REGENSBURG

Porting Simulated Quantum Algorithms to the QExa20: Challenges and Results

Bachelor Thesis

in
Computer Engineering

At

Regensburg University of Applied Science

Student:	Laura Madajczyk
Student Number:	3213681
Supervisor:	Prof. Dr. Wolfgang Mauerer
Second Examiner:	Prof. Dr. Florian Heinz

Submission Date: 29.04.2025

OTH REGENSBURG

ERKLÄRUNG
ZUR BACHELORARBEIT VON

Name: Madajczyk

Vorname: Laura

Studiengang: technische Informatik

1. Mir ist bekannt, dass dieses Exemplar der Bachelorarbeit als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht. Die Abschlussarbeit darf elektronisch gespeichert und zu Zwecken der Zitatkontrolle genutzt und unter Verwendung digitaler Hilfsmittel, insbesondere von Plagiatserkennungssoftware, auf das Vorhandensein eventueller Plagiate geprüft werden.

2. Ich erkläre hiermit, dass ich diese Bachelorarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

3. Für den Fall, dass eine elektronische Fassung der Bachelorarbeit abgegeben worden ist, bestätige ich die Übereinstimmung der elektronischen Fassung mit der gebundenen Ausfertigung der Abschlussarbeit.

Regensburg, den

28.04.2025, Madajczyk, Laura
....
Unterschrift

Diese Erklärung ist mit der Bachelorarbeit (eingeheftet) abzugeben.

Stand: 10.07.2024/Abt. III

Abstract

The access to quantum hardware is limited, which is why simulating is common practice in quantum computing research. Thanks to granted access to the QExa20 quantum computer at the Leibniz Supercomputing Centre during its pilot phase, this thesis focuses on reproducing previously simulated experiments on real quantum hardware. The QExa20 is part of the *European Quantum Computing for Exascale-HPC* project and the first of its kind connected with a SuperMUC high-performance computer. There are five more quantum computer systems planned across Europe, with the vision of making several quantum computers and simulators accessible for multiple users.
The experiments ported to the QExa20 are based on the approaches introduced in [22] and [16], presenting a non-iterative version of the Quantum Approximate Optimization Algorithm (QAOA) and a quantum reinforcement learning method for solving the join order problem. Although the system is still in an early stage of development, it faces initial challenges, such as long runtimes and unclear error responses. Because 20 qubits is a really limited source to work with, the experimental parameters needed to be scaled down to fit the number of available qubits and experiments were carefully selected to account for long execution times.
As far as this thesis examined the performance of the QExa20, a really low depolarization error could be obtained and the results for non-iterative QAOA could be confirmed on real quantum hardware. Despite all limitations, the QExa20 has proven to be capable of producing reliable results. However, integrating quantum hardware into large computer systems and controlling multi-user access remains a complex challenge. Network latencies and request management can easily erode quantum advantage.

Contents

1 Introduction

Because available quantum hardware is a limited resource, it is common practice in research to simulate quantum circuits. Because of the chance to work on real quantum hardware, this thesis concentrates on porting such simulated experiments onto the quantum system QExa20. This superconducting quantum computer from IQM is the first of its kind connected to the SuperMUC high-performance computer and is stationed in the *Leibniz Supercomputing Centre* (LSC). It is part of the *Munich Quantum Valley* (MQV) and the first of six quantum computer systems of the project *European Quantum Computing for Exascale-HPC*. The big vision is to build multiple quantum machines and simulators all over Europe and making it accessible for multiple users. The MQV created an online portal for accessing their quantum machines and simulators.
Thanks to the granted access for several research groups in the pilot phase of the QExa20, the system could be used for the experiments of this thesis. The chosen experiments come from the paper [22] that introduces a non-iterative QAOA variant through parameter approximation and from [16] giving an approach for solving the join order problem with quantum reinforcement learning.
The work concentrates on the comparison of the results and also examines the system performance and workflow for QExa20 and the management infrastructure around it. To provide some fundamentals on quantum computing, the first chapter introduces qubit representation, the visualization of quantum states on the Bloch sphere, and the application of basic quantum gates. The second chapter outlines the concept of variational quantum circuits, focusing in particular on QAOA and Quantum Reinforcement Learning, followed by the third chapter outlining the covered optimization problems and their realization as a quantum circuit. In the final chapter describes the QExa20 system and its surrounding infrastructure, evaluates its performance and discusses the obtained results.

2 Quantum Basics

This chapter is for giving a basic understanding of quantum computing. Before qubits, quantum states, gates and circuits can be discussed, it is necessary to address some mathematical basics about vectors and vector spaces. The math of quantum computing is performed in the Hilbert Spaces, which are (complex) complete vector spaces with a scalar product and in this context with finite dimensions. They are also separable, what means they have a countable basis, so the basic vectors can be numbered consecutively. In such spaces, standard vector and matrix operations can be applied mainly [28].
The architecture of a quantum computer can be built from a n-dimensional Hilbert space, in which a state is referred to as qudit. But most architectures are constructed in a two-dimensional Hilbert space where states are named qubits. The state vectors are written on the computational basis which is orthonormal, meaning they are both orthogonal and normalized. So any two basis vectors v_i and v_j with $i, j \in \{0, n-1\}$, with $n \in \mathbb{N}_{\geq 1}$ have the property [17]

$$\delta_{ij} := v_i, v_j = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

This orthogonality ensures that, upon measurement, a qubit collapses to either $|0\rangle$ or $|1\rangle$ with well-defined probability. The used notation is Dirac's *bra-ket* notation, which provides a more compact denotation for vectors in a complex vector space. Let $|v\rangle$ represent a column vector $v \in \mathbb{C}^n$ with its elements v_i for $i \in \{0, 1, ..., n-1\}, n \in \mathbb{N}$ in *ket*-notation and let $|v\rangle^\dagger$ be the dagger operation performed on v. The dagger denotes the complex conjugate of its transpose. For each vector in *ket*-notation exists a corresponding *bra*-notation [28]. So for $|v\rangle$ applies

$$|v\rangle := \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix}, \quad \langle v| := |v\rangle^\dagger := \begin{pmatrix} \overline{v_0} \\ \overline{v_1} \\ \vdots \\ \overline{v_{n-1}} \end{pmatrix}^T = (\overline{v_0}\ \overline{v_1}\ ...\ \overline{v_{n-1}})$$

Using this notation, we can now describe any qubit state as a linear combination of the basis states. Any state of the system can be written as $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, which is a linear combination of $|0\rangle$ and $|1\rangle$ or also called superposition [28]. The coefficients α and β are complex numbers that satisfy the property $|\alpha|^2 + |\beta|^2 = 1$. This ensures that the total probability of measuring the qubit in either $|0\rangle$ or $|1\rangle$ is 100%.

2.1 Bloch Sphere

Beside Dirac's notation, another common way of representing a single qubit is the Bloch sphere. This is a normalized sphere in the three-dimensional space as pictured in Figure 2.1. Since every qubit state is described by a normalized vector, which can be visualized in the Bloch sphere originating from the center of the sphere, pointing to its surface. By convention, the state $|0\rangle$ is placed at the top and $|1\rangle$ at the bottom of the sphere. To represent a general qubit state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, the complex coefficients

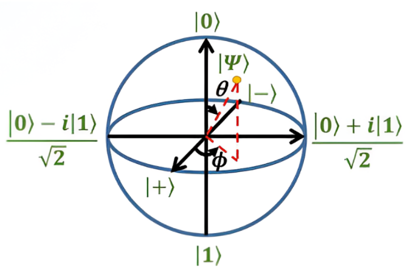

$|0\rangle$
$|\Psi\rangle$
θ
$|-\rangle$
$\frac{|0\rangle - i|1\rangle}{\sqrt{2}}$
$\frac{|0\rangle + i|1\rangle}{\sqrt{2}}$
ϕ
$|+\rangle$
$|1\rangle$

Figure 2.1: Bloch Sphere, Source:[28]

α and β can first be written in polar coordinates: $\alpha = r_1 e^{i\gamma_1}$ and $\beta = r_2 e^{i\gamma_2}$ where $\gamma_1, \gamma_2 \in [0, 2\pi]$ and $r_1, r_2 \in [0, 1]$. Due to the normalized condition $|\alpha|^2 + |\beta|^2 = 1$, there exists an angle $\theta \in [0, \pi]$ such that $r_1 = \cos(\theta/2)$ and $r_2 = \sin(\theta/2)$ where, θ represents the polar angle of the state on the Bloch sphere. This allows us to rewrite the state as [8]:

$$|\psi\rangle = \cos\frac{\theta}{2} e^{i\gamma_1} |0\rangle + \sin\frac{\theta}{2} e^{i\gamma_2}$$

Since a global phase factor $e^{i\gamma_1}$ does not affect the physical state, we can eliminate it by multiplying the entire state by $e^{-i\gamma_1}$. Letting $\phi = \gamma_2 - \gamma_1$ yield the equivalent expression:

$$|\psi\rangle = \cos\frac{\theta}{2} |0\rangle + \sin\frac{\theta}{2} e^{i\phi} |1\rangle \tag{2.1}$$

Thus, any pure qubit state can be fully described by the two real angles $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$. [20]

2.2 Quantum Gates

As described in 2.1 the state transformation of a qubit can be visualized with rotations inside the Bloch sphere. These rotations can be represented by matrices and are then called *operators* or *quantum gates.* This chapter breaks down the effect of quantum gates on qubits and breaks down their most important properties.
To perform calculations in quantum computing, the inner product and the tensor product are important operations on vectors and matrices. The inner product measures the similarity or overlap between two vectors in the same vector space. So for two vectors $v, u \in \mathbb{C}^n$ the inner product is

$$\langle u, v\rangle = \overline{u_1} v_1 + \overline{u_2} v_2 + ... + \overline{u_n} v_n$$

with, $\overline{u}$ denoting the complex conjugate of the vector u.[17]
The evolution of a quantum state can be represented by a linear operator. Any such operator can be expressed as a linear combination of tensor products. For an operator A in a n-dimensional Hilbert space with basis vectors $|i\rangle$ and $\langle j|$ for $i, j \in \{0, 1...n-1\}$ applies:

$$A = \sum_{ij} A_{ij} |i\rangle \langle j|$$

with standard basis vectors $|i\rangle$ and $\langle j|$.[20]
The tensor product of two vector spaces U and V with dimensions n and m, $n, m \in \mathbb{N}$ are of dimension $n \cdot m$. For two vectors $u \in U$ and $v \in V$, the tensor product $|u\rangle \otimes |v\rangle$ creates a new vector in the combined vector space $U \otimes V$. [20]
Extending the tensor product to matrices, the Kronecker product for two matrices A being a $n \times n$ matrix and B being a $m \times m$ matrix can be defined as [21]:

$$A \otimes B := (a_{i,j}B) := \begin{pmatrix} a_{0,0}B & a_{0,1}B & \cdots & a_{0,n-1}B \\ a_{1,0}B & a_{1,1}B & \cdots & a_{1,n-1}B \\ \vdots & \vdots & & \vdots \\ a_{n-1,0}B & a_{n-1,1}B & \cdots & a_{n-1,n-1}B \end{pmatrix}$$

where $a_{i,j}B$ is a scalar-matrix-product, where $a_{i,j}$ represent an element in A. The resulting matrix is of size $(n \cdot m) \times (n \cdot m)$.
As an example, the identity gate is a gate that induces no qubit evolution, because it consists of the basis vectors of the n qubit system. The identity matrix for a one qubit system is simply defined as $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Applying an operator onto a state includes a matrix vector multiplication. So for applying I onto $|0\rangle$ it is

$$I|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Note that I consists of both basic vectors for $|0\rangle$ and $|1\rangle$. Therefore, the result for the multiplication is always zero or one where the matrix row is the vector itself. For any quantum operator A applies that A multiplied with its inverse A^{-1} will result in the Identity operator. This property is called reversibility and is one of the essential properties for quantum operators. Some operators also fulfill the property of involutory. That is the case, if an operator applied twice, it will also result in the identity operator.

2.2.1 Projection, Hermitian and Unitary Operator

Beside general operator properties, there are three special operator types to be discussed. At first there are Hermitian operators. These are matrices with real eigenvalues. A matrix A is Hermitian if it satisfies $A = A^{\dagger}$, where $A^{\dagger}$ denotes the conjugate transpose of A. [17]
To give an example, let

$$A = \begin{bmatrix} 2 & i \\ -i & 3 \end{bmatrix}$$

By transposing A we get:

$$A^T = \begin{bmatrix} 2 & -i \\ i & 3 \end{bmatrix}$$

Further, the complex conjugate of A^T is $A^{\dagger}$, so we get:

$$A^{\dagger} = \begin{bmatrix} 2 & i \\ -i & 3 \end{bmatrix}$$

By comparison, we get $A = A^\dagger$, which means that A is Hermitian.
Secondly, a unitary operator U describes the evolution of a quantum state over time t [1]. If Matrix A is Hermitian, then for any $t \in \mathbb{R}$, e^{itA} is unitary. Conversely, every unitary matrix has the form e^{itA} of some Hermitian matrix [20]. The third operator type is the projection operator, which is more a construction method for operators than an operator itself. They utilize the basis states of the Hilbert space to project any state onto a subspace. Let ψ be a vector in a high dimensional space, then we project it to a lower dimensional space spanned by $|i\rangle$ with the projection operator P_i given by

$$P_i = |i\rangle \langle i|$$

The operator removes all orthogonal components and only retains the component in the subspace spanned by $|i\rangle$[28].
The act of measurement is described mathematically by using projection operators. Each observable is associated with a Hermitian operator A with a set of eigenvalues a_i, onto which the state ψ is projected onto. The probability of obtaining the outcome a_i is given by $p_i = \langle\psi| P_i |\psi\rangle$. After the measurement, the system collapses into state $|i\rangle$ with probability p_i.

2.2.2 Eigenvectors and Eigenstates

The possible outcomes of a measurement correspond to the eigenvalues of an observable. Let A be a linear operator and $|a\rangle$ a vector in Hilbert space. If $|a\rangle$ satisfies the equation

$$A |a\rangle = b |a\rangle$$

for some scalar b, then represents $|a\rangle$ the eigenvector of A with the eigenvalue b. This implies that applying A to $|a\rangle$ does not change the direction of the vector, It just scales it by its eigenvalue. Furthermore, when $|b\rangle$ is scaled by any complex number c it stays an eigenvector of A.

$$cA |b\rangle = ca |b\rangle \, A(c |b\rangle = a(c |b\rangle)$$

This justifies why eigenvectors are typically normalized, because normalization does not affect their eigenvalue relation. If two or more linear independent eigenvectors correspond to the same eigenvalue, these eigenvectors are called degenerated. In this case, any linear combination of these eigenvectors is also an eigenvector with the same eigenvalue. Together they span a subspace called eigenspace associated with that eigenvalue [28].

2.2.3 Pauli Rotation Gates

Three very basic operations to transform a qubit state are the Pauli Operators X, Y, Z. These are represented by the following matrices:

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

These gates are named according to the rotations they perform on the Bloch sphere. The X-gate performs a bit flip, the Z gate induces the phase flip and the Y gate combines both a bit and phase flip.

While the standard Pauli gates always flip the qubit state with a fixed angle of π, the presence of the identity operator allows for a controlled and continuous rotation by an arbitrary angle θ. This leads to a generalization of the Pauli gates into the Pauli-rotation gates [8]:

$$R_X(\theta) = e^{-i\frac{\theta}{2}X} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

$$R_Y(\theta) = e^{-i\frac{\theta}{2}Y} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

$$R_Z(\theta) = e^{-i\frac{\theta}{2}Z} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Z = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$$

2.2.4 RGate and Cphase

The RGate and the CPhase gate are the two native gates of the QExa20. Therefore, any quantum circuit must be decomposed into a sequence of these two gates to be executable on this specific hardware.
The Rotation Gate (RGate) is a single qubit gate, capable of inducing rotation on the Bloch Sphere with the rotation angles $\theta, \phi \in [0, 2\pi]$. It is defined as [12] :

$$R(\theta,\phi) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -ie^{-i\phi}\sin(\frac{\theta}{2}) \\ -ie^{i\phi}\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}$$

However, operations on single qubits are not sufficient to change the degree of entanglement in a quantum system, which is a property that fundamentally distinguishes quantum computing from classical computing. Entanglement is the central quantum mechanical effect in which at least two quantum particles are related to each other in such a way that their common properties are retained, regardless of how far apart they are in space. In the case of entanglement, both particles are in an indeterminate state. If one of the two is measured, i.e. the state collapses to $|0\rangle$ and $|1\rangle$, it is also clear from this measurement that the other particle is in the same state. Entanglement can occur over multiple qubits, can have different intensity (concurrence) and can be passed between qubits. Because it is a very complex topic on its own and not essential for this thesis, for further readings on that [19] is recommended. With applying only unitary single qubit gates entanglement can not be generated or manipulated. To overcome this, at least one entangling two qubit gate is required.
One such gate is the controlled phase (CPhase) gate. It applies a phase shift conditioned on the control qubit being in the $|1\rangle$ state. The matrix form of the CPhase gate with the real parameter $\lambda \in [0, 2\pi]$ is:

$$CPhase(\lambda) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\lambda i} \end{bmatrix}$$

[11] Together, the RGate and the CPhase gate form a universal gate set, as proven in [3]. Therefore, can any unitary operation be decomposed into a finite sequence of these two gates.

2.3 Quantum Circuits

A formalism to denote quantum circuits is the quantum gate array, introduced by Deutsch. It is the quantum generalization of acyclic combinational logic circuits that consist of quantum gates interconnected with quantum wires. These gates have the same number of input as output and carry an operation in the Hilbert space of dimension 2^n for $n \in \mathbb{N}$ input qubits[1].
The circuit shown in 2.2 consists out of the one qubit Hadamard gate and the two qubit CNOT-gate. Both together create an entangled bell state. A circuit can be quantified with the parameters circuit width and circuit depth. The circuit width is the number of qubits utilized for the circuit. The depth describes the highest amount of operations on one qubit over the whole circuit. For the example, the width and the depth of the circuit is two. To perform a quantum circuit on real quantum hardware, transpilation

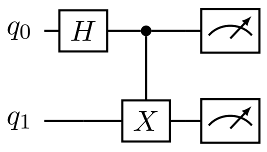
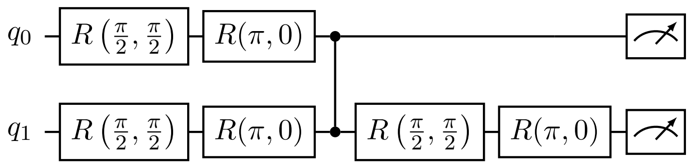


Figure 2.2: Original circuit


q_0
$R\left(\frac{\pi}{2}, \frac{\pi}{2}\right)$
$R(\pi, 0)$
q_1
$R\left(\frac{\pi}{2}, \frac{\pi}{2}\right)$
$R(\pi, 0)$
$R\left(\frac{\pi}{2}, \frac{\pi}{2}\right)$
$R(\pi, 0)$
Figure 2.3: Transpiled circuit for QExa20

needs to be performed. Transpilation is the process of bringing a circuit into a form that considers the resources of the system, like qubit count, qubit coupling and native gates. The transpiled circuit for the bell state for the QExa20 is given in 2.3. The transpilation partitioned the Hadamard gate into two R-Gates on qubit zero and the effect of the CNOT is achieved with two RGates on qubit one, followed by on CPhase gate on both qubits, followed by two more RGates on qubit one. For this simple example, the circuit width stays two, but the circuit depth increased to five. To calculate the outcome for a circuit, it is always quantum gate multiplied by the state it acts on. The result is the new state vector after the gate acted on the input qubits. To get the state vector over more than one qubit, the vector can be retrieved through the tensor product of the single qubit states. If there are qubit gates that act simultaneously on different qubits, the qubit gate for the whole system is retrieved by calculating the tensor product of the qubit gates of all qubits that are performed in parallel. If a qubit has no gate that acts on it, the identity gate is inserted. Sometimes extra qubits are needed to map a gate onto a specific quantum hardware. These helping qubits have the name ancilla qubits and store information for subcalculations if gates can not be performed directly. [17]

2.4 Ising Hamiltonians and cost functions

A Hamiltonian H represents the total energy of a system, expressed as a Hermitian operator. Because of this property, the eigenvalues of the Hamiltonian are always real and the eigenstate with the lowest energy state is called ground state. Any optimization problem can be transformed into a minimization problem. Thus, the optimal solution lies in the ground state $|\psi\rangle$ that minimizes the average energy value $\langle\psi| H |\psi\rangle$. The eigenstates of the Hamiltonian corresponds to all possible inputs $x \in X$, where $X = \{0, 1\}^n$ is the set of binary strings with length n. These binary strings represent the state vectors $|x\rangle$ of a n-qubit system. The respective eigenvalues correspond to an objective function $f(x)$. This so-called cost function encodes the problem and its minimum corresponds to the solution of the problem. Therefore, the problem Hamiltonian is a $2^n \times 2^n$ matrix

$$H = \sum_{x \in X} f(x) \, |x\rangle \langle x|$$

with its eigenvalues as diagonal elements [17].
Problem Hamiltonians can be encoded by formulating Quadratic unconstrained binary optimization (QUBO) problems. A QUBO problem is defined via a quadratic cost function over boolean variables $x_i \in \mathbb{F}_2$, $\mathbb{F}_2 := \{0, 1\}$ and is given by

$$\sum_{i \neq j} a_{i,j} x_i x_j + \sum_i a_i x_i \tag{2.2}$$

where $a_{i,j}, a_i \in \mathbb{R}$ are weights, which describe the cost contribution of a single bit or bit pair.[22]
Equivalently, problem Hamiltonians can also be constructed efficiently using the Ising model. In this case, the boolean variables in the objective cost function are transformed into spin variables $s_i = \pm 1$. A real constant $J_{i,j}$ describes the correlation between the spins s_i and s_j, while another real constant h_i introduces the incentive for each individual spin. The corresponding cost function

$$C(s) = C(s_1, ...s_n) = -\sum_{i<j} J_{i,j} s_i s_j - \sum_i h_i s_i$$

and the respective problem Hamiltonian is specified with

$$H_C = -\sum_{i<j} J_{i,j} Z^{(i)} Z^{(j)} - \sum_i h_i Z^{(i)} \tag{2.3}$$

where $Z_{(i)}$ denotes the Pauli-Z operator on the i-th qubit. [18]

3 Variational quantum algorithms

Achieving quantum advantage on NISQ devices remains a significant challenge due to constraints such as limited qubit counts, restricted connectivity between qubits, gate errors, and short coherence times. Nevertheless, variational quantum algorithms are considered a potentially sufficient strategy to obtain quantum advantage on NISQ devices. VQAs adopt a hybrid quantum-classical approach of executing a parameterized circuit on Quantum Hardware while a classical optimizer iteratively updates the parameters to minimize a predefined cost function $C(\theta)$. In this framework, the quantum circuit estimates the cost function and classical methods are used to optimize the parameters θ.

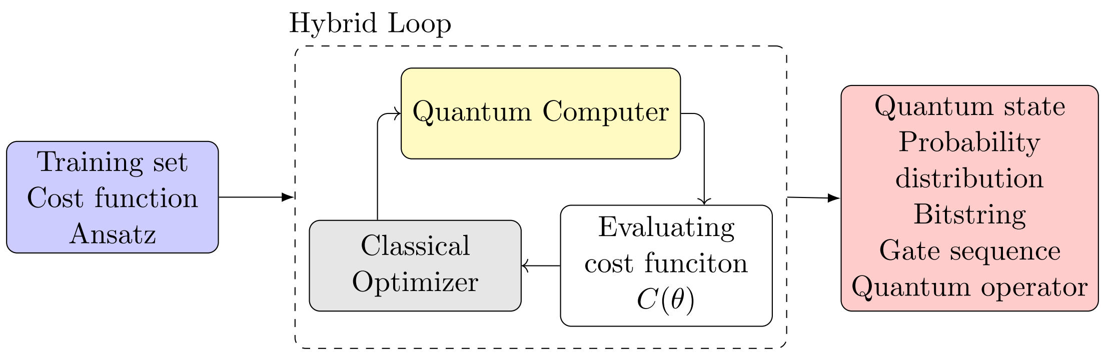


Figure 3.1: Schematic diagram of a VQA, inspired by [7]

A Quantum advantage using VQA's would be demonstrated if the minimum of the cost function cannot be computed efficiently on a classical machine but by quantum means. Additionally, for a VQA to be practical, the optimization landscape should allow an efficient optimization of the parameters θ and smaller values of $C(\theta)$ should correspond to higher-quality solutions.
Due to the limitations of NISQ hardware, the quantum circuits used in VQAs must remain shallow and avoid extensive use of ancillary qubits. The parameters θ are typically encoded in a unitary $U(\theta)$, which is applied to the input quantum state. The structure of the parameterized circuit, referred to as ansatz, determines the nature of θ and is highly task-dependent. There exist many types of ansätze and design considerations, as detailed in [7].
In this chapter, the prominent algorithm QAOA will be introduced, as well as Quantum Reinforcement Learning, which also has received growing attention in recent years.

3.1 QAOA

The Quantum Approximate Optimization Algorithm (QAOA) is a gate-based quantum algorithm designed to obtain approximate solutions to combinatorial optimization problems. It can be viewed as the discrete (trotterized) counterpart of quantum annealing. Both methods are inspired by the adiabatic theorem, which uses a time-dependent Hamiltonian, that induces a continuous transformation of the state of a quantum system from an initial to a final state. In quantum annealing, the solution is guaranteed to be

optimal, under the condition that the evolution of the system is slow enough.
The algorithm, originally introduced by [15], translates this adiabatic process into a discrete sequence of unitary gates using two Hamiltonians:

- The cost Hamiltonian C encodes the optimization problem and has the unitary

$$U(C, y) = e^{i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha}$$

- The mixer Hamiltonian B ensure exploration of the solution space. The corresponding mixer unitary is

$$U(B, \beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta \delta_j^x}$$

Each cost Hamiltonian is parameterized by angle γ and each mixer Hamiltonian parameterized by angle β. For the cost Hamiltonian $C(z) = \sum_{\alpha=1}^{m} C_\alpha$ represents the objective function defined over bit strings $z = z_1 z_2 ... z_n$ with n bits and m clauses. If clause $C_\alpha = 1$, z satisfies clause α, $C_\alpha = 0$ otherwise.
The mixer Hamiltonian with angle β running from 0 to 2π and operator $B = \sum_{j=1}^{n} \delta_j^x$ and δ_i^x is the Pauli X-operator acting on qubit j.[15] A schematic example of a QAOA circuit with $p = n$ layers is shown below, where each layer alternates between applications of the cost unitary $U_C(\gamma)$ and the mixer unitary $U_B(\beta)$: In practice, the cost function is

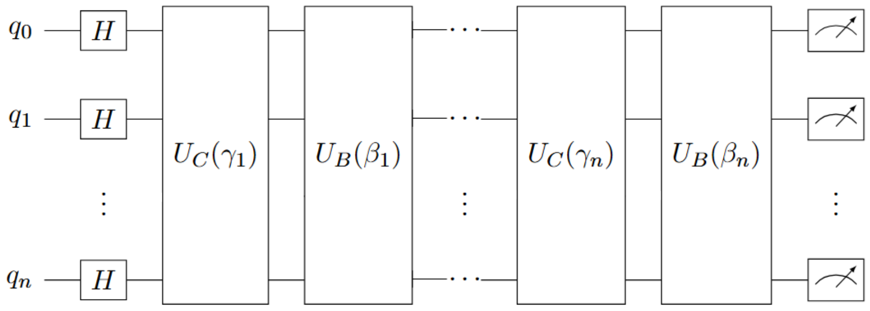


Figure 3.2: Schematic p-layer QAOA circuit for n qubits

often translated into an Ising model representation to define the cost Hamiltonian. The parameters (γ, β) are optimized classically to minimize the expectation value of the cost Hamiltonian, thus improving the quality of the approximate solution.

3.2 Quantum Reinforcement learning

For Quantum Reinforcement Learning (QRL) and also for other Quantum machine learning methods applies, that the classical neural network is changed into a variational quantum circuit, because both are universal function approximators. In QRL a decision maker, called agent is defined, who interacts in an environment. In discrete time steps t, the agent receives a state of the environment $S_t \in S$ with S being all possible states

and performs a state based action $A_t \in A(S_t)$, where $A(S_t)$ denotes the set of all available actions in the state S_t. As a consequence of this action, the agent obtains a reward $R \in R$ and transitions into a new state. The goal is to maximise the total amount of this accumulated reward over a longer period of time. The decision regarding which action to take is defined by a policy π_t, where $\pi_t(a|s)$ denotes the probability that $A_t = a$ given $S_t = s$.
In accordance with this policy, most reinforcement learning algorithms define a value function $v_\pi(s)$, that estimates the expected future rewards depending on the agent's action. The mathematically idealized form of a reinforcement learning problems are Markov Decision Problem (MDPs) [26].
QRL was first introduced by Daoyi Dong and his team in 2005. They derived the set of states S and the set of actions $A_(S_t)$ for a given state S_t from the eigenvectors of the set of the complete orthonormal bases in a Hilbert Space. A state $|S\rangle$ and an action $|A\rangle$ are extended to the set of eigenstates $|s_n\rangle$ or eigenactions $|a_n\rangle$ like

$$|S\rangle = \sum_n \alpha_n |s_n\rangle \qquad |A\rangle = \sum_n \beta_n |a_n\rangle$$

with probability amplitudes α_n and β_n, satisfying $\sum_n |\alpha_n|^2 = 1$ and $\sum_n |\beta_n|^2 = 1$. Every state and action in traditional Reinforcement Learning has a corresponding representation with eigenstates and eigenactions in QRL.
The selection policy $\pi : S \to A$ is executed by measuring $|a_s\rangle$, associated with a particular state $|s\rangle$. Upon measurement, $|a_s\rangle$ collapses to a specific action $|a\rangle$ with probability $|C_a|^2$:

$$|a_s^{(n)}\rangle = \sum_{a=0...0}^{\overbrace{1...1}^{n}} C_a |a\rangle \text{ , with } \sum_{a=0...0}^{\overbrace{1...1}^{n}} |C_a|^2 = 1$$

Here $|a_s^{(n)}\rangle$ represents the superposition of 2^n possible eigenactions and $|a\rangle$ is the eigenaction selected by measurement. The probability amplitude is updated by performing Grover iterations multiple times. For further details, refer to [14].

4 Optimization Problems on Quantum Hardware

Certain problems have the potential to be solved more efficiently using quantum algorithms over classical approaches. A promising field is that of Quanutm Machine Learning, especially in combination with VQAs. The paper [16] approaches the join order problem with quanutum reinforcement learning. It states, that even though it may not significantly outperform classical approaches for this problem, a drstic reduction in required trainable parameters could be found.
In [22] a two phase, non-iterative algorithm is introduced to approximate the optimization landscape of QAOA circuits. This algorithm consists of two steps: first, approximating a problem specific bit instance independent expected landscape and second, sampling a fixed quantum circuit based on this landscape. An optimization landscape can be described by all point to point interactions between all states in superposition. While the results apply to QUBO problems, the theoretical considerations are general and apply to arbitrary combinatorial optimization problems.
The problems from both papers are in the class of NP, which means that there is not yet known algorithm that can solve the problem in polynomial time. If a problem is at least as hard as another problem in NP, it is called NP-hard and if it belongs to NP itself, it is classified as NP-complete. Because promisingly good results could be aimed with VQAs, it gained a lot of interest in current research [25].
This chapter focuses on the transformation of three specific optimization problems into a VQA. The first two problems are boolean satisfiability (SAT) and qr-factoring. In [22] is a strategy given on how to solve these problems with QAOA. The third one describes the approach of solving the join order problem with Quanutm Reinforcement Learning.

4.1 Boolean satisfiability

In [25] the problem of satisfiability (SAT) is fittingly referred to as ẗhe mother of all NP-complete problems·̈ SAT takes a set of boolean variables V and a set of logic clauses C over these variables as input. The goal is to determine a variable assignment such that every clause is satisfied. The clauses must be expressed as a logical formula in conjunctive normal form (CNF). Although logical formulas can be also be expressed in disjunctive normal form (DNF), this would render the problem trivial. The conversion of a formula from CNF into DNF using De Morgan's laws can result in an exponential increase of terms required, making such a translation not realizable in polynomial time [25].
In the special case of k-SAT, each clause contains at most k literals. For example, an instance of 3-SAT could be expressed as:

$$(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

which is satisfied by the assignment $x_1 = true$, $x_2 = true$ and $x_3 = false$. As shown in [25] the 3-SAT problem is computationally hard, and it is known that any instance of k-SAT can be polynomially reduced to 3-SAT. This reduction places all k-SAT problems

into the class of NP-complete problems.
For solving a SAT instance with QAOA in [22] a constraint Hamiltonian is constructed. At first, a boolean formula in conjunctive normal form $C = \{c_i\}_{i=1}^{m}$ with c_i being the i-th clause is given. Let $\overline{C_i}$ be a projector onto an unsatisfied assignment of the clause c_i and $\{\overline{C_i}\}_{i=0}^{m}$, then it is possible to construct a constraint Hamiltonian $C = \prod_{i=0}^{m}(\not{1} - \overline{C_i})$ that projects onto all satisfying assignments of C.

4.2 Qr-factoring

QR-factoring is the task of decomposing a matrix A into two matrices Q and R such that $A = QR$. Here Q is an orthogonal matrix and R is an upper triangular matrix. In contrast to LU-decomposition, QR-decomposition is not limited to square matrices. It is widely used in numerical methods, particularly for solving least squares problems and for eigenvalue computations. Common construction methods for the orthogonal matrix Q include Givens rotations and Householder transformations. The decomposition process typically involves a sequence of transformations, where A is multiplied by a series of orthogonal matrices Q_i. The product of these matrices forms Q and the resulting matrix becomes upper triangular [9]. While the classical process of QR-factoring is computationally intensive, [22] proposes a different perspective by using a QR-factoring as an example of a one-way function. A one-way function is defined as an injective function $f : D \to W$ between two non-empty sets D, W, where the evaluation of $f(x)$ for all $X \in D$ can be performed efficiently, but computing the inverse $f^{-1}(y)$ for $y \in f(D)$ is computationally hard [23]. For QR-factoring in [22] instances of the problem are generated by sampling pairs by $f_{qr}^{-1}(\{(q,r)_i\}_{i=1}^{m})$ with $(q,r)_i$ sampled from $P \times P$ of a precomputed set of primes $P \subset \mathbb{P}$. The product $x = qr$ then serves as the problem instance, while finding the factors q and r represents the hard inversion task.
To represent these instances of QAOA, the binary representation $x_{(2)}$ of $x \in \mathbb{N}$ is used. Let $z \in \mathbb{F}_2^k$ denote the corresponding bit string of length k that is padded with $n - k$ leading zeros, if necessary, to obtain the fixed size n. With $n = max\{\lceil ld(q) \rceil, \lceil ld(r) \rceil$ the mapping of a solution (q,r) to the target space T is then performed via $(q,r) \mapsto p_n(q_{(2)}) \circ p_n(r_{(2)})$ where $\circ$ denotes the concatenation of two bit strings. This encoding allows for a structured quantum representation of the qr-factoring problem.

4.3 Join Order Optimization

Another interesting problem arises from the field of relational databases. The task of Join Order Optimization is to find an optimal sequence in which to join multiple relations to minimize the cost function evaluated from a query graph. An example for a query graph is given in figure 4.1 A query graph is a graph where the vertices represent base relations and the edges represent join predicates between those relations. Formally, a query graph is described as $G = (V, E)$ where V is a set of nodes (relations) and E set of edges (join predicates). This graph can be compactly represented by an adjacent matrix $G \in F_2^{r \times r}$ where r is the number of relations. For a node $v_i \in V$ with $i \in \mathbb{N}, 1 \leq i \leq |V|$ the corresponding relation is denoted by R_i. A join predicate pi, j connects two relations R_i and R_j, and each predicate is associated with a selectivity,

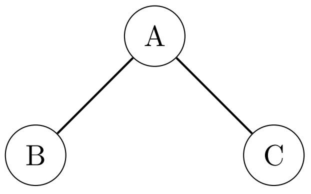


Figure 4.1: Example Query Graph

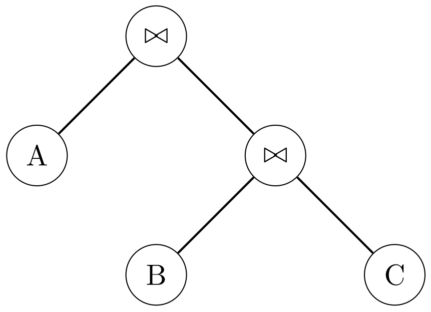


Figure 4.2: Example Join Tree

which measures the percentage of tuples surviving the join compared to the total number of tuples in the Cartesian product of the two relations. From a given query graph, one or more join trees can be constructed, representing different ways to execute the join operations. An example of a join tree is shown in figure 4.2. The leaf nodes correspond to base relations and the internal nodes represent join operations. The structure is that of a binary tree [24].

The input to the join order optimization problem consists of the join graph, a set of considered join trees and a cost function. The cost function assigns a cost value to each join tree, based on an estimate of query execution complexity. There are multiple possible cost functions with varying degrees of accuracy and computational difficulty. In the reinforcement learning approach in [16], the cost function approximates the complexity based on the cardinalities of intermediate results: $C_{out}(T) = |T| + C_{out}(T_1) + C_{out}(T_2)$ where $T = T_1 \bowtie T_2$ denotes a join of two subtrees, $|T|$ is the true cardinality (number of tuples) of the result and $C_{out}(T) = 0$ if $T \in r_1, r_2, ...$ is a leaf.

5 Experimental Setup

The Experiments will replicate parts of the experimental results of two papers on the LRZs Quantum Hardware QExa20. In the first part there is given an introduction to the hardware specifics. The goal of this chapter is to compare the Quality of the results from the Quantum Simulation with the results of the QExa20 as well as the evaluation of the performance of the system. The system has a tokenized authentication, which tokens can be generated over the online portal *Munich Quantum Portal.* An own python package called *MQP Provider* includes all functions to connect and communicate with the system. For the experiments in this thesis a slightly modified version of the *MQP Provider* v0.1.4 was used, but the modifications have been only for the purpose of logging the job responses.
The system accepts python code implemented with the python package *Qiskit.* The progress of an ongoing experiment can be watched in the online portal, giving an overview of the triggered jobs with their current status.
In experiments from the paper [16] have been already implemented with Qiskit, using the *Qiskit Aer Simulator* for simulating a quantum backend. So the Simulator could be conveniently exchanged with QExa as backend device after installing the *MQP Provider* python package. Surprisingly, no further adjustments on the Code needed to be done to run the experiments. This was a good opportunity to get used to the workflow with the system and gather first results rather quickly.
More effort took the porting of the experiments from the paper [22]. Because the work introduces two new theorems for approximating the parameter of QAOA to create a non-iterative version of this algorithm, the experiments concentrated on showing the quality of this new approach. That means, for the experiments a quantum processor was simulated with *QuTip.* QuTip is an open source tool for simulating quantum systems numerically, depending on the python packages *Numpy*, *Scipy* and *Cython* [10]. Therefore, the Code had to be translated into Qiskit, but also the parameter had to be altered to not exceeding the limited resources of the QExa20.
Most experiments took place from the November 2024 until January 2025. The mentioned issues refer to this time and do not reflect the current system state.

5.1 QExa

The QExa20 is a NISQ-Computer from IQM using superconductivity to realize qubits. The 20 qubit system, stationed in and run by the team of Leibniz Supercomputing Centre in Munich, is the first of its kind having its control electronics directly connected to a high-performance computer. The connected machine is the SuperMUC-NG, also stationed at LRZ, that has over 300.000 compute cores with a main memory of 719 TB in total and a peak performance of 26.9 Petaflops/s.[13]
The system can hold an entangled Greenberg-Horne-Zeilinger state with 19 qubits without readout error mitigation and a fully entangled 20 qubit GHZ state with readout error mitigation. It has an average one qubit gate fidelity of 99.94% (over 20 qubits and an average two qubit gate fidelity over 30 qubit pairs of 99.49%.[4]
The QExa20 system is part of the project Ëuropean Quantum Computing for Exascale-HPC", short Euro-Q-Exa, funded by EuroHPC Joint Undertaking that is funding the

construction and running of six hybrid quantum systems planned for Europe. It is also financed by the Federal Ministry of Education and Research and the *Bavarian state of ministry of science and art* as a part of the Munich Quantum Valley. [5] While QExa20s pilot phase started in June 2024, it is planned to integrate another 54 qubit system in the second half of 2025 and extend it with a 105 qubit system in 2026 into the HPC architecture. [6]
To access the QExa20, a user must have access to the web portal *Munich Quantum Portal* or a direct HPCQC access. The second access possibility is a programming interface implemented as a C library to express quantum circuits. This approach will not to be examined in this thesis. The Munich Quantum Portal allows using multiple quantum backends and simulators directly over Qiskit with the python package *MQPProvider*, which provides a basic API for communication. To utilize the backend the resource information for the qubit count, the coupling map and native instructions is directly retrieved from the addressed system. To launch a job submission the system will take the resource name, the circuit, the circuit format, shot count and a no modify flag. Each job can be identified by a unique job ID with a default value of 10,000 shots.
The waiting time for the results is strongly bounded by the time needed for data transmission, the scheduling processes inside the Munich Quantum Stack environment and the system utilization. To get some sense of time, three time stamps are given back by the system: *submitted, scheduled, completed*. Unfortunately the granularity is in seconds and not enough to make precise statements about the execution time.
The circuit result is given back in a dictionary that assigns the result bit string with the amount of occurrences of the job.
For automatically adapting to to changing physical characteristics and constraints of the system, the LRZ developed the Quantum Device Management Interface (QDMI). It is an abstraction layer that retrieves direct information from the quantum resources, manages sessions and enables device control [27].
As described in section 2.2.4, the native gate set of the QExa20 includes the Rotation- and the CPhase gate. The coupling graph for the system is shown in 5.1.

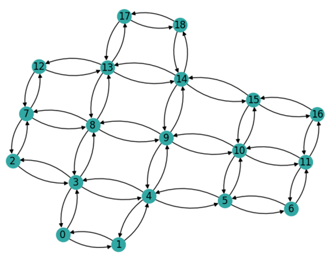


Figure 5.1: QExa20 coupling map

5.2 Join Order Optimization with QRL

The paper [16] examines the ansatz of using quantum reinforcement learning for the join order problem. The optimal join tree is discovered by multi-step quantum reinforcement learning with *Proximal Policy Optimization* as baseline for the *Markov decision process.*

For the experiments a test dataset was trained one including 12000 subqueries. Through the lack of sufficiently large quantum machine, each query includes four relations. 500 queries are used as dataset for the experiments, with 497 randomly chosen queries from the generated data set and the 3 available queries from the join order benchmark with four relations. On this Dataset a ten-fold cross validation is applied, which is a method in machine learning for evaluating a model design. Each group is set to be the test dataset once, all the others are declared to be the training dataset for this one evaluation. Then the model is fitted on the training set and evaluated on the test set [2].
For the implementation the python libraries *Tensorflow*, for machine learning specifics, *Tensorflow Quantum* to simulate an ideal quantum system and *Qiskit*, to simulate a noisy system, are used. Originally exist three configurations for the experiments. For one the variational quantum circuit (VQC) acts as critic together with a classical neural network as actor, the other has a classical critic and a VQC as an actor and the third one uses for critic and actor a VQC.
To introduce *data re-uploading* (DRU) resulting in 8, 12, 16 and 20 layers coming from training with four relations per query. The same amount of variational layers is used for the configuration without DRU, including an experiment with four layers.
For the experiments in this paper, the variational quantum circuit is employed as the agent. The experiments were done with and without data reuploading, resulting in three runs without data reuploading for four, twelve and 20 layers and two runs for 12 and 20 layers with data reupload. In figure 5.2 the results for the experiment are shown. It can be seen that the results from the QExa training is likely of the one percent depolarising error from the simulation based training process, also in the case of more variational layers. When having a closer look on figure 5.3 can be seen that the depolarising error is even below the one percent error rate. Not all possible variations of the experiments have been executed, not just because of redundancy. One experiment was taking the QExa20 system over 24 hours to finish. From start to end, there have been triggered 300 jobs, each with their maximum shot count of 10000 for each execution. Also there have been some issues with the continuity of the execution. Each night around 4:00 a.m. all jobs that were in the queue stopped proceeding further. They could not be aborted, what lead to rerunning the experiment, starting at the point where the last run stopped.

5.3 Non-iterative QAOA

These two theorems presented in [22] are applied to four different problems: uniform random sampling, clustered sampling, Boolean Satisfiability and qr-factoring as a one-way function. Uniform random sampling was used as an introducing example to demonstrate, that the structure of a given target space, with respect to Hamming distances between targets, can be modelled analytically. The expected optimization landscape can be approximately derived solely from this model. For this problem, instances in state space dimensions form $8 \leq n \leq 11$ were sampled. The upper limit of $n = 11$ was chosen to keep the level of computational cost reasonable, while $n = 8$ ensures a sufficiently large state space. The same applies to the experiment with clustered sampling.
For both experiments exceeds the lower bound of $n = 8$ the limit of the QExa20. To make the experiment still executable, n must be reduced to six or fewer. Since these two experiments were mainly in the paper to illustrate the effectiveness of the theorems

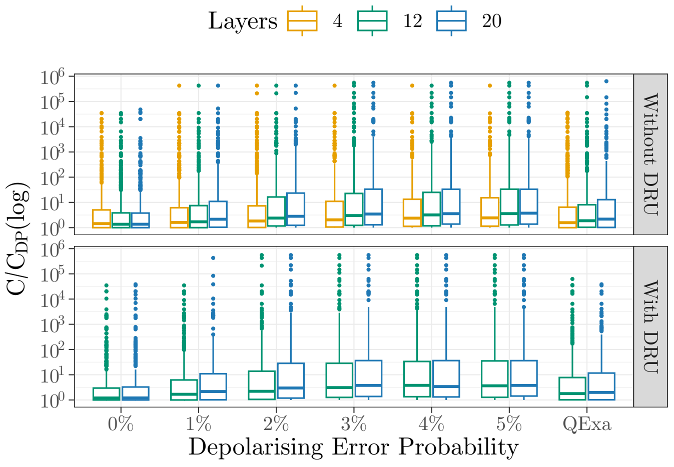


Figure 5.2: Comparison of relative cost after training with VQC as actor with different for QExa20 and Qiskit-Simulation

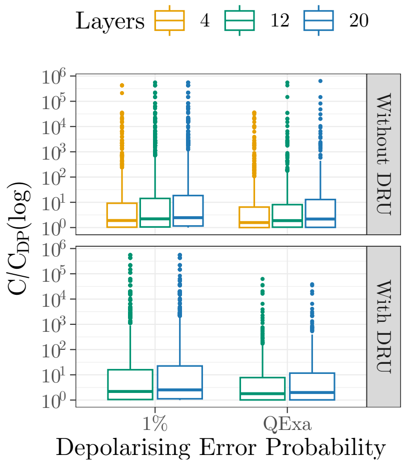


Figure 5.3: Zoom in onto simulation with 1% depolarising error and QExa20 results

, and given the extremely long execution times on QExa20 (12 to 24 hours per run for a single value of n), these introductory experiments were not reproduced.

The experiments of greater practical relevance are SAT and qr-factoring. For the SAT problem 500 random instances were sampled for $n = 6$ where n denotes the number of boolean variables and $|C| = 4n$ represents the number clauses. Figure 5.4 visualizes the results. At first glance, the simulation clearly outperforms the execution on real quantum hardware for all values of α. This is primarily due to the fact that the simulation with *QuTip* is noiseless. When only comparing the two QAOA variants for the QExa, the non-iterative QAOA achieves results comparable to the standard QAOA. In the qr-factoring experiment, the instances in the paper were sampled using all prime numbers up to 61. Since the number of primes directly influences the amount of required qubits n via $n = max\{\lceil ld(q)\rceil, \lceil ld(r)\rceil$, the set of primes was reduced to $P = \{2, 3, 5, 7\}$. Figure 5.5 shows that, overall, the solution probability is much lower than for the SAT

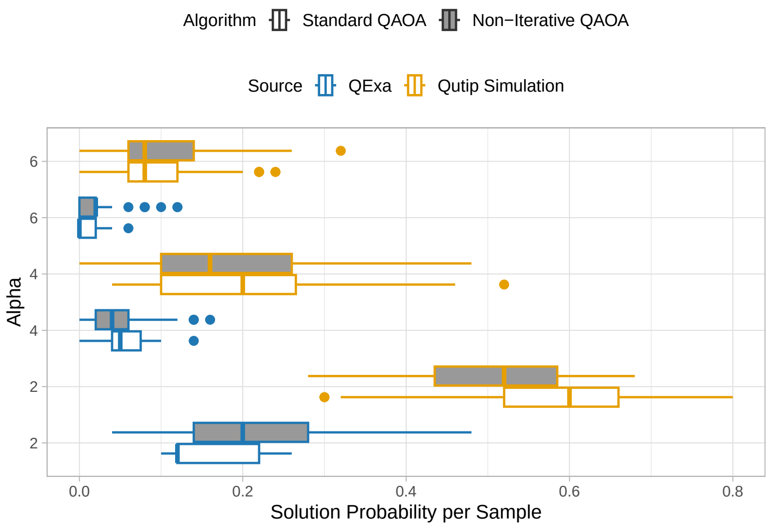


Figure 5.4: SAT experiment

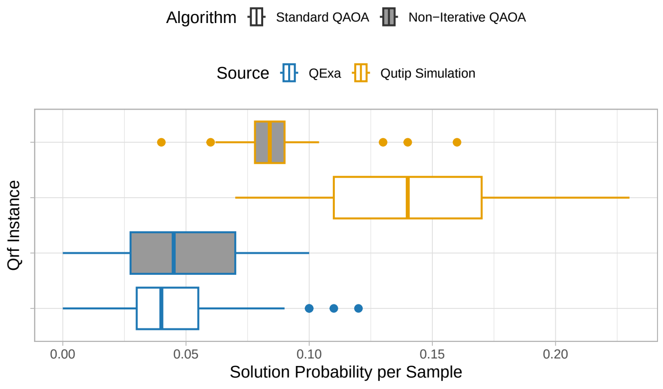


Figure 5.5: Qr-factoring experiment

experiment, yet similar patterns can be observed: The simulation outperforms the QExa results, likely due to noise. Nevertheless, both QAOA variants perform similarly, with a slightly higher median for the non-iterative QAOA in this case.
For experiments on the QExa, QAOA was implemented explicitly constructing the two Hamiltonians and applying them onto a circuit with n qubits, closely following the *QuTip*-based simulation. Due to the systems limitation, n had to be lowered to six. The constructed Hamiltonians transpile into very deep and wide circuits. For example, with $n = 6$ transpiled circuit depths range from 10.000 to 12.000 gates with widths up to 26. In comparison, trainspiled circuits with $n = 8$ exceeded 160.000 gates in depth and width increased to 28, leading to the QExa20 rejecting the circuit with the message: "Job cancelled: Circuit Execution Error on IQM Backend".
Additionally, QAOA circuits could not be executed on QExa20 using *Qiskits* built in *QAOAAnsatz()* function. Even for constructed circuits of the same size, the function failed, returning the same error as above. The issue was resolved by manually implementing the QAOA circuits.

6 Conclusion

The goal was to port the experiments of [22] and [16] onto the QExa20 and compare the results and evaluating the performance of the system. In the experiments it could be shown that the depolarizing error of the system is really low and also that non-iterative QAOA has a similar solution probability to standard QAOA. The execution time is really high and working with the system is difficult, because of very undetailed information about the system. The error messages were unprecise and all job executions will be aborted around 4 a.m.
Considering the challenges of the current system, a big field of research would be to design a more efficient interface or integration of quantum hardware. Running a network utilizing several quantum devices and simulators with multi-user accessibility is a big topic in itself, especially if the goal is to obtain and hold up the quantum advantage. For the current state of quantum hardware, it could also be interesting to design more algorithms like the non-iterative QAOA that minimizes the usage of the quantum resource.

6.1 Discussion

The access to the QExa20 over the Munich Quantum Portal is very convenient to use. Besides some start issues, like page loading failures and an automated logout after 15 minutes even while using the page, the simple design gives a straightforward overview on the currently available resources, token generation, and job execution.
With the first experiment with quantum reinforcement learning for the join order problem could be obtained, that the QExa20 has a really low depolarization error. What is a really nice result in addition to the days of execution time needed to gather all the data for the graph in 5.2. It is worth mentioning that the already high execution times for this experiments were increased by hours, because of a miracle cancelation of every executing job sequence at around 4 a.m. That the reason for this event is still unknown shows just once more, how complex the whole design and management of creating such a network with multiple quantum devices is.
Surprisingly, QExa20 could execute these experiments without any code adjustments, while the system struggled with a very low value of n as the circuit width for QAOA circuits. The transpiled unitary operators caused circuit depths of over 15.000 that was very close to the limit of circuit sizes the circuit could handle. It is also worth mentioning that the information for a non-executable circuit gives only very vague information about the failed process with "Error on IQM backend", to just cite one example. A clear error message at this point would not only improve the user experience significantly, but could also be the foundation of future system development and maintenance.
The experiments for SAT and QR-factoring show that the non-iterative QAOA can also compeed in solution probability with the standard QAOA on Quantum Hardware. What is the significant difference between these two approaches executed on the QExa is the runtime. Because the non-iterative QAOA needs only one execution of the QAOA circuit, the whole experiment execution time was around 45 minutes, while the execution of standard QAOA needs through the alternating circuit execution and optimization takes around 18 hours. This is the most significant disadvantage of utilizing the QExa.

The process of sending the job request, partitioning it into jobs, assumingly another transpilation process, waiting in the scheduler until the jobs can be executed, the actual execution and sending it back takes a significant amount of time. Because concrete information about the whole Munich quantum stack is not available, it is unclear which exact step consumes so much time. For more clarity of the processes happening before actually executing the circuit, a better log of timestamps with clear positions could be useful, especially for research purposes. The approximation approach for the non-iterative QAOA is also an interesting field of research in case of efficient resource utilization, especially for nowadays, where quantum resources are very limited.

Bibliography

[1] Adriano Barenco et al. “Elementary gates for quantum computation”. In: *Physical Review A* 52.5 (Nov. 1995), pp. 3457–3467. ISSN: 1094-1622. DOI: 10.1103/physreva.52.3457. URL: http://dx.doi.org/10.1103/PhysRevA.52.3457.

[2] Jason Brownlee. *A Gentle Introduction to k-fold Cross-Validation.* Oct. 2023. URL: https://machinelearningmastery.com/k-fold-cross-validation/.

[3] Jean-Luc Brylinski and Ranee Brylinski. *Universal quantum gates.* 2001. arXiv: quant-ph/0108062 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/0108062.

[4] Leibnitz Supercomputing Centre. *Deutschlands erster hybrider Quantencomputer am Leibnitz-Rechenzentrum.* June 2024. URL: https://doku.lrz.de/supermuc-ng-10745965.html.

[5] Leibnitz Supercomputing Centre. *Quantencomputer für Europa.* June 2023. URL: https://www.quantum.lrz.de/de/bits-von-qubits/detail/quantum-computer-for-europe.

[6] Leibnitz Supercomputing Centre. *Von IQM ausgestattet, vom LRZ gehostet: Euro-Q-Exa, der hybride Quanten-Supercomputer des EuroHPC Joint Undertaking (EuroHPC JU) für die Forschung.* Oct. 2024. URL: https://www.quantum.lrz.de/de/bits-von-qubits/detail/iqm-selected-to-deliver-two-advanced-quantum-computers-as-part-of-euro-q-exa-hybrid-system.

[7] M. Cerezo et al. “Variational quantum algorithms”. In: *Nature Reviews Physics* 3.9 (Aug. 2021), pp. 625–644. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9. URL: http://dx.doi.org/10.1038/s42254-021-00348-9.

[8] Elias F. Combarro and Samuel Gonzales-Castillo. *A Practical Guide to Quantum Machine Learning and Quantum Optimization.* Ed. by Rosal Colaco, Maran Fernandes, and Safis Editing. 1st ed. Birmingham, UK: Packt Publishing Ltd., Mar. 2023. ISBN: 987-1-80461-383-2.

[9] Wolfgang Dahmen and Arnold Reusken. *Numerik für Ingenieure und Naturwissenschaftler.* 3rd ed. Springer-Lehrbuch. Springer Spektrum Berlin, Heildelberg, Sept. 2022. ISBN: 978-3-662-65181-0. DOI: https://doi.org/10.1007/978-3-662-65181-0.

[10] QuTiP developers and contributors. *QuTip Quantum Toolbox in Python.* URL: https://qutip.org/.

[11] IBM Quantum Documentation. *Qiskit SDK v1.1, CPhaseGate.* 2024. URL: https://docs.quantum.ibm.com/api/qiskit/1.1/qiskit.circuit.library.CPhaseGate.

[12] IBM Quantum Documentation. *Qiskit SDK v1.1, RGate.* 2024. URL: https://docs.quantum.ibm.com/api/qiskit/1.1/qiskit.circuit.library.RGate.

[13] LRZ Dokumentationsplattform. *SuperMUC-NG.* Leibnitz Supercomputing Centre. n.d. URL: https://doku.lrz.de/supermuc-ng-10745965.html.

[14] Daoyi Dong et al. “Quantum Reinforcement Learning”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38.5 (Oct. 2008), pp. 1207–1220. ISSN: 1083-4419. DOI: 10.1109/tsmcb.2008.925743. URL: http://dx.doi.org/10.1109/TSMCB.2008.925743.

[15] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. Version 1. Nov. 2014. arXiv: 1411.4028. URL: https://arxiv.org/pdf/1411.4028v1.

[16] Maja Franz et al. “Hype or Heuristic? Quantum Reinforcement Learning for Join Order Optimisation”. In: *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Sept. 2024, pp. 409–420. DOI: 10.1109/qce60285.2024.00055. URL: http://dx.doi.org/10.1109/QCE60285.2024.00055.

[17] Jack D. Hidary. *Quantum Computing: An Applied Approach*. 2. Auflage. Springer, 2021. DOI: https://doi.org/10.1007/978-3-030-83274-2.

[18] Matthias Homeister. *Quantum Computing verstehen*. 6th ed. Birmingham, UK: Springer Vieweg Wiesbaden, Feb. 2022. ISBN: 978-3-658-36434-2.

[19] Ryszard Horodecki et al. *Quantum entanglement*. Version 2. arXiv:quant-ph/0702225v2. Apr. 2007. arXiv: 0702225. URL: https://arxiv.org/pdf/quant-ph/0702225v2.

[20] Antoine Jacquier and Oleksiy Kondratyev. *Quantum Machine Learning and Optimization in Finance*. Ed. by Saby D’silva et al. Birmingham, UK: Packt Publishinh Ltd., Oct. 2022. ISBN: 987-1-80181-357-0.

[21] Gregor Kemper and Fabian Reimers. *Lineare Algebra*. 1st ed. Springer-Lehrbuch. Springer Spektrum Berlin, Heildelberg, Mar. 2022. ISBN: 978-3-662-63724-1. DOI: https://doi.org/10.1007/978-3-662-63724-1.

[22] Tom Krüger and Wolfgang Mauerer. *Out of the Loop: Structural Approximation of Optimisation Landscapes and non-Iterative Quantum Optimisation*. 2024. arXiv: 2408.06493 [quant-ph]. URL: https://arxiv.org/abs/2408.06493.

[23] Burkhard Lenze. *Basiswissen Angewandte Mathematik – Numerik, Grafik, Kryptik*. 2nd ed. Springer Vieweg Wiesbaden, Aug. 2020. ISBN: 978-3-658-30028-9. DOI: https://doi.org/10.1007/978-3-658-30028-9.

[24] Manuel Schönberger. “Applicability of Quantum Computing on Database Query Optimization”. In: *Proceedings of the 2022 International Conference on Management of Data*. SIGMOD ’22. Philadelphia, PA, USA: Association for Computing Machinery, 2022, pp. 2512–2514. ISBN: 9781450392495. DOI: 10.1145/3514221.3520257. URL: https://doi.org/10.1145/3514221.3520257.

[25] Steven S. Skiena. *The Algorithm Design Manual*. 3rd edition. 6330 Cham, Switzerland: Springer, 2020. ISBN: 987-3-030-54255-9.

[26] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. second edition. The MIT Press, 2018. ISBN: 978-0-262-03924-6.

[27] Robert Wille et al. *QDMI – Quantum Device Management Interface: Hardware-Software Interface for the Munich Quantum Software Stack*. URL: https://www.cda.cit.tum.de/files/eda/2024_qce_qdmi.pdf.

[28] Hiu Yung Wong. *Introduction to Quantum Computing*. 2nd ed. 6330 Cham, Switzerland: Springer Nature, 2023. DOI: `https://doi.org/10.1007/978-3-031-36985-8`.