

}
```

Algorithm E.1: Insertion of SWAP operations into gate queue. Here, $\hat{G}_k$ denotes a gate at position k in the queue, with T_G and T_L being arrays of its global and local target qubits, and σ_k is the qubit permutation at gate $\hat{G}_k$.

E.2. MPI communication scheme

The algorithm depicted in E.2 presents the MPI communication scheme used to exchange K local and global qubits. It separates the GPU network into 2^{N_G-K} groups of 2^K communicating GPUs each. In doing so, each GPU must exchange 2^{N_L-K} state amplitudes with every GPU in its respective group. These transfers are performed in disjoint pairs, such that each GPU only communicates with one other GPU at a given time, in order to reduce the impact of MPI synchronization barriers. In doing so, each subset of exchanged local state vectors is constructed via the MPI data types: *MPI_Type_continuous*, *MPI_Type_vector* and *MPI_Type_indexed*.

```
Count I_group from 1 to 2^K {

    B_send = [];     //Amplitude buffer send to other GPU
    B_recv = [];     //Amplitude buffer recieved from other GPU

    Count I_Amplitude from 0 to 2^(N_L-K) {

        B_send[I_Amplitude] = Ψ[S(I_Amplitude, I_M^local)];

    }

    Exchange B_send with MPI-rank R ∧ S(I_group, I_M^global) into B_recv;

    Count I_Amplitude from 0 to 2^(N_L-K) {

        Ψ[S(I_Amplitude, I_M^local)] = B_recv[I_Amplitude];

    }

}
```

Algorithm E.2: MPI communication scheme for exchaning K local and global qubits. Here, I_M^{local} and I_M^{global} denote 32 bit integers with 1's at the bit positions of the target local and global qubits, respectively. E.g. exchanging the local qubits 0 and 2 with the global qubits 5 and 7 ($K = 2$) with $N_L = 4$, yields $I_M^{local} = 0101$ and $I_M^{global} = 1010$. Moreover, R is the MPI-rank of the respective GPU, and Ψ refers to the array of 2^{N_L} local state amplitudes. $S(L,\ J)$ denotes a bit-wise shift operation that returns a bitstring with the bit values of L consecutive distributed among the set bit posistions in J. E.g. $S(0011,\ 1101) = 0101$.

E.3. Creation of gate-clusters

The algorithm depicted in E.3 is used during the circuit preprocessing of SEQCS (see Fig. 1.2) to rearrange the gate-queue and form gate-clusters. Each gate-cluster can be simulated using a single kernel call, thus allowing to reuse state amplitudes stored in the shared memory during the evaluation of its gates. Consequentially, the algorithm is designed to maximize the cluster size and hence minimize the number of kernel calls. Moreover, Fig. E.8 illustrates the transfer of the amplitudes between the global memory and shared memory.

```
Clusters = [];  //Final gate-clusters

for each gate Ĝ_k in the queue {

    Qubits  = List of N qubits initialized as unobstructed
    Cluster = [Ĝ_k]    //List of gates forming the new cluster

    for each gate Ĝ_{j>k} in Queue {

        if inserting Ĝ_j into Cluster violates shared-memory
           or constant-memeory restrictions
            for each q in T_j
               Set Qubits[q] as obstructed;

        if for all q in T_j Qubits[q] is not obstructed
            Insert Ĝ_j into Cluster;
        else
            for each q in T_j
               Set Qubits[q] as obstructed;

    }

    Insert Cluster into Clusters;
    Remove Cluster from Queue;

}
```

Algorithm E.3: Creation of gate-clusters. Here, $\hat{G}_k$ denotes a gate at position k in the gate-queue, with T_k being an array of its target qubits.

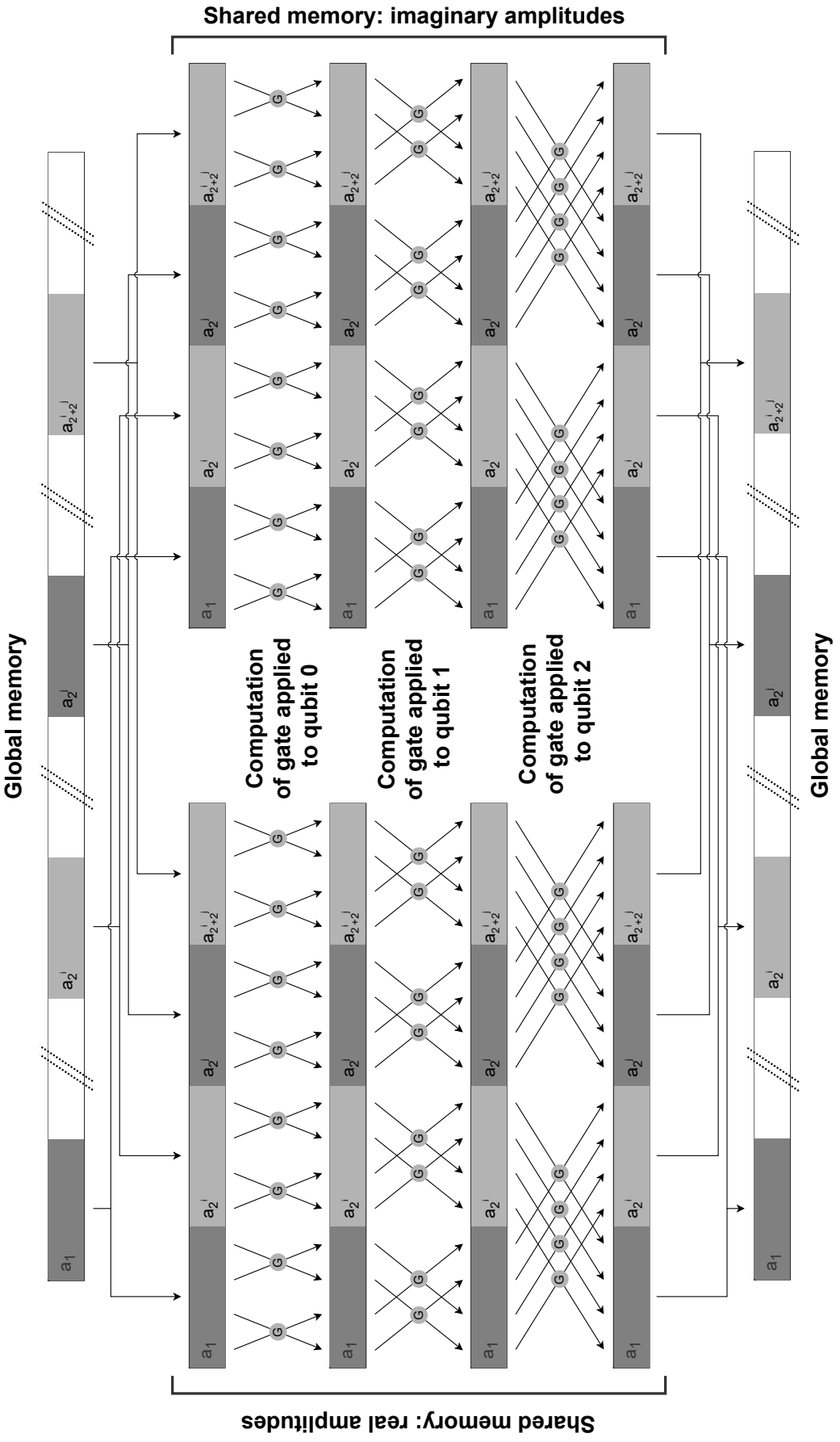


Figure E.8.: The figure depicts the proposed caching strategy of cluster-spaces in the shared memory using $N_S = 4$.

E.4. Formation of gate-groups

The algorithm depicted in E.4 is used during the circuit preprocessing of SEQCS (see Fig. 1.2) to rearrange the gate-queue and hence to form gate-groups. Each gate-group can be simulated using a single shared-memory traversal, thus reducing the number of memory transactions. Consequentially, the algorithm is designed to maximize the group size.

```
Groups   = [];  //Final gate-groups

for each gate Ĝ_k in the queue {

    Qubits  = List of N qubits initialized as unobstructed
    Group   = [Ĝ_k]    //List of gates forming the new group

    for each gate Ĝ_{j>k} in Queue {

        if inserting Ĝ_j into Group violates
           maximal group size
            break;

        if type(Ĝ_k) == type(Ĝ_j) and for all q in T_j Qubits[q] is unobstructed
        or [partially-obstructed and Ĝ_j is diagonal]
            Insert Ĝ_j into Group;

        else
            if Ĝ_j is diagonal
                 for each q in T_j
                    Set Qubits[q] as partially-obstructed;
            else
                for each q in T_j
                    Set Qubits[q] as obstructed;

    }

    Insert Group into Groups;
    Remove Group from Queue;

}
```

Algorithm E.4: Formation of gate-groups. Here, $\hat{G}_k$ denotes a gate at position k in the gate-queue, with T_k being an array of its target qubits. The algorithm is executed two times, with both nested loops iterating the gate queue in positive order first, followed by a traversal in negative order.

E.5. Out-of-order gate execution

The out-of-order execution scheme is a technique used by SEQCS to prevent shared memory bank conflicts in the evaluation of combined Hadamard, $\pm X$ and $\pm Y$ gates. It is based around the observation, that in order to fully utilize the shared memory bandwidth the threads need to access the initial state amplitudes as well as process them in a different order, potentially causing significant warp divergences. In the following, the proposed strategy will be explained with respect to two combined Hadamard and $+X$ gates:

$$H_i \otimes H_j = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \qquad +X_i \otimes +X_j = \frac{1}{2}\begin{pmatrix} 1 & i & i & -1 \\ i & 1 & -1 & i \\ i & -1 & 1 & i \\ -1 & i & i & 1 \end{pmatrix}$$

Table E.5: Tensor product matrix of two Hadamard (left) and two $+X$ (right) gates in the state-space of qubits i and j.

Consider a gate-pair applied to the first two bit-indices in the shared-memory ($i = 0,\ j = 1$), i.e. the counting bits c_i and c_j are placed among the first 5 bit-positions in J (see Eq. 1.21), which index the shared memory banks of the cached amplitudes. A key insight into the combined evaluation of such gates is that when using the same traversal of the counting bits (e.g. $c_1\, c_0 = 0\,0 \to 0\,1 \to 1\,1 \to 1\,0$) only a quarter of the shared memory banks is accessed each time a state amplitude is requested from the memory, yielding a 4-way bank conflict. To conquer this problem, SEQCS initializes c_i and c_j based on the threadID indices t_i and t_j, respectively, hence starting at different positions in the inverse-gray code. In order to prevent branch divergences during the expensive memory transactions, SEQCS stores the amplitudes ψ_{c_i,c_j} among the variables V_{kl} in the order in which they are accessed from the shared memory. In doing so, four separate thread groups emerge, each featuring a different distribution of ψ_{c_i,c_j} among V_{kl} (see Tab. E.6).

	$\psi_{0,0}$	$\psi_{0,1}$	$\psi_{1,0}$	$\psi_{1,1}$
$t_1\, t_0 = 0\,0$	V_{00}	V_{01}	V_{10}	V_{11}
$t_1\, t_0 = 0\,1$	V_{01}	V_{00}	V_{11}	V_{10}
$t_1\, t_0 = 1\,0$	V_{10}	V_{11}	V_{00}	V_{01}
$t_1\, t_0 = 1\,1$	V_{11}	V_{01}	V_{01}	V_{00}

Table E.6: Permutation of the state amplitudes ψ_{c_i,c_j} among the variables V_{kl} based on threadID indices t_i and t_j.

Since the amount of registers on each SM is limited, each thread is only able to hold up to 6 double-precision floats simultaneously without causing register spilling. Moreover, the processed amplitudes must be written back to the shared memory in a different order depending on t_i and t_j to prevent bank-conflicts, yielding four separate instruction branches. Due to these limitations, it is not possible to first reorder the amplitudes ψ_{c_i,c_j} among the variables V_{kl} and then compute the altered amplitudes ψ'_{mn} synchronously across the threads in a warp

(i.e. all threads process ψ'_{mn} at the same time), as the final amplitudes cannot be simultaneously held in the registers. To conquer this problem, SEQCS considers the real ($\psi^{real}_{c_i,c_j}$) and imaginary ($\psi^{imag}_{c_i,c_j}$) part of the amplitudes separately (hence the separation of the real and imaginary data in the shared memory; see section 1.2.3.2) and proposes an out-of-order execution scheme, allowing to process all four instruction branches simultaneously. This is done by exploiting the permutations of the amplitudes ψ_{c_i,c_j} among the variables V_{kl} and the matrix coefficients in each row (see Tab. E.5), such that each thread can execute the same instructions while processing a different amplitude ψ'_{c_i,c_j}. Tables E.7 and E.8 depict the computation-order of the amplitudes for each thread group $t_i\,t_j$.

$H_i \otimes H_j$	
$t_1\,t_0 = 0\,0$	$\psi'^{\,real}_{00} = \frac{1}{2}\,(V_{00} + V_{01} + V_{10} + V_{11})$
	$\psi'^{\,real}_{01} = \frac{1}{2}\,(V_{00} - V_{01} + V_{10} - V_{11})$
	$\psi'^{\,real}_{11} = \frac{1}{2}\,(V_{00} - V_{01} - V_{10} + V_{11})$
	$\psi'^{\,real}_{10} = \frac{1}{2}\,(V_{00} + V_{01} - V_{10} - V_{11})$
$t_1\,t_0 = 0\,1$	$\psi'^{\,real}_{01} = \frac{1}{2}\,(-V_{00} + V_{01} - V_{10} + V_{11})$
	$\psi'^{\,real}_{11} = \frac{1}{2}\,(-V_{00} + V_{01} + V_{10} - V_{11})$
	$\psi'^{\,real}_{10} = \frac{1}{2}\,(V_{00} + V_{01} - V_{10} - V_{11})$
	$\psi'^{\,real}_{00} = \frac{1}{2}\,(V_{00} + V_{01} + V_{10} + V_{11})$
$t_1\,t_0 = 1\,0$	$\psi'^{\,imag}_{10} = \frac{1}{2}\,(-V_{00} - V_{01} + V_{10} + V_{11})$
	$\psi'^{\,real}_{00} = \frac{1}{2}\,(V_{00} + V_{01} + V_{10} + V_{11})$
	$\psi'^{\,real}_{01} = \frac{1}{2}\,(V_{00} - V_{01} + V_{10} - V_{11})$
	$\psi'^{\,real}_{11} = \frac{1}{2}\,(-V_{00} + V_{01} + V_{10} - V_{11})$
$t_1\,t_0 = 1\,1$	$\psi'^{\,real}_{11} = \frac{1}{2}\,(V_{00} - V_{01} - V_{10} + V_{11})$
	$\psi'^{\,real}_{10} = \frac{1}{2}\,(-V_{00} - V_{01} + V_{10} + V_{11})$
	$\psi'^{\,real}_{00} = \frac{1}{2}\,(V_{00} + V_{01} + V_{10} + V_{11})$
	$\psi'^{\,real}_{01} = \frac{1}{2}\,(-V_{00} + V_{01} - V_{10} + V_{11})$

Table E.7: Calculation of the first half of the altered amplitudes ψ'_{c_i,c_j} for the combined Hadamard gate. The remaining amplitudes are determined identically by swapping $real \leftrightarrow imag$. Each cell depicts the processing order based on the thread group $t_i\,t_j$, e.g. $t_1\,t_0 = 0\,0$ calculates $\psi'^{\,real}_{00} \rightarrow \psi'^{\,real}_{01} \rightarrow \psi'^{\,real}_{11} \rightarrow \psi'^{\,real}_{10}$.

$+X_i \otimes +X_j$	
$t_1\,t_0 = 0\,0$	$\psi'^{\,real}_{00} = \frac{1}{2}\,(V_{00} - V_{01} - V_{10} - V_{11})$ $\psi'^{\,imag}_{01} = \frac{1}{2}\,(V_{00} + V_{01} - V_{10} + V_{11})$ $\psi'^{\,real}_{11} = \frac{1}{2}\,(-V_{00} - V_{01} - V_{10} + V_{11})$ $\psi'^{\,imag}_{10} = \frac{1}{2}\,(V_{00} - V_{01} + V_{10} + V_{11})$
$t_1\,t_0 = 0\,1$	$\psi'^{\,real}_{01} = \frac{1}{2}\,(V_{00} - V_{01} - V_{10} - V_{11})$ $\psi'^{\,imag}_{11} = \frac{1}{2}\,(V_{00} + V_{01} - V_{10} + V_{11})$ $\psi'^{\,real}_{10} = \frac{1}{2}\,(-V_{00} - V_{01} - V_{10} + V_{11})$ $\psi'^{\,imag}_{00} = \frac{1}{2}\,(V_{00} - V_{01} + V_{10} + V_{11})$
$t_1\,t_0 = 1\,0$	$\psi'^{\,real}_{10} = \frac{1}{2}\,(V_{00} - V_{01} - V_{10} - V_{11})$ $\psi'^{\,imag}_{00} = \frac{1}{2}\,(V_{00} + V_{01} - V_{10} + V_{11})$ $\psi'^{\,real}_{01} = \frac{1}{2}\,(-V_{00} - V_{01} - V_{10} + V_{11})$ $\psi'^{\,imag}_{11} = \frac{1}{2}\,(V_{00} - V_{01} + V_{10} + V_{11})$
$t_1\,t_0 = 1\,1$	$\psi'^{\,real}_{11} = \frac{1}{2}\,(V_{00} - V_{01} - V_{10} - V_{11})$ $\psi'^{\,imag}_{10} = \frac{1}{2}\,(V_{00} + V_{01} - V_{10} + V_{11})$ $\psi'^{\,real}_{00} = \frac{1}{2}\,(-V_{00} - V_{01} - V_{10} + V_{11})$ $\psi'^{\,imag}_{01} = \frac{1}{2}\,(V_{00} - V_{01} + V_{10} + V_{11})$

Table E.8: Calculation of the first half of the altered amplitudes ψ'_{c_i,c_j} for the combined $+X$ gate. The remaining amplitudes are determined identically by swapping $real \leftrightarrow imag$. Each cell depicts the processing order based on the thread group $t_i\,t_j$, e.g. $t_1\,t_0 = 0\,0$ calculates $\psi'^{\,real}_{00} \rightarrow \psi'^{\,imag}_{01} \rightarrow \psi'^{\,real}_{11} \rightarrow \psi'^{\,imag}_{10}$.

It is important to mention here that the evaluation of $H_i \otimes H_j$ demands conditioned negations of some variables V_{kl} based on $t_i\,t_j$, causing small branch divergences (see discussion in section 1.3.3.2), while $+X_i \otimes +X_j$ can be executed fully synchronized as all threads in a warp must perform the same instructions. The latter is because the permutation of ψ_{c_i,c_j} among V_{kl} compensates the permutation of the matrix elements in Tab. E.5.

F. Conversion between Ising and QUBO formulation

Equations F.1 and F.7 show the conversion between the QUBO formulation and the Ising formulation of the cost function $\mathrm{C}(Z/S)$. Here, the former uses the classical N bit binary string $Z = [z_{N-1} \ldots z_0]_2$, while the latter operates on an N dimensional array of Ising spins $S = [s_{N-1} \ldots s_0]_2$. In this thesis, the gate-based quantum computing convention is used, such that $z_i = \frac{1}{2}[1 - s_i]$. In doing so, $z_i = 0$ ($z_i = 1$) is mapped to $s_i = +1 \mathrel{\widehat{=}} \text{spin}- \uparrow$ ($s_i = -1 \mathrel{\widehat{=}} \text{spin}- \downarrow$), hence one can simply replace the spin variables s_i by the Pauli-z operators $\hat{\sigma}_i^z$ to obtain the Ising formulation of the cost Hamiltonian $\hat{H}_C$. The latter is used in quantum optimization algorithms (see section 2.2). Note that $z_i = z_i^2$, $s_i^2 = 1$, $Q_{i,j} = Q_{j,i}$ and $J_{i,j} = J_{j,i}$ are used below.

$$\mathrm{C}_{QUBO}(Z) = \sum_{i \leq j} Q_{i,j}\, z_i\, z_j \;+\; C_{QUBO} \tag{F.1}$$

$$= \sum_{i \leq j} Q_{i,j} \left(\frac{1 - s_i}{2}\right) \left(\frac{1 - s_j}{2}\right) \;+\; C_{QUBO} \tag{F.2}$$

$$= \frac{1}{4} \sum_i Q_{i,i} \left[1 - 2s_i + s_i^2\right] \;+\; \frac{1}{4} \sum_{i<j} Q_{i,j} \left[1 - s_i - s_j + s_i s_j\right] \;+\; C_{QUBO} \tag{F.3}$$

$$= -\frac{1}{4} \sum_i s_i \left[Q_{i,i} + Q_{i,i} + \sum_{i<j} Q_{i,j} + \sum_{i>j} Q_{i,j} \right] \;+\; \frac{1}{4} \sum_{i<j} Q_{i,j}\, s_i\, s_j \tag{F.4}$$
$$+\; \frac{1}{2} \sum_i Q_{i,i} \;+\; \frac{1}{4} \sum_{i<j} Q_{i,j} \;+\; C_{QUBO}$$

$$= -\sum_i s_i \underbrace{\left[\frac{1}{4} Q_{i,i} \;+\; \frac{1}{4} \sum_j Q_{i,j} \right]}_{h_i} \;+\; \sum_{i<j} s_i\, s_j \underbrace{\frac{1}{4} Q_{i,j}}_{J_{i,j}}$$
$$+\; \underbrace{\frac{1}{2} \sum_i Q_{i,i} \;+\; \frac{1}{4} \sum_{i<j} Q_{i,j} \;+\; C_{QUBO}}_{C_{Ising}} \tag{F.5}$$

$$= \mathrm{C}_{Ising}(S) \tag{F.6}$$

$$\mathrm{C}_{Ising}(S) = -\sum_i h_i\, s_i \;+\; \sum_{i<j} J_{i,j}\, s_i\, s_j \;+\; C_{Ising} \tag{F.7}$$

$$= -\sum_i h_i\, (1 - 2z_i) \;+\; \sum_{i<j} J_{i,j}\, (1 - 2z_i)\, (1 - 2z_j) \;+\; C_{Ising} \tag{F.8}$$

$$= \sum_{i<j} z_i\, z_j \underbrace{4\, J_{i,j}}_{Q_{i,j},\, i \neq j} \;+ \underbrace{\sum_i 2\, h_i\, z_i \;-\; \sum_{i<j} 2\, J_{i,j}\, (z_i + z_j)}_{\sum_i z_i \left[2h_i - 2\sum_{j \neq i} j_{i,j}\right] = \sum_i z_i^2\, Q_{i,i}} \tag{F.9}$$

$$= \mathrm{C}_{QUBO}(Z) \tag{F.10}$$

G. Exact cover problems

The complete set of exact cover problem investigated in chapter 2, including their QUBO formulations and energy spectra, can be found at `https://jugit.fz-juelich.de/qip/seqcs/-/tree/main/res/Exact-Cover`. This set contains 48 unique problem instances in total, which are equally distributed among the system sizes $N \in \{10, 12, 14, \ldots, 20\}$. Each exact cover matrix contain 64 columns and the solution binary strings contain approximately $N/3$ set bits. In doing so, the problem Hamiltonians feature a low degeneracy of the numerous energy levels. In section G.1 the generation process of the problem instances will be explained.

G.1. Exact cover generator

All exact cover problem used throughout this thesis are generated randomly using an in-house C++ program. Given the number of qubits N, number of solution bits n, the minimal number of matrix columns P and a seed to initialize the random number generator as user-inputs, the program starts by generating the P bit binary strings (initialized to 0) corresponding to the first n matrix rows (solution subsets $V_{i<n}$). This is done by iterating through each of the P bits, selecting one of the n binary strings at random with equal probability, and setting its bit to 1. Hence, for each matrix column, the set of solution rows contains only a single 1. Next, the remaining $N-n$ binary strings are generated by traversing through each bit and setting it to 1 with probability $1/n$. The latter is done to achieve a similar distribution of set bits among all matrix rows. Note that this probability can also be changed via a user input, thus allowing to modify the relative population of 1s among the non-solution binary strings compared to the solution rows. Important to mention here is that a higher population typically causes a higher interaction between the qubits, hence increasing the energy splitting, while a lower population can result in a high degeneracy of the energy levels. In order to ensure that the problem instances feature only a single solution string Z_{opt}, the program checks all 2^N combinations of matrix rows and stores them into an array S. This process is performed in parallel using OpenMP tasks [113] and a binary search tree in the matrix rows. Hence, starting at a row p, the task checks if setting bit p to 1 would violate the exact cover. If that is not the case, it spawns a new task to check row $p+1$, featuring the previous binary string with bit p set to 1. In any case, a new task regarding the row $p+1$ with the previous bit string unchanged is spawned. This continues until $p = N$ and the solution strings are then appended to S. Afterwards, the program resolves the unwanted solutions by appending additional matrix columns and assigning set bits to the matrix rows as described above until S only contains a single element (bit string with the first n bits set to 1). As a final step, the program applies a random permutation to the matrix rows and returns the exact cover matrix, its QUBO formulation, the solution string and the minimal energy. Note that the program code can be found at `https://gitlab-public.fz-juelich.de/qip/exact-cover-generator`.

H. 2-SAT problems

The complete set of 2-SAT problems studied in chapter 2 is provided at `https://jugit.fz-juelich.de/qip/seqcs/-/tree/main/res/2-SAT`, including the QUBO formulations and energy spectra. The set contains 48 problems, with 8 instances per system size $N \in \{10, 12, 14, \ldots, 20\}$. These problems are provided by *Mehta et al.* and feature a high degeneracy of the energy levels [73].

I. Distribution of interaction coefficients

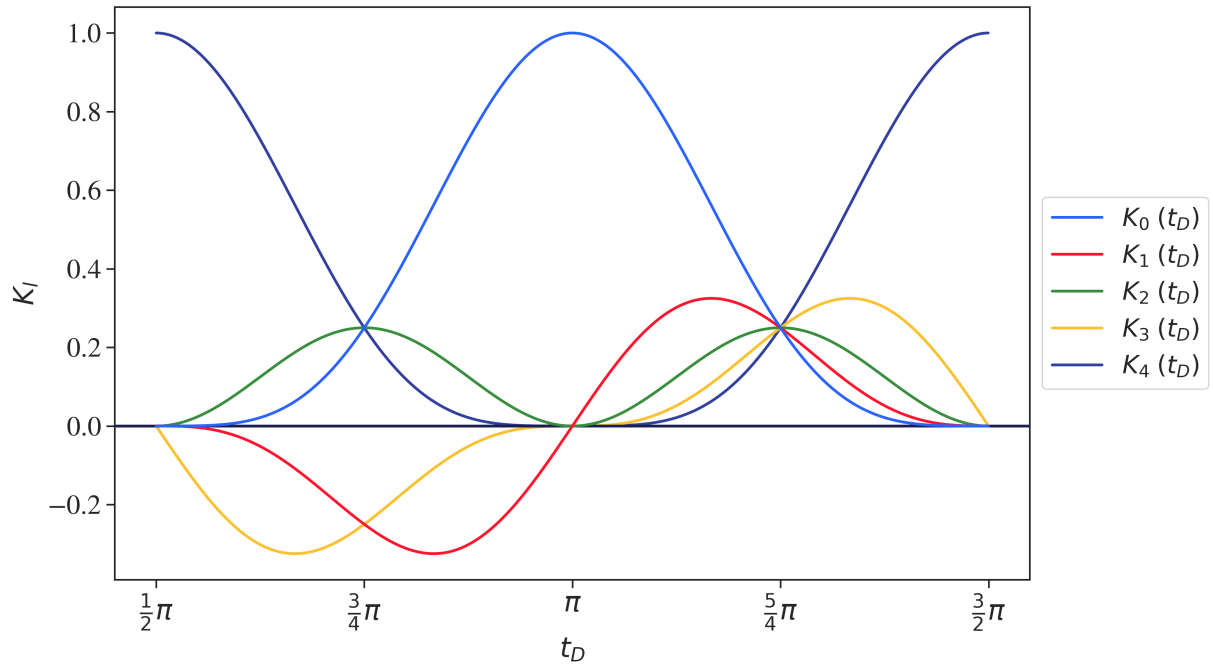


Figure I.9.: The plot shows the amplitudes of the interaction coefficients $K_l\,(t_D)$ as a function of the evolution time t_D. The data is obtained for the case of $N = 4$ qubits, yielding 5 distinct K_l. Note that the individual extrema are located at $t_l = k \cdot \pi \pm \arctan\sqrt{\frac{l}{N-l}}$ with $k \in \mathbb{N}$.

J. Additional performance results

J.1. GQW results

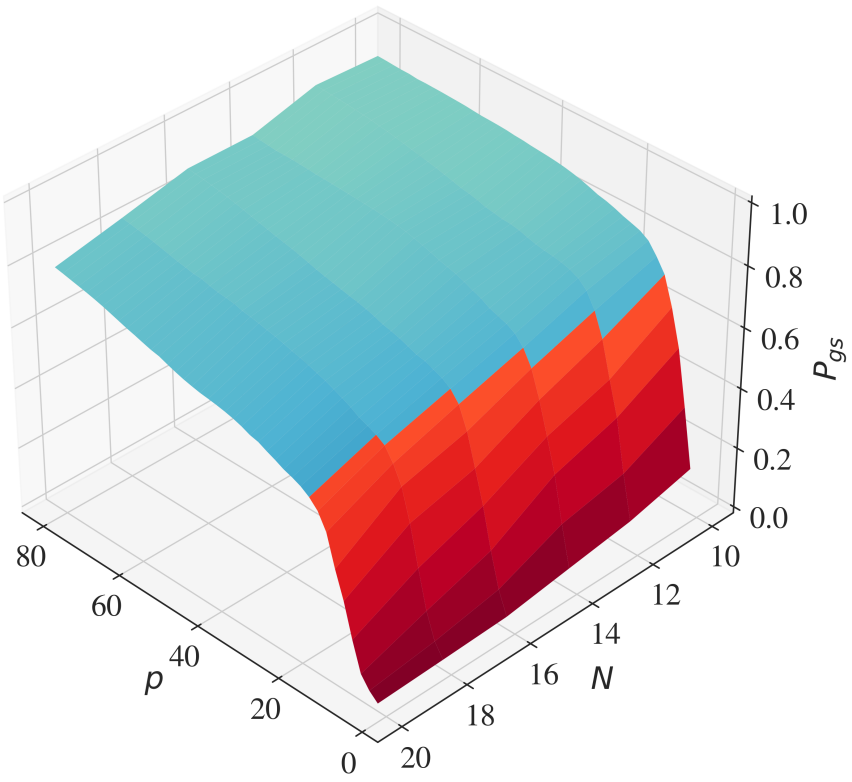


Figure J.10.: The figure presents a three-dimensional plot of the data presented in Fig. 2.12a in section 2.4.1, depicting the mean success probabilities P_{gs} across the sample groups as a function of the system size N and the number of iterations p. Note that only the *best* (highest P_{gs}) parameter sets obtained by the HGQW model throughout 200 optimization runs of the Nelder-Mead optimizer with a maximum of 4000 evaluations based on A_r on the set of exact cover problems (see appendix G) are shown. The two regions of operation are distinguished by colour, with *blue* denoting parameter configurations at $p \geq N$ and *red* representing $p < N$.

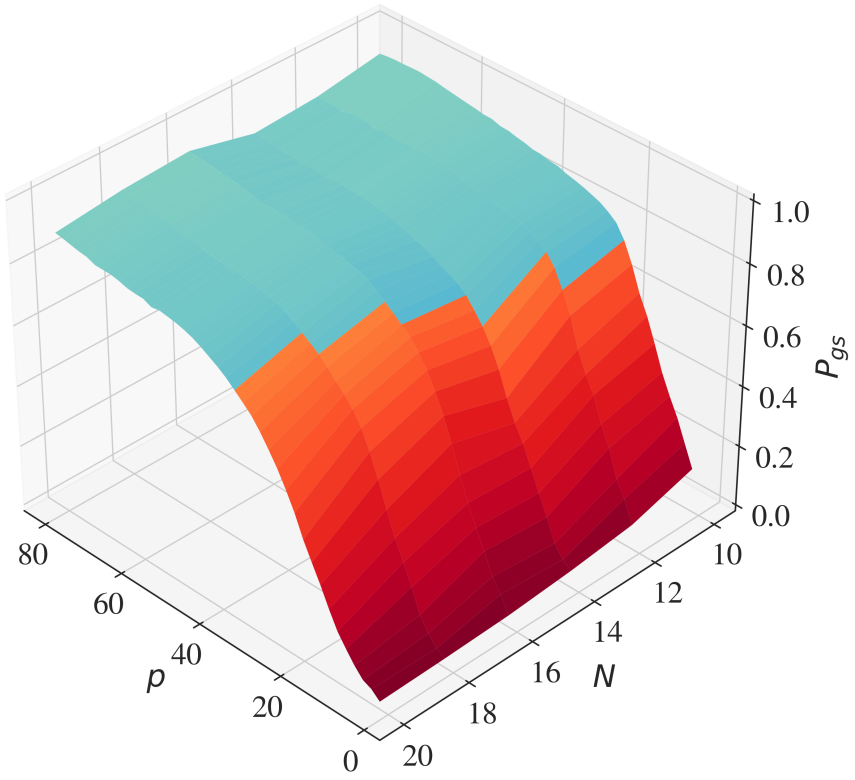
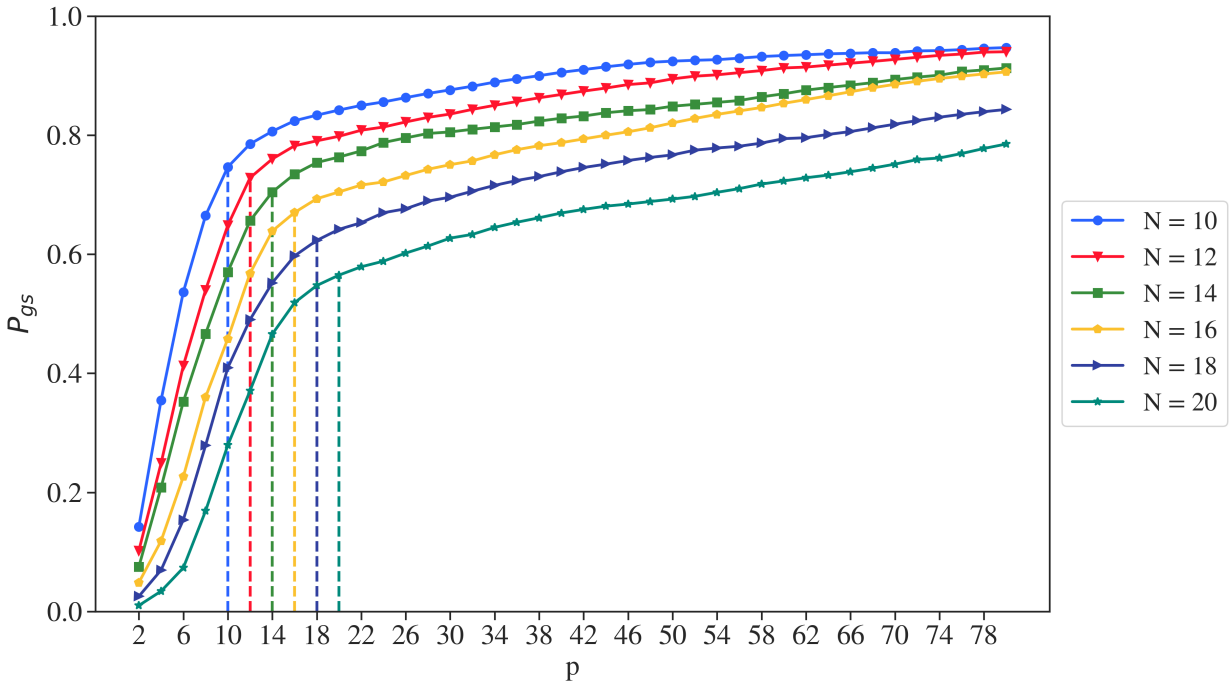
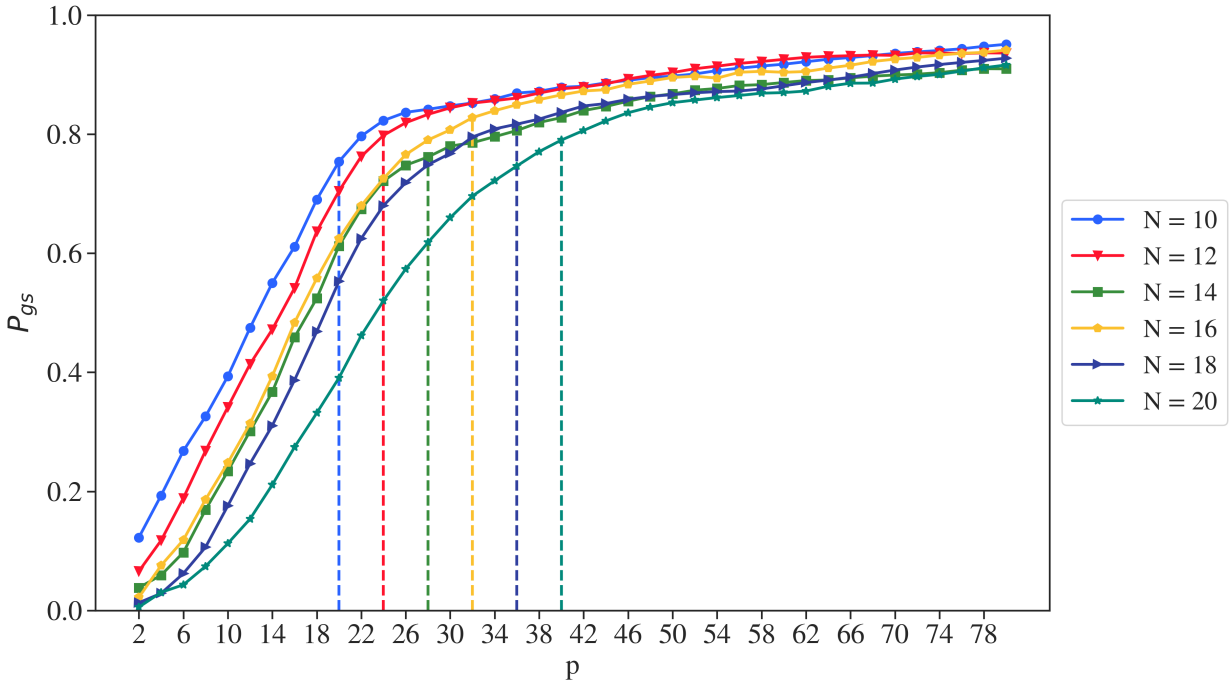


Figure J.11.: The figure presents a three-dimensional plot of the data presented in Fig. 2.13a in section 2.4.2, depicting the mean success probabilities P_{gs} across the sample groups as a function of the system size N and the number of iterations p. Note that only the *best* (highest P_{gs}) parameter sets obtained by the HGQW-A model throughout 200 optimization runs of the Nelder-Mead optimizer with a maximum of 4000 evaluations based on A_r on the set of 2-SAT problems (see appendix H) are shown. The two regions of operation are distinguished by colour, with *blue* denoting parameter configurations at $p \geq 2N$ and *red* representing $p < 2N$.



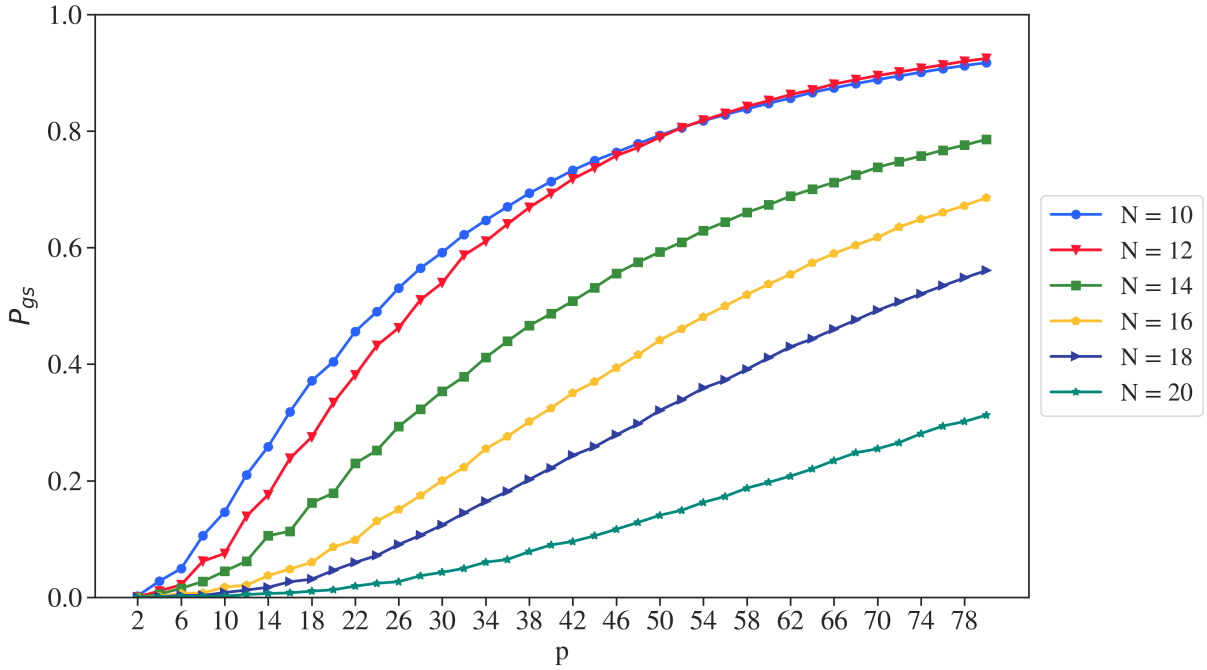
(a) HGQW model



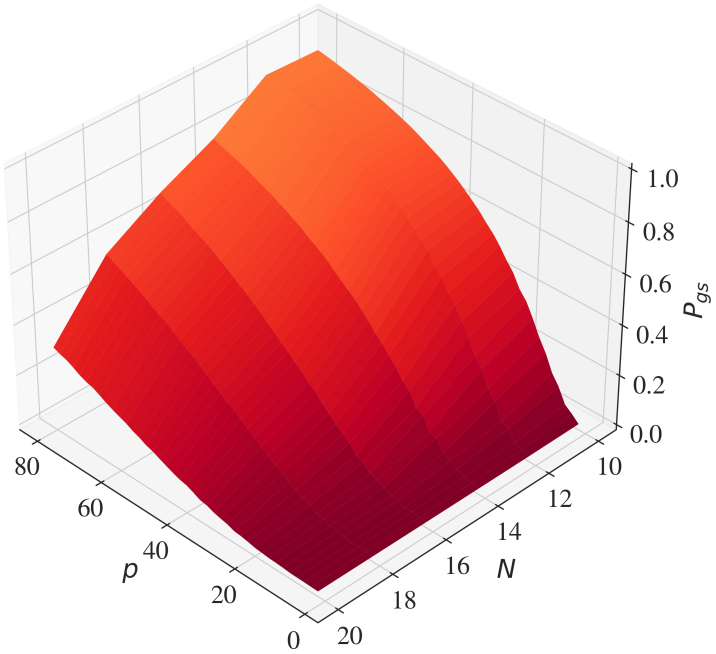
(b) HGQW-A model

Figure J.12.: The plots show the distributions of the success probability P_{gs} obtained by the HGQW model (a) and the HGQW-A model (b) as a function of the total number of iterations p for the exact cover and 2-SAT problems given in appendix G and H. The data is derived using a maximum of 4000 evaluations within 200 runs of the Nelder-Mead optimizer, focussing on an increase of P_{gs}. For each problem instance and each circuit depth p, the best (highest P_{gs}) parameter set is chosen, and the data is averaged within the sample groups (i.e. across problems of the same size N). The individual sample groups are distinguished by colour, and *dashed* lines indicate $p = N$ and $p = 2N$, respectively.

J.2. AQA results

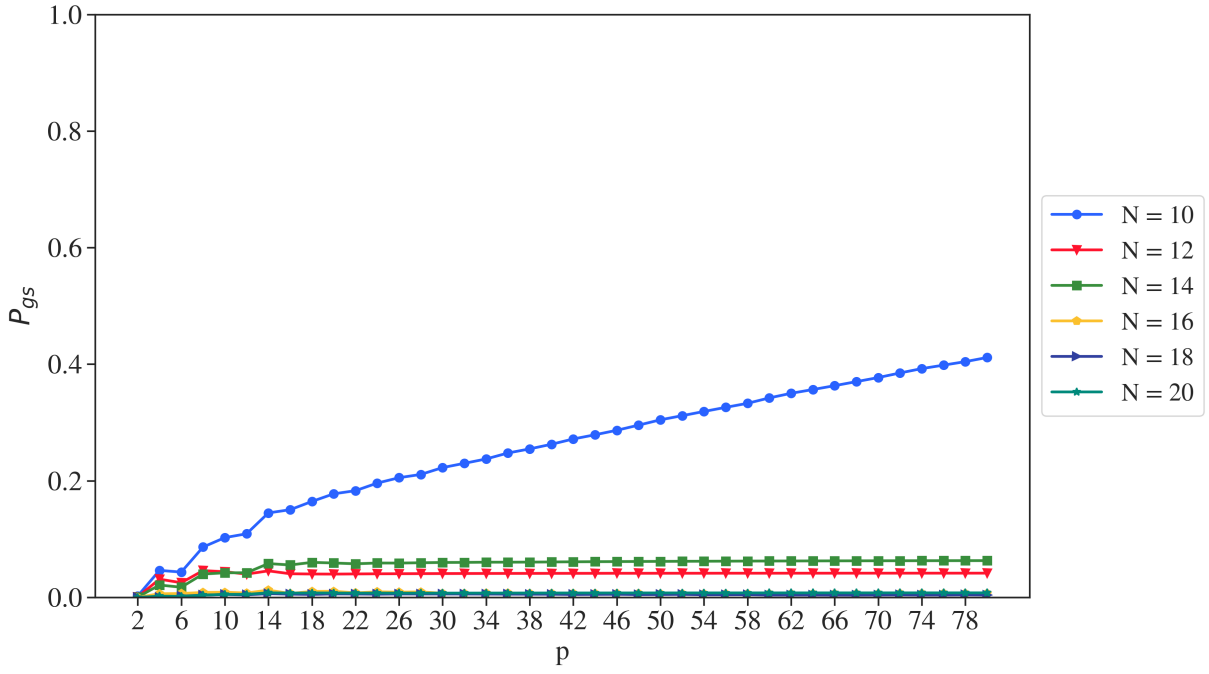


(a) 2D plot

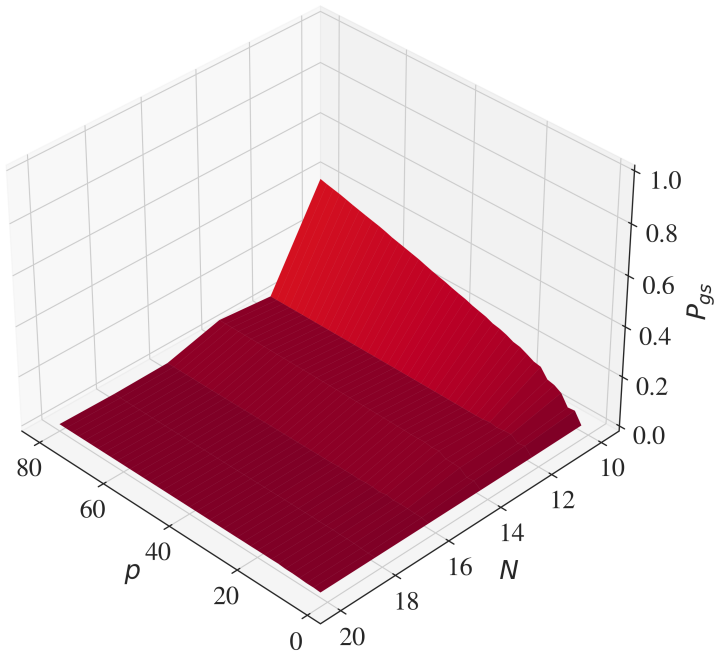


(b) 3D plot

Figure J.13.: The plots show the distribution of the mean success probabilities P_{gs} across the sample groups of the exact cover problems (see appendix G) obtained by the AQA as a function p and N. The data is derived using a maximum of 4000 evaluations within 200 runs of the Nelder-Mead optimizer, tuning the time step Δt (see Eq. 2.32) to reduce the approximation ratio A_r (see Eq. 2.7). Note that the D-Wave annealing schedule [80] is used.



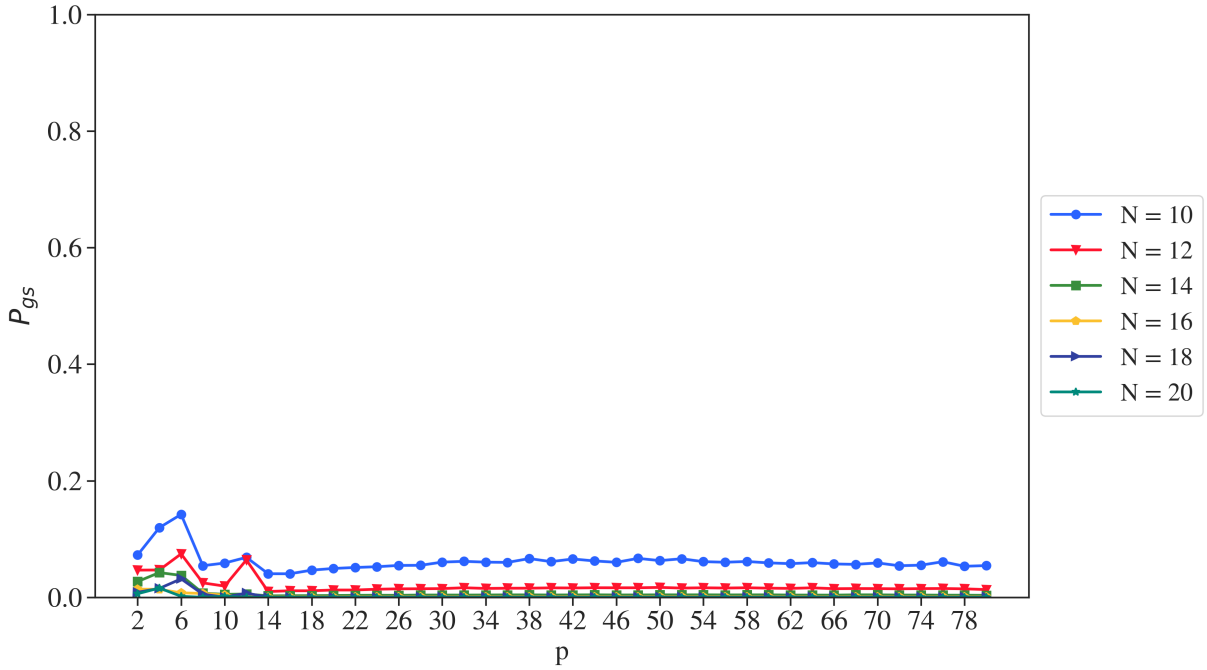
(a) 2D plot



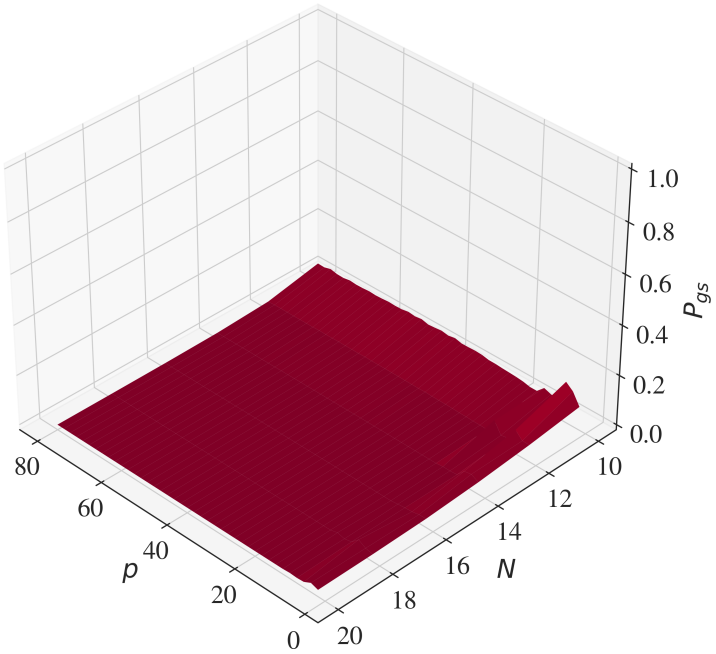
(b) 3D plot

Figure J.14.: The plots show the distribution of the mean success probabilities P_{gs} across the sample groups of the 2-SAT problems (see appendix H) obtained by the AQA as a function p and N. The data is derived using a maximum of 4000 evaluations within 200 runs of the Nelder-Mead optimizer, tuning the time step Δt (see Eq. 2.32) to reduce the approximation ratio A_r (see Eq. 2.7). Note that the D-Wave annealing schedule [80] is used and that, except for the $N = 10$ qubit problems, P_{gs} saturates in p.

J.3. QAOA results

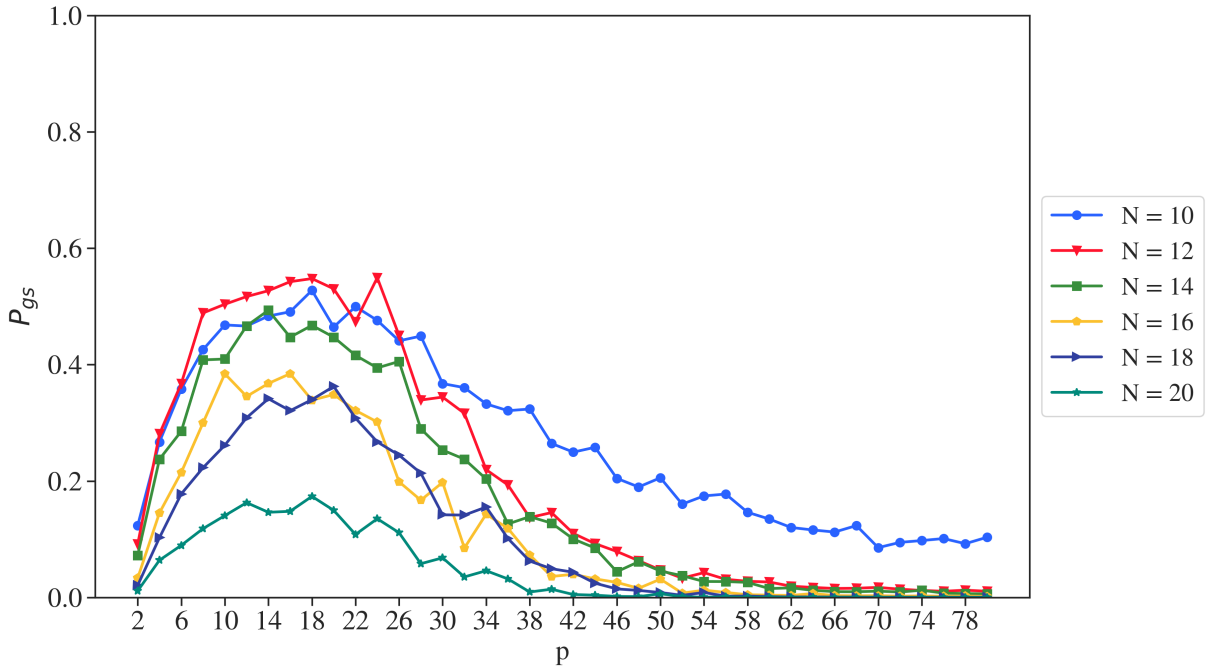


(a) 2D plot

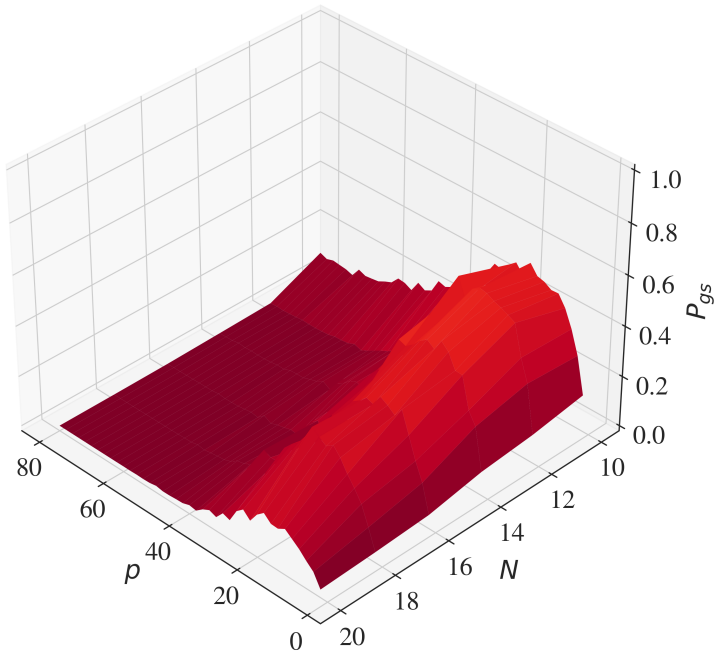


(b) 3D plot

Figure J.15.: The plots show the distribution of the mean success probabilities P_{gs} across the sample groups of the exact cover problems (see appendix G) obtained by the QAOA as a function p and N. The data is derived using a maximum of 4000 evaluations within 200 runs of the Nelder-Mead optimizer, tuning the variational parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ to reduce the approximation ratio A_r (see Eq. 2.7).



(a) 2D plot



(b) 3D plot

Figure J.16.: The plots show the distribution of the mean success probabilities P_{gs} across the sample groups of the 2-SAT problems (see appendix H) obtained by the QAOA as a function p and N. The data is derived using a maximum of 4000 evaluations within 200 runs of the Nelder-Mead optimizer, tuning the variational parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ to reduce the approximation ratio A_r (see Eq. 2.7).

Acknowledgments

An dieser Stelle möchte ich vor allem meiner Freundin Sarah-Marie Grabeck danken. Seit nun fast fünf Jahren bist du an meiner Seite, unterstützt mich bei all meinen verrückten Ideen, bist für mich da, wenn mir manchmal alles zu viel wird, bist meine beste Freundin, und zeigst mir, dass es auch noch ein Leben außerhalb der Physik gibt. Ich weiß, das vergangene Jahr war anstrengend für uns beide, aber ich freue mich darauf, endlich mit dir zusammenzuziehen und mein weiteres Leben mit dir zu verbringen.

Ebenfalls möchte ich vielmals meinen Eltern danken, die mich stets ermutigt und unterstützt haben in all meinen Vorhaben. Ich weiß, es war nicht immer einfach mit mir, aber ohne euch wäre ich jetzt nicht da, wo ich heute bin.

Auch möchte ich den Vorzeichenzauberern für ein unvergessliches Studium danken. Ohne euch hätten die letzten fünf Jahre mit Sicherheit viel weniger Spaß gemacht und ich werde die gemeinsame Zeit mit euch vermissen. Ich hoffe, dass wir weiterhin in Kontakt bleiben und wünsche euch allen viel Erfolg bei euren Promotionen.

Additionally, I would like to express my sincere gratitude towards Dennis Willsch for an excellent supervision. I really appreciate your support and the discussions we had. Many thanks go to Kristel Michielsen for giving me the opportunity to do my master thesis as well as my PhD in the group. Finally, I would like to thank the whole QIP group for the warm welcome, and I am looking forward to spending the next years with you guys.

Zentrales Prüfungsamt/Central Examination Office

Eidesstattliche Versicherung
Statutory Declaration in Lieu of an Oath

Schulz, Sebastian | 377242

Name, Vorname/Last Name, First Name | Matrikelnummer (freiwillige Angabe) Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel
I hereby declare in lieu of an oath that I have completed the present paper/Bachelor thesis/Master thesis* entitled

GPU-accelerated simulation of guided quantum walks

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Ort, Datum/City, Date | Unterschrift/Signature

*Nichtzutreffendes bitte streichen

*Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Ort, Datum/City, Date | Unterschrift/Signature