universität innsbruck

Machine Learning enhanced Quantum Optimization on Parity architectures

Beat Maximilian Haslinger

Master Thesis

University of Innsbruck
Faculty of Mathematics, Computer Science and Physics
Institute for Theoretical Physics

Supervisor: **Univ.-Prof. Dr. Wolfgang Lechner**
Co-Supervisor: **Glen Bigan Mbeng, PhD**

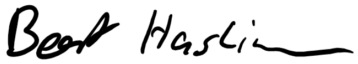
May 11, 2024

Abstract

The Quantum Approximate Optimization Algorithm (QAOA) is a prominent variational quantum algorithm designed to solve hard combinatorial optimization problems. Due to the limited qubit connectivity in current quantum hardware, obtaining accurate results with the QAOA is challenging. This thesis employs the Lechner-Hauke-Zoller (LHZ) architecture, which circumvents these hardware limitations by utilizing a novel encoding strategy that facilitates full connectivity with only local interactions. Within the QAOA this architecture introduces additional degrees of freedom, which will be exploited to improve efficiency. We apply a generalized QAOA, incorporating discrete variables, represented by gate sequences and optimized with Reinforcement Learning (RL). The study aims at identifying optimal gate sequences required to solve optimization problems on near-term quantum devices. We apply two RL algorithms: Projective Simulation (PS) and Proximal Policy Optimization (PPO). While each unitary retains the conventional QAOA-intrinsic continuous variational parameters, we formulate the order of available gates in the control sequence as a discrete optimization problem. We introduce additional gates inspired by counterdiabatic driving to enhance the expressiveness of the algorithm. We present a gate sequence, discovered through RL, significantly outperforming the application of commonly utilized sequences. Finally, we compare PS and PPO in their capacity to find sequences of interest.

Declaration

I hereby certify that this report has been written independently and that all necessary sources and references are given.



Beat Haslinger

May 11, 2024
Date

Acknowledgment

Foremost, I want to thank Wolfgang Lechner for this amazing opportunity to conduct my Master thesis within the Quantum Optimization research group. I am especially thankful for the guidance of my supervisor Glen Mbeng, who supported me during the whole process with his extensive knowledge and experience. Furthermore, great thanks also to the friendly members of the research group, who were always there for support and engaging discussions. Especially I want to mention Anita Weidinger and Kilian Ender, whose contributions were always very helpful.

Contents

List of Abbreviations

LHZ Lechner-Hauke-Zoller

QAOA Quantum Approximate Optimization Algorithm

RL Reinforcement Learning

PS Projective Simulation

PPO Proximal Policy Optimization

CO Combinatorial Optimization

TSP Traveling Salesman Problem

MaxCut Maximum Cut

NISQ Noisy intermediate-scale quantum

QA Quantum Annealing

DC Digitized counterdiabatic

ML Machine Learning

MDPs Markov Decision Processes

ECM epsiodic and compositional memory

VPG Vanilla Policy Gradient

TRPO Trust Region Policy Optimizing

SB3 Stable Baselines3

CHAPTER 0

Introduction

In today's world, optimization problems are ubiquitous, appearing in diverse sectors such as logistics, supply chain optimization, and scheduling, where they are critical for enhancing efficiency, reducing costs, and improving service delivery and product quality[1–3]. Therefore, a significant amount of research has been conducted to develop classical and quantum algorithms that improve the computation speed and quality of results, and promising algorithms have been developed in the fields of Machine Learning and Quantum Computing [4–6]. In this thesis, we will use tools from Machine Learning to further optimize the Quantum Approximate Optimization Algorithm (QAOA) for the scalable Lechner-Hauke-Zoller (LHZ) architecture [7]. Achieving accurate results with traditional QAOA often requires high qubit connectivity or a significant number of fault-intensive SWAP operations [8]. Within the LHZ architecture, just local interactions on neighboring qubits are required. For the QAOA, the LHZ approach introduces additional degrees of freedom in the variational parameters and the arrangement of quantum gates, which can be exploited to achieve better results. In this thesis, we employ decoded parity QAOA [9] operating with an enlarged set of gates. This gate set consists not only of the gates used in the original paper but also of additional gates, which are inspired by counterdiabatic driving. We aim to discover gate sequences improving the QAOA's performance. To this end, Reinforcement Learning (RL) agents are trained to predict suitable sequences, including gates from specific gate pools. We compare the quality of the learning and the discovered sequences from Projective Simulation (PS) agents and Proximal Policy Optimization (PPO) agents. Finally, the algorithm used is benchmarked with a classic random search.

0.1 Thesis Outline

The thesis is structured as follows: Chapter 1 introduces important concepts, including how to encode optimization problems in Ising spin glasses, the QAOA, the LHZ architecture, and a brief introduction to RL. Furthermore, the two RL agents (PS and PPO) are explained. The RL setup used to discover improved QAOA sequences is described in chapter 2. In chapter 3, we present challenging 10 qubit problem instances that are interesting for gate sequence optimization and discuss relevant gate pools. Our main contribution from

this work is the presentation of a gate sequence that significantly outperforms commonly utilized sequences for several problem instances. In this chapter, we also compare, discuss, and critically analyze the results obtained from PS and PPO agents.

CHAPTER 1

Theoretical Background

1.1 Solving hard optimization problems

Combinatorial Optimization (CO) is a central field of study within operations research and computer science, focusing on finding the optimal solution from a finite set of possible solutions. This domain has evolved into an interdisciplinary field spanning optimization, discrete mathematics and computer science with numerous crucial real-world applications [10]. These include, for example, logistics, supply chain optimization, and scheduling [1–3].

Usually CO (minimization) problems are defined on N-bit strings $\mathbf{z} = z_1 z_2 ... z_N$ that minimize a classical cost function $C(\mathbf{z})$, mapping the cost of a certain string to a real value [11]:

$$C(\mathbf{z}) : \{-1, 1\}^N \rightarrow \mathbb{R} \tag{1.1}$$

A CO problem can be formulated as a CO decision problem, which aims to answer the question of whether there exists an optimal solution in the set of feasible solutions such that its cost is smaller than or equal to a given value. Many CO decision problems turn out to be NP-hard in practice [10], meaning that it is not possible to find the solution efficiently, but a given solution can be easily verified. Due to the discrete nature of such CO problems, the solution space grows exponentially with the input size, making them intractable for classical computers. Examples include the Traveling Salesman Problem (TSP) [12], which involves finding the shortest possible route that visits a set of cities and returns to the origin city . Another example is the Maximum Cut (MaxCut) [13] problem, which involves finding a partition of a graph's vertices into two complementary sets, such that the number of edges between these sets is as large as possible. We introduce this problem in the following section.

In current research, NP-hard CO problems are often addressed using approximate optimization algorithms. These algorithms aim to find an approximate solution in polynomial time. The algorithm's quality is assessed by the approximation ratio r. For the minimiza-

tion problem with cost function $C(\mathbf{z})$, the approximation ratio r can be denoted as follows:

$$r = \frac{C(\mathbf{z})}{\min_{\mathbf{z}} C(\mathbf{z})} \tag{1.2}$$

where $\min_{\mathbf{z}} C(\mathbf{z})$ is the optimal value obtained by the optimal bit string $\mathbf{z}_{op}$.

Approximately solving NP-hard CO problems is still a difficult and computationally expensive task. Current research suggests that quantum computers offer advantages over classical computers for tackling these types of problems [14, 15].

A widely used and promising quantum algorithm for this purpose is the Quantum Approximate Optimization Algorithm (QAOA). As this algorithm plays a significant role in this thesis, we dedicate Sec. 1.2 to it. Before that, we introduce some basics of quantum computing and explain how NP-hard CO problems can be encoded in so-called Ising spin glasses.

1.1.1 Quantum computing basics

A quantum mechanical system is described by its quantum state $|\Phi\rangle$ [16]. This state exists in a complex vector space with an inner product, called the Hilbert space $\mathcal{H}$. In quantum computing, quantum bits are typically used to represent a state. The quantum bits (also known as qubits) are simple two-level systems with a two-dimensional complex Hilbert space $\mathcal{H} = \mathbb{C}^2$. This Hilbert space is spanned by the computational basis states $|0\rangle$ and $|1\rangle$. Since a qubit is often realized with spin systems, we use the following notations interchangeably: $|0\rangle = |\uparrow\rangle$, $|1\rangle = |\downarrow\rangle$.

Given the Hamiltonian H of a quantum system, its dynamics are completely described by the Schrödinger equation [16]. The time evolution operator for time independent Hamiltonians can be derived from the Schrödinger equation and is defined as

$$U(t_1, t_2) = \exp\left[\frac{-iH(t_2 - t_1)}{\hbar}\right]. \tag{1.3}$$

Where $\hbar$ denotes the Plank's constant, which in practice is usually absorbed in H. This operator evolves a quantum state $|\Phi\rangle$ from time t_1 to time t_2:

$$|\Phi(t_2)\rangle = U(t_1, t_2)\,|\Phi(t_1)\rangle \tag{1.4}$$

A quantum gate efficiently implements a unitary operation U and represents a discrete step in the evolution of a quantum system. Unitary operations map physical states to physical states and therefore must preserve the norm ($U^\dagger U = UU^\dagger = \mathbb{1}$). Important single

qubit rotation gates are given by the three Pauli matrices:

$$\begin{aligned}\sigma_x &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0| \\ \sigma_y &= \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i\,|0\rangle\langle 1| + i\,|1\rangle\langle 0| \\ \sigma_z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|\end{aligned} \tag{1.5}$$

The Pauli-X gate acts as a bit flip, mapping $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$. The Pauli-Z gate implements a phase flip, leaving $|0\rangle$ unchanged and mapping $|1\rangle$ to $|-1\rangle$. A simple two-qubit gate is the SWAP gate, which swaps two qubits. It acts on the four-dimensional basis $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ and has the matrix representation

$$\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1.6}$$

This gate is commonly used to address the problem of lacking connectivity in current quantum hardware. The CNOT gate is an example of a basic conditional two qubit gate. It performs a conditional NOT operation on a target qubit, flipping its state only if the control qubit is in the state $|1\rangle$. The CNOT gate is represented by the matrix

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.7}$$

This gate is used within the parity QAOA, as introduced and explained in Sec. 1.4.

1.1.2 Ising spin glass encoding

To solve (combinatorial) optimization problems on quantum hardware it is convenient to encode them in a spin model where the cost function corresponds to the system energy. Finding the ground state provides the solution to the problem. Ref. [6] shows that Ising spin glasses can mathematically formulate "all famous NP problems", including Karp's 21 NP-complete problems [13]. Here we focus on simple all-to-all connected Ising spin glasses represented by the Hamiltonian

$$H = \sum_{i=1}^{N} \sum_{j<i} J_{ij} s_i s_j \tag{1.8}$$

where J_{ij} denotes the coupling strength between spins s_i and s_j possessing the values $s = \pm 1$. The quantum Ising spin glass Hamiltonian is derived by exchanging the spins s_i and s_j with Pauli-Z matrices:

$$H_{\text{ising}} = \sum_{i=1}^{N} \sum_{j<i} J_{ij} \sigma_z^{(i)} \sigma_z^{(j)} \tag{1.9}$$

The sum covers all possible couplings among the N spins, with $\sigma_z^{(i)}$ representing the Pauli-Z matrix σ_z corresponding to the i-th spin:

$$\sigma_z^{(i)} = \underbrace{\mathbb{1} \otimes \ldots \otimes \mathbb{1}}_{(i-1) \text{ times}} \otimes \sigma_z \otimes \underbrace{\mathbb{1} \otimes \ldots \otimes \mathbb{1}}_{(N-i) \text{ times}} \tag{1.10}$$

Considering the exponential growth of the number of spin configurations as the number of spins increases, the intractability for large systems becomes problematic. Indeed, the complexity of finding the ground state of an Ising spin glass is proven to be NP-hard [17]. Next, we illustrate how to encode the Maximum Cut problem in an Ising spin glass.

1.1.3 Example: Maximum Cut

Maximum Cut (MaxCut) is a problem that has been extensively investigated by many researchers in different fields [18, 19]. It serves as a benchmark problem for quantum approximate algorithms [11, 20–22]. For this problem, given an undirected graph $\mathcal{G}(V, E)$ with vertices $i \in V$ and edges $(i, j) \in E$, the objective is to find two subsets S_0 and S_1 of V such that $S_0 \cup S_1 = V$ and $S_0 \cap S_1 = \emptyset$ [22]. This means that we aim to maximize the number of connections between the two subsets S_0 and S_1. An example graph is illustrated in Fig. 1.1. The more general version assigns weights to the edges and aims to find the cut

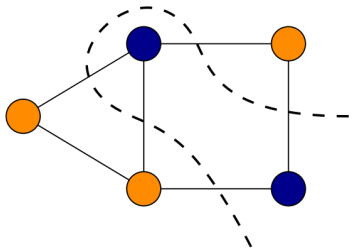


Figure 1.1: Illustration of a simple MaxCut graph with 5 vertices and 6 edges. The dashed line represents an exemplary maximum cut, separating the vertices into two disjoint subsets represented by orange and blue dots, respectively.

with the maximum total weight. This version of MaxCut is one of Karp's 21 NP-complete problems [13].

Continuing, we provide the spin glass Hamiltonian of the basic version:

$$H_{\text{MaxCut}} = \sum_{(i,j) \in E} \sigma_z^{(i)} \sigma_z^{(j)} \tag{1.11}$$

The sum ranges over all edges E. The expectation value E_{MaxCut} is inversely proportional to the number of cut edges, and minimizing this value yields to the maximum cut. The state $|\psi\rangle$ that minimizes the expectation value E_{MaxCut} with respect to the MaxCut Hamiltonian

$$E_{\text{MaxCut}} = \min_{|\psi\rangle} \langle\psi| H_{\text{MaxCut}} |\psi\rangle \tag{1.12}$$

is the ground state $|\psi_{GS}\rangle$ and thus encodes the solution of the given MaxCut problem.

1.2 The Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA), introduced in 2014 by E. Farhi et al. [21], is a hybrid algorithm that uses classical and quantum resources in order to find approximate solutions for CO problems. It is designed to take advantage of quantum computing, even though current quantum hardware is noisy and not scalable [23]. In this section, we describe the theory of the QAOA and introduce an improved scheme inspired by counterdiabatic driving, known as Digitized counterdiabatic (DC-QAOA) [24].

The goal of the QAOA is to approximate the ground state $|\psi_{GS}\rangle$ of the problem Hamiltonian H_P. H_P encodes the objective function

$$f(\mathbf{z}) : \{0,1\}^N \to \mathbb{R}. \tag{1.13}$$

Here $\mathbf{z}$ is an N-bit string with binary values. H_P acts diagonally on the computational basis states of the Hilbert space, which is of dimension 2^N. The eigenvalues of the problem Hamiltonian encode the objective values $f(\mathbf{z})$:

$$H_P |\mathbf{z}\rangle = f(\mathbf{z}) |\mathbf{z}\rangle . \tag{1.14}$$

Following Ref. [21], we introduce the phase operator, also called problem unitary, which is specified by parameter γ:

$$U_p(\gamma) = e^{i\gamma H_P} \tag{1.15}$$

Further the mixer Hamiltonian is defined as

$$H_M = \sum_{i=1}^{N} \sigma_x^{(i)}, \tag{1.16}$$

where $\sigma_x^{(i)}$ denotes the Pauli-X operator σ_x acting on the i-th qubit (see Eq. (1.10)). The mixer unitary, dependent on parameter β is:

$$U_x(\beta) = e^{i\beta H_M} = \prod_{i=1}^{N} e^{-i\beta\sigma_x^{(i)}}. \tag{1.17}$$

The state of the m-level QAOA is given by sequentially applying the above defined unitaries to the initial state $|\psi_0\rangle$,

$$|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle = U_x(\beta_1)U_p(\gamma_1)...U_x(\beta_m)U_p(\gamma_m) |\psi_0\rangle \tag{1.18}$$

where the initial state is an equal superposition of the computational basis states:

$$|\psi_0\rangle = |+\rangle^{\otimes N} = \frac{1}{\sqrt{2^N}} \sum_{\mathbf{z}} |\mathbf{z}\rangle \tag{1.19}$$

and the QAOA state $|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle$ is parameterized by $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_m)$ and $\boldsymbol{\beta} = (\beta_1, ..., \beta_m)$. A classical optimizer aims to find the best parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ by minimizing the expectation value of the Hamiltonian, which corresponds to the energy of the objective function:

$$E(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \langle f \rangle_{(\boldsymbol{\beta}, \boldsymbol{\gamma})} = \langle \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) | \, H_p \, | \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) \rangle \tag{1.20}$$

Going forward, we will refer to this expectation value as QAOA energy $E(\boldsymbol{\beta}, \boldsymbol{\gamma})$.

In Fig. 1.2 the working principle of the QAOA is illustrated. This feedback loop

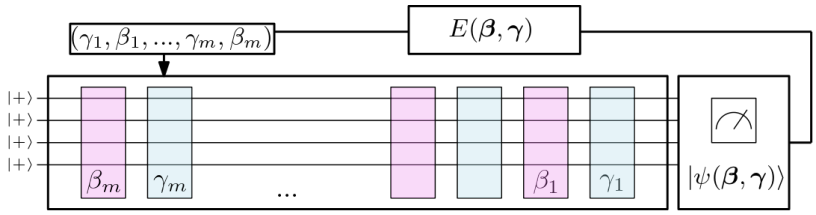


Figure 1.2: Illustration of the QAOA workflow. The colored boxes indicate the quantum gates $U_p(\gamma)$ and $U_x(\beta)$ dependent on the QAOA angles β and γ. One cycle stands for one iteration of the classical optimizer aiming to improve the QAOA angles. The iterative process includes: 1. Execution of the quantum algorithm, 2. Measurement of the resulting state $|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle$, 3. Calculation of the expectation value $E(\boldsymbol{\beta}, \boldsymbol{\gamma})$, 4. Adjustment of the QAOA angles.

between the classical optimizer and the quantum circuit is performed until the energy of the objective function converges. When the algorithm executes successfully, the solution to the optimization problem is encoded in the final state of the quantum system. For $m \to \infty$ the adiabatic theorem [25] ensures that the algorithm finds an exact solution to the corresponding classical CO problem, as discussed in Ref. [21]. Current Noisy intermediate-scale quantum (NISQ) hardware faces two issues. Firstly, the low coherence times of qubits restrict the circuit depth leading to low m. Secondly, the qubit connectivity often does not align with the necessary interactions in the problem Hamiltonian H_P. Consequently, to simulate higher connectivity, qubits must be swapped using SWAP gates (see Eq. (1.1.1)). However, these gates introduce errors and increase the circuit depth.

1.2.1 QAOA's connection to quantum annealing

Quantum Annealing (QA) aims to find the ground state of Ising models in order to solve hard CO problems [26, 27]. For this, the system is prepared with the mixer Hamiltonian H_M and initialized in its ground state. Then, the system undergoes adiabatic evolution, which is governed by the annealing Hamiltonian

$$H_a(t) = [1 - \lambda(t)]H_M + \lambda(t)H_P, \tag{1.21}$$

where $\lambda(t) \in [0,1]$ is the annealing schedule for $t \in [0,T]$. According to the adiabatic theorem [28], a quantum system remains in its ground state if the Hamiltonian is varied slowly enough and there is a sufficient energy gap between the ground state and the next higher eigenstates. Usually QA protocols follow a linear schedule $\lambda(t/T)$, however it can be advantageous to use more appropriate nonlinear schedules [29]. Considering that $H_a(t)$ can be decomposed into M k-local terms we can make use of the trotterized time evolution operator, derived in Ref. [30],

$$U(0,T) \approx \prod_{j=1}^{m} \prod_{n=1}^{M} \exp[-iH_n(j\Delta t)\Delta t], \tag{1.22}$$

where the terms $H_n(t)$ have k-body interactions at most. With the unitaries $U_x(\beta)$ and $U_p(\gamma)$ introduced in Sec. 1.2, we can parameterize $U(0,T)$ as [24]

$$U(\boldsymbol{\gamma},\boldsymbol{\beta}) = U_x(\beta_m)U_p(\gamma_m)...U_x(\beta_1)U_p(\gamma_1). \tag{1.23}$$

This describes the digital circuit ansatz for the QAOA, as previously described (see Eq. (1.18)). We observe that the the parameter m is directly correlated with the annealing time T. According to Ref. [24], m can be significantly decreased by including additional terms in the problem Hamiltonian as discussed in the following section.

1.2.2 From counterdiabatic driving to DC-QAOA

Within the adiabatic scheme, the most severe problem lies in the long annealing time required to leave the system in its ground state if the energy gap between the ground state and the first excited state of the evolved Hamiltonian $H_a(t)$ (see Eq. (1.21)) is small. Ref. [31] proposed a counterdiabatic driving protocol, allowing for fast changes in the Hamiltonian $H_a(t)$ without exciting transitions (also see [32, 33]). For a basic understanding of counterdiabatic driving, we refer the reader to the aforementioned paper. The key principle is to vary the parameter $\lambda(t)$ and counteract any unwanted diabatic excitations by adding an auxiliary term to the annealing Hamiltonian

$$H_{CD}(t) = H_a(t) + \dot{\lambda}(t)A_\lambda, \tag{1.24}$$

where A_λ denotes the adiabatic gauge potential [34]. Ref. [34] introduces a variational approach to find the best possible approximate protocols given certain physical constraints, like locality. In this work we use the approximate gauge potential, $A_\lambda^{(l)}$, introduced in Ref. [35]:

$$A_\lambda^{(l)} = i\sum_{k=1}^{l} \alpha_k \underbrace{[H_a,[H_a,...[H_a}_{2k-1},\partial_\lambda H_a]]] \tag{1.25}$$

The natural number l denotes the expansion's order and $\{\alpha_1,\alpha_2,...,\alpha_l\}$ is a set of l variational coefficients. In this equation we use the derivative of H_a with respect to λ. For example, the derivative of the annealing Hamiltonian Eq. (1.21) is

$$\partial_\lambda H_a = -H_M + H_P. \tag{1.26}$$

Digitized counterdiabatic (DC) QAOA (DC-QAQA), as introduced in Ref. [24], makes use of the potential $A_\lambda^{(l)}$ to design additional QAOA unitaries parameterized by a set of new parameters $\boldsymbol{\alpha} = \alpha_1, \alpha_2, ...$ These unitaries are

$$U_D(\boldsymbol{\alpha}) = e^{iA_\lambda^{(l)}}. \tag{1.27}$$

The new QAOA unitary (Eq. (1.23)), parameterized by $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and $\hat{\boldsymbol{\alpha}}$ becomes

$$U(\boldsymbol{\gamma}, \boldsymbol{\beta}, \hat{\boldsymbol{\alpha}}) = U_x(\beta_m)U_p(\gamma_m)U_D(\boldsymbol{\alpha}_m)...U_x(\beta_1)U_p(\gamma_1)U_D(\boldsymbol{\alpha}_1), \tag{1.28}$$

where $\hat{\boldsymbol{\alpha}} = \{\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_m\}$. According to Ref. [24], these additional parameters increase the degree of freedom, making it possible to access broader parts of the Hilbert space of the Hamiltonian. This can efficiently reduce the circuit depth, allowing for faster and more fault tolerant computations. Efforts have recently been made to improve QAOA protocols by adopting this approach [24, 36, 37].

1.3 LHZ architecture

As explained in Sec. 1.1.2 CO problems can be encoded in all-to-all connected Ising spin glasses, such that the QAOA (see Sec. 1.2) can find an approximate solution encoded in the final QAOA state. The connectivity of qubits in current quantum hardware is generally incompatible with the necessary interactions for the quantum gate that realizes the problem unitary. This is because usually only nearest neighbor interactions can be implemented, and implementing the problem unitary requires either an all-to-all connected quantum system or multiple additional gates to compensate for the missing connections.

W. Lechner et al. proposed a novel qubit architecture "with full connectivity, which can be implemented with local interactions only"[7], known as LHZ architecture or parity architecture. This scalable architecture was originally developed for the purpose of quantum annealing and relies only on local four-body interactions. However, this comes at a cost: the number of qubits increases quadratically. In the following chapter we will introduce the underlying concept and explain how to translate an Ising model into the LHZ scheme.

The LHZ-model introduces a novel set of physical qubits $\tilde{\sigma}$, representing the parity of two interacting qubits within the Ising model, referred to as logical qubits σ. The physical qubits can be implemented on a regular 2D lattice. Consequently, there are a total $K = N(N-1)/2$ of physical qubits, one for each interaction of the Ising Hamiltonian (Eq. (1.9)). With this translation, the interaction matrix elements J_{ij} are transformed into local magnetic fields acting on the physical qubits, thereby facilitating the implementation in the laboratory. Two interacting logical qubits with a parallel (antiparallel) alignment are respectively mapped to 1 (0), as illustrated in Fig. 1.3. The Ising Hamiltonian (Eq. (1.9)) is thus cast into the LHZ Hamiltonian encoding the optimization problem [7]:

$$\tilde{H}_p = \tilde{H}_z + \tilde{H}_c \tag{1.29}$$

The Hamiltonian is the sum of two components: the local field Hamiltonian

$$\tilde{H}_z = \sum_{i=1}^{K} \tilde{J}_i \tilde{\sigma}_z^{(i)} \tag{1.30}$$

and a constraint Hamiltonian

$$\tilde{H}_c = \sum_{l=1}^{K-N+1} C_l. \tag{1.31}$$

The sum of the local field component ranges over the K physical qubits, each multiplied by the new interaction matrix element $\tilde{J}_{i+(N-i)j} = J_{ij}$. The constraint Hamiltonian compensates for the increased degree of freedom with energy penalties for physical states that do not represent any logical states. For this purpose, $K - N + 1$ constraints, C_l, are introduced. These are constructed based on the consistency of the relative spin alignment on closed loops in the logical configuration. An example is depicted in Fig. 1.3 as an orange loop bonding the logical spins $\sigma_z^{(1)}\sigma_z^{(4)} \rightarrow \sigma_z^{(2)}\sigma_z^{(4)} \rightarrow \sigma_z^{(2)}\sigma_z^{(3)} \rightarrow \sigma_z^{(1)}\sigma_z^{(3)}$. This loop can be considered as a constraint plaquette in the LHZ architecture, as indicated by an orange square.

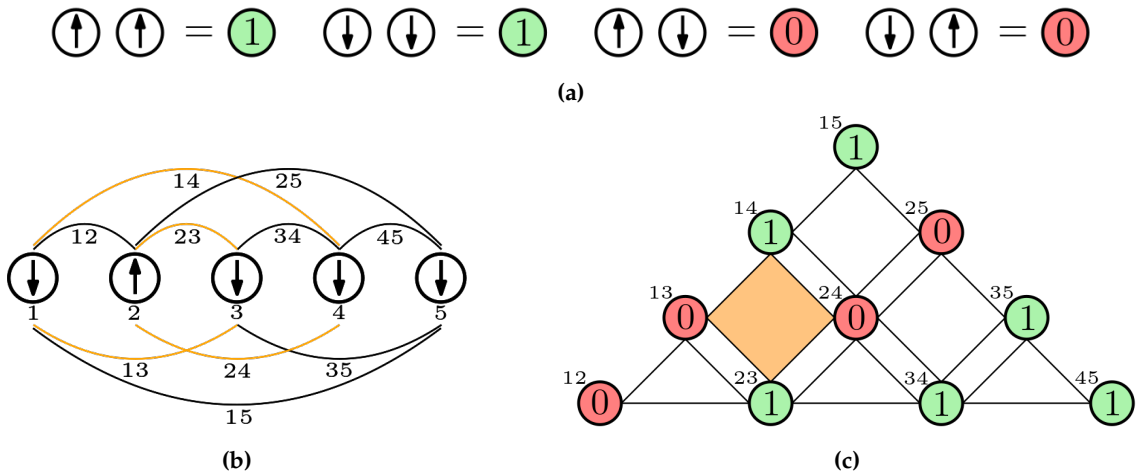


Figure 1.3: Illustrated example of the translation of a 5 (logical) qubit configuration with all-to-all connections (Panel (b)) in the LHZ architecture consisting of 10 physical qubits denoting the parity of the logical qubits (Panel (c)). The logical spins are depicted by arrows. Panel (a) provides the translation table from the Ising scheme in the LHZ scheme. An antiparallel (parallel) alignment of logical spins is represented with a physical qubit holding value 1 (0) and colored green (red), respectively. One exemplary constraint plaquette, resulting from a closed loop bonding 4 logical qubits (orange lines), is indicated as an orange square in the LHZ architecture. The represented configuration does not violate any parity constraints.

To ensure consistency in the relative spin alignment within a loop, an even number of logical spins that are antiparallel is necessary. This prevents contradictions when translating back from physical to logical qubits. For instance, the number of 0'th in the plaquette spanned by the four physical qubits $\tilde{\sigma}_z^{(14)}, \tilde{\sigma}_z^{(24)}, \tilde{\sigma}_z^{(23)}, \tilde{\sigma}_z^{(13)}$ must be even. Furthermore, it is required that the constraints cover all physical qubits and the number of constraints should

be at least $K - N$ [7]. The constraints C_l can be written as

$$C_l = -C\tilde{\sigma}_z^{(l,n)}\tilde{\sigma}_z^{(l,e)}\tilde{\sigma}_z^{(l,w)}[\tilde{\sigma}_z^{(l,s)}], \tag{1.32}$$

where C denotes the constraint strength and the Pauli-Z Matrices $\tilde{\sigma}_z^{(l,i)}$ with $i \in \{n, e, w, [s]\}$ (north, east, west, [south]) are associated with the four [three] qubits of plaquette l. If a constraint includes an odd number of 0's it is violated. For each violated constraint the constraint term contributes positively to the energy of H_p, thus penalizing every constraint violation. This is clarified by considering the eigenvalues of the physical qubits. A qubit with value 0 (1) has eigenvalue -1 ($+1$), respectively. Therefore, an odd number of qubits with a value of 0 result in positive contributions to the respective constraint C_l.

1.4 Parity QAOA

As discussed in Sec. 1.2, the quantum component of the traditional QAOA is limited by the qubit connectivity of current quantum hardware. Connecting the idea of the LHZ architecture, introduced in Sec. 1.3, and the QAOA, W. Lechner introduced a parallelizable QAOA scheme with local connectivity [38]. Continuing, we will refer to this concept as parity QAOA. By applying the LHZ architecture within the QAOA, only single qubit rotation and pairwise CNOT gates (see Eq. (1.1.1)) are required for the implementation on quantum hardware. In the following section, we will briefly explain this concept. Furthermore, we will introduce an improved and fault-tolerant method, known as decoded parity QAOA [9]. Lastly, we will briefly discuss the equivalence of two problem instances in the context of parity QAOA.

The working principle of the parity QAOA remains the same as that of the traditional QAOA; only the unitary operators change to account for the qubit remapping. By inserting $\tilde{H}_p$ (Eq. (1.29)) in Eq. (1.15), we obtain a unitary that can be split into two unitaries: $\tilde{U}_z(\gamma)$ and $\tilde{U}_c(\Omega)$. We also consider the mixer Hamiltonian

$$\tilde{H}_x = \sum_{i=1}^{K} \tilde{\sigma}_x^{(i)}, \tag{1.33}$$

where K denotes the number of physical parity qubits. The following three unitary operators act as building blocks for parity QAOA:

$$\begin{aligned}
\tilde{U}_x(\beta) &= \prod_{i=1}^{K} \mathrm{e}^{-i\beta\tilde{\sigma}_x^{(i)}} \\
\tilde{U}_z(\gamma) &= \prod_{i=1}^{K} \mathrm{e}^{-i\gamma\tilde{J}_i\tilde{\sigma}_z^{(i)}} \\
\tilde{U}_c(\Omega) &= \prod_{l=1}^{K-N+1} \mathrm{e}^{-i\Omega C\tilde{\sigma}_z^{(l,n)}\tilde{\sigma}_z^{(l,e)}\tilde{\sigma}_z^{(l,w)}[\tilde{\sigma}_z^{(l,s)}]}
\end{aligned} \tag{1.34}$$

When talking about gate sequences, we will neglect the angles and respectively refer to these unitaries as $\hat{\mathrm{X}}$ (for $\tilde{U}_x(\beta)$), $\hat{\mathrm{Z}}$ (for $\tilde{U}_z(\gamma)$), and $\hat{\mathrm{C}}$ (for $\tilde{U}_c(\Omega)$). With these unitaries, the variational parity QAOA state is written out as follows:

$$|\tilde{\psi}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})\rangle = \prod_{k=1}^{m} \tilde{U}_x(\beta_k)\tilde{U}_c(\gamma_k)\tilde{U}_z(\gamma_k) \, |\psi_0\rangle \tag{1.35}$$

The unitaries, dependent on the QAOA angles $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_m)$, $\boldsymbol{\beta} = (\beta_1, ..., \beta_m)$, and $\boldsymbol{\Omega} = (\Omega_1, ..., \Omega_m)$, are illustrated in Fig. 1.4 as colored boxes. Similar to the conventional QAOA,

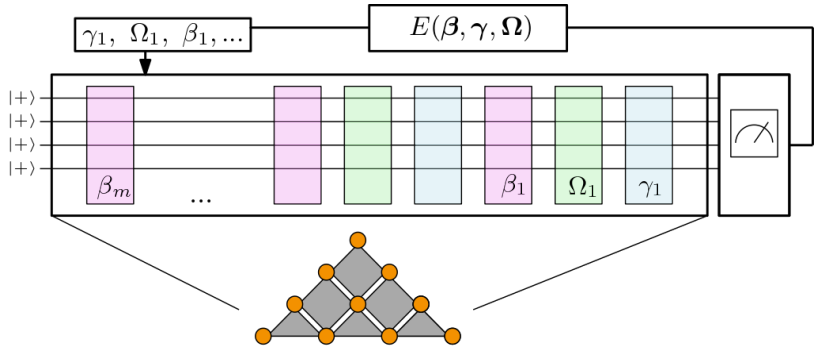


Figure 1.4: Illustration of the parity QAOA. The sketch of the LHZ architecture represents the qubit arrangement on which the quantum algorithm operates. The colored boxes represent the unitaries $\tilde{U}_x(\beta), \tilde{U}_z(\gamma)$ and $\tilde{U}_c(\Omega)$. One cycle stands for one iteration of the classical optimizer aiming to improve the QAOA angles. The iterative process includes: 1. Execution of the quantum algorithm, 2. Measurement of the resulting state $|\tilde{\psi}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})\rangle$, 3. Calculation of the expectation value $E(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})$, 4. Adjustment of the QAOA angles.

the classical optimizer tunes the QAOA angles to minimize the expectation value

$$E(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega}) = \langle\tilde{\psi}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})| \, \tilde{H}_p \, |\tilde{\psi}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})\rangle \tag{1.36}$$

After the optimization, the algorithm returns a set of final QAOA angles $(\boldsymbol{\beta}_f, \boldsymbol{\gamma}_f, \boldsymbol{\Omega}_f)$ and the final objective value

$$E_f = E(\boldsymbol{\beta}_f, \boldsymbol{\gamma}_f, \boldsymbol{\Omega}_f). \tag{1.37}$$

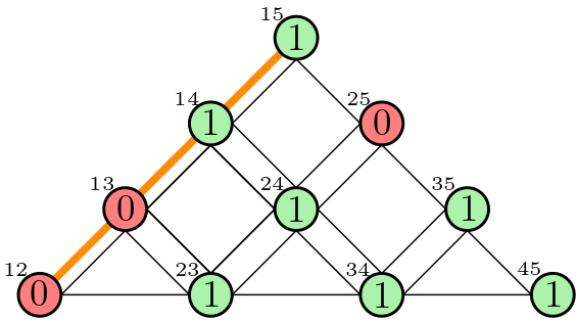
Typically, this process is repeated multiple times with randomly selected QAOA angles, as finding the global minimum is difficult when multiple local minima are present. The goal is to approximate the ground state $|\psi_{GS}\rangle$, which is given by the state minimizing $\langle\psi| \, \tilde{H}_p \, |\psi\rangle$. We denote the associated ground state energy by

$$E_{GS} = \langle\psi_{GS}| \, \tilde{H}_p \, |\psi_{GS}\rangle \, . \tag{1.38}$$

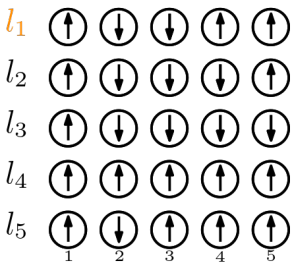
Given that for parity QAOA there are three unitaries instead of two, a natural question arises: Are there better unitary sequences than the standard sequence (Eq. 1.35)? In an effort to answer this question, we will later train Reinforcement Learning agents to learn improved sequences. Before that, in the following section, we introduce an improved parity QAOA scheme called decoded parity QAOA.

1.4.1 Decoded parity QAOA

Recently, A. Weidinger et al. proposed a novel error mitigation technique for QAOA based on the LHZ architecture [9], called decoded parity QAOA. While parity QAOA aims to minimize the energy from the physical state, decoded parity QAOA aims to minimize the average energy of a set of decoded logical states. These decoded states result from a remapping of the physical qubit configuration back into the all-to-all connected Ising model. In this thesis, we consider the decoding of all the so-called logical lines in the physical configuration, which represent a special set of spanning trees [7, 9, 39]. A logical line, denoted by l_i, includes all physical parity qubits containing the logical qubit i. The logical line corresponding to qubit 1 is depicted in Fig. 1.5a as an orange line. Decoding the physical configuration with this logical line using the translation table provided in Fig. 1.3 yields the upper logical spin configuration corresponding to the logical line l_1 in Fig. 1.5b. The remaining decoded logical configurations derive from the other logical lines. All de-



(a) physical configuration



(b) decoded logical configurations for the logical lines

Figure 1.5: Illustration of physical qubit configuration (a) decoded into the logical scheme (b). Table of all decoded logical spin configurations (up to a global spin flip) resulting from the spanning trees (b). The shown physical qubit configuration (a) violates the three four-body constraints. The first logical line corresponding to qubit 1 is indicated with an orange line, and its logical decoding is denoted by an orange l_1.

coded configurations would be identical if the physical configuration satisfies all parity constraints. However, in the presented physical qubit configuration, the three four-body constraints are violated (odd number of 0's), resulting in multiple decoded configurations.

The decoded parity QAOA decodes the physical state of LHZ-QAOA, represented by $|\tilde{\psi}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})\rangle$ (refer to Eq. (1.35)) in each QAOA iteration, and hence determines the average logical energy $\mathcal{C}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})$ using the following equation:

$$\mathcal{C}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega}) = \frac{1}{M} \sum_{i=1}^{N} \langle D_{l_i} \tilde{\psi}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})| \, H_{ising} \, |D_{l_i} \tilde{\psi}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})\rangle \tag{1.39}$$

where $D_{l_i} \tilde{\psi}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})$ stands for the decoded logical state associated with the logical line i. In Ref. [9], the best logical state with the lowest energy is selected as the approximate final solution. For subsequent calculations, it turns out to be favorable to use the mean energy $\mathcal{C}(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Omega})$ as final QAOA energy E_{qaoa} (see Appendix B). We refer to this method

as "decoded parity QAOA with final mean decoding". Whereas, the standard method is referred to as "decoded parity QAOA with final min decoding". For our purposes, decoded parity QAOA is particularly suitable, as it does not require knowledge of the constraint strength C (see Eq. (1.32)). This simplifies the RL algorithm used.

1.5 Reinforcement Learning

Machine Learning (ML) plays a significant role in the modern world due to the increasing number of applications, from image classification [40] to the large language model ChatGPT [41], which is accessible to the general public. There are three basic types of ML algorithms: Supervised Learning, Unsupervised Learning and Reinforcement Learning (RL). In the field of quantum optimization and quantum control, RL is demonstrating promising results [42–45]. For traditional QAOA with additional unitaries, similar approaches have already been examined by Yao et al. in Refs. [37, 44, 46].

RL is a dynamic process where an agent navigates an environment and learns to formulate a policy over time that maximizes cumulative long-term rewards. It is a very intuitive approach to machine learning wherein the agent learns by interacting with its environment, similar to the way human beings and animals do. Each step in this process includes the agent receiving an observation, selecting an action based on its established policy (a defined behavioral map that associates observations with actions), and executing the chosen action. Afterwards, the environment responds by issuing a scalar reward and transitioning into a new state, governed by its intrinsic dynamics. This iterative interaction between the agent and environment is the heart of RL.

RL problems are typically framed as Markov Decision Processes (MDPs) defined by a tuple $(\mathcal{S}, \mathcal{A}, p, \lambda)$. During an agent-environment interaction, the agent can select an action a from the action space $\mathcal{A}$ and execute it on the environment. Subsequently, the environment transitions to a new state $s \to s' \in \mathcal{S}$, where $\mathcal{S}$ represents the state space. This transition is governed by the inherent transition dynamics of the environment: $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$. Additionally, the agent receives a reward $\lambda : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, which it uses to determine future actions. The agent selects an action according to the policy $\pi(a|s)$, which is updated during the learning process in order to enhance the agent's success. The goal is to find an optimal policy that maximizes the total expected reward. Given a policy, the system evolves with trajectory $\zeta = (s_1, a_1, ..., a_q, s_{q+1})$, where q refers to the trajectory- or episode length.

In this work, we will discover gate sequences that improve the performance of decoded parity QAOA. To this end, RL agents will be trained to find gate sequences that result in a low QAOA energy. Continuing, we will introduce and discuss two RL models: Projective Simulation and Proximal Policy Optimization.

1.5.1 Projective Simulation

Projective Simulation (PS) is a model of agency that can also function as a RL-agent. It was introduced by H. J. Briegel et al. in 2012 [47]. PS has competed well against other RL algorithms like Q-Learning and SARSA [48, 49]. An important feature of PS is its ability to project the agent into future scenarios based on previous experiences. For that, the PS agent utilizes an epsiodic and compositional memory (ECM), which helps it learn and generalize. It is composed of so-called clips, labeled as c, which represent basic units of memory. In our application, the ECM consists of two layered decision trees, with each tree corresponding to one observed gate sequence. Fig. 1.6 shows the decision tree for a single observation. Percept clip s_1 represents an observation, while the three action clips a_1, a_2, and a_3 represent the available actions. The edges of this decision tree are represented by

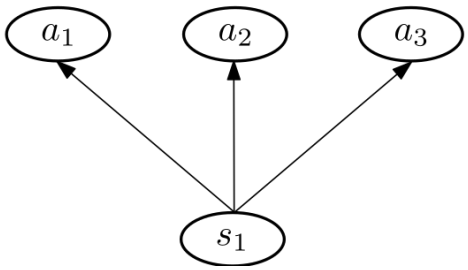


Figure 1.6: Example for a two layered decision tree with one observation clip s_1 and three action clips a_1, a_2 and a_3.

the arrows. They carry weights represented by h_{ij}, which are utilized in conjunction with a softmax distribution function to compute the transition probability at time t ($p_{ij}^{(t)}$):

$$p_{ij}^{(t)} = p^{(t)}(c_i \to c_j) = \frac{e^{\beta \tilde{h}_{ij}^{(t)}}}{\sum_k e^{\beta \tilde{h}_{ik}^{(t)}}} \tag{1.40}$$

Here β represents the softmax parameter. $\tilde{h}_{ij}^{(t)}$ is calculated with

$$\tilde{h}_{ij}^{(t)} = h_{ij}^{(t)} - \max_{k \in \mathcal{N}_i} h_{ik}^{(t)}. \tag{1.41}$$

where $\mathcal{N}_i$ refers to the set of all clips linked to clip c_i. Eq. (1.41) prevents calculation errors caused by large numbers. The h-values are initialized at a value of one ($h_{ij}^{(0)} = 1$) and updated according to

$$h_{ij}^{(t+1)} = h_{ij}^{(t)} + g_{ij}^{(t)} \cdot \lambda^{(t)} \tag{1.42}$$

at each time step t. The reward received at the current time step t is denoted as $\lambda^{(t)}$, and the glow value, representing the agent's short-term memory, is labeled as $g_{ij}^{(t)}$. Like each edge having an h-value, they also possess a g-value initialized at zero ($g_{ij}^{(0)} = 0$) and is subsequently set to one upon transition. This g-value is discounted at every time step with

$$g_{ij}^{(t+1)} = g_{ij}^{(t)}(1 - \eta), \tag{1.43}$$

where $0 \leq \eta \leq 1$ is called glow damping parameter. The glow value ensures that edges that are temporally closer to the reward are given more weight than those in the past. The values of β and η are hyperparameters and have a significant impact on both the learning speed and the quality of the results. Therefore, hyperparameter optimization must be conducted to fully exploit the algorithm's potential.

We may intuitively think of the ECM as the agent's brain. Whenever a new observation is made, a corresponding percept clip with a decision tree is added to the ECM. In every interaction with the environment, the ECM updates all h and g values according to Eq. (1.42) and Eq. (1.43). It's important to note that even if the reward is zero, the network undergoes changes due to the g-values. The g-values "trace" the path to a reward, and the higher the reward, the more the path is strengthened. Intuitively, the "glow" represents the agent's ability to recall the actions that led to the reward. Each edge's relevance decreases exponentially as it moves further into the past, causing it to "glow out". The softmax parameter β regulates the agent's balance between exploration and exploitation, with a lower value resulting in more thorough exploration of the observation space.

1.5.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a cutting-edge deep RL algorithm that has recently gained considerable attention in the fields of computer science, robotics, and physics [50, 51]. Introduced by Schulman et al. (2017) [52], PPO provides a compelling balance between sample complexity, ease of implementation, and computational cost, enabling efficient deployment in practical applications [50]. In 2021, Yao et al. [37] successfully employed a slightly adapted version of PPO to predict efficient gate sequences for QAOA. Refs. [45, 51] successfully applied the PPO algorithm in order to optimize the variational QAOA parameters. Since PPO is rooted in the basic principles of policy gradient algorithms, we first provide an introduction to such algorithms.

In policy gradient algorithms, deep neural networks serve as highly expressive function approximators to help parameterize the policy π_θ with variational parameters θ. These are represented by the weights and biases of the neural network. In one learning epoch (see Fig 1.7), the parameters θ are updated via stochastic gradient ascent according to

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k}), \tag{1.44}$$

where $J(\pi_{\theta_k})$ denotes the total expected return and α the learning rate. For Vanilla Policy Gradient (VPG), the expected return $J(\pi_{\theta_k})$ (also called objective or loss) is [53, 54]

$$J^{VPG}(\pi_{\theta_k}) = \mathop{\mathbb{E}}_{\zeta \sim \pi_\theta} \left[\sum_{t=0}^{q} \log \pi_\theta(a_t|s_t) A_t^{\pi_\theta} \right]. \tag{1.45}$$

It is the expectation value of over a bunch of trajectories ζ sampled from the current policy π_θ. The advantage estimate function $A_t^{\pi_\theta}$ quantifies the advantage of choosing a particular

action a in state s over randomly selecting future actions according to the current policy π_θ. Mathematically, we obtain this value by subtracting the state-value function V^{π_θ} [55] from the action-value function Q^{π_θ}:

$$A_t^{\pi_\theta} = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t) \tag{1.46}$$

The state-value function $V^{\pi_\theta}(s)$ calculates the expected return from starting in state s and always acting on policy π_θ. The action-value function $Q^{\pi_\theta}(s, a)$ computes the expected return from starting in state s, picking an arbitrary action a and acting on the current policy π_θ afterwards. A neural network is used to parameterize both the state-value function and the policy in deep RL. The training process is sketched in Fig. 1.7. For each epoch k, the agent experiences N_{epi} trajectories with length q. Thereby, the agent follows a fixed policy π_{θ_k} and receives corresponding rewards λ. The weights of the policy and

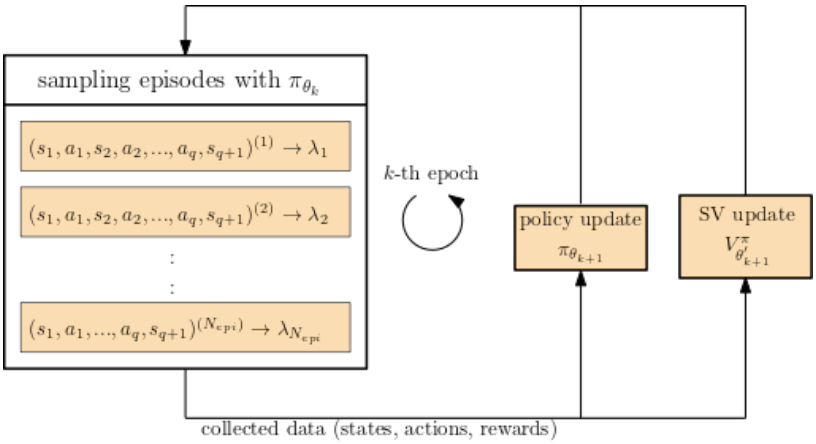


Figure 1.7: Illustration of a single learning epoch k inspired by Ref. [45]. With a fixed policy the agent experiences N_{epi} episodes and receives resulting rewards λ. The policy network π_θ and state-value network $V^{\pi}_{\theta'}$ are updated using the collected data from the preceding episodes.

state-value network are updated using gradient ascent (Eq. (1.44)) with a batch of collected data.

2017 Schulman et al. [52] proposed a novel policy gradient method known as Proximal Policy Optimization (PPO). This approach is based on Trust Region Policy Optimizing (TRPO) [56], which updates policies to maximize performance improvement while limiting the allowed deviation between new and old policies. PPO is more general, easier to implement, and provides some advantages over TRPO [52]. In PPO, the training objective is given by

$$J^{PPO}(\pi_\theta) = \mathop{\mathbb{E}}_{\zeta \sim \pi_\theta} \left[\sum_{t=0}^{q} \min(r_t(\theta) A_t^{\pi_\theta}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) A_t^{\pi_\theta}) \right] \tag{1.47}$$

with the probability ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}. \tag{1.48}$$

The probability ratio $r_t(\theta)$ measures how well the new policy performs compared to the old policy. The clipping function is defined as follows:

$$\text{clip}(x, a, b) = \begin{cases} a & \text{if } x < a \\ x & \text{if } a \leq x \leq b \\ b & \text{if } x > b \end{cases} \tag{1.49}$$

It prevents $r_t(\theta)$ from moving beyond the range of $[1 - \epsilon, 1 + \epsilon]$. We will refer to the hyperparameter ϵ as the clip parameter. The parameter ϵ therefore determines how much the new policy can deviate from the old one while still enhancing the training objective.

CHAPTER 2

Methods and Implementation

In this chapter, we will present the RL agent's learning scheme for discovering optimal gate sequences for decoded parity QAOA. As stated in Sec. 1.2.2, DC-QAOA introduces additional unitaries inspired by counterdiabatic driving. In prior research [37, 44], the generalized QAOA ansatz [37] is implemented in the context of RL.

For our calculations, we adapt the problem Hamiltonian for parity QAOA (see Sec.1.4 and Eq. (1.29)) slightly:

$$\tilde{H}_p = \sum_{i=1}^{K} \tilde{J}_i \tilde{\sigma}_z^{(i)} + C \sum_{l=1}^{K-N+1} \frac{1 - \tilde{\sigma}_z^{(l,n)} \tilde{\sigma}_z^{(l,e)} \tilde{\sigma}_z^{(l,w)} [\tilde{\sigma}_z^{(l,s)}]}{2} \tag{2.1}$$

In this form, each constraint that is violated imposes a positive energy penalty of C, and a constraint that is satisfied has no effect on the energy. In this study, the Python library Qutip [57] is utilized, and quantum states are calculated precisely.

2.1 Generalized QAOA for Reinforcement Learning

In 2019, Hadfield et al. introduced the Quantum Alternating Operator Ansatz, which is more versatile as it allows for alternating between more general families of unitaries within the QAOA. The generalized QAOA ansatz, introduced in Ref. [37], constructs a variational quantum circuit with a sequence of parameterized unitaries as follows:

$$U(\boldsymbol{\alpha}, \boldsymbol{\tau}) = \prod_{j=1}^{q} U(\alpha_j, \tau_j) = \prod_{j=1}^{q} \exp(-i\alpha_j H_{\tau_j}) \tag{2.2}$$

The unitary $U(\boldsymbol{\alpha}, \boldsymbol{\tau})$ depends on a discrete sequence of Hamiltonians $\boldsymbol{\tau} = (\tau_1, \tau_2, ..., \tau_q)$ with sequence/gate length q, as well as the continuous variables $\boldsymbol{\alpha} = \{\alpha_j\}_{j=1}^{q}$ representing the variational parameters (or angles) of the QAOA. The Hamiltonians H_{τ_j} are further selected from a fixed Hamiltonian pool $\mathcal{H} = \{H_1, H_2, ..., H_{|\mathcal{H}|}\}$. Also, with the constraint that consecutive gates are not repeated, the number of potential unitary sequences increases exponentially with the gate length q, making random search intractable [44]. Consequently, we will define an optimization problem and employ RL agents to find suitable unitary

sequences in the following section. The objective function for the generalized QAOA becomes:

$$E(\boldsymbol{\alpha}, \boldsymbol{\tau}) = \langle \psi_0 | U^\dagger(\boldsymbol{\alpha}, \boldsymbol{\tau}) \tilde{H}_p U(\boldsymbol{\alpha}, \boldsymbol{\tau}) | \psi_0 \rangle \tag{2.3}$$

We define the QAOA energy E_{qaoa} as QAOA energy resulting from a full QAOA routine operating with a fixed gate sequence $\hat{\boldsymbol{\tau}}$

$$E_{qaoa} = E(\boldsymbol{\alpha}_f, \hat{\boldsymbol{\tau}}), \tag{2.4}$$

where $\boldsymbol{\alpha}_f$ denotes the optimized QAOA angles. A QAOA routine involves performing several randomly initialized runs of the classical optimizer and selecting the best final QAOA energy E_f.

2.2 Reinforcement learning setup

2.2.1 Learning scheme

In our study the agent's goal is to learn gate sequences enhancing the performance of parity QAOA. We formalize the concept by introducing some mathematical notions. A time

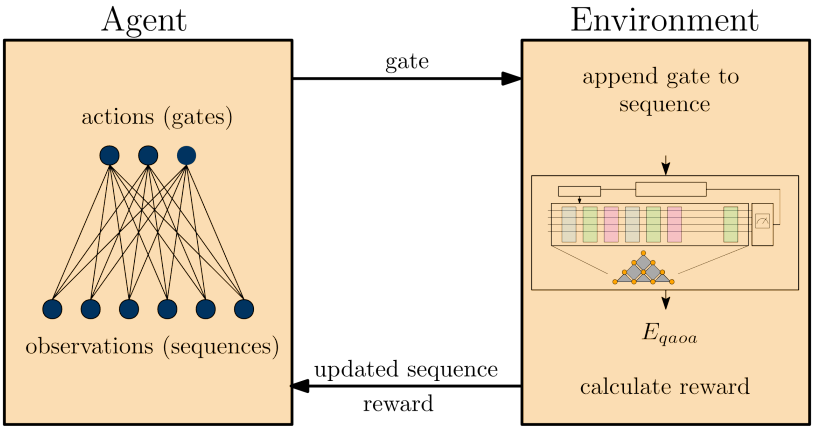


Figure 2.1: Illustration of one interaction step between the environment encapsulation the parity QAOA routine and the RL agent, visualized by a two layered neural network.

step, denoted by t, represents a single interaction between the agent and the environment within a given episode e. This interaction includes the agent choosing an action and the environment performing a QAOA routine and returning a reward (see Fig. 2.1). Following, we will explain these concepts in detail:

actions: The action space $\mathcal{A}$ consists of a fixed set of Hamiltonians. At time step j the agent has to choose one action $a_j = H_j$. Due to mathematical properties of the quantum circuit, some restrictions apply to this selection, as explained in Sec. 2.2.2.

observations: The observation space (or state space) $\mathcal{S}$ contains all possible combinations of Hamiltonians within a maximal gate length q. A single observation S_t is

the concatenation of all previous actions $a_1, ..., a_t$ up to time step t throughout one episode e:

$$S_t^{(e)} = (a_1^{(e)}, a_2^{(e)}, ..., a_t^{(e)}) \tag{2.5}$$

rewards: In RL, the design of the reward function is crucial for effective learning. Here, the reward is calculated according to

$$f(E_{qaoa}, t) = \begin{cases} -\frac{E_{qaoa}}{K} & \text{if } |S_t| = q \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

where K stands for the number of physical parity qubits. Thus, the agent receives a reward equal to the negative energy density at the end of each episode ($|S_t| = q$). This reward function is particularly suitable because no information about the physical system is required. Furthermore, the QAOA routine has to be run only at the end of each episode, significantly decreasing the computational cost.

Following Ref. [37], we formulate the optimization problem as a two-level optimization scheme, including a low-level optimization and a high-level optimization.

Low-level optimization

The low-level optimization is performed within the environment. It includes the decoded parity QAOA routine (see Sec. 1.4) optimizing the variational parameters $\boldsymbol{\alpha}$ for a fixed gate sequence $\hat{\boldsymbol{\tau}}$. Due to the complex structure of the energy landscape spanned by the variational parameters $\boldsymbol{\alpha}$, it is not possible to find the global energy minima in most cases. Therefore, several optimizations initialized with randomly chosen variational parameters uniformly distributed in the interval $[0, 2\pi]$ are performed to find a particularly low local minimum. Finally, the lowest local minimum is selected, and its energy is denoted as E_{qaoa}. For the classical parameter optimization, we employ several classical optimizers, including a Monte Carlo search and optimizers implemented in the Python package SciPy [58].

High-level optimization

The optimization of the gate sequence $\boldsymbol{\tau}$ is formulated as a RL problem. In Fig. 2.1, the basic interaction between the agent and environment is sketched. During one time step t of the learning process, the agent selects an action that corresponds to a Hamiltonian from the pool $\mathcal{H}$. The associated unitary is added to the unitary (gate) sequence, and the low-level optimization is performed in order to optimize the QAOA parameters and find an approximate energy E_{qaoa}. The reward is calculated according to Eq. (2.6). After this, the agent receives both the updated gate sequence and the reward. Using the reward as feedback, the agent adjusts its decision network. The next action is then chosen based on this updated network and the observed gate sequence. This process continues until a specified gate length q is reached, signaling the start of the next episode. One episode starts with an empty gate sequence and ends with a sequence of a predetermined gate length q.

2.2.2 Action masking

Masking invalid actions can have a positive impact on learning [59]. In our case, there are certain limitations on the possible gate sequences:

1. The first gate must not be an $\hat{X}$.

2. Equal unitaries can't be applied sequentially.

3. Following two unitaries that commute, a different unitary must be applied.

Since the initial state is the plus state $|+\rangle^{\otimes K}$ and the $\hat{X}$ stands for the unitary $\tilde{U}_x(\beta)$, which applies single qubit rotations about the x-axis of the Bloch sphere, the state remains unchanged. Selecting identical unitaries for two consecutive time steps is forbidden because the unitaries are defined by exponential functions, which can be merged into a single unitary by adding up their respective angles. The third point follows directly from this: while the commutation of two unitaries leaves the energy unchanged, it could violate the second restriction if one of the commuted unitaries were to follow immediately.

2.2.3 Applied Reinforcement Learning models

In this work, two RL models, PS and PPO, are applied. Both models are trained on the same environment, which includes the low-level optimization and reward function (see Sec. 2.2.1). The action masking, as explained in Sec. 2.2.2, is also applied in both models. In Sec. 3.3.5, we compare the performance of PS and PPO with a basic random search. Following, we provide details on the implementation of both algorithms.

Projective Simulation

The PS algorithm was coded from scratch, implementing the theory explained in Section 1.5.1. We used a simple ECM, which includes two layered, all-to-all connected decision trees. The transition probability is given by the softmax function (Eq. (1.40)) and the weights are updated according to Eq. (1.42) and Eq. (1.43). The PS results are presented in Sec. 3.3.3.

Proximal Policy Optimization

For the implementation of PPO (see Sec 1.5.2), we adopt the Stable Baselines3 (SB3) module, a high-quality open-source implementation of PPO and other RL algorithms [60]. The module "MaskablePPO" from SB3 is used, employing the corresponding policy called "MaskableActorCriticPolicy", initialized with default parameters (Appendix E). We analyze and discuss the PPO results in Sec. 3.3.4.

2.3 Performance measure

For approximation algorithms, the quality of the solution is usually quantified with the approximation ratio r, as introduced in Sec. 1.1. We evaluate the quality of the resulting QAOA energy after a low-level optimization using the approximation ratio

$$r = E_{qaoa}/E_{GS}, \tag{2.7}$$

where E_{GS} stands for the ground state energy. For decoded parity QAOA with final mean decoding, which is applied during this research, the expectation value $\langle\psi_0| \tilde{H}_p |\psi_0\rangle$ of the initial state $|\psi_0\rangle = |+\rangle^{\otimes K}$ has an approximation ratio of $r = 0$.

CHAPTER 3

Learning optimal parity QAOA gate sequences for 10 qubit systems

This work aims to identify optimal gate sequences, improving the performance of QAOA on parity architectures. For this purpose, we focus on a system of 10 parity qubits representing an all-to-all connected Ising spin glass with 5 logical qubits. Due to long simulation times, it was not possible to increase the system size further. We also present results for a system with 6 physical qubits in Appendix C. We start with the identification of hard problem instances for the RL simulations. Next, to enhance QAOA's performance even further and enlarge the action space of the RL agents we incorporate additional unitaries, inspired by counterdiabatic driving. Decoded parity QAOA is employed as it has been shown to improve the performance of QAOA [9]. Furthermore, a comparison of several classical optimizers is presented. Having decided on a particular hard 10 qubit problem instance, we continue to employ PS and PPO agents to identify improved QAOA gate sequences. Furthermore, the performance of RL learning is compared to a simple random search. Finally, we discuss a gate sequence pattern found by RL and compare the parity QAOA performance utilizing this pattern on several nontrivial problem instances.

3.1 10 qubits parity architecture

Here, we describe the selection of 10 qubit problem instances and introduce important definitions and notations. For the purpose of consistency and understanding, the qubit numbering of the problem configuration is depicted in Fig. 3.1. The interactions $\tilde{J}_i$ are ex-

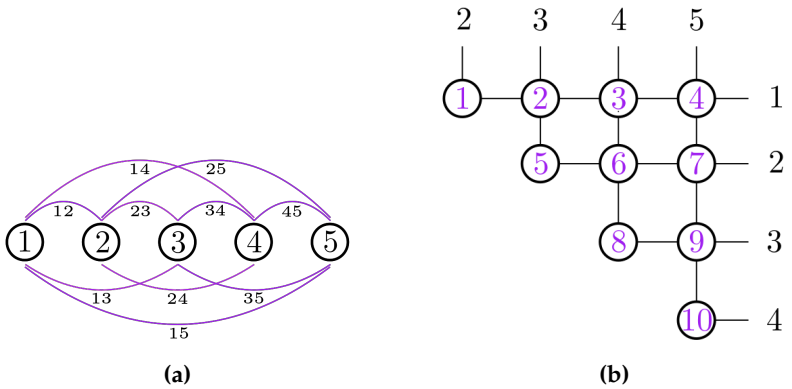


Figure 3.1: Panel (a) illustrates a basic layout of five logical qubits, numbered from 1 through 5. Interactions between them are indicated by purple lines. Panel (b) represents the corresponding physical system. Each combination of logical qubits, marked with black numbers, corresponds to a physical qubit. These physical qubits are labeled with purple numbers. There are 10 physical qubits, numbered from 1 to 10.

pressed in a simple array form, $\tilde{J} = [J_1, J_2, ..., J_K]$, where K is the total number of physical qubits. A problem instance is completely described by its interaction array $\tilde{J}_i$. We call an instance trivial if the spins can be aligned with the local fields without violating any parity constraints. In such a case, the QAOA can easily solve the problem with just two layers of single qubit gates (i.e. the gate sequence $\hat{Z}\hat{X}$). Stated mathematically, a problem instance governed by the problem Hamiltonian $\tilde{H}_p = \tilde{H}_z + \tilde{H}_c$ is considered trivial if the groundstate $|\psi_{GS}\rangle$ minimizes each term in the problem Hamiltonian $\tilde{H}_p$ ($\langle\psi_{GS}|\,\tilde{H}_z\,|\psi_{GS}\rangle = E_{min}$ and $\langle\psi_{GS}|\,\tilde{H}_c\,|\psi_{GS}\rangle = 0$). As a measure of frustration, we assign the variable v to the number of violated parity constraints for the spin configuration, minimizing $\tilde{H}_z$. In the following, we only consider local fields in the set $\tilde{J}_i \in \{-1, 1\}$. We keep only one instance from a group of equivalent problem instances resulting from certain spin flips, as described in Appendix A. Specifically, these instances are those where the spin configuration aligned with the local fields violates the same parity constraints. Additionally, we keep only one instance from the group of reflection-symmetric instances in terms of the violated parity constraints, as sketched in Fig. 3.2. It shows two exemplary problem instances with 5 violated parity constraints each. These instances are equivalent due to the reflective symmetry in the violated parity constraints around the axis connecting the third and fifth parity qubit. What remains are 54 unique problem instances. Later in Sec. 3.3.1, we will further reduce the set of problem instances in order to find a hard problem instance suitable for the RL simulations.

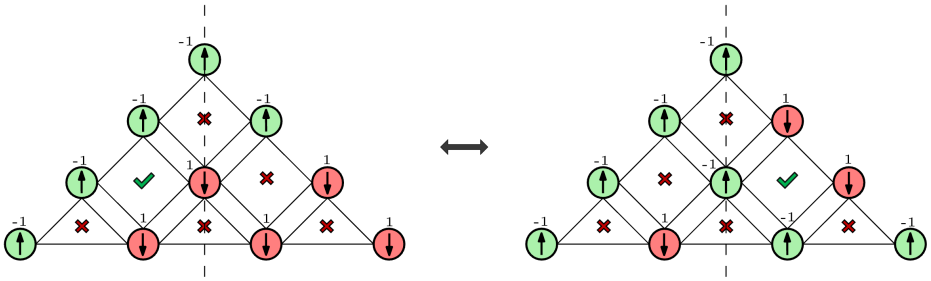


Figure 3.2: Sketch of the parity triangle of two equivalent 10 qubit problem instances with these local fields: $\tilde{J} = [-1, -1, -1, -1, 1, 1, -1, 1, 1, 1]$ (left) and $\tilde{J} = [-1, -1, -1, -1, 1, -1, 1, -1, 1, -1]$ (right). The physical qubit configuration minimizing $\tilde{H}_z$ for both instances is indicated by arrows, respectively. Violated parity constraints are indicated with red crosses and satisfied ones with green ticks. A parity constraint is satisfied, if the number of down spins is even. The dashed lines illustrate the parity triangle's reflection symmetry.

3.2 Parity QAOA setup

In this section, we present gate pools available for decoded parity QAOA and the RL agents. There is evidence that additional gates could increase the performance of QAOA [24, 37]. Furthermore, several classical optimizers for the parity QAOA routine are compared. For the best performing optimizer, we also analyze the accuracy of the results for an increasing number of random QAOA angle initializations.

3.2.1 Gate pools and relevant gate sequences

In this section, we present three Hamiltonian pools $\mathcal{H}$ and the corresponding QAOA unitaries for the RL simulations. As discussed in Sec. 2, we apply the generalized QAOA ansatz (see Sec. 2.1). In this work, we approximate sums of non-commuting operators in the exponent with the first order of the Trotter product formula ($e^{A+B} \approx e^A e^B$) [61]. Considering terms up to the second order of Eq. (1.25), along with the mixer Hamiltonian $\tilde{H}_x$, the local field Hamiltonian $\tilde{H}_z$, and the constraint Hamiltonian $\tilde{H}_c$ the following Hamiltonian pools $\mathcal{H}$ arise:

$$
\begin{aligned}
\mathcal{H}_0 &= \{\tilde{H}_x, \tilde{H}_z, \tilde{H}_c\} \\
\mathcal{H}_1 &= \{\tilde{H}_x, \tilde{H}_z, \tilde{H}_c, [\tilde{H}_x, \tilde{H}_z], [\tilde{H}_x, \tilde{H}_c]\} \\
\mathcal{H}_2 &= \{\tilde{H}_x, \tilde{H}_z, \tilde{H}_c, [\tilde{H}_x, \tilde{H}_z], [\tilde{H}_x, \tilde{H}_c], \\
&\quad [\tilde{H}_x, [\tilde{H}_x, [\tilde{H}_x, \tilde{H}_c]]], [\tilde{H}_c, [\tilde{H}_c, [\tilde{H}_x, \tilde{H}_c]]], \\
&\quad [\tilde{H}_z, [\tilde{H}_x, [\tilde{H}_x, \tilde{H}_c]]], [\tilde{H}_x, [\tilde{H}_c, [\tilde{H}_x, \tilde{H}_c]]], \\
&\quad [\tilde{H}_z, [\tilde{H}_c, [\tilde{H}_x, \tilde{H}_c]]]\}
\end{aligned}
$$

Continuing, we work with the following three gate pools corresponding to the Hamiltonian pools $\mathcal{H}_0, \mathcal{H}_1$, and $\mathcal{H}_2$:

$$
\begin{aligned}
l = 0 &: \{\hat{\mathrm{X}}, \hat{\mathrm{Z}}, \hat{\mathrm{C}}\} \\
l = 1 &: \{\hat{\mathrm{X}}, \hat{\mathrm{Z}}, \hat{\mathrm{C}}, \hat{\mathrm{U}}_1, \hat{\mathrm{U}}_2\} \\
l = 2 &: \{\hat{\mathrm{X}}, \hat{\mathrm{Z}}, \hat{\mathrm{C}}, \hat{\mathrm{U}}_1, \hat{\mathrm{U}}_2, \hat{\mathrm{U}}_3, \hat{\mathrm{U}}_4, \hat{\mathrm{U}}_5, \hat{\mathrm{U}}_6, \hat{\mathrm{U}}_7\}
\end{aligned}
$$

Here, l donates the order of the expansion from the adiabatic gauge potential (Eq. (1.25)). Accordingly, the gates have the following unitary representations:

$$\begin{aligned}
\hat{\mathrm{X}} : \tilde{U}_x(\beta) &= \mathrm{e}^{-i\beta\tilde{H}_x} \\
\hat{\mathrm{Z}} : \tilde{U}_z(\gamma) &= \mathrm{e}^{-i\gamma\tilde{H}_z} \\
\hat{\mathrm{C}} : \tilde{U}_c(\Omega) &= \mathrm{e}^{-i\Omega\tilde{H}_c} \\
\hat{\mathrm{U}}_1 : \tilde{U}_1(\alpha_1) &= \mathrm{e}^{-\alpha_1[\tilde{H}_x,\tilde{H}_z]} \\
\hat{\mathrm{U}}_2 : \tilde{U}_2(\alpha_2) &= \mathrm{e}^{-\alpha_2[\tilde{H}_x,\tilde{H}_c]} \\
\hat{\mathrm{U}}_3 : \tilde{U}_3(\alpha_3) &= \mathrm{e}^{-\alpha_3[\tilde{H}_x,[\tilde{H}_x,[\tilde{H}_x,\tilde{H}_c]]]} \\
\hat{\mathrm{U}}_4 : \tilde{U}_4(\alpha_4) &= \mathrm{e}^{-\alpha_4[\tilde{H}_c,[\tilde{H}_c,[\tilde{H}_x,\tilde{H}_c]]]} \\
\hat{\mathrm{U}}_5 : \tilde{U}_5(\alpha_5) &= \mathrm{e}^{-\alpha_5[\tilde{H}_z,[\tilde{H}_x,[\tilde{H}_x,\tilde{H}_c]]]} \\
\hat{\mathrm{U}}_6 : \tilde{U}_6(\alpha_6) &= \mathrm{e}^{-\alpha_6[\tilde{H}_x,[\tilde{H}_c,[\tilde{H}_x,\tilde{H}_c]]]} \\
\hat{\mathrm{U}}_7 : \tilde{U}_7(\alpha_7) &= \mathrm{e}^{-\alpha_7[\tilde{H}_z,[\tilde{H}_c,[\tilde{H}_x,\tilde{H}_c]]]}
\end{aligned}$$

When talking about gates, we neglect the angles and refer to the additional gates as $\hat{\mathrm{U}}_1$ through $\hat{\mathrm{U}}_7$. As previously stated, in this work we only consider local fields from the set $\tilde{J}_i \in \{-1, 1\}$. Consequently, Eq. (1.25) simplifies, and only the given Hamiltonians can be derived. In Appendix G, we discuss the implementation of the considered unitaries.

As noted in Ref. [38], the initial parity QAOA implementation employed the gate sequence $\hat{\mathrm{Z}}\hat{\mathrm{C}}\hat{\mathrm{X}}\hat{\mathrm{Z}}\hat{\mathrm{C}}\hat{\mathrm{X}}...\hat{\mathrm{Z}}\hat{\mathrm{C}}\hat{\mathrm{X}}$, which we introduced in Sec. 1.4, and is thus referred to as the "standard gate sequence". Meanwhile, the gate sequence $\hat{\mathrm{Z}}\hat{\mathrm{X}}\hat{\mathrm{Z}}\hat{\mathrm{X}}...\hat{\mathrm{Z}}\hat{\mathrm{X}}$ is referred to as the "trivial gate sequence", as it only involves single qubit rotation gates. Quantum algorithms that only use single qubit gates can be simulated efficiently classically [62]. Therefore, they cannot provide quantum advantage.

3.2.2 QAOA angle optimization

Throughout this work, we employ decoded parity QAOA with final mean decoding, as explained in Sec. 1.4.1. Decoded parity QAOA with final min decoding is already able to find the groundstate energy for non-trivial 6 and 10 qubit instances with the standard gate sequence (see Appendix B). In our effort to find a good balance between speed and quality of results, we compare different classical optimizers for decoded parity QAOA with final mean decoding. The aim is to identify an optimizer that can yield a high approximation ratio, denoted as r, in a short amount of time. We utilize the following classical optimizers that are available in the Scipy library [58]:

- Broyden-Fletcher-Goldfarb-Shanno (BFGS): Iterative method for solving unconstrained nonlinear optimization problems.

- Limited-memory BFGS with box constraints (L-BFGS-B): Optimization algorithm in the family of quasi-Newton methods that approximates the Broyden-Fletcher-Goldfarb-Shanno algorithm using a limited amount of computer memory.
- Constrained Optimization BY Linear Approximations (COBYLA): Numerical optimization method for constrained problems where the derivative of the objective function is not known.
- Sequential Least Squares Programming (SLSQP): Numerical optimization algorithm for constrained problems where both the objective function and the constraints are differentiable.

We tested all four SciPy optimizers using their default parameters, as well as the Monte Carlo algorithm with a maximum of 200 total iterations. We examine a non-trivial 10 qubit problem instance governed by the following local fields $\tilde{J} = [-1,-1,-1,-1,1,-1,1,-1,1,-1]$, which is shown in Fig. 3.6. The mean and

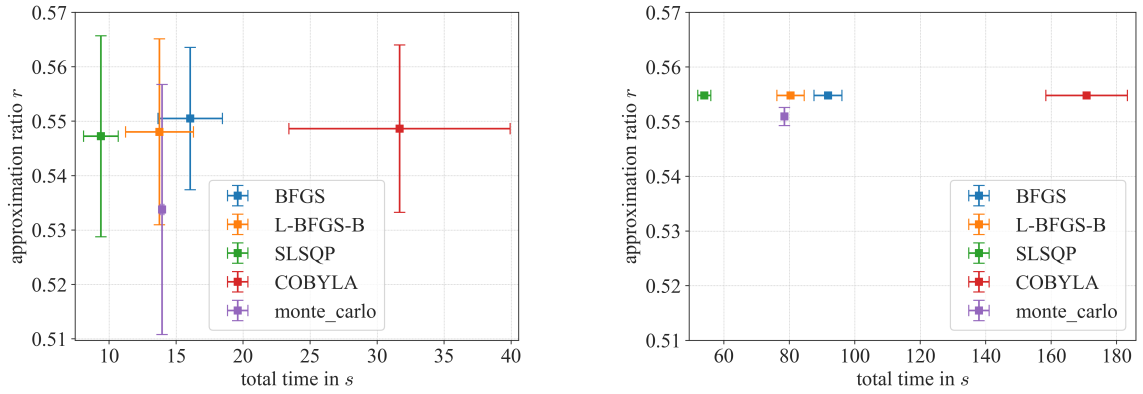


Figure 3.3: Scatterplots of the mean approximation ratio r and the total execution time of QAOA experiments with 10 (a) and 100 (b) random initializations for various classical optimizers. The considered 10 qubit problem is presented in Fig. 3.6. The errorbars indicate the standard deviation from 100 simulations. Decoded parity QAOA with mean decoding is used. QAOA operates with the standard gate sequence of length $q = 6$.

standard deviation of the approximation ratio r and the total execution time, t, across 100 executions are displayed in Fig. 3.3. Here, a single execution describes the process of a specified number of classical optimizer cycles, each initialized with randomly selected QAOA angles uniformly distributed in the range of $[0, 2\pi]$. The approximation ratio r corresponds to the highest obtained from 10 (a) or 100 (b) randomly initialized QAOA cycles and the time represents the total time of these cycles.

Timewise, the SLSQP solver outperforms all other optimizers. All Scipy optimizers achieve similar approximation ratios, while the Monte Carlo optimizer performs slightly worse. In the rest of the thesis, we employ the SLSQP solver as it is the fastest optimizer.

Moving further, we analyze the performance of the SLSQP optimizer using varying numbers of random initializations for the discussed problem instance. As can be seen in Fig. 3.4, the approximation ratio r increases slightly with an increasing number of random initializations of the initial QAOA angles. This is expected because the optimizer explores the

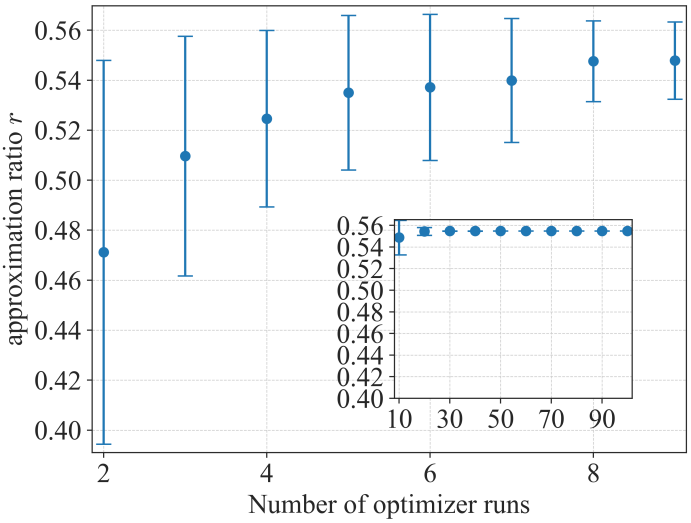


Figure 3.4: Dependence of the mean approximation ratio r on the number of randomly initialized optimizer cycles. The errorbars indicate the standard deviation from 100 simulation. Decoded parity QAOA with mean decoding operating with the standard gate sequence of length $q = 6$ is employed.

energy landscape spanned by the QAOA angles more thoroughly. The inset presents the approximation ratio for a higher number of random initializations in the range of 10 to 100. We observe a decrease in the standard deviation, indicating a higher level of precision in the results. However, the computation time increases linearly with the number of random initializations. Therefore, it is necessary to strike a balance between the accuracy and quality of the results and the computation time by carefully selecting the number of initializations.

3.3 Learning optimal gate sequences

In this section, our primary focus will be on 10 qubit problem instances derived from the set of instances discussed in Sec. 3.1. For the QAOA subroutine, we will employ decoded parity QAOA with the final mean decoding strategy (see. Sec. 1.4.1). The overall goal is to train both RL agents (PS and PPO) to find gate sequences which enhance the performance of the QAOA algorithm. As a strategy to lessen computation time, the QAOA energy E_{qaoa}, resulting from a specific gate sequence, is stored and not recalculated should the agent encounter the sequence again. Both RL agents deal with the same environment, encapsulating the update of the observation, the parity QAOA subroutine, and the reward function.

3.3.1 Hard problem instances

Here, we reduce the set of 54 unique problem instances (see. Sec. 3.1) further to such instances for which QAOA, employing the standard gate sequence, does not already find the ground state with energy E_{GS}. Additionally, it is required that the standard gate sequence outperforms the trivial gate sequence. Fig. 3.5 presents the approximation ratio r for all

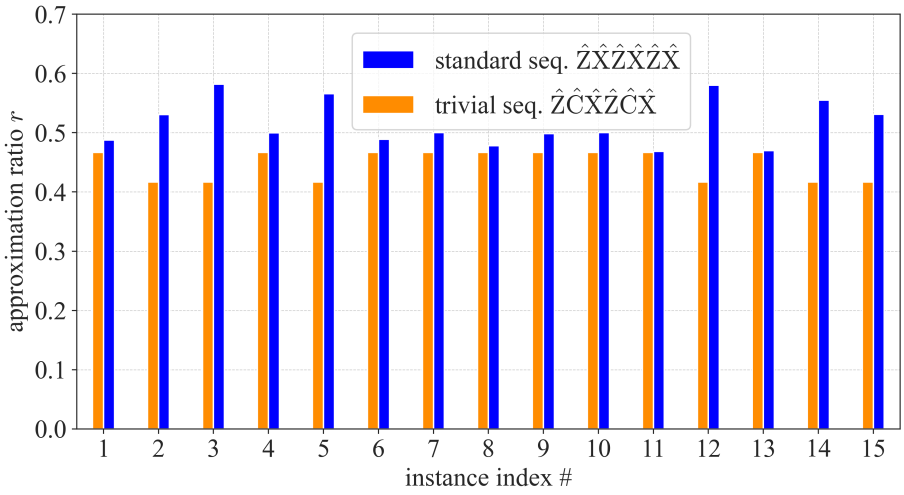


Figure 3.5: Histograms of the approximation ratios r resulting from simulations of all selected unique 10 qubit problem instances with local fields in the set $\tilde{J}_i \in \{-1, 1\}$ for the standard gate sequence and the trivial gate sequence of gate length $q = 6$. Decoded parity QAOA with mean decoding with 100 random initializations is employed.

of these unique 10 qubit problem instances. The local fields with the corresponding number of violated constraints can be found in Fig. D.1 in Appendix D. We choose problem instance number 14 with $\tilde{J} = [-1, -1, -1, -1, 1, -1, 1, -1, 1, -1]$ for the RL simulations because it is the instance with the largest energy difference between the standard and trivial gate sequence from the two instances with the most frustrated constraints ($v = 5$). Fig. 3.6 illustrates the corresponding parity triangle. Similar simulations are performed for the 6 qubit instances in Appendix C, and an hard problem instance for the RL learning on 6

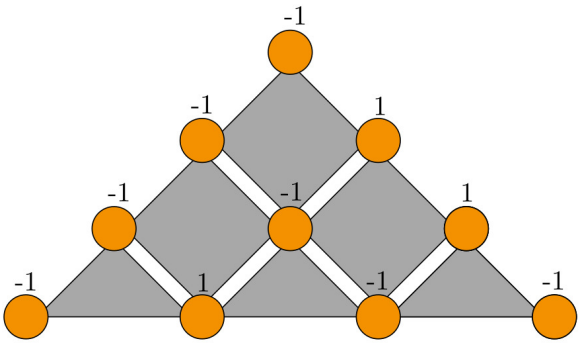


Figure 3.6: Sketch of the considered 10 qubits instance of the parity triangle with local fields of the chosen problem instance. The orange dots represent the physical qubits and the gray areas the parity constraints. The integer numbers represent the local field value assigned to each qubit.

qubits is chosen.

3.3.2 Energy landscape

Keeping in mind that the overall goal of this work is to find gate sequences for parity QAOA to enhance its performance, in the following, we present energy landscape plots that show the performance of certain gate sequences with length $q = 6$ on the chosen 10 qubit problem instance with local fields $\tilde{J} = [-1, -1, -1, -1, 1, -1, 1, -1, 1, -1]$ (see Fig. 3.6). We consider the gate pool for $l = 1$ following Sec. 3.2.1. Furthermore, only

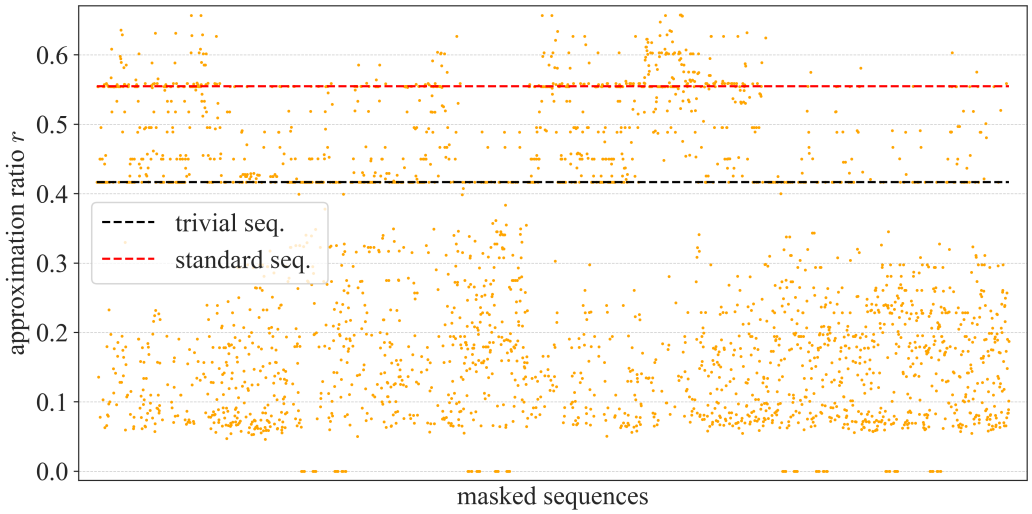


Figure 3.7: Approximation ratio r for all masked gate sequences with a gate length of $q = 6$ resulting from the gate pool $\{\hat{X}, \hat{Z}, \hat{C}, \hat{U}_1, \hat{U}_2\}$ ($l = 1$). Decoded parity QAOA with mean decoding and 10 random initializations is employed. The approximation ratios resulting from the standard gate sequence ($\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$) and the trivial gate sequence ($\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}\hat{X}$) with 100 random initializations are indicated with dashed lines.

gate sequences following the three requirements presented in Sec. 2.2.2 are displayed. Additionally, equivalent gate sequences under the commutation of $\hat{C}$ and $\hat{Z}$ are not taken into account. Accordingly, there are 3105 gate sequences left, for which Fig. 3.7 presents the

approximation ratio r resulting from (mean) decoded parity QAOA simulations with 10 random initializations. The performance of the trivial gate sequence and the standard gate sequence are represented by red dashed lines. The results show that the standard gate sequence outperforms the trivial gate sequence, as anticipated. Furthermore, multiple gate sequences perform better than the standard gate sequence, which is desirable for the RL simulations. For $l = 2$, there are 501760 masked gate sequences, which renders obtaining the landscape too computationally expensive.

3.3.3 Projective Simulation

In this section, learning results of PS agents, as introduced in Sec. 1.5.1, are presented. A simple, two-layered decision tree guides the PS agents. Each observation clip, which represents a gate sequence, is linked to the action clips via edges that hold g and h-values. The action clips correspond to the gates that can be added to the current gate sequence according to the action masking restrictions (see. Sec. 2.2.2). With the goal of comparing the performance of the three gate pools corresponding to $l = 0$, $l = 1$, and $l = 2$ (see. Sec. 3.2.1), we perform a hyperparameter optimization on the case of $l = 1$. We find that the hyperparameter combination with glow damping parameter $\eta = 0.2$ and softmax parameter $\beta = 0.5$ leads to the overall best performance for this particular case (see. Appendix E, Fig. E.1). Applying these hyperparameters, we train the PS agents on the three gate pools mentioned. The results are presented in Fig. 3.8. We observe that only the agents operating

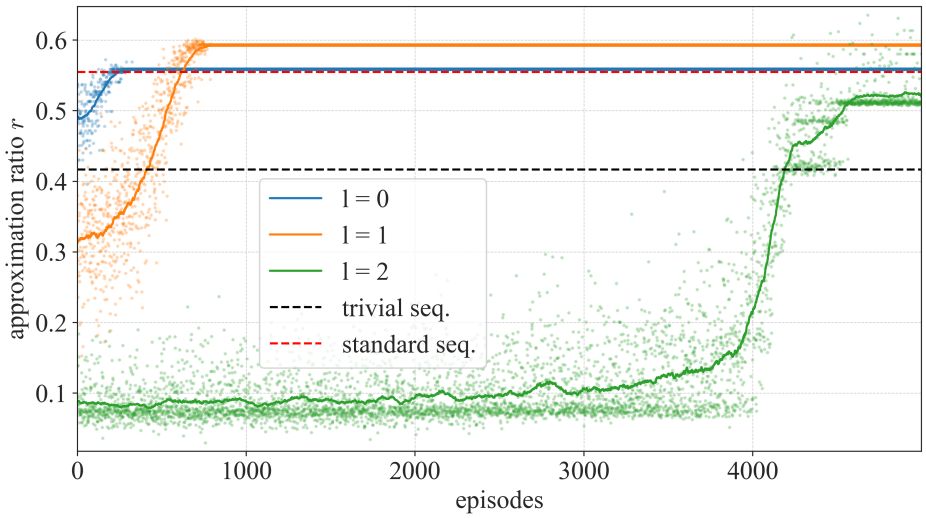


Figure 3.8: Learning curves for various gate pools each averaged over 10 PS agents for $l = 0$ and $l = 1$ and 5 PS agents for $l = 2$. We consider the chosen 10 qubit problem instance (Fig. 3.6) with a maximal gate length of $q = 6$. The solid lines represent the running average of the mean approximation ratio r, donated by points. Decoded parity QAOA with mean decoding and 10 random initializations is employed. The approximation ratios resulting from the standard gate sequence ($\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$) and the trivial gate sequence ($\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}\hat{X}$) with 100 random initializations are indicated with dashed lines.

with the gate pool for $l = 1$ are capable of finding gate sequences with a significantly higher

approximation ratio r than the standard gate sequence for the considered hyperparameter combination. Furthermore, we see that for $l = 2$, the agents are starting to learn late compared to the other agents. There are two potential reasons for this. The hyperparameters found may not be transferable, requiring hyperparameter optimization for individual setups. This is computationally quite expensive and not desired. Additionally, the expanded observation space for $l = 2$ significantly increases the difficulty of the learning process.

Learned gate sequences

As we continue, we list the found gate sequences in Tab. 3.1 for $l = 0$ (a), $l = 1$ (b), and $l = 2$ (c) after 5000 episodes and the corresponding approximation ratio r for the discussed 10 qubit instance (Sec. 3.3.1) in Fig. 3.1. For the discovered gate sequences, we perform QAOA simulations with 100 random initializations and denote the best obtained approximation ratio with r_{100}. This simulation is repeated 10 times and the resulting standard deviation is provided. We merge gate sequences that are equivalent under the commutation of $\hat{C}$ and $\hat{Z}$. The standard gate sequence $\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$ is found by three agents for $l = 0$. For this case, the gate sequence $\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}$ obtains the overall best approximation ratio, but was found just once. For $l = 1$, every agent found a different gate sequence, most of them outperforming the standard gate sequence. We could not observe a comprehensive pattern. Gate sequence $\hat{U}_1\hat{C}\hat{X}\hat{U}_1\hat{C}\hat{U}_1$ exhibits the highest approximation ratio r_{100} among all gate pools. For the last two gate sequences of Tab. 3.1c, the approximation ratio after 100 random initializations is lower than the approximation ratio found by the PS agent with only 10 random initializations. This can be attributed to a large standard deviation in the approximation ratio for gate sequences with additional gates from pool $l = 2$, which is also evident in Table 3.3d.

3.3.4 Proximal Policy Optimization

In this section, we present results obtained by PPO agents, as introduced in Sec. 1.5.2. The module "MaskablePPO" from Stable Baselines3 (SB3) is used, employing the corresponding policy called "MaskableActorCriticPolicy" [60]. Between the policy updates, 50 environment steps are performed. This choice is justified in Appendix E with a hyperparameter optimization showcased in Fig. E.3. All the other hyperparameters are set to their default values (see. Appendix E, Tab. E.1). We compare the average learning curves of 10 agents each for the case with no additional unitaries ($l = 0$) with the set of first ($l = 1$) and second-order ($l = 2$) unitaries (see Sec. 3.2.1). Each parity QAOA subroutine is randomly initialized 10 times, the best result is chosen, and its approximation ratio is denoted as r. Fig. 3.9 shows the evolution of the approximation ratio over 2000 episodes. As a benchmark, the performance of decoded parity QAOA employing the standard and trivial gate sequence with 100 initializations is drawn. For $l = 0$, there is a slight performance improvement compared to the standard gate sequence. The agents using the action space resulting from first-order terms with $l = 1$ find gate sequences which outperform the stan-

Gate sequence	r_{final}	count	r_{100}
$\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$	0.555 ± 0.000	3	0.555 ± 0.0
$\hat{Z}\hat{X}\hat{C}\hat{Z}\hat{X}\hat{C}$	0.554 ± 0.000	2	0.554 ± 0.0
$\hat{Z}\hat{X}\hat{C}\hat{X}\hat{Z}\hat{C}$	0.554 ± 0.000	2	0.554 ± 0.0
$\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}$	0.597	1	0.652 ± 0.014
$\hat{Z}\hat{X}\hat{Z}\hat{C}\hat{X}\hat{Z}$	0.554	1	0.554 ± 0.0
$\hat{Z}\hat{C}\hat{X}\hat{C}\hat{X}\hat{Z}$	0.554	1	0.555 ± 0.0
standard seq. $\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$	-	-	0.555

(a)

Gate sequence	r_{final}	count	r_{100}
$\hat{U}_1\hat{X}\hat{C}\hat{U}_1\hat{C}\hat{U}_1$	0.657	1	0.657 ± 0.0
$\hat{U}_1\hat{Z}\hat{X}\hat{C}\hat{Z}\hat{U}_1$	0.554	1	0.554 ± 0.0
$\hat{U}_1\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}$	0.594	1	0.657 ± 0.0
$\hat{U}_1\hat{C}\hat{X}\hat{C}\hat{X}\hat{U}_2$	0.545	1	0.606 ± 0.008
$\hat{U}_1\hat{X}\hat{C}\hat{Z}\hat{X}\hat{C}$	0.554	1	0.554 ± 0.0
$\hat{U}_1\hat{Z}\hat{C}\hat{X}\hat{C}\hat{U}_2$	0.590	1	0.636 ± 0.014
$\hat{U}_1\hat{C}\hat{Z}\hat{U}_1\hat{C}\hat{U}_1$	0.657	1	0.657 ± 0.0
$\hat{U}_1\hat{C}\hat{X}\hat{C}\hat{U}_2\hat{C}$	0.585	1	0.613 ± 0.014
$\hat{Z}\hat{U}_1\hat{C}\hat{X}\hat{C}\hat{X}$	0.594	1	0.657 ± 0.0
$\hat{U}_1\hat{C}\hat{U}_1\hat{Z}\hat{U}_2\hat{C}$	0.602	1	0.602 ± 0.0

(b)

Gate sequence	r_{final}	count	r_{100}
$\hat{U}_1\hat{C}\hat{X}\hat{U}_1\hat{C}\hat{U}_1$	0.661	1	0.661 ± 0.000
$\hat{U}_1\hat{Z}\hat{X}\hat{Z}\hat{C}\hat{X}$	0.554	1	0.554 ± 0.000
$\hat{U}_1\hat{Z}\hat{C}\hat{U}_1\hat{C}\hat{Z}$	0.554	1	0.554 ± 0.000
$\hat{U}_1\hat{X}\hat{Z}\hat{C}\hat{X}\hat{U}_4$	0.715	1	0.611 ± 0.131
$\hat{U}_1\hat{X}\hat{U}_1\hat{Z}\hat{X}\hat{U}_7$	0.417	1	0.583 ± 0.000

(c)

Table 3.1: Found gate sequences for the 10 qubit problem instance after 5000 episodes of 10 PS agents for $l = 0$ and $l = 1$ and 5 PS agents for $l = 2$ with the corresponding approximation ratio r_{final} for a maximal gate length of $q = 6$. Equivalent gate sequences under the commutation of $\hat{C}$ and $\hat{Z}$ are merged. If multiple agents found the same gate sequence, the count is provided, and the mean and standard deviation of the approximation ratio r_{final} are shown. Tab. (a) corresponds to the case of $l = 0$, Tab. (b) to $l = 1$, and Tab. (c) to $l = 2$. r_{100} stands for the mean approximation ratio resulting from repeated QAOA simulations for each gate sequence with 100 random initializations. The standard deviation is derived from 10 executions of such 100 QAOA repetitions.

dard gate sequence significantly. With the action space including also gates from the gate pool for $l = 2$, the agent's performance is worse again, apparently due to the large observation space (526.513 sequences). Next, the discovered gate sequences are presented for

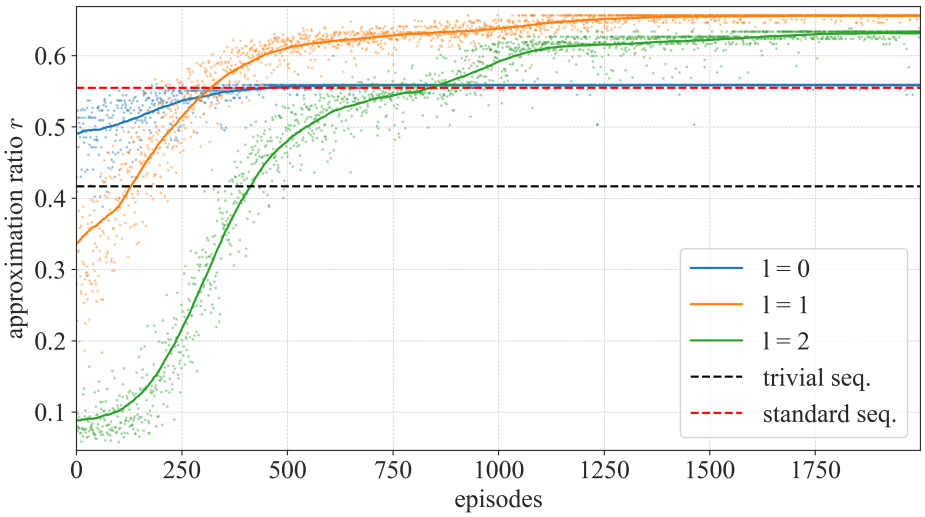


Figure 3.9: Learning curves for various gate pools each averaged over 10 PPO agents for the chosen 10 qubit problem instance (Fig. 3.6 with a maximal gate length of $q = 6$. The solid lines represent the running average of the mean approximation ratio r, donated by points. Decoded parity QAOA with mean decoding and 10 random initializations is employed. The approximation ratios resulting from the standard gate sequence ($\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$) and the trivial gate sequence ($\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}\hat{X}$) with 100 random initializations are indicated with dashed lines.

each gate pool.

Learned gate sequences

Tab. 3.2 shows the learned gate sequences and the corresponding approximation ratio r after 2500 RL learning episodes. For the discovered gate sequences, we perform QAOA simulations with 100 random initializations and denote the best obtained approximation ratio with r_{100}. This simulation is repeated 10 times, and the resulting standard deviation is provided. For $l = 0$, the standard gate sequence is found most frequently by 7 PS agents. For $l = 1$ and $l = 2$, the most frequently found sequence is

$$\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1.... \tag{3.1}$$

We will analyze this particular gate sequence in detail later. Interestingly, also for the 6 qubit instance and a maximal gate length of $q = 9$, the most frequently discovered gate sequence has the same pattern (see. Appendix C, Fig. C.2).

3.3.5 Random search vs. Reinforcement Learning

In this section, we present the results of a basic random search and compare the performance of random search and RL on the discovered 10 qubit problem instance for the gate pool $l = 2$. We define a quantum call as one execution of the QAOA routine, which includes several optimizations of the variational QAOA parameters. The number of random initializations of the QAOA angles is fixed. During the random search, we utilize decoded

Gate sequence	r_{final}	count	r_{100}
$\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$	0.555 ± 0.0	7	0.555 ± 0.0
$\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}$	0.604 ± 0.006	2	0.657 ± 0.0
$\hat{C}\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}$	0.555	1	0.555 ± 0.0
standard seq. $\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$	-	-	0.555 ± 0.0

(a)

Gate sequence	r_{final}	count	r_{100}
$\hat{Z}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$	0.657	1	0.657 ± 0.0
$\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}$	0.656 ± 0.0	3	0.656 ± 0.0
$\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$	0.656 ± 0.0	2	0.656 ± 0.0
$\hat{U}_1\hat{C}\hat{Z}\hat{U}_1\hat{C}\hat{U}_1$	0.657 ± 0.0	2	0.657 ± 0.019
$\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{X}$	0.658 ± 0.0	2	0.658 ± 0.0

(b)

Gate sequence	r_{final}	count	r_{100}
$\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}$	0.638 ± 0.031	4	0.656 ± 0.0
$\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{Z}$	0.656 ± 0.0	3	0.656 ± 0.0
$\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$	0.606 ± 0.036	3	0.656 ± 0.0

(c)

Table 3.2: Found gate sequences for the 10 qubit problem instance after 2500 episodes of 10 PPO agents with the corresponding approximation ratio r_{final} for a maximal gate length of $q = 6$. Equivalent gate sequences under the commutation of $\hat{C}$ and $\hat{Z}$ are merged. If multiple agents found the same gate sequence, the count is provided, and the mean and standard deviation of the approximation ratio r_{final} are shown. Tab. (a) corresponds to the case of $l = 0$, Tab. (b) to $l = 1$, and Tab. (c) to $l = 2$. r_{100} stands for the mean approximation ratio resulting from repeated QAOA simulations for each gate sequence with 100 random initializations. The standard deviation is derived from 10 executions of such 100 QAOA repetitions.

parity QAOA with final mean decoding and 10 random initializations using a randomly picked gate sequence. We keep track of the best resulting energy as E_{min} and repeat the process 500 times. Whenever QAOA detects a gate sequence with lower energy, we update the minimum energy E_{min} accordingly. Additionally, any previously discovered gate sequence cannot be picked again. The resulting curves for the three gate pools resulting from $l = 0, l = 1$, and $l = 2$ are displayed in Fig. 3.10. Due to the increasing number of possible gate sequences, the curve converges first for $l = 0$ and lastly for $l = 2$. Interestingly, after 500 quantum calls, all curves achieve matching approximation ratio values within the error.

Next, we compare the performance of RL to the random search for $l = 2$. To do this, we adapt the RL results as follows: whenever the agent discovers a new gate sequence with improved energy, this energy is stored as E_{min}. The approximation ratio of the minimal energy $r_{min} = E_{min}/E_{GS}$ averaged over 10 executions for each random search, PPO and

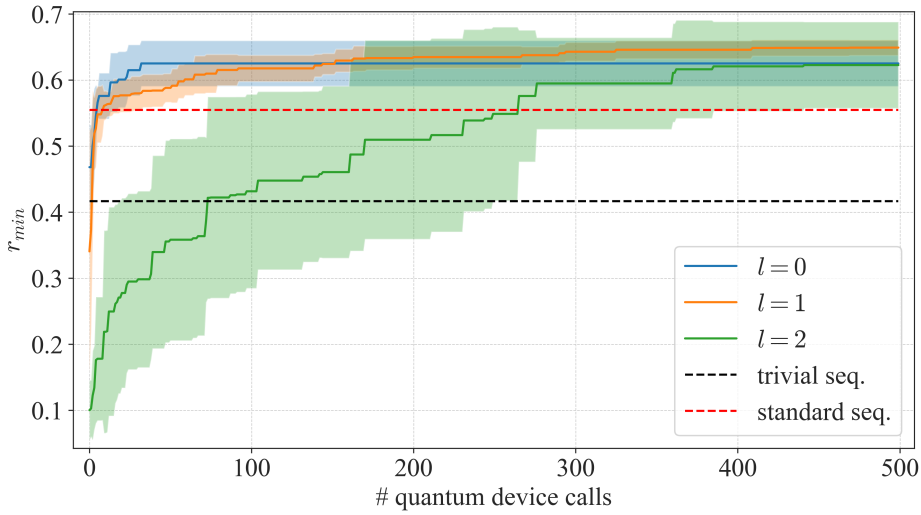


Figure 3.10: Classical random search for three gate pools ($l = 0$, $l = 1$ and $l = 2$) for QAOA gate sequences with a gate length of $q = 6$. Decoded parity QAOA with mean decoding an 10 random initializations is employed. The simulations are performed for the chosen 10 qubit problem instance (Sec. 3.3.1).

PS is presented in Fig. 3.11. It is apparent that for this specific problem, both random search, PS, and PPO demonstrate similar performance. This could be due to the relatively small number of gate sequences (526.513). Further improving the performance of PPO may also require a careful optimization of hyperparameters or policy tweaks. Remember that PS did not learn good gate sequences for $l = 2$, but apparently, it competes similarly against random search. This is because when an agent fails to learn, it essentially performs a random search, randomly exploring the observation space. Conclusively, it is evident

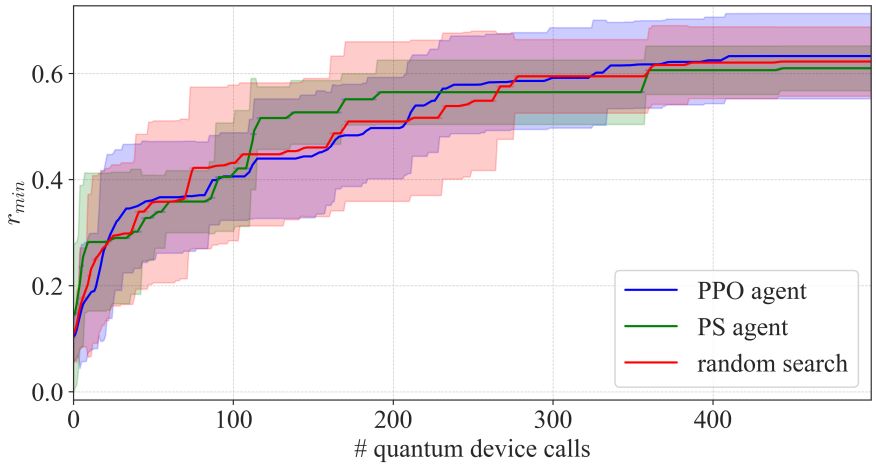


Figure 3.11: Comparison of the performance of the PPO agent and a basic random search. The approximation ration corresponds to the current minimal energy $r = E_{min}/E_{GS}$. The results are averaged across 10 executions each, and the standard deviation is indicated through shaded areas. The solid lines represent the moving average with a window length of 100. Decoded parity QAOA with 10 random initializations is employed.

that RL is unable to surpass a random search on this specific problem instance using the applied RL algorithms and their hyperparameters.

3.4 Patterns in RL gate sequences

We explore the performance of the observed gate sequence pattern $\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$.... (see Sec. 3.3.4) on the non-trivial 10 qubit problem instances, presented in Sec. 3.1. Fig. 3.12

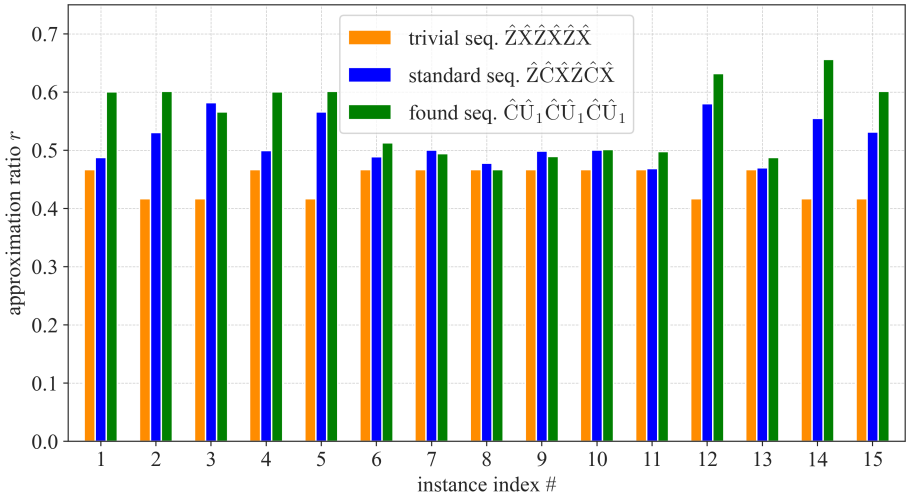


Figure 3.12: Approximation ratios r resulting from simulations of all selected unique 10 qubit problem instances with local fields in the set $\tilde{J}_i \in \{-1, 1\}$ for the standard ($\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$), trivial ($\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}\hat{X}$) and found gate sequence ($\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$) of gate length $q = 6$. Decoded parity QAOA with mean decoding and 100 random initializations is employed.

compares the performance of the found pattern against the performance of the standard and trivial gate sequence on the selected non-trivial 10 qubit problem instances (see. 3.3.1). In 11 out of 15 instances, the discovered gate sequence with a length of $q = 6$ outperforms the standard gate sequence. For instance number 3 ($v = 2$), 7, 8 ($v = 3$) and 9 ($v = 4$) the found gate sequence performs slightly worse. For the 6 qubit instances, we observe similar results (see. Appendix F, Tab. F.1).

3.5 Overall best gate sequences

Tab. 3.3 displays the gate sequences achieving the best approximation ratio r_{best} found by random search, the PS and PPO agents for the gate pools $l = 0$ (a), $l = 1$ (b) and $l = 2$ (c) (see Sec. 3.2.1) over the complete learning process. Furthermore, we perform 10 QAOA simulations with 100 random initializations for each gate sequence and donate the mean with r_{100}. For comparison, we compute this value also for the standard gate sequence, the trivial gate sequence, and the found gate sequence pattern (see. Sec. 3.4) and present the results in Tab. 3.3 (d). We recognize that the standard deviation is close to zero for every gate sequence from the pool $l = 0$ and two gate sequences from $l = 1$ partially exhibiting the shape of the observed gate sequence pattern $\hat{C}\hat{U}_1\hat{C}\hat{U}_1$.... The approximation ratio r_{100} for the found gate sequence $\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$ has a small standard deviation compared to gate sequences not exhibiting the found pattern of the pools $l = 1$ and $l = 2$. This could explain

Gate sequence	r_{best}	r_{100}	method
$\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}$	0.657	0.656 ± 0.005	PPO
$\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}$	0.657	0.656 ± 0.005	PS
$\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}$	0.657	0.656 ± 0.005	random search

(a)

Gate sequence	r_{best}	r_{100}	method
$\hat{U}_1\hat{X}\hat{C}\hat{U}_2\hat{C}\hat{U}_2$	0.667	0.464 ± 0.221	PS
$\hat{U}_1\hat{C}\hat{U}_1\hat{X}\hat{C}\hat{U}_1$	0.661	0.661 ± 0.0	PPO
$\hat{U}_1\hat{C}\hat{U}_1\hat{Z}\hat{C}\hat{U}_1$	0.658	0.658 ± 0.0	random search

(b)

Gate sequence	r_{best}	r_{100}	method
$\hat{U}_1\hat{C}\hat{U}_1\hat{X}\hat{U}_4\hat{Z}$	0.707	0.646 ± 0.106	random search
$\hat{U}_1\hat{C}\hat{U}_4\hat{U}_1\hat{C}\hat{U}_1$	0.699	0.53 ± 0.131	PPO
$\hat{U}_1\hat{U}_7\hat{X}\hat{U}_1\hat{C}\hat{U}_1$	0.646	0.624 ± 0.05	PS

(c)

Gate sequence	r_{100}
trivial seq. $\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}\hat{X}$	0.417 ± 0.0
standard seq. $\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$	0.555 ± 0.0
found seq. $\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$	0.654 ± 0.011

(d)

Table 3.3: Gate sequences achieving the best approximation ratio r_{best} found by random search, the PS and PPO agents for the gate pools $l = 0$ (a), $l = 1$ (b) and $l = 2$ (c) over a complete learning process. r_{100} stands for the mean approximation ratio resulting from repeated QAOA simulations for each gate sequence with 100 random initializations. The standard deviation from 100 executions per gate sequence is given. Tab. (d) shows this value for the standard gate sequence, the trivial gate sequence, and the found gate sequence.

why the PPO agent learns this pattern instead of a random gate sequence, which could lead to a higher approximation ratio solely by chance. Fig. 3.13 shows the distribution of the final approximation ratio resulting from 100 executions. We notice that the approximation ratio r_{100} of gate sequence $\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}$ resulting from pool $l = 0$ is within the error of the found gate sequence $\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$ and the data has a similar distribution.

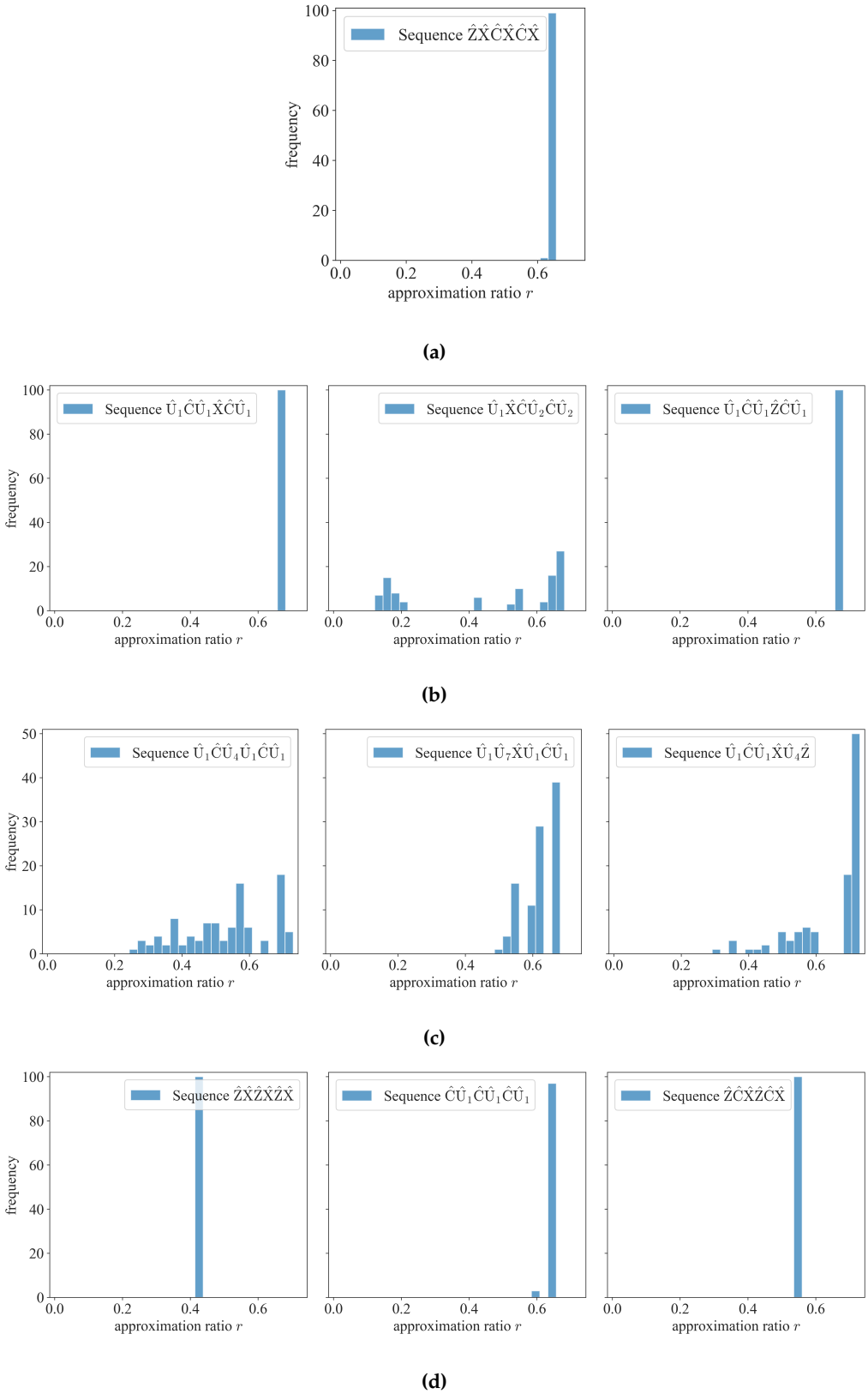


Figure 3.13: Approximation ratio distribution of the best gate sequences from gate pools $l = 0$ (a), $l = 1$ (b) and $l = 2$ (c). Tab. (d) shows the distribution for the standard gate sequence, the trivial gate sequence and the found gate sequence. The distributions result from 100 QAOA routines with 100 random initializations, each.

CHAPTER 4

Outlook and Conclusion

In recent literature, the exploration of Reinforcement Learning (RL) capabilities in quantum control, particularly in the context of the Quantum Approximate Optimization Algorithm (QAOA), has gained attention. A notable contribution by Yao et al. in 2021 introduced the generalized continuous-discrete QAOA, incorporating additional unitaries inspired by counterdiabatic driving and leveraging RL agents to discover effective gate sequences [37].

In this study, we extended these ideas to the realm of the parity QAOA, operating within the novel LHZ-architecture. Using CD methods, we derived seven additional locally implementable unitaries to enhance the expressiveness of the QAOA. Three gate pools, derived from the adiabatic gauge potential, were considered to enrich the QAOA. A key contribution of our work lies in demonstrating the applicability of RL algorithms to identify advantageous gate sequences for the decoded parity QAOA.

We successfully implemented and trained Projective Simulation (PS) and Proximal Policy Optimization (PPO) agents to find gate sequences that enhance QAOA's performance on challenging 6 and 10 qubit problem instances. To identify these instances, we compared the performance of the trivial gate sequence and the standard gate sequence and analyzed the frustration of the trivial configuration. Our results suggest that the sequence pattern

$$\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1.... \tag{4.1}$$

can outperform the standard gate sequence. Remarkably, this pattern exhibited superiority in 11 out of 15 unique hard 10 qubit instances, with similar trends observed for the 6 qubit instances. Notably, the implementability of gate U_1 with a single qubit rotation gate further adds to the attractiveness of the discovered sequence. We anticipate that the decoded parity QAOA, operating with this identified gate sequence, will yield more accurate results across a broad spectrum of challenging problem instances.

PPO agents played a pivotal role in discovering the aforementioned gate sequence, showcasing significantly better overall learning performance compared to PS. This dis-

parity is attributed to the increased expressiveness of the deep neural network employed in PPO, in contrast to the simpler two-layered decision tree used in PS. A suggestion for future investigations involves applying the recently introduced deep PS for a fairer comparison [63].

The observed trend reveals an elevated standard deviation in the approximation ratio when considering gate sequences that incorporate gates from the set $\{U_2, U_3, ..., U_7\}$ across 100 random initializations of the classical optimizer. This may account for the consistent recurrence of the mentioned gate sequence in the RL results, which notably excludes these gates. An open question remains if the RL agents could identify beneficial sequences, including these gates, with an increased number of random initializations of the classical optimizer. In this work, this could not be explored due to computational limitations.

Finally, we compared the RL approach against a random search algorithm, finding no significant performance advantage for the considered gate pools and problem instances. This may be attributed to the large standard deviation in the approximation ratio of some gate sequences, including higher-order gates. However, the PPO agents demonstrated the ability to identify a useful pattern, distinguishing themselves from random search. Future research could continue with careful hyperparameter optimizations of the PPO algorithm and improve the applied policy.

Ultimately, a repetition of the simulations incorporating noisy quantum measurements is needed to enhance fidelity in replicating experiments on real quantum hardware.

APPENDIX A

Equivalence of LHZ Hamiltonians

Two problem instances determined by their interaction arrays $\tilde{J}_1$ and $\tilde{J}_2$ are equivalent, if there exists a unitary V with $VV^\dagger = V^\dagger V$, which maps the Hamiltonian of problem 1, $\tilde{H}_p^{(1)}$, into the Hamiltonian of problem 2, $\tilde{H}_p^{(2)}$. In the context of QAOA, this must also hold for the mixer Hamiltonian $\tilde{H}_x$. Mathematically, we can construct equivalent sets of Hamiltonians for QAOA by identifying unitaries V, which satisfy the following relations:

$$\tilde{H}_x^{(1)} = V\tilde{H}_x^{(2)}V^\dagger = \tilde{H}_x^{(2)} \tag{A.1}$$

$$\tilde{H}_c^{(1)} = V\tilde{H}_c^{(2)}V^\dagger = \tilde{H}_c^{(2)} \tag{A.2}$$

$$\tilde{H}_z^{(1)} = V\tilde{H}_z^{(2)}V^\dagger \tag{A.3}$$

Let the unitary U be a multi local spin flip, acting on a fixed subset of physical qubits, denoted as $\mathcal{P}$:

$$U = \sum_{i\in\mathcal{P}} \tilde{\sigma}_x^{(i)} \tag{A.4}$$

It is easy to show that relation (A.1) holds for any U. The equation (A.2) is satisfied, if in each parity constraint none, or an even number of spins are flipped. This is the case, for example, when all spins corresponding to a logical line are flipped. Having found such a unitary, we apply it to $\tilde{H}_z^{(1)}$ and derive local fields $\tilde{J}_2$ such that $U\tilde{H}_z^{(1)}U^\dagger = \tilde{H}_z^{(2)}$.

The same can be done for SWAP operations (see Eq. (1.1.1), which are also unitary. In this case, the constraint Hamiltonian remains invariant when the qubits are swapped around the vertical symmetry axis of the LHZ triangle. For instance, in Fig. 1.3c, the vertical axis passes through qubits $\tilde{\sigma}_z^{(15)}$ and $\tilde{\sigma}_z^{(24)}$.

APPENDIX B

Details on decoded parity QAOA

Here, we compare the results of decoded parity QAOA for two final decoding strategies using both the standard and trivial gate sequences, covering gate lengths ranging from 3 to 15. We distinguish between taking the mean and the minimum energy of the final decoded state, as explained in Sec. 1.4.1. Fig. B.1 showcases the approximation ratio r for various gate lengths for the non-trivial 6 qubit instance with local fields $\tilde{J} = [-1, -1, -1, 1, -1, -1]$. The same calculations are performed for the non-trivial 10 qubit instance with local fields $\tilde{J} = [-1, -1, -1, -1, 1, -1, 1, -1, 1, -1]$, and the results are presented in Fig. B.2.

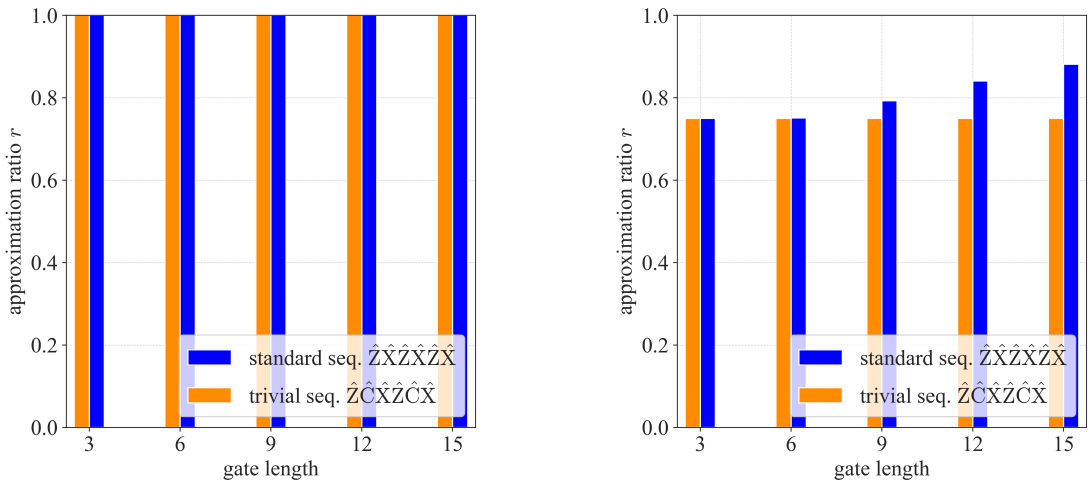
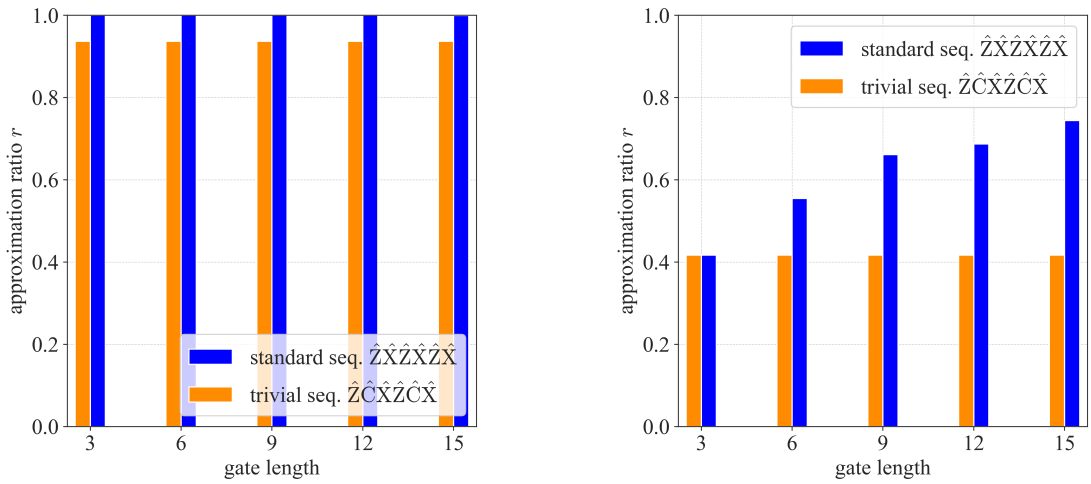


Figure B.1: Comparison of the final min decoding strategy (a) and the final mean decoding strategy (b) of decoded parity QAOA operating with the standard or the trivial gate sequence for various gate length. The parity QAOA angles are randomly initialized 100 times. The results were obtained from the non-trivial 6 qubit instance with local fields $\tilde{J} = [-1, -1, -1, 1, -1, -1]$.

Apparently, for 6 and 10 qubits, decoded parity QAOA with the minimum decoding strategy easily finds the ground state using both the trivial and the standard gate sequence. Therefore, for the RL simulations, we use the mean decoding strategy with a gate length of $q = 9$ for 6 physical qubits and $q = 6$ for 10 physical qubits, as for these the standard gate sequence outperforms the trivial one. Increasing the gate length further would lead



(a) **(b)**

Figure B.2: Comparison of the final min decoding strategy (a) and the final mean decoding strategy (b) of decoded parity QAOA operating with the standard or the trivial gate sequence for various gate length. The parity QAOA angles are randomly initialized 100 times. The results were obtained from the non-trivial 10 qubit instance with local fields $\tilde{J} = [-1, -1, -1, -1, 1, -1, 1, -1, 1, -1]$.

to exponentially larger observation spaces, hence making the learning process more challenging.

APPENDIX C

RL on 6 qubit instances

Tab. C.1 shows the QAOA results of unique 6 qubit problem instances, for which QAOA operating with the standard gate sequence does not find the optimal solution and $r^{standard} > r^{trivial}$. There are 6 unique problem instances for 6 physical qubits in total. We choose a gate sequence length of $q = 9$ because, for $q = 6$, the standard gate sequence never outperforms the trivial one significantly (see Fig. B.1). For the RL, we investigate

#	$\tilde{J}$	v	$r^{standard}$	$r^{trivial}$
1	$[-1,-1,-1,1,1,-1]$	1	0.778	0.75
2	$[-1,-1,-1,-1,1,1]$	1	0.774	0.75
3	$[-1,-1,-1,1,-1,1]$	2	0.785	0.75
4	$[-1,-1,-1,1,-1,-1]$	3	0.792	0.75

Table C.1: The table displays properties of all unique problem instances for 6 physical qubits with local fields in the set $\tilde{J}_i = \{-1, 1\}$, for which the QAOA with the standard gate sequence does not find the optimal solution and $r^{standard} > r^{trivial}$. One exemplary set of local fields $\tilde{J}$ per instance, the number of frustrated constraints v and the approximation ratio obtained with the standard gate sequence $r^{standard}$ and the trivial gate sequence $r^{trivial}$ with a sequence length of $q = 9$ are shown. Decoded parity QAOA with final mean decoding and 100 random initializations is employed.

problem instance number 4 with $\tilde{J} = [-1,-1,-1,1,-1,-1]$, because all the constraints are frustrated and the energy difference between the standard and the trivial gate sequence is the largest.

Proximal Policy Optimization

Following on, we explore the performance of the PPO agent on the discussed problem instance for the three gate pools corresponding to $l = 0$, $l = 1$, and $l = 2$, as discussed in Sec. 3.2.1. Fig. C.1 depicts the learning curves over 2500 episodes averaged across 10 PPO agents. We observe that only for $l = 0$ and $l = 1$, the agents are able to find gate sequences outperforming the standard gate sequence. The agents operating with the gate pool for $l = 2$ with 10 gates are not able to find gate sequences outperforming the standard gate sequence on average. The reason for this may be the large observation space.

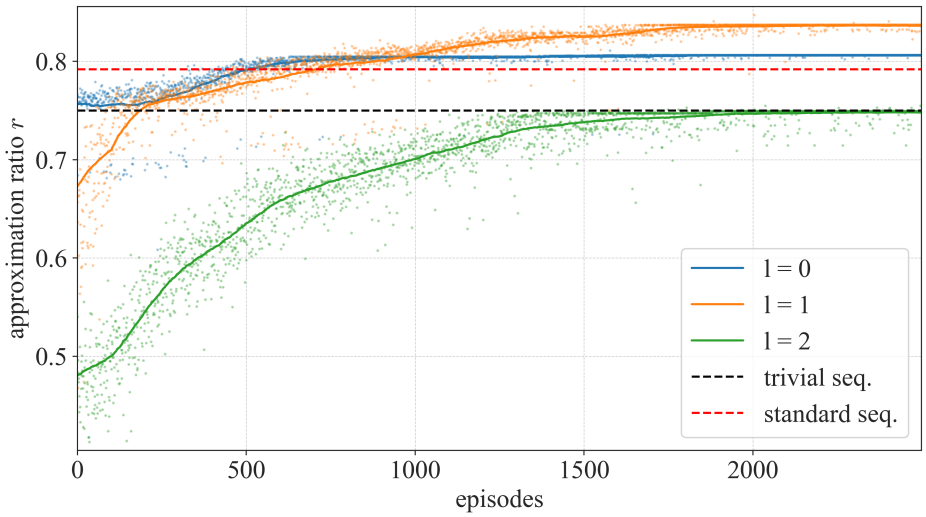


Figure C.1: Learning curves for various gate pools each averaged over 10 PPO agents for the chosen 6 qubit problem instance with a maximal gate length of $q = 9$. The solid lines represent the running average of the mean approximation ratio r, donated by points. Decoded parity QAOA with mean decoding and 10 random initializations is employed. The approximation ratios resulting from the standard gate sequence ($\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$) and the trivial gate sequence ($\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}\hat{X}$) with 100 random initializations are indicated with dashed lines.

Found gate sequences

We also present the learned gate sequences after 2500 episodes for the three gate pools in Tab. C.2. As for the 10 qubits problem instance, we observe the gate sequence pattern $\hat{C}\hat{U}_1\hat{C}\hat{U}_1...$ for $l = 1$ (b). In contrast to the case of 10 qubits and a gate length of 6, here the most frequently found pattern starts with the gate $\hat{U}_1$. Note that for $l = 2$, the approximation ratio r_{100} exhibits lower values than the final approximation ratio after learning. This is caused by a large standard deviation of r_{100} for gate sequences exhibiting additional gates from the pool $l = 2$.

Overall best gate sequences

Here, we present the gate sequences achieving the best approximation ratio r_{best} over all agents and the complete learning process. Tab. C.3 shows these gate sequences and the corresponding approximation ratio for the gate pools $l = 0$, $l = 1$, and $l = 2$ (see Sec. 3.2.1). For every gate sequence, we also present the approximation ratio r_{100} resulting from 100 randomly initialized QAOA runs with the standard deviation obtained from 10 executions. For comparison, this value is also computed for the trivial gate sequence, the standard gate sequence, and the just found gate sequence. Overall, the results look similar to the case of 10 qubits (see Sec. 3.5). We see that the gate sequence resulting from gate pool $l = 2$ exhibits the largest standard deviation. The gate sequence $\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{Z}\hat{X}\hat{C}\hat{X}$ achieves an approximation ratio r_{100} that is within the error range of the same value obtained from the discovered gate sequence.

Gate sequence	r_{final}	count	r_{100}
$\hat{C}\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}$	0.804	1	0.819 ± 0.019
$\hat{C}\hat{Z}\hat{X}\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$	0.792	1	0.790 ± 0.008
$\hat{Z}\hat{C}\hat{X}\hat{C}\hat{Z}\hat{X}\hat{C}\hat{Z}\hat{X}$	0.792	1	0.790 ± 0.008
$\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}\hat{C}$	0.810 ± 0.005	2	0.819 ± 0.018
$\hat{Z}\hat{C}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}\hat{C}\hat{Z}$	0.792	1	0.792 ± 0.000
$\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{Z}\hat{X}\hat{C}\hat{X}$	0.855	1	0.842 ± 0.019
$\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$	0.813	1	0.830 ± 0.017
$\hat{C}\hat{Z}\hat{X}\hat{C}\hat{Z}\hat{X}\hat{C}\hat{Z}\hat{X}$	0.792	1	0.792 ± 0.000
$\hat{Z}\hat{C}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}$	0.805	1	0.821 ± 0.017
standard seq. $\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$	-	-	0.792 ± 0.000

(a)

Gate sequence	r_{final}	count	r_{100}
$\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$	0.834 ± 0.016	4	0.846 ± 0.017
$\hat{U}_1\hat{C}\hat{U}_1\hat{X}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$	0.855	1	0.843 ± 0.017
$\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{X}\hat{U}_1\hat{C}\hat{U}_1$	0.833	1	0.836 ± 0.004
$\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{Z}\hat{U}_1\hat{C}\hat{Z}\hat{U}_2$	0.870	1	0.859 ± 0.009
$\hat{U}_1\hat{X}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$	0.816	1	0.828 ± 0.022
$\hat{U}_1\hat{C}\hat{Z}\hat{X}\hat{C}\hat{Z}\hat{X}\hat{C}\hat{X}$	0.855	1	0.827 ± 0.018
$\hat{U}_1\hat{C}\hat{X}\hat{C}\hat{X}\hat{C}\hat{X}\hat{C}\hat{U}_2$	0.852	1	0.863 ± 0.011

(b)

Gate sequence	r_{final}	count	r_{100}
$\hat{U}_5\hat{U}_7\hat{U}_5\hat{U}_7\hat{U}_5\hat{U}_7\hat{U}_5\hat{U}_3\hat{U}_5$	0.736	1	0.720 ± 0.034
$\hat{Z}\hat{U}_1\hat{U}_2\hat{U}_3\hat{U}_2\hat{U}_3\hat{U}_2\hat{U}_1\hat{U}_2$	0.795	1	0.745 ± 0.047
$\hat{U}_1\hat{U}_3\hat{U}_4\hat{U}_3\hat{U}_4\hat{U}_3\hat{U}_4\hat{U}_3\hat{U}_1$	0.750	1	0.754 ± 0.011
$\hat{U}_5\hat{U}_7\hat{U}_5\hat{U}_7\hat{U}_5\hat{U}_6\hat{U}_5\hat{U}_6\hat{U}_5$	0.733	1	0.706 ± 0.033
$\hat{U}_5\hat{U}_4\hat{U}_5\hat{U}_7\hat{U}_5\hat{U}_7\hat{U}_5\hat{U}_7\hat{U}_5$	0.794	1	0.699 ± 0.034
$\hat{U}_7\hat{U}_6\hat{U}_7\hat{U}_6\hat{U}_7\hat{U}_6\hat{U}_7\hat{U}_4\hat{U}_7$	0.731	1	0.702 ± 0.033
$\hat{U}_1\hat{U}_4\hat{U}_1\hat{U}_4\hat{U}_1\hat{U}_4\hat{U}_1\hat{U}_4\hat{U}_1$	0.772	1	0.764 ± 0.040
$\hat{U}_1\hat{U}_4\hat{U}_1\hat{U}_3\hat{U}_1\hat{U}_3\hat{U}_1\hat{U}_3\hat{U}_1$	0.846	1	0.759 ± 0.034
$\hat{U}_7\hat{U}_6\hat{U}_7\hat{U}_6\hat{U}_7\hat{U}_6\hat{U}_7\hat{U}_6\hat{U}_7$	0.706	1	0.715 ± 0.023
$\hat{U}_1\hat{U}_3\hat{U}_1\hat{U}_3\hat{U}_1\hat{U}_3\hat{U}_1\hat{U}_3\hat{U}_1$	0.749	1	0.765 ± 0.025

(c)

Table C.2: Found gate sequences for the 6 qubit problem instance after 2500 episodes of 10 PPO agents with the corresponding approximation ratio r for a maximal gate length of $q = 9$. If multiple agents found the same gate sequence, the count is provided, and the mean and standard deviation of the residual energy value are given. Tab. (a) corresponds to the case of $l = 0$, Tab. (b) to $l = 1$ and Tab. (c) to $l = 2$. r_{100} stands for the average approximation ratio obtained from multiple QAOA simulations for each gate sequence with 100 random initializations. The standard deviation is derived from 10 executions of such 100 QAOA repetitions.

gate pool	gate sequence	r_{best}	r_{100}
$l = 0$	$\hat{Z}\hat{X}\hat{C}\hat{X}\hat{C}\hat{Z}\hat{X}\hat{C}\hat{X}$	0.855	0.843 ± 0.018
$l = 1$	$\hat{U}_1\hat{C}\hat{U}_1\hat{Z}\hat{C}\hat{U}_1\hat{C}\hat{Z}\hat{U}_2$	0.899	0.855 ± 0.008
$l = 2$	$\hat{U}_6\hat{U}_2\hat{U}_1\hat{U}_2\hat{U}_6\hat{U}_3\hat{U}_1\hat{C}\hat{U}_4$	0.902	0.823 ± 0.058

(a)

Gate sequence	r_{100}
trivial seq. $\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}$	0.75 ± 0.0
standard seq. $\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$	0.79 ± 0.008
found seq. $\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$	0.85 ± 0.021

(b)

Table C.3: Gate sequences achieving the best approximation ratio r_{best} found by the PPO agents for the gate pools $l = 0$, $l = 1$, and $l = 2$ over a complete learning process (Tab. (a)). r_{100} stands for the mean approximation ratio resulting from repeated QAOA simulations for each gate sequence with 100 random initializations. The standard deviation from 10 executions per gate sequence is given. Tab. (b) shows this value for the standard gate sequence, the trivial gate sequence, and the found gate sequence.

APPENDIX D

Additional details on 10 qubit instances

Tab. D.1 lists the local fields and the number of frustrated constraints of all unique 10 qubit instances for which QAOA operating with the standard gate sequence does not find the optimal solution and $r^{standard} > r^{trivial}$. These problem instances are the relevant ones for the RL simulations.

#	$\tilde{J}$	v
1	$[-1,-1,-1,-1,1,1,1,-1,-1,1]$	2
2	$[-1,-1,-1,-1,1,1,1,-1,1,1]$	2
3	$[-1,-1,-1,-1,1,1,1,1,1,-1]$	2
4	$[-1,-1,-1,-1,-1,1,1,1,-1,1]$	2
5	$[-1,-1,-1,-1,1,1,1,1,-1,1]$	3
6	$[-1,-1,-1,-1,-1,-1,1,1,-1,-1]$	3
7	$[-1,-1,-1,-1,-1,1,1,-1,1,1]$	3
8	$[-1,-1,-1,-1,1,-1,-1,1,1,1]$	3
9	$[-1,-1,-1,-1,1,-1,1,-1,1,1]$	4
10	$[-1,-1,-1,-1,1,1,-1,1,1,-1]$	4
11	$[-1,-1,-1,-1,1,-1,1,1,1,-1]$	4
12	$[-1,-1,-1,-1,1,1,-1,1,-1,-1]$	4
13	$[-1,-1,-1,-1,1,-1,-1,-1,-1,1]$	4
14	$[-1,-1,-1,-1,1,-1,1,-1,1,-1]$	5
15	$[-1,-1,-1,-1,1,-1,1,1,1,1]$	5

Table D.1: The table displays one exemplary set of local fields $\tilde{J}$ per instance and the number of frustrated constraints v of all relevant and unique problem instances for 10 physical qubits with local fields in the set $\tilde{J}_i = \{-1, 1\}$.

APPENDIX E

RL Hyperparameter

E.1 Projective Simulation

In this section, we demonstrate PS's sensitivity to changes in the hyperparameters η, which denotes the glow damping parameter, and β, the softmax parameter. For this, we simulate the learning of 10 agents for $\eta \in \{0.05, 0.1, 0.2\}$ and $\beta \in \{0.5, 1.0, 2.0\}$, respectively. In Fig E.1, the moving averages of four averaged learning curves for different hyperparameter combinations are presented. Here we look at the 10 qubit instance with local fields $\tilde{J} = [-1, -1, -1, -1, 1, -1, 1, -1, 1, -1]$ (see Fig. 3.6) and search for gate sequences with a length of $q = 6$ with gates drawn from the gate pool $l = 1 : \{\hat{X}, \hat{Z}, \hat{C}, \hat{U}_1, \hat{U}_2\}$. We observe

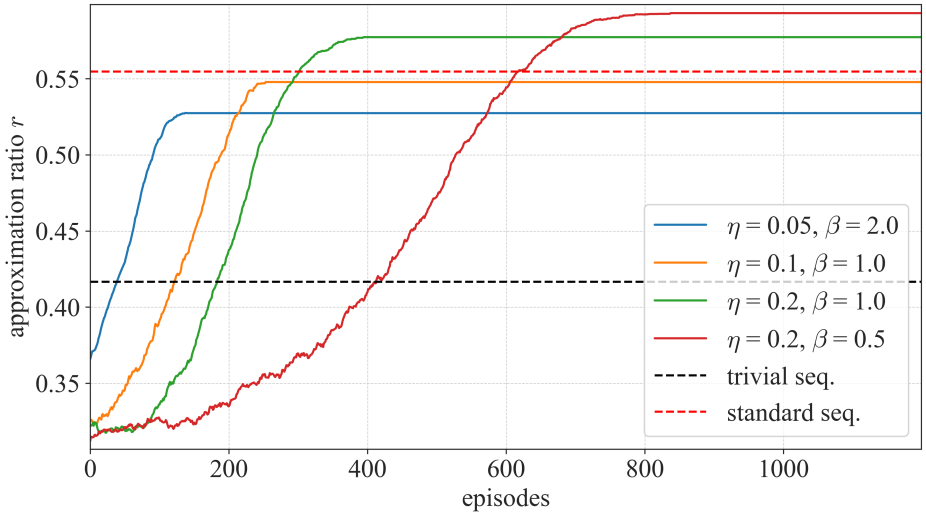


Figure E.1: Comparison of the influence on the learning of PS agents on certain combinations of the hyperparameters η and β. Each learning curve represents the moving average of the learning data averaged over 10 agents. Decoded parity QAOA with mean decoding and 10 random initializations is employed. The approximation ratios r resulting from the standard gate sequence ($\hat{Z}\hat{C}\hat{X}\hat{Z}\hat{C}\hat{X}$) and the trivial gate sequence ($\hat{Z}\hat{X}\hat{Z}\hat{X}\hat{Z}\hat{X}$) with 100 random initializations are indicated with dashed lines.

that a lower η and a higher β leads to faster convergence but a worse resulting approximation ratio r. In contrast, agents operating with high η and low β need more time to learn a gate sequence, but achieve better results. Notice that just two of the four combinations

even outperform the standard gate sequence. Among the tested combinations of η and β, the agents with $\eta = 0.2$ and $\beta = 0.5$ showed the overall best learning performance. Consequently, we use this combination to compare the performance of the three gate pools $l = 0$, $l = 1$ and $l = 2$ in Fig. 3.8 in Sec. 3.3.3. Actually, independently optimizing hyperparameters for each gate pool would be ideal, but this is computationally demanding. It becomes apparent, that PS is very sensitive to changes in hyperparameters, and moreover, the combinations found are not transferable to different problem setups.

E.2 Proximal Policy Optimization

For the PPO simulations, the Python framework from Stable Baselines 3 (SB3) is utilized. The module "MaskablePPO" is employed using the corresponding policy called "MaskableActorCriticPolicy", initialized with feedforward neural networks representing the policy and value function (see Sec. 1.5.2). Each of these networks comprises two hidden layers with 64 neurons. In the module "MaskablePPO", we only adjusted the number of steps between the policy updates while maintaining default settings for all other parameters. Tab. E.1 lists the values of important hyperparameters. By default, the policy updates occur every 2048 timesteps, which proves unsuitable in our case due to the computationally expensive nature of one QAOA subroutine call. Thus, we aim to determine a more appropriate value for the discussed non-trivial 6 (Fig E.2) and 10 (Fig E.2) qubit instances. For both cases, the choice of a number of steps of 50 seems appropriate.

Parameter	**Default value**
Learning Rate	0.0003
Batch Size	64
Number of Epochs	10
Gamma	0.99
GAE Lambda	0.95
Clip Range	0.2
Entropy Coefficient	0.0
Value Function Coefficient	0.5

Table E.1: Default settings of hyperparameters governing the fine-tuning dynamics of the PPO algorithm within the Stable Baselines 3 framework.

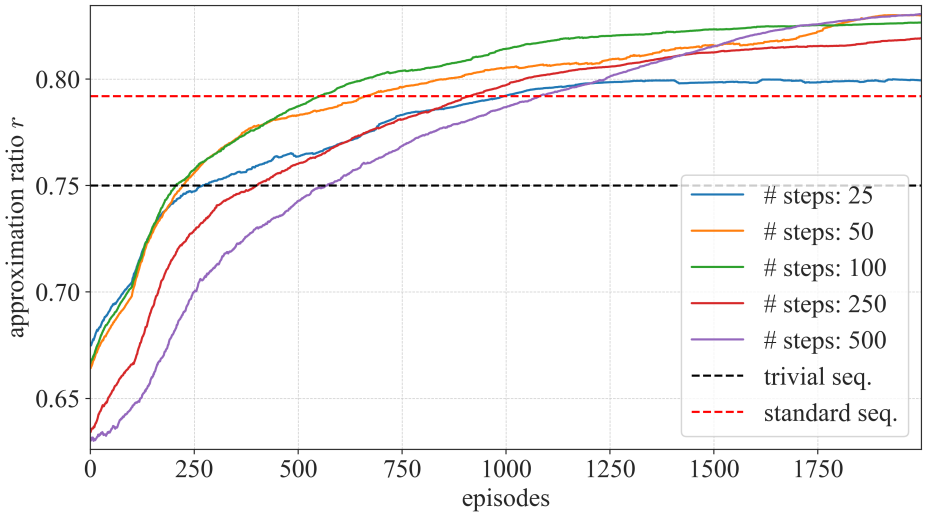


Figure E.2: Comparison of varying numbers of agent environment interaction steps between policy updated for a non-trivial 6 qubit instance with local fields $\tilde{J} = [-1, -1, -1, 1, -1, -1]$. Decoded parity QAOA with mean decoding is employed to find gate sequences of length $q = 9$. The lines display the moving average of the results averaged across 10 PPO agents.

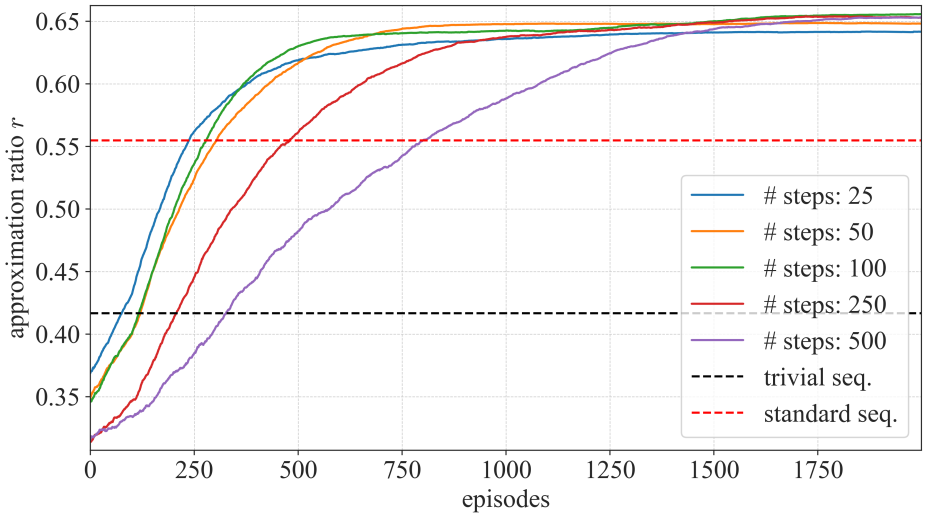


Figure E.3: Comparison of varying numbers of agent environment interaction steps between policy updated for a non-trivial 10 qubit instance with local fields $\tilde{J} = [-1, -1, -1, -1, 1, -1, 1, -1, 1, -1]$. Decoded parity QAOA with mean decoding is employed to find gate sequences of length $q = 9$. The lines display the moving average of the results averaged across 10 PPO agents.

APPENDIX F

Patterns in RL sequence

For completeness, we provide results for the performance of decoded parity QAOA operating with the discovered gate sequence $\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1\hat{C}\hat{U}_1$ with a length of $q = 9$ on the non-trivial 6 qubit problem instances. We see that, except for the problem instance number 1, the discovered gate sequence outperforms the standard gate sequence significantly.

#	$\tilde{J}$	v	$r^{trivial}$	$r^{standard}$	r^{found}
1	$[-1,-1,-1,1,1,-1]$	1	0.75	0.778	0.75
2	$[-1,-1,-1,-1,1,1]$	1	0.75	0.774	0.837
3	$[-1,-1,-1,1,-1,1]$	2	0.75	0.784	0.812
4	$[-1,-1,-1,1,-1,-1]$	3	0.75	0.792	0.866

Table F.1: The table displays properties of all unique problem instances for 6 physical qubits with local fields in the set $\tilde{J}_i = \{-1, 1\}$, for which QAOA with the standard gate sequence does not find the optimal solution and $r^{standard} > r^{trivial}$. One exemplary set of local fields $\tilde{J}$ per instance, the number of frustrated constraints v, and the approximation ratios obtained with the standard gate sequence $r^{standard}$ and the trivial gate sequence $r^{trivial}$ with a gate sequence length of $q = 9$ are shown. Decoded parity QAOA with mean decoding and 100 random initializations is employed.

APPENDIX G

Implementation of the QAOA unitaries

Here, we discuss the implementation of the additional gates $\hat{U}_1, \hat{U}_2, ..., \hat{U}_7$ applied in the RL simulations. Gate $\hat{U}_1$ can be implemented locally with single qubit y-rotation gates σ_y. Fig. G.1 presents a sketch of the gates required for implementation of the unitaries used for the simulations. A square represents a parity plaquette, and an arrow indicates that the implementation with the two gates swapped is also required. We see that $\hat{U}_2$, $\hat{U}_3$, and $\hat{U}_5$ are locally implementable on one parity constraint plaquette. $\hat{U}_6$ and $\hat{U}_7$ are local on

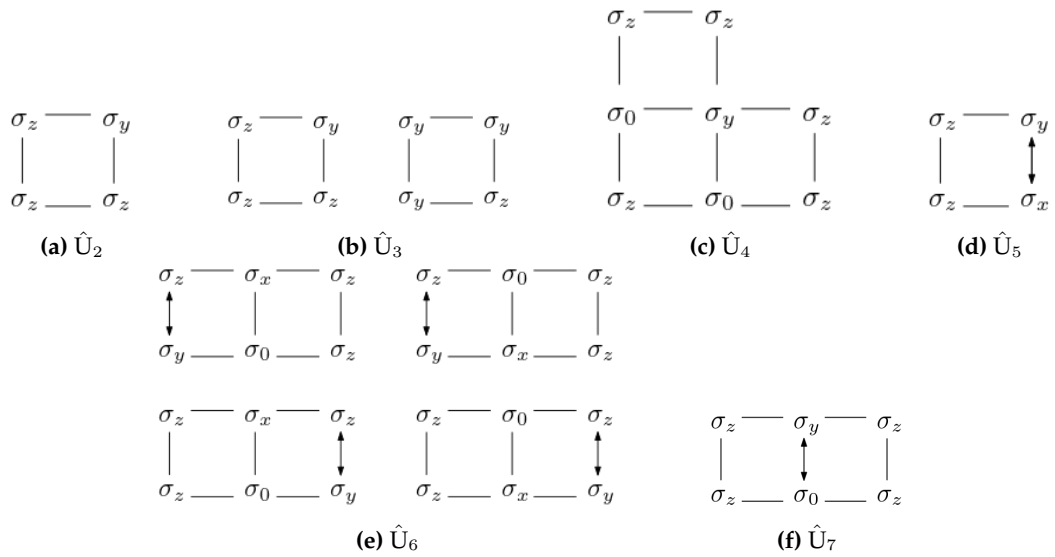


Figure G.1: Sketch of the required gates for implementing relevant unitaries.

two neighboring parity constraint plaquettes and $\hat{U}_4$ is local on three neighboring parity constraint plaquettes.

Bibliography

[1] B. Korte and J. Vygen, *Combinatorial optimization: theory and algorithms*, 2000.

[2] M. Eskandarpour, P. Dejax, J. Miemczyk, and O. Péton, "Sustainable supply chain network design: an optimization-oriented review", Omega **54**, 11 (2015).

[3] A. Sbihi and R. Eglese, "Combinatorial optimization and green logistics", Annals of Operations Research **175**, 254 (2010).

[4] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon", European Journal of Operational Research **290**, 405 (2021).

[5] A. Montanaro, "Quantum algorithms: an overview", npj Quantum Information **2**, 15023 (2016).

[6] A. Lucas, "Ising formulations of many np problems", 10.3389/fphy.2014.00005 (2013).

[7] W. Lechner, P. Hauke, and P. Zoller, "A quantum annealing architecture with all-to-all connectivity from local interactions", Science Advances **1**, 10.1126/sciadv.1500838 (2015).

[8] G. G. Guerreschi and M. Smelyanskiy, *Practical optimization for hybrid quantum-classical algorithms*, 2017.

[9] A. Weidinger, G. B. Mbeng, and W. Lechner, "Error mitigation for quantum approximate optimization", (2023).

[10] Q. Cappart et al., "Combinatorial optimization and reasoning with graph neural networks", (2021).

[11] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices", 10.1103/PhysRevX.10.021067 (2018).

[12] G. Laporte, "The traveling salesman problem: an overview of exact and approximate algorithms", European Journal of Operational Research **59**, 231 (1992).

[13] R. M. Karp, "Reducibility among combinatorial problems", in *Complexity of computer computations: proceedings of a symposium on the complexity of computer computations, held march 20–22, 1972, at the ibm thomas j. watson research center, yorktown heights, new york, and sponsored by the office of naval research, mathematics program, ibm world trade corporation, and the ibm research mathematical sciences department*, edited by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger (Springer US, 1972).

[14] M. Szegedy, *Quantum advantage for combinatorial optimization problems, simplified*, Dec. 2022.

[15] N. Pirnay, V. Ulitzsch, F. Wilde, J. Eisert, and J.-P. Seifert, *An in-principle superpolynomial quantum advantage for approximating combinatorial optimization problems*, 2023.

[16] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, U.K., 2000).

[17] F. Barahona, "On the computational complexity of ising spin glass models", Journal of Physics A: Mathematical and General **15**, 3241 (1982).

[18] S. Poljak and F. Rendl, "Solving the max-cut problem using eigenvalues", Discrete Applied Mathematics **62**, 249 (1995).

[19] B. R. Jumaa and A. S. Al-Jilawi, "Solving max-cut optimization problem", Journal of Physics: Conference Series **1591**, 012051 (2020).

[20] B. G. Sarmina, *Comparison between the iterative local search and exhaustive search methods applied to qaoa in max-cut and ising spin model problems*, 2022.

[21] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm", (2014).

[22] M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, "Benchmarking the quantum approximate optimization algorithm", Quantum Information Processing **19**, 10.1007/s11128-020-02692-8 (2020).

[23] J. Preskill, "Quantum computing in the nisq era and beyond", Quantum **2**, 79 (2018).

[24] P. Chandarana et al., "Digitized-counterdiabatic quantum approximate optimization algorithm", Physical Review Research **4**, 10.1103/PhysRevResearch.4.013141 (2022).

[25] A. Messiah, *Quantum mechanics* (Dover Publications, 1999).

[26] A. Rajak, S. Suzuki, A. Dutta, and B. K. Chakrabarti, "Quantum annealing: an overview", Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **381**, 10.1098/rsta.2021.0417 (2022).

[27] T. Albash and D. A. Lidar, "Adiabatic quantum computation", Reviews of Modern Physics **90**, 10.1103/revmodphys.90.015002 (2018).

[28] A. Messiah, *Quantum mechanics* (Courier Corporation, 2014).

[29] G. B. Mbeng, R. Fazio, and G. Santoro, *Quantum annealing: a journey through digitalization, control, and hybrid quantum variational schemes*, 2019.

[30] D. Poulin, A. Qarry, R. Somma, and F. Verstraete, "Quantum simulation of time-dependent hamiltonians and the convenient illusion of hilbert space", Physical Review Letters **106**, 10.1103/physrevlett.106.170501 (2011).

[31] M. Demirplak and S. A. Rice, "Adiabatic population transfer with control fields", The Journal of Physical Chemistry A **107**, 9937 (2003).

[32] E. Torrontegui et al., "Shortcuts to adiabaticity", Advances in Atomic, Molecular, and Optical Physics **62**, 117 (2013).

[33] D. Guéry-Odelin et al., "Shortcuts to adiabaticity: concepts, methods, and applications", Reviews of Modern Physics **91**, 10.1103/revmodphys.91.045001 (2019).

[34] D. Sels and A. Polkovnikov, "Minimizing irreversible losses in quantum systems by local counterdiabatic driving", Proceedings of the National Academy of Sciences **114**, 10.1073/pnas.1619826114 (2017).

[35] P. W. Claeys, M. Pandey, D. Sels, and A. Polkovnikov, "Floquet-engineering counterdiabatic protocols in quantum many-body systems", Physical Review Letters **123**, 10.1103/physrevlett.123.090602 (2019).

[36] P. Chandarana et al., *Meta-learning digitized-counterdiabatic quantum optimization*, 2022.

[37] J. Yao, L. Lin, and M. Bukov, "Reinforcement learning for many-body ground-state preparation inspired by counterdiabatic driving", Physical Review X **11**, 10.1103/PhysRevX.11.031070 (2021).

[38] W. Lechner, "Quantum approximate optimization with parallelizable gates", IEEE Transactions on Quantum Engineering **1**, 1 (2021).

[39] M. Fellner, A. Messinger, K. Ender, and W. Lechner, "Universal parity quantum computing", 10.1103/PhysRevLett.129.180503 (2022).

[40] Òscar Lorente, I. Riera, and A. Rana, *Image classification with classic and deep learning techniques*, 2021.

[41] P. P. Ray, "Chatgpt: a comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope", Internet of Things and Cyber-Physical Systems **3**, 121 (2023).

[42] H. P. Nautrup, N. Delfosse, V. Dunjko, H. J. Briegel, and N. Friis, "Optimizing quantum error correction codes with reinforcement learning", Quantum **3**, 215 (2019).

[43] F. Albarrán-Arriagada, J. C. Retamal, E. Solano, and L. Lamata, "Measurement-based adaptation protocol with quantum reinforcement learning", Physical Review A **98**, 10.1103/physreva.98.042315 (2018).

[44] J. Yao, H. Li, M. Bukov, L. Lin, and L. Ying, *Monte carlo tree search based hybrid optimization of variational quantum circuits*, 2022.

[45] G. B. Mbeng, R. Fazio, and G. Santoro, "Quantum annealing: a journey through digitalization, control, and hybrid quantum variational schemes", (2019).

[46] J. Yao, P. Köttering, H. Gundlach, L. Lin, and M. Bukov, *Noise-robust end-to-end quantum control using deep autoregressive policy networks*, 2020.

[47] H. J. Briegel and G. D. L. Cuevas, "Projective simulation for artificial intelligence", Scientific Reports **2**, 10.1038/srep00400 (2012).

[48] Øystein Førsund and M. G. Parker, "Projective simulation compared to reinforcement learning", (2015).

[49] J. Mautner, A. Makmal, D. Manzano, M. Tiersch, and H. J. Briegel, "Projective simulation for classical learning agents: a comprehensive investigation", 10.1007/s00354-015-0102-0 (2013).

[50] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, *Benchmarking reinforcement learning algorithms on real-world robots*, 2018.

[51] S. Khairy, R. Shaydulin, L. Cincio, Y. Alexeev, and P. Balaprakash, "Learning to optimize variational quantum circuits to solve combinatorial problems", Proceedings of the AAAI Conference on Artificial Intelligence **34**, 2367–2375 (2020).

[52] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms", (2017).

[53] B. Grooten, J. Wemmenhove, M. Poot, and J. Portegies, *Is vanilla policy gradient overlooked? analyzing deep reinforcement learning for hanabi*, 2022.

[54] "Proximal policy optimization algorithms", (2017).

[55] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction* (MIT press, 2018).

[56] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, *Trust region policy optimization*, 2017.

[57] J. Johansson, P. Nation, and F. Nori, "Qutip: an open-source python framework for the dynamics of open quantum systems", Computer Physics Communications **183**, 1760–1772 (2012).

[58] P. Virtanen et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python", Nature Methods **17**, 261 (2020).

[59] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms", 10.32473/flairs.v35i.130584 (2020).

[60] A. Raffin et al., "Stable-baselines3: reliable reinforcement learning implementations", Journal of Machine Learning Research **22**, 1 (2021).

[61] J. E. Cohen, S. Friedland, T. Kato, and F. P. Kelly, "Eigenvalue inequalities for products of matrix exponentials", Linear Algebra and its Applications **45**, 55 (1982).

[62] D. Gottesman, *The heisenberg representation of quantum computers*, 1998.

[63] S. Jerbi, L. M. Trenkwalder, H. Poulsen Nautrup, H. J. Briegel, and V. Dunjko, "Quantum enhancements for deep reinforcement learning in large spaces", PRX Quantum **2**, 10.1103/prxquantum.2.010328 (2021).