TU WIEN Informatics

Benchmarking Of Quantum Edge On The Computational Continuum

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Dominik Jandl
Matrikelnummer 11808611

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Vincenzo De Maio, Ph.D.
Mitwirkung: Univ. Prof. Dr. Ivona Brandić

Wien, 30. Jänner 2024

Dominik Jandl	Vincenzo De Maio

TU WIEN Informatics

Benchmarking Of Quantum Edge On The Computational Continuum

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Dominik Jandl
Registration Number 11808611

to the Faculty of Informatics

at the TU Wien

Advisor: Vincenzo De Maio, Ph.D.
Assistance: Univ. Prof. Dr. Ivona Brandić

Vienna, 30th January, 2024

Dominik Jandl	Vincenzo De Maio

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Dominik Jandl

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 30. Jänner 2024

Dominik Jandl

Danksagung

Ich möchte diese Gelegenheit nutzen, um meinem Betreuer Vincenzo De Maio, Ph.D. und meiner Co-Betreuerin Univ. Prof. Dr. Ivona Brandić, für die äußerst angenehme Zusammenarbeit und die problemlose Kommunikation während dieser letzten Phase meines Studiums zu danken. Deren Hingabe und technische Unterstützung, begleitet von schnellem, konstruktivem Feedback, haben eine bedeutende Rolle im Fortschritt meiner Masterarbeit gespielt. Durch die fachliche Expertise und die Möglichkeit mit neuesten Technologien zu arbeiten, wurde für mich eine ganz neue Sicht auf die wissenschaftliche Herangehensweise an Probleme geschaffen.

Weiterer Dank gebührt meinen Studienkollegen und sehr guten Freunden, Sven Bombera, Michael Martinek und Maximilian Sutrich. Ohne eure Unterstützung und den starken Zusammenhalt wäre die Absolvierung des Masterstudiums in einem so erfolgreichen Rahmen nicht möglich gewesen. Der stetige Austausch und die motivierenden Worte in herausfordernden Zeiten sind wesentliche Aspekte, für die ich sehr dankbar bin. Die zahllosen schlaflosen Nächte haben uns zu einem untrennbaren Team gemacht. Darauf bin ich sehr stolz und hoffe unsere Freundschaft bleibt noch lange bestehen.

Abschließend möchte ich meiner Familie, insbesondere meinen Eltern Sabine und Johannes Jandl, einen besonderen Dank aussprechen. Sie haben mir das Studium an der Technische Universität Wien ermöglicht. Ihr Glaube an mich und ihre Unterstützung haben sich nicht nur während meiner gesamten Studienzeit, sondern über die letzten 25 Jahre in allen Lebensbereichen gezeigt. Ohne die aufmunternden Worte und die stets optimistische Einstellung wäre ich nicht in der Position, in der ich mich heute befinde. Zum Schluss: Bitte entschuldigt die gelegentlich etwas gereizten Antworten meinerseits während stressiger Prüfungszeiten.

Vielen Dank.

Acknowledgements

I would like to take this opportunity to sincerely thank my advisor Vincenzo De Maio, Ph.D. and co-advisor Univ. Prof. Dr. Ivona Brandić, for the extremely pleasant collaboration and delightful communication during the final phase of my studies. Their dedication and technical support, accompanied by prompt and constructive feedback, played a significant role in the progress of my master's thesis. Through their expertise and the opportunity to work with the latest technologies, a completely new perspective on the scientific approach to problems has been created for me.

Further thank you to my fellow students and very dear friends, Sven Bombera, Michael Martinek, and Maximilian Sutrich. Without your support and strong team spirit, the completion of the master's program in such a successful manner would not have been possible. Continuous exchange and motivating encouragement during challenging times are essential aspects for which I am very grateful. Countless sleepless nights have made us an inseparable team. I am very proud of that unique relationship and hope our friendship will continue for a long time.

Lastly, my special thanks go to my family, especially my parents, Sabine and Johannes Jandl, who made it possible for me to study at Vienna University of Technology. Their belief in me and their support have not only been evident throughout my entire academic journey but have been spanning over the past 25 years in all aspects of life. Without their encouraging words and their consistently optimistic attitude, I would not be in the position I find myself today. In conclusion, please excuse my occasional somewhat irritable responses during stressful exam periods.

Thank you very much.

Kurzfassung

In letzter Zeit hat es aufgrund der Fortschritte im Bereich des Quantencomputings ein bemerkenswertes Interesse an wissenschaftlichen Anwendungen gegeben, insbesondere in Bereichen wie der Quantenchemie. Quantencomputing bietet eine vielversprechende Möglichkeit, komplexe Probleme zu lösen. Darunter fällt zum Beispiel die Simulation komplizierter Experimente, die darauf abzielen, Einblicke in Naturphänomene zu gewinnen. Insbesondere in der Quantenchemie ist es gut vorstellbar, dass diese neue technologische Möglichkeit konventionelle Berechnungsverfahren revolutioniert. Daher birgt diese neue technologische Möglichkeit das Potenzial, wertvolle Ergebnisse, beispielsweise bei der Unterstützung der Arzneimittelentwicklung und der Erforschung neuer molekularer Verbindungen, zu liefern.

Allerdings sind derzeitige Quantenmaschinen hinsichtlich Verfügbarkeit und Zuverlässigkeit limitiert. Während auf einige Quantencomputer über die *Cloud* zugegriffen werden kann, bringt Cloud Computing seine eigenen Herausforderungen, wie erhöhte Latenz und Netzwerküberlastung, mit sich. Die Nutzung von Quantumcomputing in der *Cloud* stellt daher für Anwendung welche kurze Reaktionszeiten erwarten keine Option dar. Als Lösung für diese Herausforderungen könnte das bereits weit verbreitet Edge-Geräte Paradigma dienen. Am Netzwerkrand platziert, haben diese Geräte das Potenzial, Latenz und Netzwerküberlastung zu reduzieren.

Variational Quantum Algorithms (VQAs) sind ein vielversprechender Ansatz zur Lösung bestimmter Probleme im Zusammenhang mit der Quantenchemie. VQAs werden über verschiedene Hyperparameter konfiguriert, die ihre Ergebnisse erheblich beeinflussen. Darüber hinaus spielt die Wahl des zugrunde liegenden *Backends* eine wichtige Rolle für das Endergebnis.

In unserer Studie kombinieren wir den Einsatz von VQAs zur Problemberechnung mit dem Paradigma des *Edge Computing*. VQAs werden auf lokalen und Edge-Quantensimulatoren ausgeführt, um Einblicke in ihr Verhalten zu gewinnen. Das Hauptziel dieser Arbeit besteht darin, verschiedene Hyperparametereinstellungen für VQAs zu vergleichen und ihre Leistung anhand von Metriken wie Genauigkeit, Laufzeit und Skalierbarkeit zu bewerten. Schlussendlich sollen die optimalen Konfigurationen für die verschiedenen VQAs ermittelt werden.

Die Ergebnisse zeigen, dass die Ausführungszeit für alle drei Algorithmen und Experimente eng mit der Größe der Eingabeproblemstellung zusammenhängt und Hardwareressourcen

eine entscheidende Rolle spielen. Darüber hinaus zeigen die Berechnungsergebnisse aus den Quantensimulatoren, dass sie Annäherungen an reale Ergebnisse darstellen und daher in Anwendungen nützlich sind, die keine exakte Werte erfordern, sondern mit approximierten Ergebnissen arbeiten können.

Abstract

In recent times, there has been a notable interest in scientific applications, especially within domains like quantum chemistry, driven by advancements in the area of quantum computing. Quantum computing presents a promising opportunity for addressing complex challenges, such as the simulation of complex natural phenomena. Its potential to revolutionize conventional computational methods, particularly in quantum chemistry, promises valuable real-world applications like drug design and the exploration of new molecular compounds.

However, quantum machines currently face limitations in terms of availability and reliability. While some quantum computers are accessible via the cloud, cloud computing introduces its own challenges, including latency and network congestion. Consequently, quantum computing in the cloud is not suitable for applications requiring fast response times. As a solution to these challenges, edge devices have become popular in current computing paradigms. Placed at the network edge, these devices have the potential to reduce latency and network congestion.

Variational Quantum Algorithms (VQAs) are promising approaches to address certain quantum chemistry-related problems. VQAs are configured through various hyperparameters that significantly impact the result output. Additionally, the choice of the underlying backend, where the execution takes place, influences the final result.

In our study, we combine the use of VQAs for problem computation with the computing paradigm of edge computing. VQAs are executed on local and edge quantum simulators to gather insights into their behavior. The primary objective of this thesis is to compare different hyperparameter settings for VQAs and evaluate their performance using metrics such as accuracy, runtime, and scalability to determine the optimal configurations.

Our results indicate that, for all three algorithms and experiments, execution time is closely related to the size of the input problem instance and hardware resources play a crucial role. Furthermore, the calculation results from quantum simulators show that they provide approximations to real results and are useful in applications that do not require exact values but can work with approximated results.

Contents

CHAPTER 1

Introduction

1.1 Introduction

As scientific problems and applications become increasingly complex, we are entering in the Post-Moore era [Șt21], where classical computers may no longer provide the extensive resources required to address challenges where the time for solving the problem grows exponentially [EW06]. Quantum computing is a rapidly developing field that has the potential to revolutionize the way we approach certain computational problems [dLMPL19]. Quantum hardware can offer significant computational advantages and guarantee an speedup for different computational problems, e.g., integer factorization [Sho97], simulating real quantum systems such as Fermi Systems [AL97], search in unsorted databases [Gro96]. Various fields, such as finance, logistics, physics, and chemistry could experience significant improvements in solving complex computational problems [GKS+22] such as estimating stock risks, storage usage, simulating molecules and drug discovery. Quantum computers leverage quantum mechanical principles such as superposition and entanglement to perform calculations in a fundamentally different way than classical computers. Moreover quantum computers interact on quantum bits (qubits), which can be in multiple states simultaneously, rather than on classical bits [NC10].

One important area where quantum computing is showing promise is in the field of quantum chemistry [MEAG+20], where quantum computers can be used to simulate the behavior of molecules and materials. This deeper understanding holds the potential to refine existing knowledge and even facilitate the design of compounds with both scientific and industrial applications in the future. One promising approach could be the utilization of Variational Quantum Algorithms (VQAs) for chemistry problems.

As availability of quantum computers is limited, moving them to the cloud is a promising perspective. Quantum machines can be accessed from anywhere via the cloud and a

physical machine can be shared between multiple parties. That would offer cost efficient possibilities for scientists to run experiments without the need to deploy their own quantum systems [RSGC21]. For the deployment of quantum computers, specialized software and hardware are needed, which can be both expensive and challenging to develop and maintain [Kom20]. Therefore, many different Quantum Cloud Platforms are available for utilizing quantum machines in the cloud. A few of them are listed below:

- IBM Quantum Experience[1]
- Google Quantum AI [2]
- Microsoft Quantum Development Kit[3]
- Amazon Braket[4]
- Rigetti Forest and Cloud Computing Services (OCS)[5]

Listing 1.1: Overview of some Quantum Cloud Platforms

1.1.1 Problem Statement

A major problem with current quantum computers is limited availability. While quantum computers hold potential for solving computational challenges across various scientific domains, such as quantum chemistry, their effectiveness is constrained to a very limited extent. Since not everyone can currently access a quantum computer, Quantum Cloud Platforms as mentioned in 1.1 are an example of a way to use quantum computers in the cloud. This arrangement allows multiple users to utilize a single quantum machine, reducing the need for each user to handle the operation of such machines individually. However, the utilization via a cloud environment has disadvantages when it comes to solving problem instances, as latency and congestion can increase runtime [MYZ+17]. Furthermore, the ever-growing amount of data and the strain on network bandwidth contribute to the issues mentioned above [AFG+09]. Near real-time responses are critical in many applications and therefore the possibility of increased latency is a concern for many applications that use the cloud. These problems are already known from current cloud applications and have led to edge computing being established as a solution to these problems [RSGC21]. For this reason, in this work we take up the principle of edge computing and adapt it for quantum computers.

[1] https://quantum.ibm.com/
[2] https://quantumai.google/
[3] https://learn.microsoft.com/en-us/azure/quantum/overview-what-is-qsharp-and-qdk
[4] https://aws.amazon.com/de/braket/
[5] https://pyquil-docs.rigetti.com/en/v2.7.2/

1.1.2 Aim of Work

The objective of this thesis is to assess the feasibility of employing quantum simulators at the network edge with the intention of investigating the potential for offloading certain computational tasks to edge devices. We intend to build upon the observation that practical access to quantum computers remains limited, while exclusively relying on local processing might not be practically viable. Consequently, our focus lies in exploring the possibilities of offloading quantum computing operations to the network edge. In more detail we take a deeper look in Variational Quantum Algorithms (VQAs) and compare their results under different hyperparameter settings. Results will always be carried out remote and locally to analyze the difference in accuracy, speed and scalability. Our thesis focus on investigating the following research questions:

- Which hyperparameter settings lead to the best results for the different VQAs?
- How does the choice of hyperparameters, such as the backend, ansatz and optimizer, impact the runtime, accuracy and performance of VQAs?
- How does the runtime and accuracy of VQAs changes with increasing size of the problem instance?
- Is it feasible to offload quantum computation to the network edge to leverage the benefits of edge computing in the quantum domain?

To achieve the objective of benchmarking edge quantum simulators, this thesis compares the performance of edge and local quantum computing simulators on a range of certain problems and evaluate their effectiveness through benchmark tests. It examines the runtime and error of both architectures, considering recent research and developments in quantum algorithms, hardware, and software. Benchmarking of quantum algorithms is crucial because the hyperparameters employed within the algorithm can significantly impact both the runtime and the accuracy of the results. By systematically varying and analyzing these hyperparameters, it is possible to gain insights into their effects on the algorithm's performance. Through benchmarking, it becomes possible to identify the hyperparameter configurations that yield the best outcome, reducing runtime and minimizing errors.

1.1.3 Expected Contributions

This study aims to establish a proof-of-concept for offloading quantum computations to edge devices, thereby yielding valuable insights into how edge quantum simulators operate and perform. The concept of offloading has the potential to offer notable advantages in this domain by reducing latency and the risk for congestion between the application and the edge quantum simulator. Scientific problems play a crucial role in quantum computing, offering significant applications for this technology. We aim to demonstrate how the different hyperparameters impact the accuracy and runtime for different VQAs, which can be used for quantum chemistry problem solving.

Based on our results, this thesis will offer first insights in the possibility of using VQAs for different problem instances. Due to our findings we have provided benefits of using VQAs at the network edge and on the other hand point out current limitations.

We expect this thesis to provide a comprehensive evaluation of edge and local quantum simulators for certain problems, and to offer practical guidance for selecting the most appropriate algorithm settings for a given problem. By doing so, this thesis will contribute to the ongoing effort to develop practical applications for quantum computing and to understand the role that this technology can play in solving some of the most challenging problems in science and engineering.

1.1.4 Structure of Thesis

This thesis is structured as follows: Chapter 2 discusses the background of this thesis, beginning with an explanation of the main differences between quantum computers and classical computers. It covers important quantum mechanical principles, quantum bits, quantum circuits and gates, Variational Quantum Algorithms, and quantum error correction. Chapter 3 provides a overview of other studies with similar research goals that have already been conducted. These studies focus on hybrid classical-quantum systems, quantum computing, and Variational Quantum Algorithms. Chapter 4 covers the methodology used for obtaining all the benchmark results. Moving forward, Chapter 5 describes the implementation in detail. We proceed with presenting the results of this thesis and evaluating the implementation in Chapter 6, and conclude with a discussion of potential future work in Chapter 7.

CHAPTER 2

Background

2.1 Quantum Computing Primer

The following chapter will cover the main terms, technologies, and techniques of quantum computing, which are relevant for this thesis and will be briefly described.

2.1.1 Quantum Bits (Qubits)

In quantum computing, the qubit represents the basic unit of information, featuring a two-dimensional state space. Its state vector $|\psi(t)\rangle$ can be expressed as a combination of orthonormal basis states $|0\rangle$ and $|1\rangle$, using complex coefficients $C1(t)$ and $C2(t)$ [NC10]. These two-state quantum systems may arise naturally or be intentionally crafted, like the electron's spin degrees of freedom, leading to spin-up and spin-down states. In the domain of quantum computing, the basis states $|0\rangle$ and $|1\rangle$ are also denoted as computational basis states and qubits serve as the building blocks of quantum computers. The crucial distinction between classical and quantum information lies in quantum's capacity to superpose base states, while classical bits can only hold distinct states 0 or 1. With N qubits, the Hilbert space dimension expands to 2^N, necessitating $2^{N+1} - 2$ real numbers to represent an arbitrary superposition due to normalization and the constant phase factor ($e^{i\phi}$). Consequently, an N qubit system requires $2^{N+1} - 2$ classical bits to describe an arbitrary superposition [SMS23].

The behavior of qubits in quantum computing can be visualized as a sphere, known as the Bloch sphere, that is a common representation in quantum algorithms and quantum circuit analysis, aiding in the understanding of qubit dynamics. The Bloch sphere is a unit sphere, where the north pole represents the $|0\rangle$ state, and the south pole represents the $|1\rangle$ state. The equator of the sphere represents the equal superposition states, where the qubit is in a linear combination of $|0\rangle$ and $|1\rangle$. Any pure state of a qubit can be represented by a point on the surface of the Bloch sphere. For example, the state

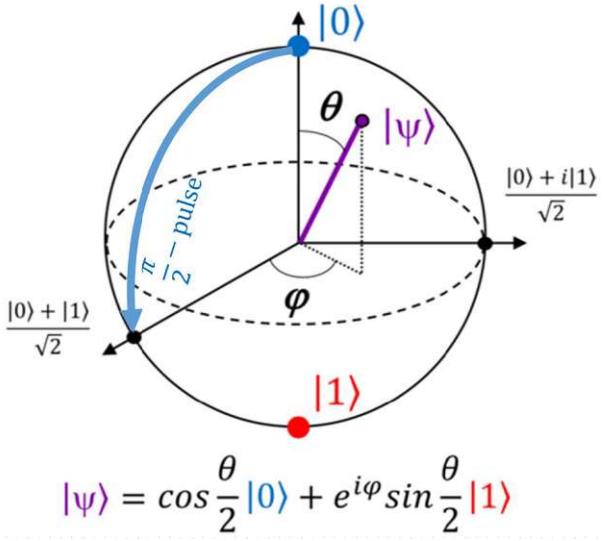


Figure 2.1: Schematic representation of a quantum state in a Bloch sphere. [JBTS19]

$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$ (to ensure normalization), corresponds to a point on the surface of the sphere. The angles θ and ϕ determine the position of the point on the sphere, and they are related to the complex coefficients α and β [JBTS19, Wil11]. The Bloch sphere provides an intuitive way to understand qubit operations. Quantum gates, which are transformations applied to qubits, correspond to rotations of the qubit's state vector on the Bloch sphere. A example illustration of a Bloch sphere is shown in Figure 2.1.

Superposition

Superposition [Pos21, dLMPL19] is a fundamental principle of quantum mechanics that allows a quantum system to exist in multiple states simultaneously. In the quantum world, particles such as qubits can be in a state that represents a linear combination of two or more basis states. For example, a qubit can be in a superposition of the states $|0\rangle$ and $|1\rangle$, represented as $\alpha\,|0\rangle + \beta\,|1\rangle$, where α and β are complex probability amplitudes. The probabilities of measuring the qubit in state $|0\rangle$ or $|1\rangle$ are set by the square of α and β given by

$$|\alpha|^2 + |\beta|^2 = 1.$$

The matrix representations of the vectors $|0\rangle$ and $|1\rangle$ are usually given by

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Superposition is a key concept in quantum computing that enables quantum systems to perform multiple computations simultaneously, providing a computational advantage

over classical systems. This principle is a cornerstone of quantum computing algorithms and allows for the potential of exponential speedup in certain computations[YM08].

Entanglement

In quantum mechanics entanglement is a phenomenon where two or more particles become intrinsically connected in such a way that the quantum state of one particle is dependent on the state of the others, regardless of the spatial separation between them. When particles become entangled, their individual quantum states can no longer be described independently. Their combined state forms a single entangled state [EPR35].

As an example of entanglement, let us consider a two-qubit register $|b_1 b_2\rangle$ initially in the state $|00\rangle$ [Hom22b]. In this scenario, we apply the Hadamard transformation to the first qubit and then the CNOT operation to the two qubits. This results in the following:

$$|00\rangle \xrightarrow{H\otimes I_2} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

$$\xrightarrow{CNOT} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

Now, when we measure the first qubit, the result of this measurement is either $|0\rangle$ or $|1\rangle$, each with a probability of $\frac{1}{2}$. In the first case, the subsequent state is $|00\rangle$. If we observe a measurement outcome of $|1\rangle$ for the first qubit, the two-qubit quantum register transitions into the state $|11\rangle$. When we subsequently measure the second qubit, we obtain the same result as for the first qubit: either both qubits are 0, or both are 1.
Before measuring $|b1\rangle$, the outcome of a measurement on $|b2\rangle$ was uncertain, both outcomes were equally likely. However, if $|b1\rangle$ has already been measured, the result of the measurement on $|b2\rangle$ is now predetermined. The same phenomenon occurs when we first measure $|b2\rangle$. The state generated in this process is also known as a Bell state, of which there are four different varieties [dLMPL19]:

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$
$$\left|\Phi^-\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$
$$\left|\Psi^+\right\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$
$$\left|\Psi^-\right\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

Entanglement can occur between qubits in quantum systems, leading to a correlation between their measurement outcomes, even when separated by large distances. This property of entanglement plays a crucial role in quantum information processing, enabling tasks like quantum teleportation and quantum communication. Moreover, entanglement

is essential in quantum algorithms, such as the quantum search algorithm (Grover's algorithm) [CBAK13] and quantum error correction [LB13]. Harnessing entanglement is a key goal in quantum technologies, as it offers the potential for enhanced computation, communication, and secure information transfer.

Measuring a Quantum State

In quantum computing, measuring a quantum circuit is a mandatory step to obtain a result or outcome from a quantum computation. The process of measurement in quantum mechanics is fundamentally different from classical measurements and is related to the principles of quantum superposition and collapse of the wave function. When a quantum circuit is executed, qubits are manipulated through quantum gates, allowing them to be in a superposition of states, representing both $|0\rangle$ and $|1\rangle$ simultaneously. However, upon measurement, the quantum state collapses to one of the basis states ($|0\rangle$ or $|1\rangle$) with a certain probability. The outcome of the measurement is a classical bit, representing either 0 or 1.The probability of obtaining a particular measurement outcome is determined by

the state of the quantum system just before the measurement. In particular, for a qubit in a superposition of states $|0\rangle$ and $|1\rangle$, the probabilities of obtaining the measurement outcomes $|0\rangle$ and $|1\rangle$ are given by the square of the magnitudes of the coefficients of $|0\rangle$ and $|1\rangle$ in the superposition. For example, consider a qubit in the state $\alpha|0\rangle + \beta|1\rangle$. Here, α and β are complex probability amplitudes. Than the quantities $|\alpha|^2$ and $|\beta|^2$ represent the probabilities of measuring the qubit in the states $|0\rangle$ and $|1\rangle$ [dLMPL19, Hom22b].

After the measurement, the quantum state collapses to the measured state, and subsequent measurements will always give the same result. This is known as the collapse of the wave function in quantum mechanics. The Schrodinger's cat thought experiment [Sch35] describes the behaviour of quantum measuring. It places a cat inside a sealed box with a radioactive atom that may or may not decay, triggering a poison release. According to quantum theory, until observed, the cat is considered both alive and dead simultaneously *(superposition state)*. Until opening the box *(measuring)* it is not possible to decide whether the cat is still alive *(qunatum state)*. It is important to note that the measurement

in quantum computing is probabilistic, meaning that we can only obtain one of the possible measurement outcomes with a certain probability. Repeated measurements on the same quantum circuit may give different outcomes due to the probabilistic nature of quantum mechanics [JJ12].

No-Cloning Theorem

In classical computing, it is possible to create copies of information by simply duplicating bits. However, in the area of quantum computing, the No-Cloning Theorem stands as a fundamental principle that restricts the direct copying of arbitrary quantum states. Proposed by Wooters and Zurek in 1982, this theorem asserts that an arbitrary unknown quantum state cannot be perfectly cloned [WZ82]. This has profound implications for

the nature of quantum information and distinguishes quantum systems from classical ones.

The No-Cloning Theorem has important consequences for various aspects of quantum computing, such as quantum cryptography, quantum teleportation, and quantum communication [Wei09]. Quantum cryptographic protocols, like the Quantum Key Distribution (QKD) schemes, rely on the principle that eavesdropping on quantum states would disturb their integrity, making them detectable [AGM06]. The No-Cloning Theorem provides a foundational basis for the security of such quantum communication protocols.

In essence, the No-Cloning Theorem underpins the delicate nature of quantum states and highlights a profound difference between classical and quantum information. While classical bits can be freely copied, in quantum computing qubits cannot be copied. This unique principle raises challenges and opportunities in the development of quantum algorithms and technologies [Hom22c].

Mathematical Formulation: We consider a quantum system in the state

$$|\phi\rangle \otimes |s\rangle \,,$$

where $|\phi\rangle$ is arbitrary and $|s\rangle$ is arbitrary but fixed. There is no unitary transformation that transforms this system for every $|\phi\rangle$ into the state

$$|\phi\rangle \otimes |\phi\rangle \,.$$

2.1.2 Quantum Circuits

Quantum Circuits

Quantum circuits serve as the fundamental framework underpinning quantum computing, playing a crucial role in executing quantum computations. Analogous to classical circuits that manipulate classical bits through logic gates, quantum circuits operate on qubits using quantum gates to perform quantum operations[MM12]. For a classical circuit the process is facilitated through logical gates like AND, OR, and NOT, which manipulate the binary bits to perform computations and problem-solving tasks. The set of logical gates can be further reduced, as OR can be expressed in terms of AND and NOT. By utilizing these logical gates, a circuit is constructed to compute results[dLMPL19].

A quantum circuit is constructed by applying a series of quantum gates to qubits as shown in Figure 2.2. Each quantum gate corresponds to a specific quantum operation, such as rotations, flips, or entanglement, influencing the quantum state of the qubits. The arrangement and order of quantum gates influence the overall computation executed by the quantum circuit. By initializing the circuit with known initial qubit states, quantum gates control the evolution of qubit states to perform computations. The operations performed by the gates are always reversible [Wit14, dLMPL19].

Quantum circuits follow the rules of quantum mechanics, including the concept of superposition, which allows qubits to exist in multiple states simultaneously. Additionally, quantum circuits harness entanglement, which correlates the states of entangled qubits, even when physically separated. Knowing these quantum phenomena is vital for devising efficient quantum algorithms and unlocking the potential of quantum computing.

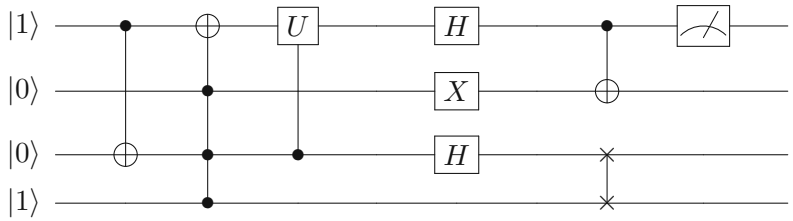
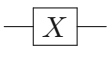


Figure 2.2: A sample quantum circuit with different quantum gates

Quantum Gates

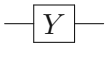
Quantum gates are the elementary building blocks of quantum circuits and are responsible for performing specific quantum operations on qubits. Each quantum gate represents a unitary transformation, which is a reversible operation that preserves the normalization and overall quantum state of the qubits [Kas21a].

There are various types of quantum gates, each with its unique function. Some of the commonly used quantum gates together with their gate symbols and transformation matrices are listed below [Hid19]:
Pauli-X (NOT) gate: Flips the state of a qubit from $|0\rangle$ to $|1\rangle$ or vice versa.



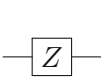
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Pauli-Y gate: Rotates the state of a qubit around the y-axis in the Bloch sphere.



$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Pauli-Z gate: Rotates the state of a qubit around the z-axis in the Bloch sphere, introducing a phase flip.

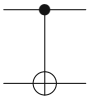


$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Hadamard gate: Puts a qubit into a superposition state, creating an equal probability of measuring $|0\rangle$ or $|1\rangle$.

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

CNOT (Controlled-NOT) gate: A two-qubit gate that performs a NOT operation on the target qubit if and only if the control qubit is in state $|1\rangle$.

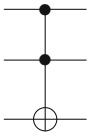


$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$$

SWAP gate: Exchanges the states of two qubits.

$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Toffoli gate: A three-qubit gate that performs a NOT operation on the target qubit if both control qubits are in state $|1\rangle$.



$$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$$

These gates, along with other more complex gates, enable the implementation of quantum algorithms and computations. Quantum gates exploit the unique properties of qubits, such as superposition and entanglement, to perform parallel computations and achieve quantum speedup for specific problems.

2.1.3 Hybrid Classic-Quantum Systems

The potential of quantum computers holds promise for accelerating problem-solving. A combination of classical and quantum systems can be envisioned as a viable solution. The main advantage lies in harnessing the strengths of classical computers for specific tasks, such as error correction, while leveraging the benefits of quantum machines, such as quantum parallelism [SLM+21]. Hybrid Classical-Quantum Systems involve a series of steps to enable interoperability between the two systems which are shown in Figure

2.3. Initially, data is pre-processed on the classical system (1) before being executed on the quantum system. Next, the quantum state is prepared based on the provided input (2). Utilizing quantum circuits to model the required computations (3), the state is manipulated, and subsequently measured (4). Finally, the measured state is transferred back to the classical system, where post-processing (5) takes place [CDMB+22].

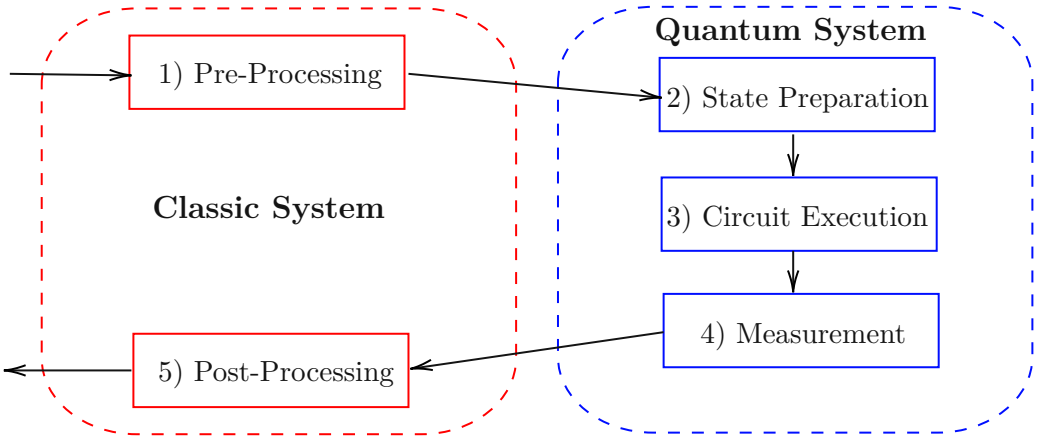


Figure 2.3: Hybrid Classic/Quantum Systems [CDMB+22]

2.1.4 Quantum Algorithms

Quantum algorithms leverage characteristics of quantum systems to address complex computational challenges with potentially greater efficiency compared to classical methods. Quantum algorithms are modeled through quantum circuits which moreover are build by using quantum gates. The design and optimization of these circuits play a crucial role in realizing the potential quantum speedup for specific computational tasks [BAAM20].

Variational quantum algorithms

Variational Quantum Algorithms (VQAs) use a classical optimizer to employ a quantum circuit with adjustable parameters, as illustrated in Figure 2.4. This approach has become popular for overcoming the limitations discussed previously [CAB+21].
In the following two variational quantum algorithms and one linear solver algorithm closely aligned with the focus of this thesis, are described.

Variational Quantum Eigensolver Variational Quantum Eigensolver (VQE) [PMS+14] yields the ground-state energy of a molecular Hamiltonian as result. VQE operates by leveraging the principles of variational optimization, where a parameterized quantum circuit is systematically adjusted to minimize the energy expectation value. The algorithm optimizes a function with the aim to retrieve the lowest possible expectation value of an observable relating to a trial wavefunction. To elaborate further, when a Hamiltonian operator $\hat{H}$ and a trial wavefunction $|\psi\rangle$ are provided, it becomes evident

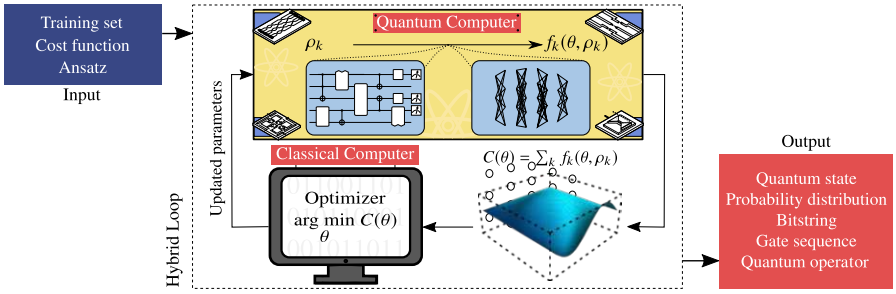


Figure 2.4: Schematic diagram of a Variational Quantum Algorithm (VQA) [CAB+21]

that the ground state energy E_0 linked to this specific Hamiltonian is confined within the bounds of

$$E_0 \leq \frac{\langle\psi|\,\hat{H}\,|\psi\rangle}{\langle\psi|\psi\rangle}.$$

The key distinguishing goal of VQE is that it is restricted to finding the eigenstate of a quantum observable, which is not necessarily the case of other VQAs (such as Quantum Approximation Optimization Algorithms or the Variational Quantum Linear Solver). Quantum advantage, achievable through VQE, depends on surpassing classical accuracy and computational time. However, challenges arise from the pre-factor of the VQE runtime, optimization complexity, and potential sensitivity to noise. VQE's resilience to noise contributes to its success on current quantum devices, but its scalability to larger experiments is uncertain. Achieving quantum advantage with VQE necessitates demonstrating higher accuracy and faster computation for specific systems compared to conventional methods, while acknowledging limitations in exactness due to parameterization [TCC+22].

Quantum Approximate Optimization Algorithm Quantum Approximate Optimization Algorithm (QAOA) is an algorithm especially designed to find approximate solutions for combinatorial optimization problems [FGG14]. The algorithm's performance is contingent upon an integer value $p \geq 1$, with improved approximation quality corresponding to higher p values. The quantum circuit responsible for executing the algorithm is composed of unitary gates, each with a locality no greater than that of the objective function whose optimum is sought.

The unitary operation $U(\beta, \gamma)$ possesses a distinct structure, comprising a combination of two distinct unitaries denoted as $U(\beta) = e^{-i\beta H_B}$ and $U(\gamma) = e^{-i\gamma H_P}$. Here, H_B represents the mixing Hamiltonian, while H_P corresponds to the problem Hamiltonian. The parameters β and γ determine the amplitude of evolution under each operator, forming a key aspect of the algorithm's optimization process. The algorithm aims to return optimal parameters such that the quantum state reveal the solution to the problem.

The quantum state is generated by employing these unitaries in an alternating sequence, with each block consisting of the two unitaries applied p times, where $|\psi_0\rangle$ indicates the

inital state:

$$|\psi(\beta,\gamma)\rangle = \underbrace{U(\beta)U(\gamma)...U(\beta)U(\gamma)}_{p \text{ times}} |\psi_0\rangle$$

QAOA has found applications across a spectrum of fields, including graph partitioning, Max-Cut problems, and portfolio optimization [KW22, CWY+23].

Harrow-Hassidim-Lloyd Algorithm

Linear equations are a mathematical problem which occur in practically all areas of science and engineering and therefore a lot of real-life applications are build upon them. The Harrow-Hassidim-Lloyd algorithm [HHL09a] has a improved complexity of logarithmically in N, whereas the classical algorithms for such problems can only be solved in polynomial time [CAB+21].
Consider a linear equation in the form of:

$$A|x\rangle = |b\rangle$$

A is a Hermitian matrix of size $N \times N$ and $|x\rangle$ represents the solution vector of the corresponding length N. By mapping the two normalized vectors $\vec{b}$ and $\vec{x}$ the quantum states $|b\rangle$ and $|x\rangle$ are build.
The HHL algorithm can be divided in four main steps or in other word also called subroutines [HHL09b, aZW23]:

1. Prepare the initial quantum state $|\psi_0\rangle$
2. Run Quantum Phase Estimation (QPE) estimate the eigenvalues of the matrix A [Kit95]
3. Rotate the ancilla bit for encoding relevant information in the amplitude
4. Run Inverse Quantum Fourier Transform (QFT) transfering the quantum state in measurement probabilities [aZW23]
5. Measurement the quantum state

2.1.5 Noisy Intermediate-Scale Quantum Devices

In the near future, quantum hardware will be based on the Noisy Intermediate-Scale Quantum (NISQ) technology which is characterized by a qubit count up to a few hundred qubits. NISQ devices operate in the presence of noise, resulting in errors during quantum operations. The noise can be caused by interaction with the environment or other qubits [Pre18]. These constraints encompass a restricted qubit count and the noise that limit circuit depth. As a result, the practical deployment of fault-tolerant quantum computers is likely to remain an aspiration in the near future [LLSK22].

2.1.6 Quantum Error Correction

Quantum Error Correction (QEC) is an important topic in the field of NISQ devices, addressing a critical challenge that arises from the inherent susceptibility of qubits to noise and decoherence [Kas21b]. As NISQ systems become more complex and computations involve larger numbers of qubits, the sensitive nature of quantum states makes them prone to disturbances from their environment. This susceptibility can lead to errors in quantum computations, impacting the reliability and accuracy of quantum algorithms.

In classical computing, error correction techniques have been developed and refined over decades, allowing for robust data manipulation and transmission. Using classical computing solutions are not suitable in the quantum world, since the no-cloning theorem prevents us, form taking this approach. In the quantum field errors raise a unique and complex challenge. Due to the principles of superposition and entanglement, errors can propagate and spread through quantum systems in ways that are fundamentally different from classical systems. This phenomenon, known as quantum error propagation, highlights the need for specialized error correction strategies that are tailored to the principles of quantum mechanics.

One of the primary causes of errors in quantum computing is decoherence, which arises from interactions between qubits and their surrounding environment. These interactions lead to loss of coherence and entanglement, resulting in the degradation of quantum states [ZuRJ+22]. Additionally, imperfections in hardware components, such as gates and measurements, can introduce errors into quantum operations. These errors accumulate as computations progress and potentially affect the final output to be meaningless [Hom22a].

Quantum Error Correction has emerged as a powerful tool to combat these challenges and ensure the reliability of quantum computations. The main principle underlying QEC is the encoding of quantum information redundantly across multiple qubits [Bac13]. By distributing the information in an error-resistant manner, errors can be detected and corrected, preserving the integrity of quantum states throughout the computation process. Quantum codes, analogous to classical error-correcting codes, enable the identification and correction of errors without the need for direct measurement of the quantum states, which would otherwise risk collapsing their delicate superposition [Kas21b].

Incorporating Quantum Error Correction into quantum algorithms and systems is essential for practical and scalable quantum computing. The field of QEC continues to evolve, with researchers developing new codes, techniques, and error-detection mechanisms that contribute to the stability and robustness of quantum computations. As quantum computing technologies advance, QEC remains an important component as it enables us to unlock the full potential of quantum computation while mitigating the impact of noise and errors [MH21].

CHAPTER 3

Related Work

The following chapter deals with similar topics that are relevant for this master thesis and serve as a basis for it. Approaches and results from various other literature sources are described and considered in more detail below. The section starts by discussing Hybrid Classical-Quantum Systems and the Computing Continuum, continuous with an in-depth exploration of Quantum Computing methodologies, and concludes with ongoing researches in Variational Quantum Algorithms.

3.1 Hybrid Classic-Quantum Systems

The utilization of hybrid systems has garnered significant interest in medical fields due to its potential to enhance existing approaches, as described in a recent paper [MBMRMRAE23]. Experts in medical applications seek new alternatives to address challenges beyond the capabilities of current Artificial Intelligence programs for Staging of Invasive Ductal Carcinoma of Breast. The experiment was based on a classical subroutine and a quantum subroutine to determined the current stage of cancer. The authors' findings indicate that quantum computing does not replace classical computers but rather extend them, as input and output data must still be processed traditionally. With quantum computing's parallelism, the results are promising, as problems that would take a lifetime to solve on quantum hardware can be addressed in seconds, hours, or days.

Another paper [BWM+16] investigates the application of a hybrid quantum-classical approach for materials simulations, particularly focusing on Density-Functional Theory (DFT), the current method widely used for such simulations. The proposed hybrid approach combines classical and quantum algorithms, integrating DFT with Dynamical Mean-Field Theory (DMFT). DFT, being computationally less expensive, is executed on classical machines, while the more computationally intensive DMFT is performed on quantum hardware. The authors demonstrate that even small quantum computers can be employed alongside their hybrid quantum-classical algorithm to simulate larger

systems, especially strongly correlated crystalline materials or complex molecules. This presents an opportunity for material simulations, which often rely on supercomputing facilities, to benefit significantly from the availability of quantum computers.

Furthermore, the potential benefits of utilizing hybrid classical-quantum systems for image classification, as highlighted in a paper [TSR+21]. The authors introduced a Hybrid Neural Network (HNN) that combines quantum computing and classical computing to perform image classification tasks. The proposed HNN is built on *Ry* quantum circuits with 2-4 qubits, each having one trainable parameter. Their findings demonstrate that the HNN can effectively handle both complex datasets (e.g., color images) and simpler datasets (e.g., greyscale images with basic content like digits or simplified clothing icons). However, it is important to note that at its current stage the HNN exhibits significantly lower accuracy compared to its classical counterparts. The authors emphasize that a direct comparison of metrics between hybrid classical quantum models and classical models might be premature, considering the early stages of quantum computing development.

An interesting study [VROVR+23] focus on the bin packing problem in various dimensions, with particular attention to the three-dimensional bin packing problem (3 dBPP) common in industrial settings. The authors present a hybrid classical computing framework for the 3 dBPP, leading to three main conclusions. Firstly, longer time limits lead to lower energy and improved solution quality. Secondly, the deviation around mean values remains stable across different time limits and problem instances. Lastly, despite varying time limits, the optimization process predominantly relies on the solver's heuristic module, evident from consistent Quantum Processing Unit (QPU) access times. The results demonstrate the effectiveness of the hybrid approach, offering feasible solutions for the studied instances.

Quantum Software Engineering (QSE) has emerged as a novel research area, as described in an article by Ricardo Pérez-Castillo and Mario Piattini [PCP22]. Quantum software programming techniques have been experimentally proposed in an ad-hoc manner as the authors stated in their article. Their findings highlight the need for new techniques that encompass both classical computers and their software, along with quantum software. Merely focusing on quantum software development will not suffice, as some simple problems can be efficiently computed using classical software at a lower cost. Ignoring this capability would lead to inefficiency in utilizing available computing resources. They propose a quantum Unified Modeling Language (UML) profile for the analysis and design of hybrid classical/quantum systems. The applicability of the profile is demonstrated through various structural and behavioral diagrams, including use case, class, sequence, activity, and deployment diagrams. The authors' work provides valuable insights into managing relationships between classical and quantum software and how to represent these relationships in abstract designs.

The Satellite Mission Planning Problem (SMPP) aims to efficiently schedule satellite resources for earth surface imaging as explained in a paper [QKBW23]. Variational

quantum algorithms like VQE and QAOA show potential for solving this combinatorial optimization problem. The experiments compared performance on noise-free and noise-aware simulations. In noise-free simulations, VQE was notably faster for smaller instances, but as the number of locations increased, result quality degraded for all algorithms, with VQE showing the most significant decrease. On the noise-aware simulator, computation times increased exponentially with more locations, and W-QAOA (warm-start variant - parameters pre-determined instead of randomly initialized) was the fastest for smaller instances, while VQE was faster for larger ones. Result quality decreased with more locations, and the drop was more significant in noise-aware simulations. The SPSA optimizer performed poorly, indicating that the Cobyla optimizer may be more suitable for SMPPs.

3.1.1 Quantum Computing

Quantum Computing is an emerging paradigm with the potential to provide substantial computational advantages for various problems. Many challenging tasks, currently requiring extensive resources and lengthy execution times, can be significantly accelerated on quantum hardware. Ongoing research in different fields has shown promising improvements, pointing to the transformative impact of quantum computing [GKS+20].

A paper [dW17] published in 2017 explores the potential societal impact of quantum computing. The authors highlight a significant threat to cryptography. This field, crucial for our online communication and e-commerce, faces potential vulnerabilities due to Quantum Computers. These advanced computers can efficiently break encryption algorithms such as RSA, utilizing Shor's algorithm to find prime factors of large numbers. Additionally, quantum computing can revolutionize optimization problems and large search challenges, enabling efficient resource allocation for governments, companies, and organizations in their daily operations. Quantum computers offer accelerated simulations of quantum systems, facilitating tasks like drug design through parallel exploration of vast lists of possible molecules to identify suitable properties. However, the paper also briefly touches upon the ethical implications of quantum computing, such as its potential impact on online privacy and the potential for power imbalances between countries or companies. These aspects highlight the transformative potential of quantum computing in various spheres and the need for careful consideration of its implications.

In the work of Upama et al. [UFN+22], an examination of various quantum computer tools and software currently available is carried out. These tools serve different use cases and offer a wide range of functionalities, accessible even from personal laptops for experimentation with quantum machines. The author listed several tools and software, including Cirq[1], TensorFlow Quantum[2], ProjectQ[3], CirqProjectQ[4], Microsoft Quantum

[1] https://quantumai.google/cirq
[2] https://www.tensorflow.org/quantum
[3] https://github.com/ProjectQ-Framework/ProjectQ
[4] https://pypi.org/project/CirqProjectQ/

Development Kit[5], IBM Quantum Experience[6], Rigetti Forest and Cloud Computing Services (OCS)[7], Quantum Computing Playground[8], Strawberry Fields[9], and Wolfram Quantum Framework[10]. In addition, the work [UFN+22] highlights some challenges related to the physical space required for quantum computers. Currently, quantum computers occupy an entire room, yet their qubit configurations are not sufficient to promise significant speedup for economically viable quantum computing. Therefore increasing the number of qubits presents a challenge due to the consequent growth in the size of quantum computers. Furthermore, the authors identified a lack of collaboration and exchange between industry and academia as a significant obstacle in the advancement of Quantum Computing. Addressing these challenges will be important for the future progress of quantum computing technologies.

In their research [GS21], the authors investigated Quantum Computing's impact on Supply Chain Finance using IBM Qiskit and three algorithms: Minimum Eigenvalue, VQE, and QAOA. They compared algorithmic results with manual selection, finding no significant difference. However, the authors stated as quantum hardware advances, the finance industry could benefit in the future.

According to Crispin H. V. Cooper's paper [Coo22], transportation simulation and planning benefits significantly from the application of quantum hardware. The author explores various existing quantum computing research related to transportation, particularly focusing on developments in network analysis, shortest path computation, multi-objective routing, optimization, and calibration. Among these areas, the latter three show particular promise for future research. Given that these computing problems often involve extensive input data, quantum hardware has the potential to outperform classical computers in handling such complexity.

In a comprehensive survey paper [YZJ23], the authors identified several prevailing challenges and limitations in both universal and annealer quantum computing. These challenges are categorized into five groups:

- Hardware challenges, encompassing issues like unreliability due to decoherence and noise, large hardware size, high design complexity, and incomplete theory.
- Connectivity challenges, including short distances between qubits, imperfect teleportation, lossy links, limited topology, and the reliance on trusted nodes.
- Security challenges, involving low key rates, vulnerability to Denial of Service attacks, limited key efficiency, and classical alternatives.

[5] https://learn.microsoft.com/en-us/azure/quantum/overview-what-is-qsharp-and-qdk
[6] https://quantum-computing.ibm.com/
[7] https://pyquil-docs.rigetti.com/en/v2.7.2/
[8] http://www.quantumplayground.net/#/home
[9] https://strawberryfields.readthedocs.io/en/stable/
[10] https://resources.wolframcloud.com/PacletRepository/resources/Wolfram/QuantumFramework/

- Data analysis challenges, such as limited data types, low compatibility with classical approaches, and the lack of collaboration strategies.
- Pragmatism challenges, including high cost, large size, diverse programming styles, and limited resources.

Addressing these challenges, the authors proposed four research groups: 1) Quantum computers, focusing on hardware improvements; 2) Quantum networks, aimed at enhancing connectivity; 3) Quantum cryptography, addressing security issues; and 4) Quantum machine learning, targeting advancements in data analysis. These research groups serve to tackle the limitations and pave the way for advancements in the field of quantum computing.

Quantum benchmarking is a critical aspect of quantifying the capabilities of a given quantum computer, as it provides a reliable measure of its performance and potential. In the paper [WGS22], the authors offer valuable guidelines on how to establish benchmarks specifically tailored for quantum computers. The benchmarks are thoughtfully categorized into three groups: physical benchmarks, aggregated benchmarks, and application-based benchmarks. The physical benchmarks focus on essential factors such as the number of qubits, energy relaxation time $T1$, decoherence time $T2$, qubits' connectivity, and gate fidelity. Meanwhile, the aggregated benchmarks encompass metrics like quantum volume, algorithmic qubits, and the circuit layer operations per second (CLOPS). Finally, the application-based benchmarks are designed to assess the performance of a quantum computer when tackling real-world quantum applications. The paper underscores the significance of quantum benchmarks, given their widespread use in high-performance computing.

3.1.2 Variational Quantum Algorithms

Variational Quantum Algorithms (VQAs) are a class of quantum algorithms that leverage quantum computers to solve optimization problems [AMR+22]. Unlike some other quantum algorithms that offer exponential speedup for specific tasks, VQAs are designed to be implemented on near-term, noisy quantum devices [LWW+21]. They combine quantum and classical computation, making them well-suited for current quantum hardware capabilities.

A paper [CAB+21] provides an in-depth exploration of Variational Quantum Algorithms (VQAs) and highlights their promising potential for various future applications. VQAs take advantage of classical optimizers to train parameters, allowing the utilization of relatively small quantum hardware. This approach is particularly beneficial in dealing with the current limitations of available quantum resources. The authors emphasize that VQAs are considered a leading candidate for achieving quantum advantage in near-term quantum computing. Their applications encompass tasks like finding ground states of molecules, simulating quantum system dynamics, and solving linear systems of equations. However, addressing challenges related to trainability, accuracy, and efficiency when

applying VQAs to large-scale problems is crucial for future advancements. As VQAs transition from the proposal and development phase to implementation, it is anticipated that researchers will increasingly utilize them for real-world problems rather than limited toy problems.

In a recent study [AF23], the authors explored the application of Variational Quantum Algorithms (VQAs), including Variational Quantum Eigensolver (VQE) and Quantum Approximate Optimization Algorithm (QAOA), to solve the Vehicle Routing Problem (VRP). The VRP is a combinatorial optimization problem that aims to find the optimal routes for multiple vehicles to travel between various destinations and return to the starting location. The experiment involved executing the algorithms on different datasets and comparing their results with those obtained using classical approaches. The findings indicated that currently, VQAs can handle only small instances of VRP. Furthermore, QAOA demonstrated higher accuracy in delivering solutions compared to VQE.

The paper [SMM23] titled "Benchmarking of Different Optimizers in Variational Quantum Algorithms for Quantum Chemistry Applications" conducted various experiments on different optimizers to assess their accuracy in optimizing performance. The optimizers were categorized into gradient-based methods, gradient-free methods, and quantum-hardware-specific methods. Quantum simulations were performed on simple molecules, including hydrogen, lithium hydride, beryllium hydride, water, and hydrogen fluoride, both with noise-free and noisy quantum circuits. Under ideal conditions, L_BFGS_B, CG, and SLSQP were identified as the best gradient-based optimizers for ground state energy evaluation accuracy, while COBYLA and POWELL excelled among gradient-free optimizers. Noise in quantum circuits impacted the performance of many optimizers, but their ground state energy error remained similar. For dissociation energy, CG and SLSQP performed well under ideal conditions, while SPSA, GD, POWELL, and COBYLA outperformed others in the presence of noise. Regarding dipole moment, L_BFGS_B, CG, and SLSQP remained the best gradient-based optimizers, while SPSA, POWELL, CG, and COBYLA showed better performance under noise.

Another study [SWA21] investigates variational quantum algorithms, particularly the variational quantum eigensolver (VQE) method, to estimate the ground state energy of small molecules on noisy quantum devices. The study explores hardware-efficient ansatz families to reduce circuit depth, mitigate noise-induced barren plateaus, and enhance performance on quantum chemistry problems using noisy quantum hardware. The results indicate that the optimal ansatz family choice depends on the noise level and hardware type, highlighting the importance of evaluating circuit families on noisy quantum simulators or real quantum devices for accurate decisions. Furthermore, the study examines the expressibility measure for characterizing ansatz families and finds that it does not correlate with the circuit's performance in finding the ground state using VQE. Therefore it may not be the best criterion for selecting suitable ansatz families for chemistry applications. The paper suggests further investigations, including extensive analyses involving various ansatze and noise models, as well as exploring machine learning

techniques to predict appropriate ansatz families based on problem characteristics and hardware noise.

A paper [LCS+23] investigates the time-scaling performance of Variational Quantum Algorithms (VQAs) and their potential for achieving quantum advantages. The lack of inter-layer quantum state recording in Quantum Neural Networks (QNNs) was identified as a limitation for scalability. The estimated running time of VQAs grows polynomially. However, achieving quantum advantages at a regular time scale becomes challenging. In ideal cases, the running time may even reach the 1-year barrier. Nonetheless, the authors recognize the potential of VQAs and Noisy Intermediate-Scale Quantum (NISQ) algorithms and propose optimizations such as more efficient sampling strategies and parameter-saving ansatzes. They also emphasize the importance of exploring a more natural approach to Quantum Machine Learning tasks, moving away from a direct replacement of classical Neural Network models with QNNs.

3.1.3 Quantum Algorithm Benchmarking

Quantum algorithm benchmarking is a crucial area of research aimed at evaluating and quantifying the performance of quantum algorithms on different hardware platforms. This section provides an overview of the state of the art in quantum algorithm benchmarking, highlighting key research studies and advancements in the field.

Researchers have proposed various benchmarking methodologies to assess the performance of quantum algorithms. These include approaches such as randomized benchmarking [MGJ+12], cross-entropy benchmarking [BIS+18], and gate set tomography [BKGN+13]. These techniques provide quantitative measures of the accuracy, fidelity, and efficiency of quantum algorithms, enabling comparative evaluations across different quantum hardware platforms. Quantum algorithm benchmarking aims to evaluate the performance of specific quantum algorithms in solving well-defined problems. For example, several studies have focused on the algorithms performance such as the Quantum Approximate Optimization Algorithm (QAOA) for combinatorial optimization problems [WWJ+20] and the Variational Quantum Eigensolver (VQE) for molecular simulations [HLL+22]. Quantum algorithm benchmarking also includes assessing the impact of errors and noise on the performance of quantum algorithms. Researchers have explored various techniques for error characterization and mitigation, such as quantum error correction [Got97] and error mitigation using classical post-processing techniques [TBG17]. These studies investigate the effectiveness of error mitigation strategies in improving the reliability and accuracy of quantum algorithm results. Researchers have also conducted studies comparing the performance of algorithms on different quantum systems, such as superconducting qubits [AASG19], trapped ions [MK13], and topological qubits [AS11]. These comparative benchmarking studies provide insights into the strengths and limitations of different hardware platforms, aiding the development and optimization of quantum algorithms for specific systems.

CHAPTER 4

Methodology

Given the potential for future speedup in scientific computing through the utilization of quantum devices, our objective is to benchmark quantum simulators deployed at the network edge. Quantum algorithms are influenced by various parameters that can impact result accuracy. Therefore, it is crucial to run all algorithms employed in this thesis under various hyperparameter configurations to identify the set that produces the optimal results.

4.1 Benchmarks

The thesis focuses on Variational Quantum Algorithms (VQAs), specifically the Variational Quantum Eigensolver (VQE) and the Quantum Approximate Optimization Algorithm (QAOA). Additionally, it includes benchmarking of the Harrow-Hassidim-Lloyd (HHL) quantum algorithm for solving linear equations. These algorithms are directly related to scientific computing problems and have potential applications in various scientific fields, such as quantum chemistry.

Since the field of scientific research is rapidly evolving, the software development method rapid prototyping is a good option. Firstly, it allows for the efficient development of benchmarking methodologies, enabling quick experimentation and comparison of quantum systems with local machines and edge devices in different scenarios. On the other hand, it allows for flexibility in adapting methodologies based on new insights, continuous advancements in hardware and software or feedback which is crucial in such a new scientific field. Finally, rapid prototyping can save time and resources by identifying potential issues or limitations early on, allowing for timely adjustments and improvements. Overall, rapid prototyping yields a method which improves a fast and adaptive development process, resulting in early valuable findings and results.

4.2 Benchmarking Methodology

In the context of benchmarking, careful consideration is given to the selection and tuning of hyperparameters, crucial elements that significantly influence alogrithms performance. For VQE the hyperparameters cover a range of choices, including the configuration of the ansatz, choice of optimizers, and selection of backends. Optimizers are employed to iteratively adjust the ansatz parameters, converging towards an optimal solution. Meanwhile, the selection of different backends, representing the underlying quantum hardware or simulators, introduces variability into the experiments. Beside VQE, QAOA is affected by hyperparameters such as the optimizer and the the chosen backend. HHL underlying functionality is different to VQE and QAOA and therefore HHL is tested with a variety of different backends. Moreover, for HHL it is also tested if transpiling the circuit affects the final results in matter of computation time or accuracy.

All benchmark tests are conducted using publicly accessible IBM quantum computers[1]. Executed quantum computing operations are performed through the open-source quantum computing framework called Qiskit [qisa], developed by IBM. Since direct executions on the actual machines are still associated with long waiting times, proper simulators are used as alternative for performing appropriate tests. Later during the thesis and in the final stage of performing suitable benchmark tests the execution run on local simulators and a rasqberry simulator. The two end devices are a classic computer and a Raspberry Pi (Raspberry Pi 4 Model B, 8GB RAM), which acts as an edge device. The Raspberry Pi, is also called `RasQberry` [rasb] because it integrates Qiskit. For this purpose, the corresponding computational tasks are offloaded to the remote device or executed locally.

In the beginning, the primary focus was to conduct all benchmark tests for the algorithms on local simulators. Local simulators were chosen for their ease of accessibility, and the direct output and debugging processes proved to be more efficient and quicker. Qiskit was employed to load the relevant backend providers for the simulators used for benchmarking and for programming the corresponding algorithms, along with their hyperparameters. Each individual benchmark was initiated in isolation, ensuring that each experiment represented a distinct standalone instance on the local simulator. This approach guaranteed that the various benchmark tests remained independent and could not interfere with or influence one another.

A similar principle was adopted to execute identical benchmark tests on remote simulators. The key distinction lies in the fact that individual experiments were transmitted to the remote simulator via Transmission Control Protocol (TCP) [CK74]. Each instance was processed there, and the final results were returned through the same socket connection. Once again, each test was transmitted individually to obtain interference-free results.

The open-source quantum computing framework Qiskit is a quantum computing library written in Python [pyt]. Therefore, all additional software engineering tasks are performed using Python 3.7 and later. The Python library `MatplotLib` [mat] is used for

[1] `https://quantum-computing.ibm.com/`

visualization tasks due to its extensive capabilities for creating high-quality plots, charts, and visualizations. Experiment data is collected and processed using Python libraries for numerical computing and data manipulation, such as `NumPy` [Num] and `Pandas` [pan].

Qiskit provides a comprehensive set of tools, libraries, and functions for designing, simulating, and executing quantum circuits on real quantum hardware or simulators. Mainly, simulators are used for execution due to the long waiting queues and limited resources of currently accessible IBM quantum computers. Otherwise it may result in delays and unreliable performance, potentially leading to inaccurate or unrepresentative benchmark results. Qiskit provides several simulators for quantum circuit simulations, which offer a range of options for simulating quantum circuits with different levels of accuracy and noise models, allowing for a thorough investigation of quantum algorithms and their performance. All experiments utilize the Aer provider backend [Sim] from Qiskit to execute the tasks.

4.2.1 Variational Quantum Eigensolver

VQE [PMS+14] operates as a quantum algorithm designed for computing minimal eigenvalues of hermitian matrices in various dimensions. This method uses the principles of variational optimization in quantum mechanics, employing a parameterized quantum circuit known as the ansatz. Careful selection of the ansatz is essential to obtaining a final solution that is close to the true state of interest [TCC+22].

4.2.2 Quantum Approximate Optimization Algorithm

QAOA [FGG14] instead, is a algorithm for combinatorial problems. It employs a parameterized quantum circuit, known as the initial state, to encode the problem's objective function. In our case, the initial state is chosen randomly. QAOA iteratively adjusts these parameters, aiming to find the optimal configuration that minimizes the objective function. By leveraging quantum superposition and entanglement, QAOA explores multiple potential solutions simultaneously. The algorithm's performance is influenced by the used optimizer and backend which are hyperparameters. For this reason QAOA is executed on randomly generated connected graphs, with a specified number of nodes, calculating the Max-Cut problem by using different optimizers and backends. Moreover, a brute-force approach is employed to compare the QAOA solution to get insights in accuracy and computational time. The Max-Cut problem [Com09] is a classical combinatorial optimization problem. In this problem, the objective is to partition a given undirected edge-weighted graph structure into two disjoint sets of nodes to maximize the sum of edge weights between the two sets. Similar to VQE, QAOA instance are executed with varying graph sizes and on different backends using various optimizers as input parameter. Due to the significant size of QAOA circuits, additional experiments are conducted to reduce the circuit depth.

Here's a brief explanation of the problem:

1. Input Graph: We have an undirected graph where the nodes represent specific elements or objects, and the edges between nodes are weighted to represent the relationships between the elements.

2. Subset Formation: The goal is to partition the nodes of the graph into two disjoint subsets. The idea is that the elements in the two subsets are chosen in a way that maximizes the sum of the weights of the edges connecting the two sets (referred to as cut edges).

3. Maximum Weight Sum: The problem is to find the node partition that results in the maximum possible sum of weights of the cut edges. This separation aims to create the most pronounced division between the subsets by cutting the most heavily weighted connections between them.

4.2.3 Harrow-Hassidim-Lloyd Algorithm

HHL [HHL09a] is an algorithm used for solving linear equations represented in the form of $Ax = b$. During the thesis, different linear equations are generated in a random manner. In the individual experiments, the problem instance's dimensions are varied. Furthermore, the impact of trying to simplify the circuit via transpile-functions is tested.

4.2.4 Data

In this thesis, artificial synthetic data is used for benchmarking purpose to ensure consistency and control over the experimental conditions. Synthetic data allows for precise manipulation of the input data parameters, such as size, complexity, and distribution, to systematically evaluate the performance of each type of computer. In case of VQE random hermitian matrices are generated. Each experiment is executed with a newly created hermitian matrix. The dimensions for the output hermitian matrix can be specified via the corresponding input parameter for the generation function. QAOA needs a input graph to solve the Max-Cut problem. For this reason a random undirected graph with a chosen number of nodes is generated. The probabilities to generate edges between nodes can be specified too. As a result a undirected graph with n nodes and m edges is returned. Generating linear equations for HHL like $Ax = b$ can be achieved via the same approach as for VQE. The hermitian matrix A is generated in a random manner based on the given size for the dimensions. The same procedure is used for the vector b. Vector b is randomly generated based on the dimension. Since the linear equation system must have same matrix and vector dimension, the corresponding value is used for both generation steps.

The main data sampling approach employed in this thesis is random sampling, which involves randomly selecting data points from the available data set. This approach helps to ensure that the benchmarking experiments are conducted on a diverse and

representative sample of data. Additionally, systematic sampling, which involves selecting data points at regular intervals from the data set, is utilized as an additional method to complement the random sampling approach. These data sampling methods are used to generate benchmarking data that accurately reflects real-world scenarios and provides meaningful insights into the performance of local machines, edge devices, and quantum simulators in solving various computational problems.

4.2.5 Evaluation

The execution time for benchmarking the local machine and edge device are measured using the `time()` module of Python. This module provides a simple and accurate way to measure the time taken for a specific code block to run. The corresponding benchmarks for a wide variety of tasks are analyzed. At the end, the benchmark tests are used to try to identify which tasks can be solved faster given quantum hardware. We measure the accuracy, speed and scalability of each algorithm and analyze the results using methods such as measuring computation speed and compare exact values from classical approaches with the received quantum result. Moreover, computational time and deviation between exact and quantum result are used to analyse the quantum algorithm under different instances varying in the number of dimensions or size. Furthermore, the evaluation involves comparing the accuracy and execution time for different input sizes on both quantum hardware and classical architecture. An overview of the different main metrics is listed below:

- Execution Time Measurement
  - *Metric:* Time used on the local simulator and quantum edge simulator.
  - *Method:* Uses the `time()` module in Python to measure the execution time for solving the input problem.
- Scalability Vs. Time
  - *Metric:* Time usage compared to problem dimensions is analyzed.
  - *Method:* Uses the `time()` module in Python to measure the execution time for problems with increasing complexity.
- Input Size Vs. Qubits
  - *Metric:* How does the input size affect the qubit usage by different algorithms.
  - *Method:* Different input sizes are randomly generated, with the problem dimensions increasing each time. These instances are executed on the quantum simulators. The behavior of qubit usage is monitored in comparison to the input size. Used qubits can be retrieved via: *quantumCircuit.num_qubits* [2]

[2] `https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.QuantumCircuit#num_qubits`

- Absolute Error
  - *Metric:* The deviation between the exact value and the quantum result are calculated. Moreover the deviation from the exact value is calculated for several repetitions of the experiment.
  - *Method:* Absolute Error $= |x_{\text{estimated}} - x_{\text{exact}}|$
    Mean Absolute Error $= \frac{1}{n}\sum_{i=1}^{n} |\text{Exact Value}_i - \text{Estimated Value}_i|$
- Repetition Accuracy
  - *Metric:* The algorithm is repeated several times with the same input instance.
  - *Method:* A initial input instance is generated randomly and the algorithm is executed. The result is noted and the algorithm is executed again with the initial input. This behaviour is repeated several times, each single round the result is monitored. In the end the mean absolute percentage error is calculated. Mean Absolute Percentage Error $= \frac{1}{n}\sum_{i=1}^{n} |\frac{\text{Exact Value}_i - \text{Estimated Value}_i}{\text{Exact Value}_i}| \times 100\%$

To provide decision-makers with a realistic understanding of the performance of both types of computers, the benchmark tests aim to simulate real-world scenarios. The analysis intends to identify the strengths and weaknesses of each type of computer and offer decision-makers a set of guidelines for selecting the appropriate type of computer for a given task.

CHAPTER 5

Implementation

The following chapter outlines pivotal implementation procedures necessary for conducting benchmark tests on the respective devices. In this context, we take a closer look at the utilized architecture of the experiment and subsequently address crucial details of the algorithms used for this study. Apart from the explained implementation procedures, it is noteworthy to acknowledge that a substantial portion of the thesis is dedicated to establishing a foundational understanding of quantum computers. A considerable amount of new knowledge and insights into quantum mechanics are imperative to comprehend the utilization of each algorithm and to understand the impact of diverse configurations on the ultimate outcomes.

5.1 Architecture

In order to conduct experiments in a correspondingly realistic environment, a suitable setup must be established that allows interacting with edge devices and running some experiments on local devices. For this reason, a laboratory setup was created at Vienna University of Technology, representing the edge device. This edge device is a Raspberry Pi 4 Model B [Rasa], which can be accessed from the outside via a VPN. Facilitating this objective involved configuring a static IP address for a VPN connection to the internal network, followed by implementing suitable port forwarding. As a result, it becomes possible to establish a TCP connection with the Raspberry Pi, allowing the transmission of data packets from external origins. For the local device, a conventional laptop (Dell Precision 7560 mobile workstation), no further configurations are necessary as the relevant experiments can be directly conducted on the device. A schematic representation of the experimental setup can be seen in Figure 5.1. Moreover, the specific device resources are shown in Table 5.1.

Both of the devices are capable of executing IBM Qiskit [Qisb] code to function as quantum simulators. Installing Qiskit on the local device is straightforward using *pip*

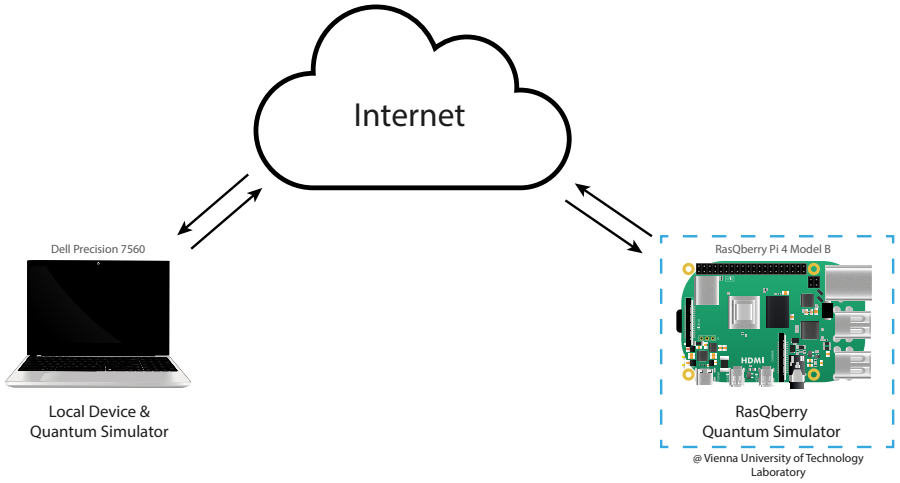


Figure 5.1: Topology used for this thesis

Device Name	CPU	RAM
Raspberry Pi 4 Model B	Broadcom BCM2711 - Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz	8GB LPDDR4-3200 SDRAM
Dell Precision 7560	11th Gen Intel® Core™ i7-11850H @ 2.50GHz 8 Cores	64GB

Table 5.1: Device specifications used for experiments

install as described on the official website [Get]. Since Qiskit is a Python package, a Python version greater than or equal to 3.7 is required.

The installation process on a Raspberry Pi differs somewhat from that on a conventional Windows computer. The Qiskit image cannot be directly installed via *pip install* for this purpose. Fortunately, Jan Lahmann has provided an accessible method for installing Qiskit on an Raspberry Pi. A detailed guide can be found in the corresponding GitHub repository [Git]. Now that the Raspberry Pi with Qiskit installation essentially resembles a quantum computer, it is referred to as `RasQberry` within the community.

An important insight gained during the setup of the test environment is that installing Qiskit on a Raspberry Pi operating as a virtual machine was not feasible for us. In our attempts, the installation process consistently terminated. Despite our efforts to resolve the encountered errors, the installation terminated again with a new error code. It could be possible that the installation routine checks for the use of a virtual machine and terminates the process accordingly.

5.2 Algorithms

To conduct our experiments, we must implement various Variational Quantum Algorithms (VQAs) using Qiskit and subject them to a range of diverse benchmark tests. In the following section, we enumerate the three specific algorithms employed for this thesis, along with their corresponding implementations and parameter configurations. To execute experiments for the algorithms, we require the incorporation of three essential Python packages:

- `numpy` [Num] - This package facilitates streamlined manipulation of vectors and matrices, among other functions.
- `time` - Used for measuring the duration of executions.
- `matplotlib` [mat] - This package offers effective tools for visualizing the benchmark experiments.

5.2.1 Quantum Simulator

In Qiskit, the execution of an algorithm requires the utilization of a backend on which the algorithm is to be executed. This may involve employing a physical quantum computer or, as in the context of this study, using quantum simulators. Through the Aer provider [Sim], Qiskit assembles a variety of high-performance simulators that can be utilized for various simulation methodologies. A listing of the different Aer backends can be generated using the command provided in Listing 5.1. In this work only the Aer-Simulator backend from the Aer provider is used.

```
Aer.backends()
    [AerSimulator('aer_simulator'),
     AerSimulator('aer_simulator_statevector'),
     AerSimulator('aer_simulator_density_matrix'),
     AerSimulator('aer_simulator_stabilizer'),
     AerSimulator('aer_simulator_matrix_product_state'),
     AerSimulator('aer_simulator_extended_stabilizer'),
     AerSimulator('aer_simulator_unitary'),
     AerSimulator('aer_simulator_superop'),
     QasmSimulator('qasm_simulator'),
     StatevectorSimulator('statevector_simulator'),
     UnitarySimulator('unitary_simulator'),
     PulseSimulator('pulse_simulator')]
```

Listing 5.1: Qiskit aer provider backend

Having opted for a backend, the subsequent step follows the creation of a Quantum Instance. This Quantum Instance uses the previously selected backend and further incorporates a *shots* parameter. The *shots* parameter determines how frequently a corresponding quantum circuit is to be reiterated. Given that a quantum computer

operates probabilistically, it is necessary to calibrate these parameters suitably. Through multiple circuit executions, the correct solution can be deduced with an enhanced likelihood. An illustrative example of invoking a quantum circuit is demonstrated in Listing 5.2.

```python
from qiskit_aer.primitives import Sampler as AerSampler
from qiskit import QuantumCircuit

#Create Circuit
circuit = QuantumCircuit(4)
circuit.h(range(2))
circuit.cx(0,1)
circuit.measure_all() # measurement!

sampler = AerSampler(run_options= {"method": "statevector"})

result = sampler.run(circuit, shots=1024).result()
quasi_dists = result.quasi_dists

# Convert the output to bit strings
binary_quasi_dist = quasi_dists[0].binary_probabilities()
print("binary_quasi_dist: ", binary_quasi_dist)
```

Listing 5.2: Qiskit quantum circuit simulation [Cir]

5.2.2 Optimizers

As described in Chapter 2, the construction of Variational Quantum Eigensolver and Quantum Approximate Optimization Algorithm relies on classical local optimizers to iteratively refine the parameters of the quantum circuit. Consequently, for benchmark tests, the utilization of appropriate optimizers is also imperative. The respective selection of the optimizer significantly influences the efficiency and performance of algorithms. Qiskit, an open-source quantum computing framework developed by IBM, provides two distinct categories of optimizers: local optimizers, which seek an optimal value within the boundaries of the neighboring region of a candidate solution, and global optimizers, which strive to find an optimal value among all feasible solutions. For this study, the following local optimizers provided by Qiskit were employed, since the cover a widely range of different optimization approaches:

- ADAM [KB17, ADA]: It is a gradient-based optimization algorithm. ADAM depends on adaptive estimates of lower-order moments. The algorithm is invariant to diagonal rescaling of the gradients and utilizes little memory.
- SPSA [Spa98, SPS]: It is a gradient-descent method for multivariate optimization problems. The algorithm uses only two measurements of the objective function, regardless the dimensions of the input instance.

- SLSQP [Kra88, SLS]: The algorithm is well suited for mathematical problems where the objective function and the constraints are twice continuously differentiable. SLSQP minimizes a function of one or multiple variables subject to general equality and inequality constraints and any combination of bounds.
- L_BFGS_B [ZBLN97, LBF]: The algorithm aims to minimize a differentiable scalar function f. It is used for solving large nonlinear optimization problems with simple bounds of the variables.
- NELDER_MEAD [NM65, NEL]: It is a algorithm used for multidimensional unconstrained optimization problems. The algorithm is used to find the minimum or maximum of an objective function in a multidimensional space.
- COBYLA [Pow07, COB]: It is a numerical optimization method for constrained problems where the derivative of the objective function is not known. COBYLA uses iterative approximation to convert the original constrained optimization problem into a series of linear programming problems.

5.2.3 Variational Quantum Eigensolver

Variational Quantum Eigensolver (VQE) falls within the category of Variational Quantum Algorithms (VQAs). In this work, VQE was employed to determine and yield the minimum eigenvalue of a matrix. Since in VQE the input matrix or in other words also called Hamiltonian, needs to be transfered to Pauli strings, it is required that the input matrix is of dimension 2^n. Due to the fact that quantum circuits interact with logical gates, this requirement is important. All products of any calculation with Pauli matrices result in dimensions of 2^n. Moreover it is important that any given input must be an hermitian matrix [TCC+22]. The mathematical definition of a hermitian matrix [Str16] is

$$A = A^\dagger$$

Here, $A^\dagger$ represents the conjugate transpose of matrix A.

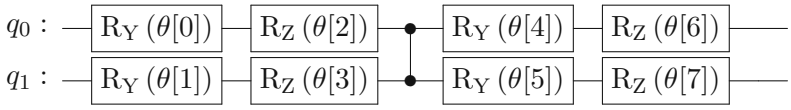
To achieve this goal, a random matrix fulfilling these specific conditions was generated for all tests. For this purpose, a function was used to create and provide a corresponding hermitian matrix of size n. This matrix can then be utilized further and passed to VQE for calculating the minimum eigenvalue. Qiskit offers a function named `random_quadratic_hamiltonian` in the `qiskit_nature_testing.random` package that generates such a matrix. A sample invocation of this method to obtain the actual values of the hermitian matrix is demonstrated as shown in Listing 5.3.

```
1 matrix = random_quadratic_hamiltonian(2).hermitian_part.real
```

Listing 5.3: Create a random hermitian matrix of size n using a function from Qiskit Nature[qisc]

VQE requires an initial point and an ansatz. The initial point refers to the initial set of parameters that define the quantum circuit's structure before optimization begins. This initial set of parameters determines the initial state of the quantum circuit, which is then iteratively adjusted to minimize the expectation value of the problem hamiltonian. For the benchmark tests, a randomized initial point is employed. Within the circuit, the ansatz specifies the manipulation and entanglement of qubits, with its selection depending on the problem's characteristics and the hardware resources available [PMS+14]. The ansatz parameter is chosen based on the test from one of the following ansatz methods:

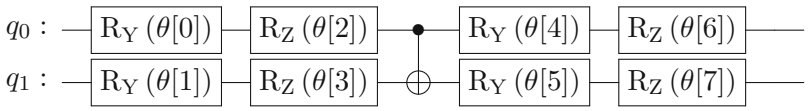
- TwoLocal




```
TwoLocal(num_qubits=2, rotation_blocks=["ry", "rz"],
↪  entanglement_blocks="cz", reps=1).decompose().draw('mpl')
```

Listing 5.4: Qiskit Code for drawing ansatz TwoLocal for 2 qubit circuit

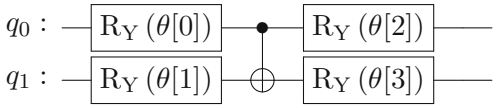
- EfficientSU2




```
EfficientSU2(num_qubits=2, reps=1,
↪  entanglement='linear').decompose().draw('mpl')
```

Listing 5.5: Qiskit Code for drawing ansatz EfficientSU2 for 2 qubit circuit

- RealAmplitudes



```
RealAmplitudes(num_qubits=2, entanglement='linear',
↪  reps=1).decompose().draw('mpl')
```

Listing 5.6: Qiskit Code for drawing ansatz RealAmplitudes for 2 qubit circuit

Noise Model and Error Mitigation

In this thesis, most experiments are conducted on ideal simulators without the use of a noise model. To best emulate environmental noise for real quantum machines, this section explores the addition of a noise model to a simulator and attempts to reduce it through an error mitigation strategy. In our case, we have chosen the `FakeVigo()` fake backend [fak] in Qiskit, which, as part of the fake provider module, replicates IBM Quantum systems using system snapshots containing essential information such as coupling maps, basis gates, and qubit properties. This choice is instrumental for testing noisy circuits and conducting simulations with noise [Noi].

In addition to the corresponding fake backend used for our simulator, we also require error mitigation for our quantum instance to reduce noise in the final results. For the simulator backend, the *resilience level* can be set to perform error mitigation. In our case, we have set it to a value of 1, as it represents the minimum mitigation cost [Con].

Finally, all these configurations were used to execute VQE without a noise model, with a noise model, and with a noise model and error mitigation. Since the implementation only required different parameter settings, this was quickly achieved by adding a fake backend and, consequently, a noise model to the existing solution. Subsequently, the *resilience level* was changed from 0 to 1 to activate error mitigation.

5.2.4 Quantum Approximate Optimization Algorithm

Another algorithm from the group of Variational Quantum Algorithms, Quantum Approximate Optimization Algorithm (QAOA) deals with solving combinatorial problems. In this study, we focus on the Max-Cut Problem, which was discussed in more detail in Chapter 4.

To effectively represent the Max-Cut Problem, a corresponding connected graph with a node set of size n is randomly generated for each benchmark test. Furthermore, it is possible to specify a probability for the creation of an edge between two nodes. In our tests graphs are not weighted. An illustrative example of a connected graph with 6 nodes and an edge creation probability of 40%, generated randomly through the method call for QAOA, is depicted in Figure 5.2. Graphs are generated through the python package `NetworkX` [Net].

To compare the performance of QAOA, a brute-force approach was implemented too. This approach calculates all possible solutions to the Max-Cut Problem and ultimately selects the best one.

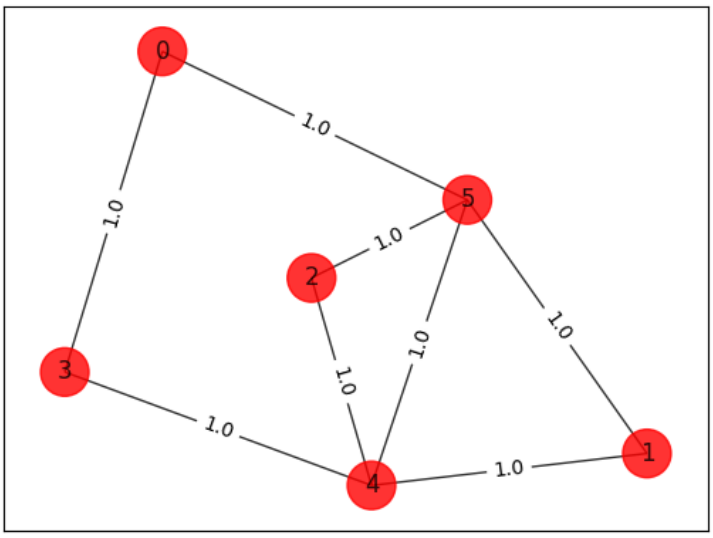


Figure 5.2: A example graph for the max-cut problem with six nodes and edge creation probability of 40%

For the generated graph to be transformed into a quantum instance, a weight matrix must be created from the graph. The resulting matrix can subsequently be converted into a quadratic program. This quadratic program can then be transformed into a Hamiltonian, facilitating the complete computation by a quantum computer. The corresponding code for this process is shown in Listing 5.7.

```
graph = random_graph(6, 0.4)

# weight Matrix
w = weighted_Matrix(graph)

# preapre
max_cut = Maxcut(w)
qp = max_cut.to_quadratic_program()

# quadratic program to ising hamiltonian
qubitOp, offset = qp.to_ising()
```

Listing 5.7: Qiskit code for transforming a generated graph to a hamiltonian

Similar to VQE, QAOA also requires an optimizer and a quantum instance as parameters. For QAOA, the initial point is randomly generated. Moreover, there is an additional

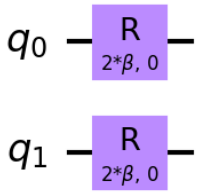


Figure 5.3: A example mixer circuit for a graph with two nodes

mixer parameter [BFL22] which should be set. The mixer parameter is used to address cases where the trial state is an eigenstate of the cost Hamiltonian. The mixer allows transitioning out of this state in order to get not stuck at a local minimum. A sample mixer circuit for a graph with $n = 2$ is shown in Figure 5.3. Furthermore, QAOA requires a *reps* parameter, which correlates with the parameter p specified in Chapter 2. The *reps* parameter controls the number of layers in the quantum circuit and, consequently, the likelihood of achieving a more accurate result. In the benchmark tests conducted in this study, the value of the *reps* parameter (or *reps*) was set to the value of $reps = 2$. The value for $reps = 2$ was chosen because of its better lower bound approximation ratio compared to $reps = 1$ as mentioned by Wurtz et al. [WL21].

5.2.5 Harrow-Hassidim-Lloyd Algorithm

The Harrow-Hassidim-Lloyd (HHL) algorithm is used for the resolution of linear equation systems. Unlike the VQE and QAOA approaches, the HHL algorithm uses only a quantum instance as a parameter input, along with the linear equation formulated in the matrix and vector representation.

To achieve a spectrum of diverse and realistic outcomes, both the vector and matrix components for the HHL algorithm are generated randomly. Similar to previous benchmarks, we construct a hermitian matrix using the functionality provided by Qiskit Nature.

To prove the accuracy and correctness of the acquired outcomes, we used the `NumPyLinearSolver` from Qiskit's linear_solver package. The `NumPyLinearSolver` employs a classical algorithm not using any quantum mechanical principles. This methodology empowers the computation of an exact solution, thereby facilitating a direct comparison with the outcomes received by the HHL algorithm.

As the circuit depth in the HHL algorithm growth rapidly with the size of the inputs, the benchmark tests cover next to some small instance experiments tests of circuit optimization. Specifically, Qiskit offers tools to simplify an input circuit via their transpiler mechanism [Tra]. This procedure tailors the circuit to suit a particular quantum device, thereby optimizing it for execution. Alternatively, as used in this study,

it offers a toolset to optimize a circuit for execution on quantum systems. In the context of the HHL algorithm, we undertake an investigation into the behavior of a transpiled HHL instance and evaluate how the optimization of such instances influences their circuit properties.

5.3 TCP-IP Connection

In order to replicate the principle of offloading to edge quantum devices it is necessary for these devices to be accessible via the internet. To ensure that client requests can be processed effectively by the edge devices, both the client and the edge device need to be properly prepared for this purpose.

We are therefore assuming the following scenario: A local device wants to solve a task based on user input, which can potentially be better solved using principles of quantum mechanics. Let's assume the requirement is to solve a system of linear equations. The local device, also referred to as the client, sends a request to the edge quantum simulator, also known as the server, asking it to solve the provided system of linear equations.

This principle works by establishing a TCP/IP socket connection from the client [Soc] to the server with the request to solve a system of linear equations. It transmits the problem instance to the server. Based on the transmitted byte stream, the server recognizes the type of problem, reads the problem instance, and starts computing the solution. Once the server completes the calculation, it transmits the solution back through the TCP/IP connection. The client can then retrieve the received result and subsequently proceed with the next steps in the program routine. A visual representation of the mentioned scenario is shown in Figure 5.4.

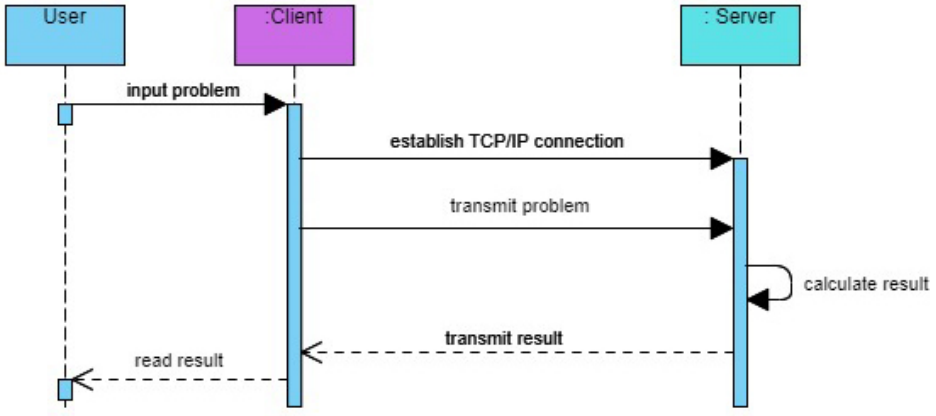


Figure 5.4: A sequence diagram of the offloading between client and server (edge quantum simulator)

CHAPTER 6

Evaluation

The central objective of this research was the thorough evaluation of quantum simulators within the edge computing paradigm. The primary focus was on assessing the viability of offloading computational tasks to quantum simulators at the network edge. To accomplish this, an in-depth analysis was conducted by benchmarking three quantum algorithms: Variational Quantum Eigensolver (VQE), Quantum Approximate Optimization Algorithm (QAOA), and Harrow-Hassidim-Lloyd (HHL) quantum algorithm. These algorithms and their implementation for this thesis were introduced and discussed comprehensively in Chapter 5.

A essential aspect of this evaluation apply to the comparative analysis of executing these algorithms locally and remotely. The fundamental metrics for this analysis are computing latency and accuracy. These metrics play a fundamental role in determining the suitability of offloading specific problem instances to the network edge. This selection is crucial to obtain results within a reasonable timeframe while upholding the desired level of precision. The comprehensive results and insights obtained from the extensive evaluation have been precisely documented in the dedicated Chapter 6. Finding provide a substantial understanding of the feasibility and implications of leveraging quantum algorithms on quantum simulators at the network edge.

6.1 General

6.1.1 Shots Parameter

Testing the influence of the *shots* parameter on the precision of final outcomes is important for quantum computing, particularly within the Qiskit framework. The *shots* parameter signifies the number of times a quantum circuit is executed to accumulate statistics for measurement outcomes. Recognizing and understanding how the parameter affects the accuracy of results is essential to benefit from the full potential of quantum algorithms.

Quantum systems inherently introduce probabilistic aspects due to the probabilistic nature of quantum measurements. As such, assessing the effect of the *shots* parameter on result accuracy is instrumental in understanding the statistical nature of quantum outcomes. By systematically altering the *shots* parameter while maintaining consistent experimental conditions, benchmark testing provides a clear picture of how the distribution of measurement outcomes evolves. This elucidates the convergence of results towards the actual probability distribution and shows potential inaccuracies caused by limited *shots*. Moreover, testing the impact of the *shots* parameter facilitates a efficient allocation of computational resources. A higher number of *shots* generally yields more accurate results by mitigating the effects of statistical fluctuations. However, this enhancement comes at the expense of increased computational time. This phenomenon is illustrated in Figure 6.1 tested on a two qubit circuit with each qubit manipulated by a hadamard gate to place both qubits in a superposition state. When running an instance with a small value provided for the *shot* parameter, the results obtained are not accurate. Increasing the number of *shots* leads to improved performance of the probability functions. Consequently, setting the shot parameter to around 1024 could serve as a suitable initial value. A value of 1024 is also the default setting in Qiskit when no other explicit value is specified. Based on our testing, the trade-off between performance and runtime appears favorable for achieving reasonably accurate results. Opting for higher shot numbers would significantly extend the runtime. Therefore, the specific hardware employed influences the execution times since more powerful hardware resources could finish execution in a shorter period of time. While the edge device indicates notably slower performance for larger shot values, the local machine completes the computations within a considerably shorter time frame.

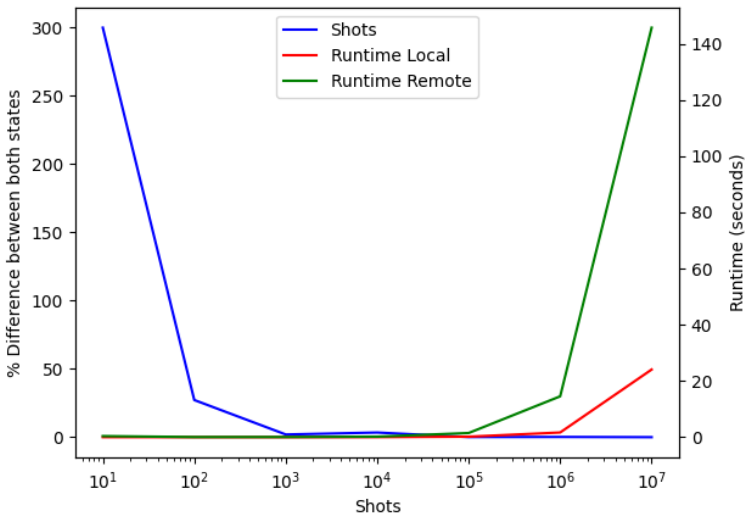


Figure 6.1: Impact of shots parameter on a 2 qubit circuit compared to accuracy and runtime

6.1.2 Network Latency

Network latencies represent a crucial criterion for real-time applications, aiming to return results in a short amount of time. In the context of employing quantum edge simulators, data is transmitted over the network and processed at the network edge. The speed and quality of returning requested results for given problems are affected upon the network's quality. As mentioned in section 5.3, requests such as TCP are transmitted to the edge device. To gain insight into the existing network latencies within our testing environment, we conducted latency measurements using the Python package `tcp-latency` [tcp]. The measurements involved dispatching 500 TCP packets at one-second intervals and measuring the resulting latency, as illustrated in Figure 6.2. Among the 500 transmitted packets, 61 were lost, which in an actual system scenario lead to packet retransmission.

The findings reveal two distinct spikes around packet numbers 209 to 230 and 440 to 448. These correspond to the transfer of a 28MB image and a 9.5MB PDF-file. Smaller spikes are caused from regular browser activity and package updates. Evaluating the mean of all measured values yields a latency of 40.85ms, and considering the median value gives a latency of 31ms. The minimum latency was measured at 15ms, while the maximum latency reached 556ms. These measurements offer insights into the network's dynamic behavior and are important for estimating its suitability for real-time applications, particularly those require rapid data transfer and processing.

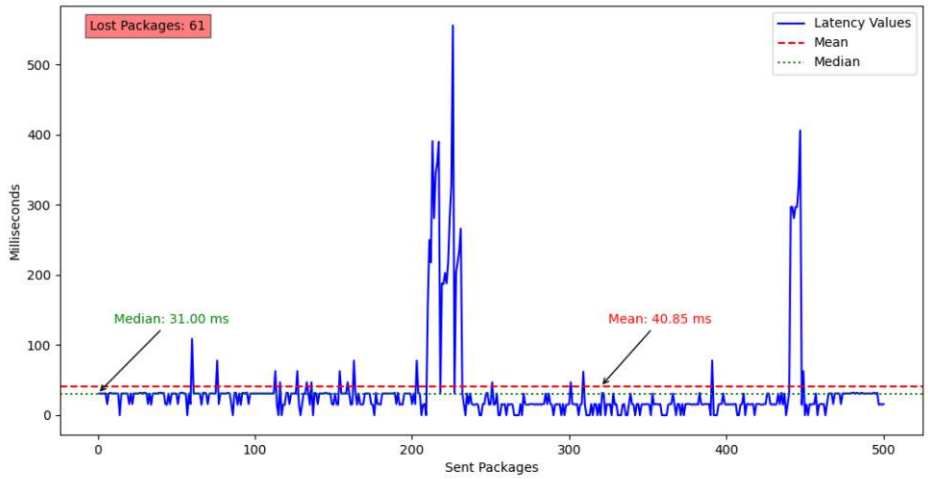


Figure 6.2: Latency measurements between local machine and quantum edge simulator

6.2 Variational Quantum Eigensolver

Variational Quantum Eigensolver (VQE) was employed in this thesis to compute the corresponding minimal eigenvalue for matrices utilized as input to the algorithm. The number of qubits necessary for computing the eigenvalues of matrices follows a direct relationship with the matrix dimensions. Specifically, an input matrix with a dimension of 2×2 requires one qubit, while a matrix with dimensions of 4×4 requires two qubits, and this pattern continues for larger dimensions. This relationship is visually represented in Figure 6.3.

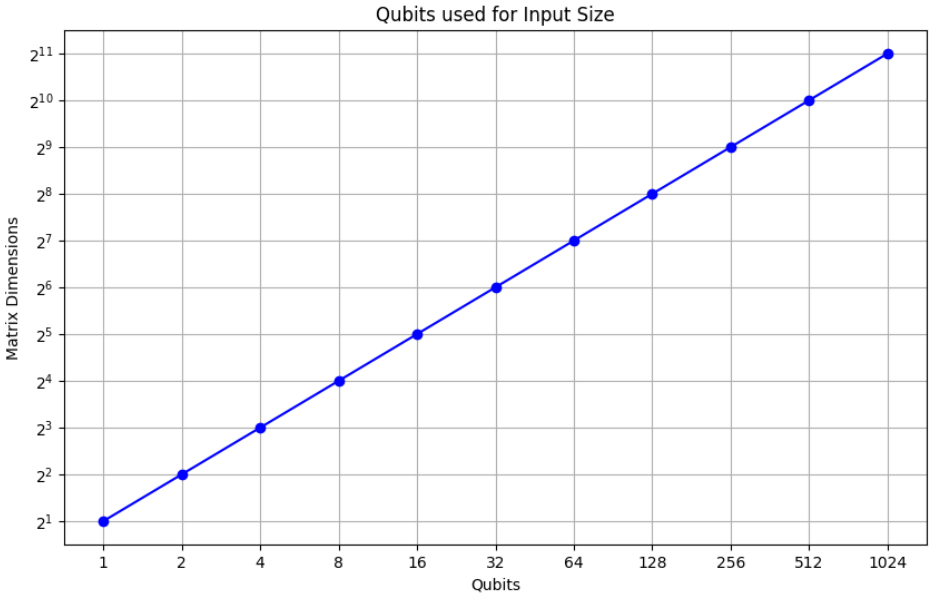
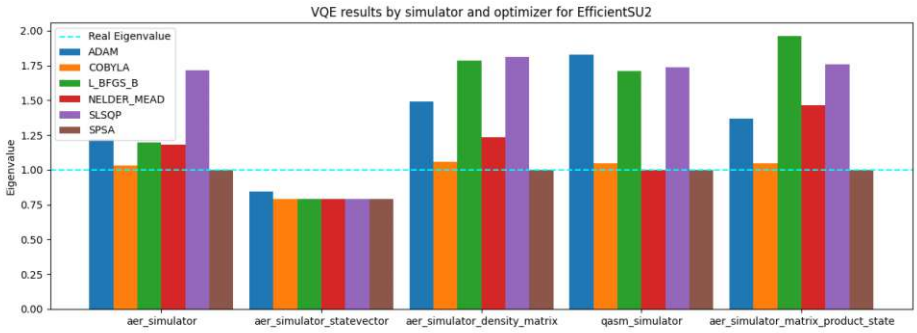
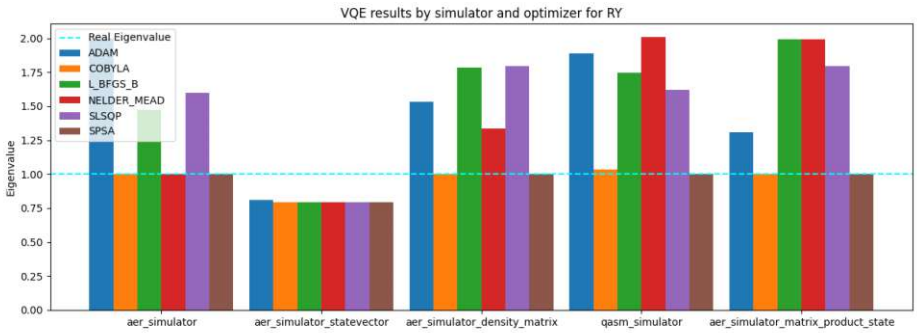


Figure 6.3: Comparison of VQE qubit usage for different matrix dimensions

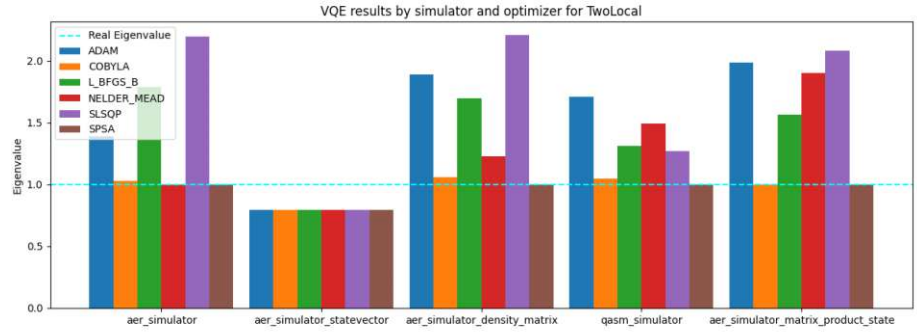
In order to gain an overview of various computational parameters, the simulator backends shown in Figure 6.4 were tested with different optimizers using three distinct ansatz methods. For this particular case, the input matrix for the algorithm had a size of 2×2. From the tests, it is evident which simulator backend yields particularly favorable results and which backend is less suitable for such problem instances. The results clearly indicate that the SPSA optimizer consistently provides the most accurate calculation of the eigenvalue of 1 for the given input size. Furthermore, it is apparent that the *aer_simulator_statevector* backend does not yield the correct results for any ansatz method across various optimizers. In multiple runs, there was no correct match for the *aer_simulator_statevector* backend, making it less suitable for eigenvalue computations of matrices. Additionally, the results demonstrate that the RealAmplitude ansatz method delivers the highest number of accurate matches across different backends.



(a) Ansatz methode EfficentSU2



(b) Ansatz methode RealAmplitude



(c) Ansatz methode TwoLocal

Figure 6.4: VQE results by simulator and optimizer for different ansatz methodes

Another crucial question in addition to the appropriate backend and the associated ansatz method with a suitable optimizer is now the accuracy of VQE. In this regard, a random 2×2 matrix was created, and the VQE algorithm was applied 100 times consecutively to the same input instance. It became evident that the results for three distinct matrices exhibit substantial variation in accuracy, as illustrated in Figure 6.5. However, it is

also obvious that the values do not deviate significantly from the exact value but rather remain closely aligned. All three tests reveal outliers that are distinctly noticeable from the graphs. In graph 6.5a, an 11.11% deviation from the exact value is observed, while in graph 6.5b, nine out of the total 100 measurements notably deviate from the precise value. For graph 6.5c, the results vary a lot between the measurements, and the exact values were less frequently recorded compared to the other graphs. The deviation for graph 6.5c from the exact value is relatively small, only changing in the second decimal place.

These observations underscore the nuanced interplay between the VQE algorithm and the generated matrices. Moreover, the findings indicate both the potential for high accuracy and the occasional presence of deviations that need careful consideration.

In order to provide a more detailed examination of VQE's accuracy, we focused on evaluating the deviation of VQE-computed results from the exact value across various matrix dimensions. This analysis included results obtained from randomly generated matrices of dimensions 2×2, 4×4, 8×8, 16×16, and 32×32. Through multiple iterations of experimentation, a clear pattern emerged, highlighting a direct relationship between matrix dimension and the difference between computed and exact values.

Graphical representations in Figure 6.6 illustrate this trend. For matrices of sizes 2×2 and 4×4, the outcomes closely aligned with the exact values. However, as the dimension increased to 8×8, notable deviations became evident. Particularly unexpected was the observation for matrices of size 32×32, where a considerable discrepancy emerged between computed and exact values, often resulting in significant numerical differences.

To further assess the accuracy of VQE, the mean absolute error was computed across three distinct matrices of dimensions 2×2, 4×4, and 8×8. This evaluation was performed over 50 repeated executions. Here again, a clear trend emerges: the mean absolute error increases as the input size grows. While the error remains at 0.0006 for the 2×2 matrix, it notably raises for the other two input sizes, as depicted in Figure 6.7.

Subsequently, we investigate the execution times for the computation of $N \times N$ matrices using VQE with the SPSA optimizer and the RealAmplitudes ansatz. To initiate this assessment, it is important to determine where the experiments are executed. Therefore, we employed two distinct devices: a local laptop and an edge device functioning as a quantum simulator. The comprehensive specifications of these diverse devices are listed in Table 5.1. Time measurements on the local machine solely encapsulate the elapsed time for the VQE computation. For computations on the edge device, the network latency is additionally factored in alongside the elapsed time for the VQE computation. Figure 6.8 shows the execution time comparison between the local and remote devices. Evidently, the resource demands of VQE computations are significant, as the edge device already exhibits substantially longer computation times, even for minimal matrix dimensions, reflecting a difference of approximately 6.1 seconds. Upon investigating the temporal

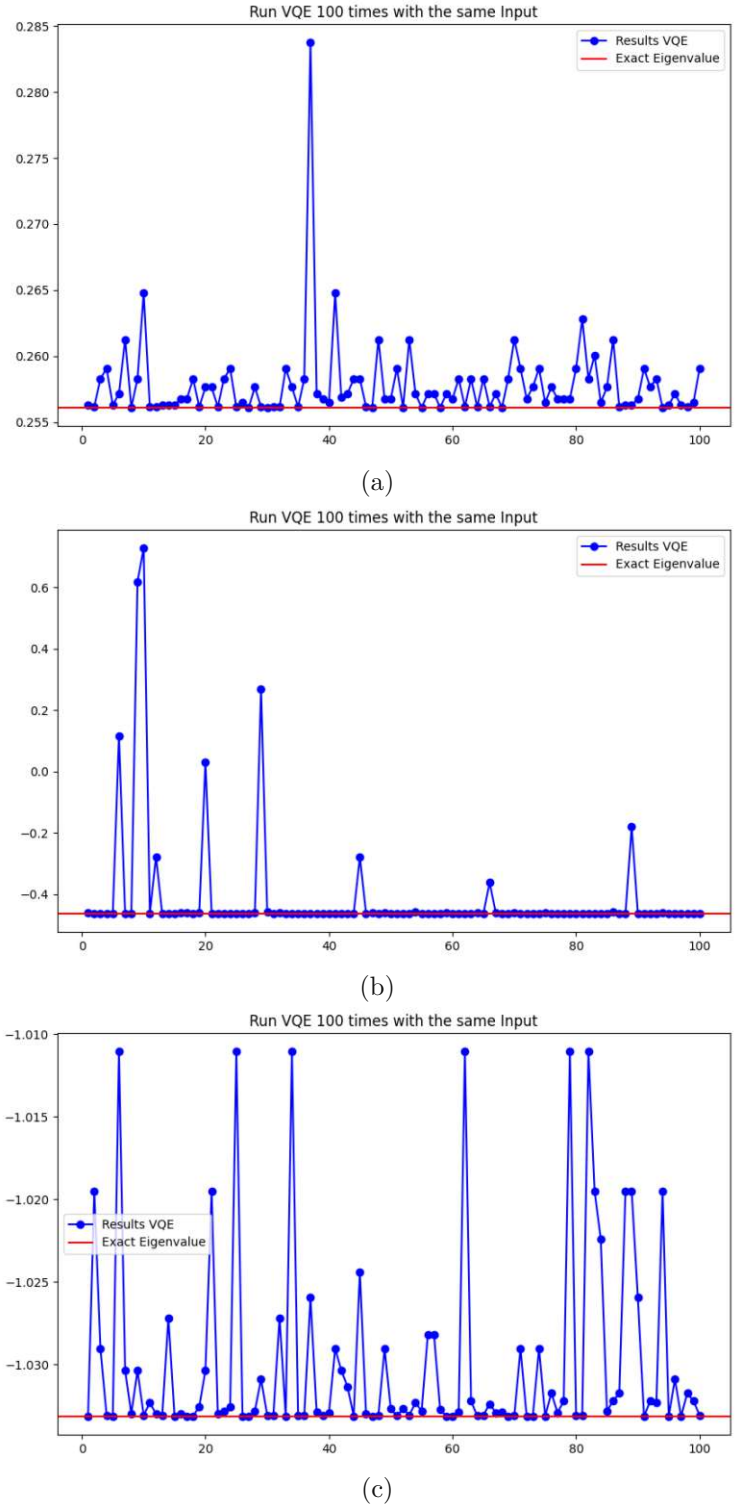


Figure 6.5: VQE accuracy for 100 executions on the same input instance

difference for a matrix size of 128×128, a considerable time discrepancy of 16.5 seconds emerges. Further analysis of the results reveals that the temporal computational overhead escalates significantly on the edge device. Furthermore, from Figure 6.9, it is evident that the RealAmplitude ansatz method exhibits the shortest computational time, consequently leading to the fastest calculation of a conclusive result. It is noticeable that the execution times for the other two ansatz methods are closely aligned. Given that the execution times are in the order of seconds, the impact of network latency, which operates in the millisecond range, remains inconsequential and can thus be effectively ignored.

6.2.1 VQE with noise model and error mitigation

So far, we have been working with backend simulators that are noise-free and, therefore, deliver ideal results. However, real-world quantum devices are exposed to environmental noise, which can lead to failure in calculations. For this reason, we computed minimal eigenvalues for hermitian matrices using an ideal simulator without noise. Subsequently, a noise model was added to the simulator to simulate environmental noise. The exact procedure for this approach is detailed explained in Section 5. Additionally, VQE was executed with a noise model and a error mitigation strategy, to minimize the impact of environmental noise on the final result. Table 6.1 provides the eigenvalue results for the various input matrices. It can be observed that the noise-free simulator backend yields the best results relative to the exact outcome. Nevertheless, it is worth noting that error mitigation can reduce the error in the results and demonstrates significant improvements compared to using only the noise model. Moreover, beside error mitigation another approach could be error correction strategies to further prevent systems for possible faulty results at a different stage during quantum processing.

input matrix	real eigenvalue	no noise	with noise	with noise and error mitigation
$\begin{bmatrix} 0.02542625 & 0.45665469 \\ 0.45665469 & -0.36131127 \end{bmatrix}$	-0.66385	-0.36131	0.22597	-0.25849
$\begin{bmatrix} 2.19979252 & 0.39313646 \\ 0.39313646 & -0.14372983 \end{bmatrix}$	-0.20792	-0.14373	0.21490	-0.00960
$\begin{bmatrix} 1.12137458 & -0.60837019 \\ -0.60837019 & -0.42483515 \end{bmatrix}$	-0.63550	-0.63549	-0.63298	-0.63512
$\begin{bmatrix} 0.90421152 & 0.04931608 \\ 0.04931608 & 0.15863771 \end{bmatrix}$	0.15539	0.15864	0.23450	0.17915
$\begin{bmatrix} 0.29751243 & 0.87265917 \\ 0.87265917 & -1.11624112 \end{bmatrix}$	-1.53240	-1.11624	-0.57915	-0.83989

Table 6.1: Comparative analysis of eigenvalue calculations for random hermitian matrices: classical and quantum approaches without and with noise modeling and error mitigation using `CompleteMeasFitter`.

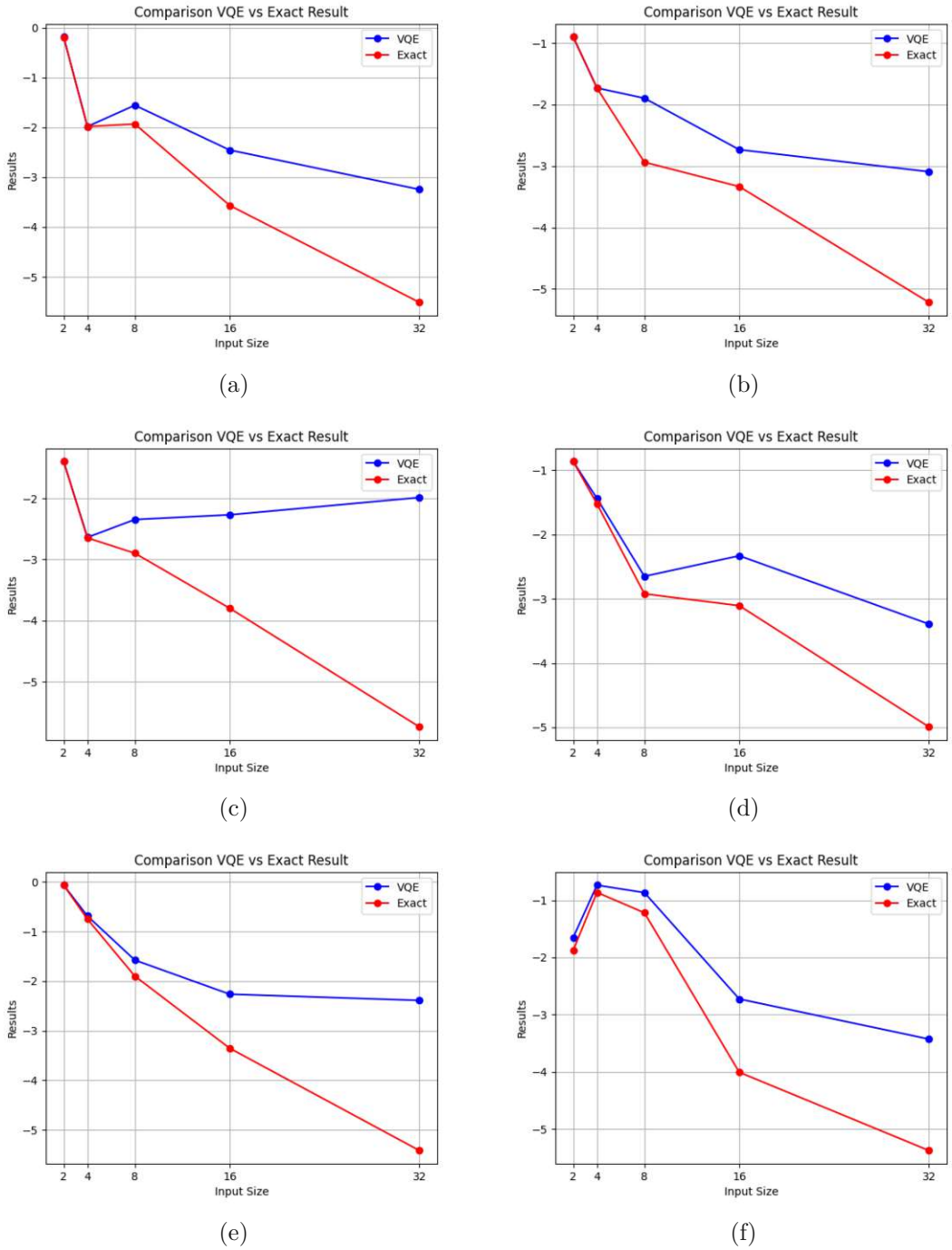


Figure 6.6: Comparison VQE result vs exact result for different matrix dimensions

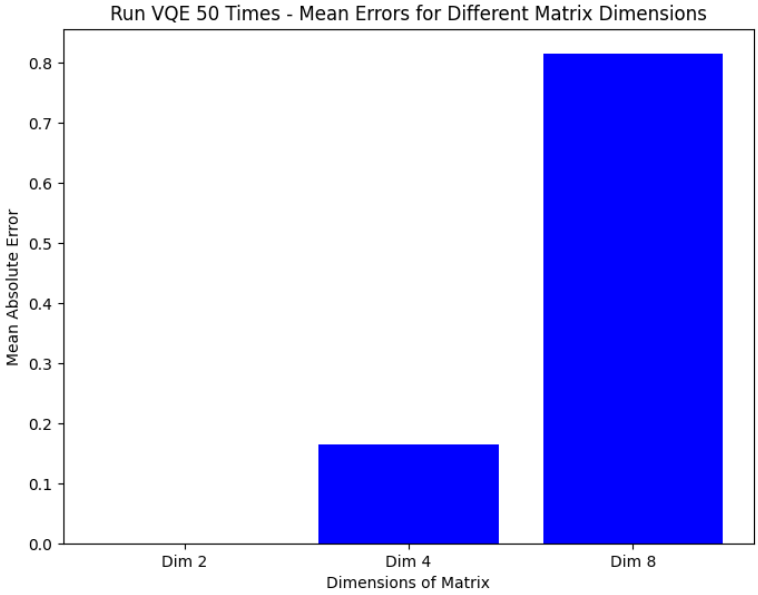


Figure 6.7: Mean absolute error after 50 VQE executions on same input instance for different matrix dimensions

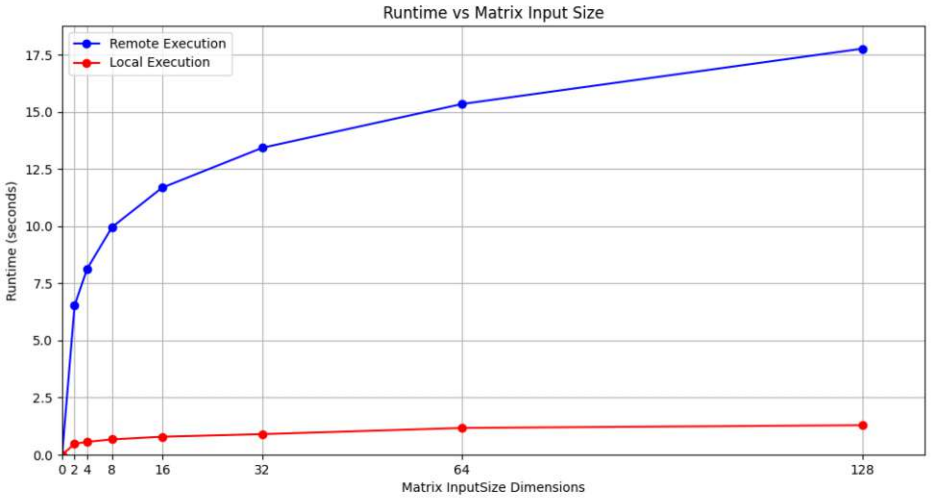


Figure 6.8: Computational time comparison of VQE calculation on local machine and edge quantum simulator

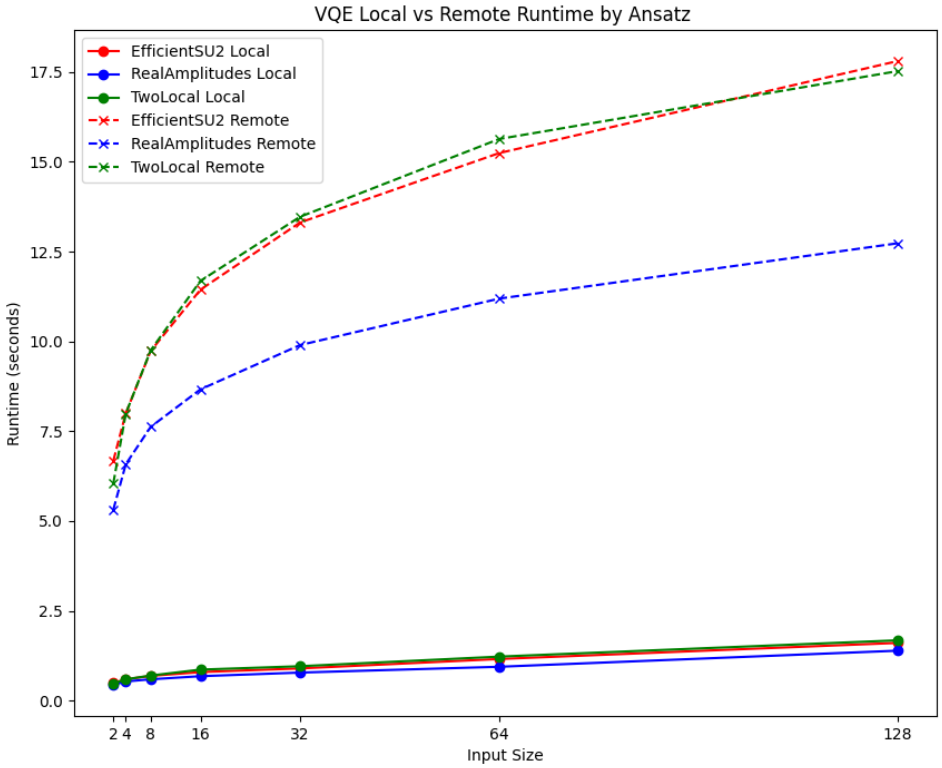
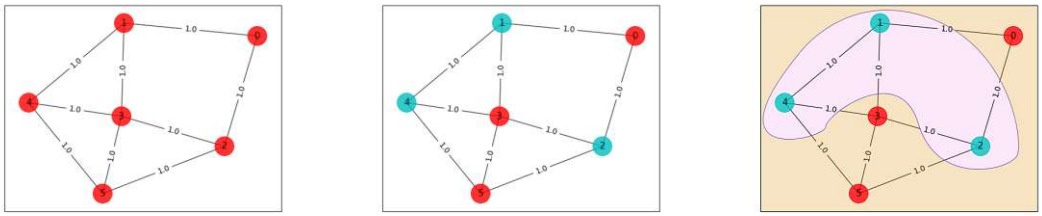


Figure 6.9: VQE execution on local machine and edge quantum simulator for different input size with different ansatz methods

6.3 Quantum Approximate Optimization Algorithm

With the aim of confirming the precision and accuracy of outcomes derived from a Quantum Approximate Optimization Algorithm (QAOA) instance, a brute force approach was utilized as a validation strategy. As part of various tests, random graphs with n nodes were generated, and subsequent computations were performed using both a brute force approach and the QAOA approach. An illustrative example of such a graph, featuring 6 nodes, along with the corresponding Max-Cut solution, is shown in Figure 6.10. In this context, the two distinct node colors showcased in the figure denote distinct subsets. This division aims to maximize the cumulative weight of edges traversing between the two subsets. The edge weights across all edges in the graph are set to 1 in the presented scenario. In Figure 6.10c, the boundary of the two subsets is visually emphasized for enhanced clarity.



(a) Max-Cut problem instance as graph

(b) Max-Cut solution with categorized nodes

(c) Max-Cut solution as two subsets

Figure 6.10: Max-Cut problem instance and final solution of node division in two subsets

Before having a deeper look into the performance aspects of QAOA, it is essential to comprehend the qubit requirements for a graph of size n. In a classical implementation, such as the one employed in our case, each node within the graph corresponds precisely to a single qubit. This correspondence arises from the binary nature of qubits, which can exist in states denoted as $|0\rangle$ or $|1\rangle$. Each of these states effectively references one of the two subsets in which a node can located within the context of the Max Cut problem. The principle of superposition allows for the simultaneous exploration of multiple states. When combined with quantum operations, it becomes a powerful tool for determining the correct outcome. In particular, it results in identifying the solution associated with the highest edge weight. Through the mechanism of measurement, carried out on the state of each individual qubit upon the completion of the QAOA procedure, it becomes feasible to determine the allocation of nodes to respective subsets. To summarize, it is evident that the quantity of qubits is directly linked to the number of nodes present within the graph. This assertion finds direct application in our initial illustration, where a graph containing 6 nodes corresponds to the utilization of 6 qubits, as also depicted in Figure 6.11.

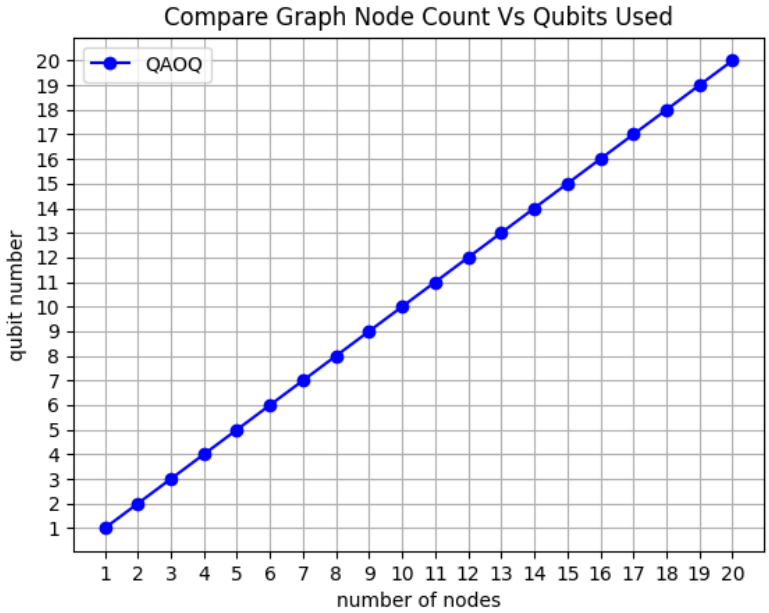


Figure 6.11: Comparison of QAOA qubit usage for different graph size

QAOA can also be executed on different backend simulators. Furthermore, various optimizers can be chosen for the respective hyperparameter tuning to obtain an optimal or near-optimal solution for the combinatorial problem. For this reason, the QAOA algorithm was executed on a problem instance with 6 nodes across different backends using various optimizers. The results are shown in Figure 6.12.

It is evident that the COBYLA optimizer consistently produces the correct result across all five utilized backends. Additionally, it can be observed that the ADAM optimizer fails to yield a correct result across all five backends. Similarly, the SPSA and SLSQP optimizers are not recommended as they predominantly generate incorrect results.

To make an well-founded decision about which backend simulator synergizes best with the COBYLA optimizer, the next step involves analyzing the execution times of individual QAOA instances across different backend and optimizer configurations. Figure 6.13 illustrates the computational times in comparison to the backend and optimizer choices.

The fusion of promising accuracy and rapid execution already establishes significant criteria. It is obvious that the computation using the *aer_simulator_matrix_product_state* backend requires relatively more time, making it a less favorable candidate for Max-Cut instances. On the other hand, the *aer_simulator_statevector* backend reveal the shortest average computation times and therefore it could be a good choice for solving these kind of problem instances.

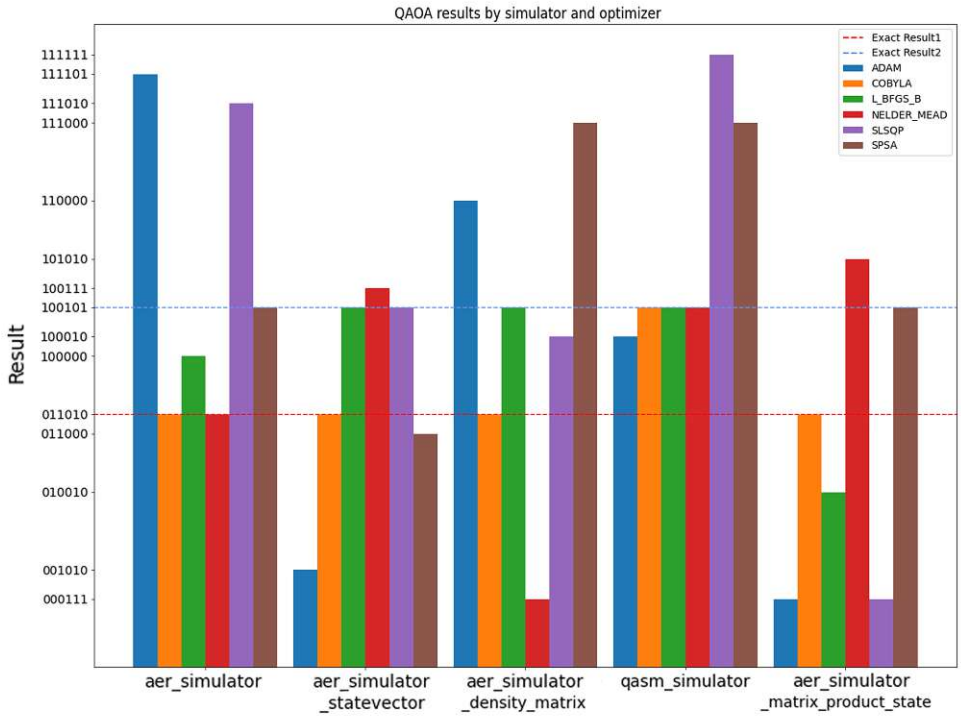


Figure 6.12: Results of QAOA execution on different backends using various optimizers. Two correct solutions are marked, reflecting the bitwise inversion between outcomes 100101 and 011010.

The accuracy of QAOA is a crucial determining factor in deciding whether and when the algorithm can be employed for Max-Cut instances. Therefore, it is essential to identify how frequently QAOA produces the correct result on a graph with 6 nodes. To achieve this, a graph is randomly generated, and subsequently, QAOA is applied to the same Max-Cut instance ten and one hundred consecutive times. Ideally, the correct results should be returned in a significant number of cases. Given that previous results have already identified COBYLA as the best optimizer and the *aer_simulator_statevector* as the optimal simulator backend, this configuration was applied for the following experiment with the problem instance.

It's remarkable to observe, as shown in Figure 6.14, that in all three cases, the correct result is returned in over 50% of all executions. In the instance given in Figure 6.14c, the correct result is even delivered with an 80% probability. To further validate the significance of the results, the same scenario was repeated 100 times. In Figure 6.14d, it's evident that 81 out of 100 instances yield the correct result, resulting in the correct outcome being produced over 80% of the time.

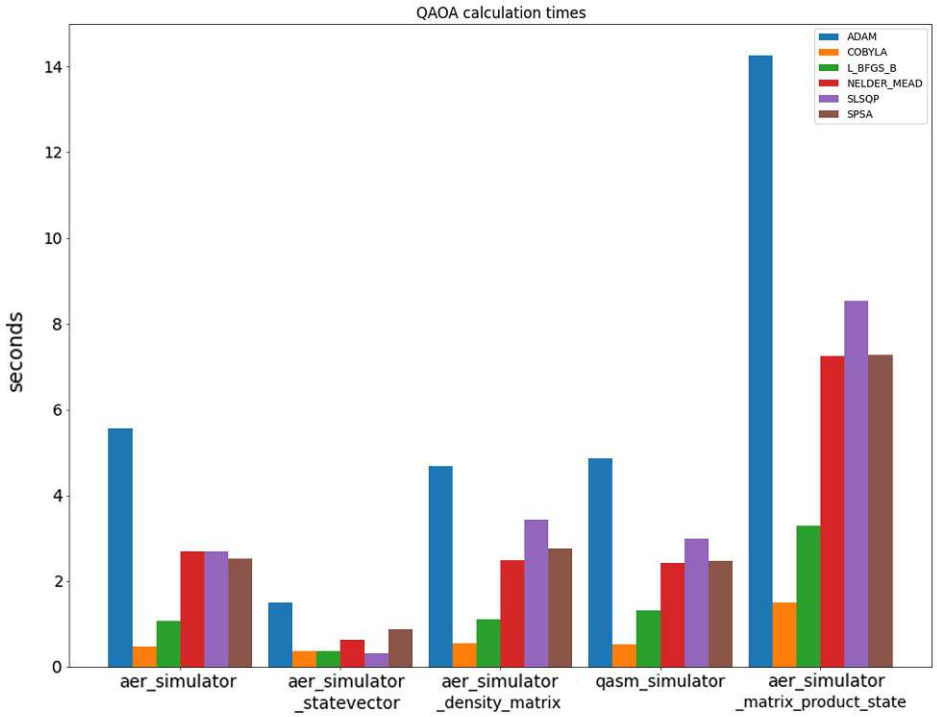
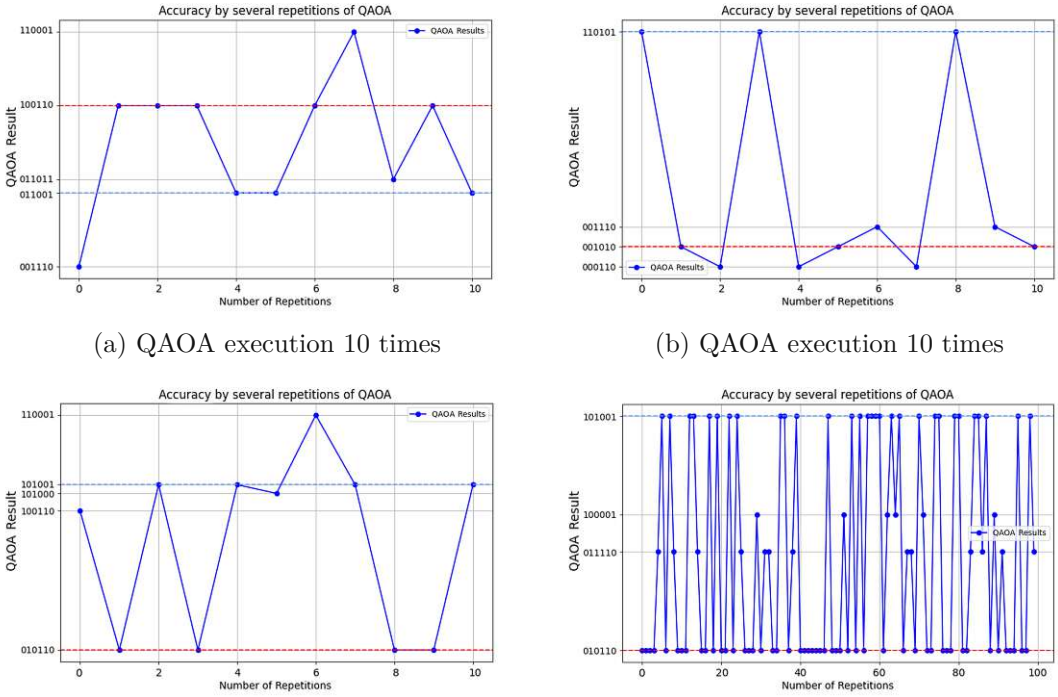


Figure 6.13: Computational times of QAOA instances on different backends with various optimizers

The Quantum Approximate Optimization Algorithm, as previously mentioned, can also be computed using a brute-force approach. This enables the calculation of the optimal solution. However, beyond a certain number of nodes in the graph, this computation can become highly time-consuming. Exploiting the principle of superposition, QAOA allows for multiple calculations to be performed simultaneously. As a result, Figure 6.15 illustrates the temporal behavior of QAOA for a problem instance of up to 10 nodes and another instance with up to 22 nodes. It can be observed that the brute force approach initially delivers results more quickly for small problem cases, but the differences remain less than a second up to a graph size of $n = 7$.

With an increasing number of nodes, the computation of QAOA becomes progressively slower for the given problem instance, while the brute-force variant maintains nearly consistent speed in the beginning, as depicted in Figure 6.15b. From a size of $n = 19$ onward, the computation time for the brute-force approach becomes noticeably raised. At this point, the brute-force and QAOA curves intersect for the first time, marking a turnover point beyond which QAOA's calculation becomes faster. The computation duration for the brute-force method escalates significantly, slowing considerably with the



(a) QAOA execution 10 times

(b) QAOA execution 10 times

(c) QAOA execution 10 times

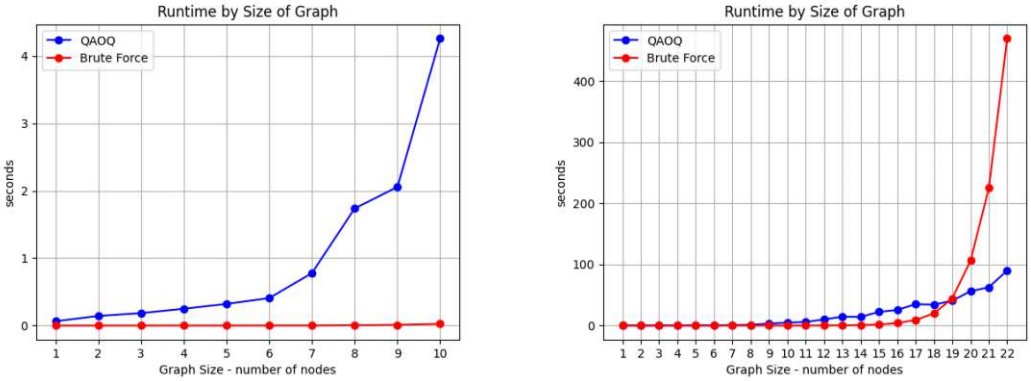
(d) QAOA execution 100 times

Figure 6.14: QAOA accuracy on multiple execution with same input instance

addition of each node. For $n = 20$, the computation time is 106.2 seconds, more than doubling to 225 seconds for $n = 21$. In the case of $n = 22$, there is again more than a doubling of time to 469.87 seconds, equivalent to 7.83 minutes.

Finally, we investigate the computation times of QAOA on the local device and the quantum edge simulator. These calculations provide insights into whether QAOA computation for larger instances could be possible at the network edge. To assess this, we conducted tests up to a maximum problem instance of $n = 15$. The results, presented in Figure 6.16, reveal that the computation time for the remote endpoint increases remarkably, particularly from a graph size of $n = 9$ onwards. Outcomes suggest a strong correlation between hardware resources and computation duration, given that the local machine has significantly more resources than the remote device.

In conclusion, an investigation was conducted to explore the feasibility of simplifying the QAOA circuit for the given problem instance using the Qiskit function *transpile*. To achieve this, a circuit was generated for a randomly constructed problem instance with a dimension of $n = 6$. Several experiments were made to simplify the circuit using the *transpile* function at various optimization levels. While optimization level 0 indicates the original circuit without any optimization, the level 3 is the highest possible optimization setting. The results of the experimentation clearly indicate that parameters such as



(a) Problem instance up to 10 nodes (b) Problem instance up to 22 nodes

Figure 6.15: BruteForce calculation runtime compared to QAOA approach

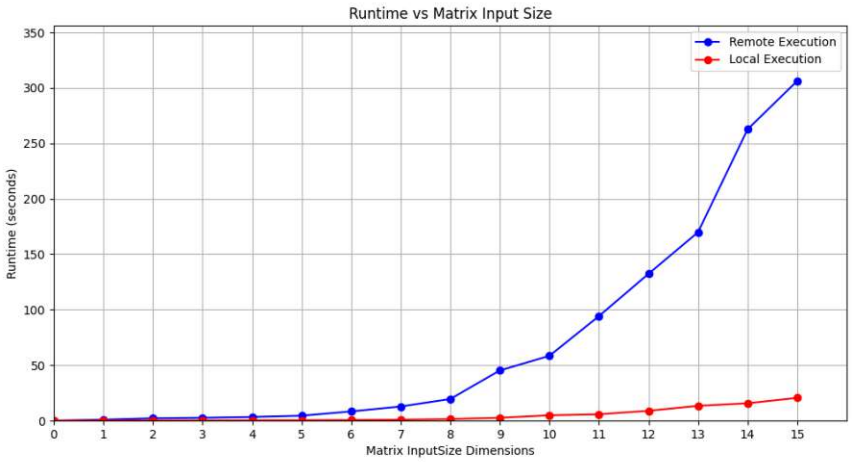
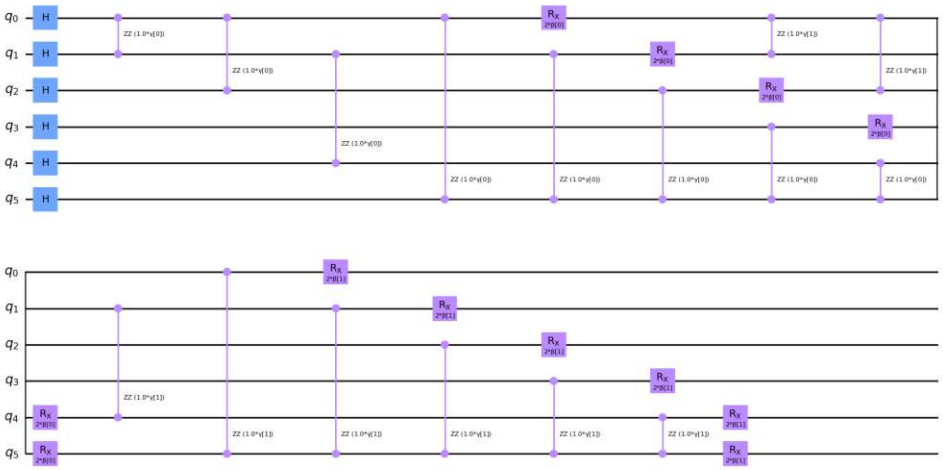


Figure 6.16: QAOA computational times compared for local machine and quantum edge simulator

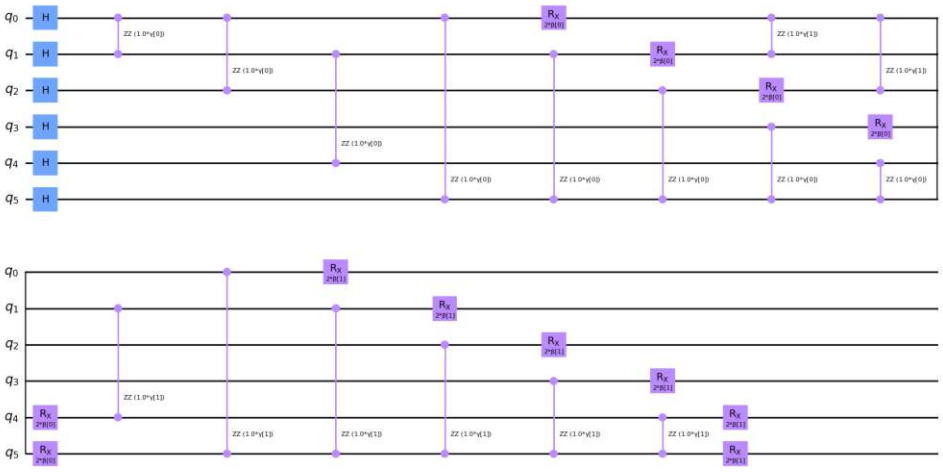
the number of qubits, the total count of instructions within the circuit, and the circuit depth (longest path in the circuit) remain unchanged regardless of the optimization value applied to the *transpile* function. Detailed outcomes are presented in Table 6.2, while Figure 6.17 illustrates the original circuit with the one transpiled using optimization level 3.

level	number of qubits	number of circuit instructions	depth
0	6	34	15
1	6	34	15
2	6	34	15
3	6	34	15

Table 6.2: QAOA circuit parameters for different transpile levels



(a) Transpile level 0



(b) Transpile level 3

Figure 6.17: Initial QAOA circuit compared to transpiled circuit

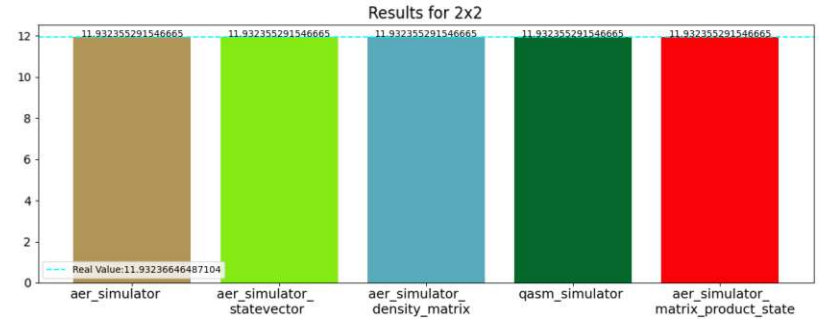
6.4 Harrow-Hassidim-Lloyd Algorithm

The solution to linear equation systems of the form $Ax = b$ can be obtained using the Harrow-Hassidim-Lloyd (HHL) algorithm on quantum hardware, aiming to potentially gain advantages in computational time for problem instances. In our case, we applied the HHL algorithm to various equation systems with dimensions 2×2, 4×4, and 8×8 and compared the results. Initially, we investigated the accuracy of different backend simulators with the mentioned problem instances. As illustrated in Figure 6.18, it can be clearly observed that, for all backend simulators, the HHL algorithm constantly produces the same results. This is a positive indication as it demonstrates consistency across various backend simulators. It is also evident that regardless of the chosen input size, the results remain consistent across the five backends. Deviations from the exact result are within an acceptable range, with noticeable discrepancies occurring only in the second decimal place for the 4×4 instance in our tests. For the other two problem instances, the absolute error is even smaller, amounting to just 0.39% for the 8×8 instance and 0.00009% for the 2×2 instance.

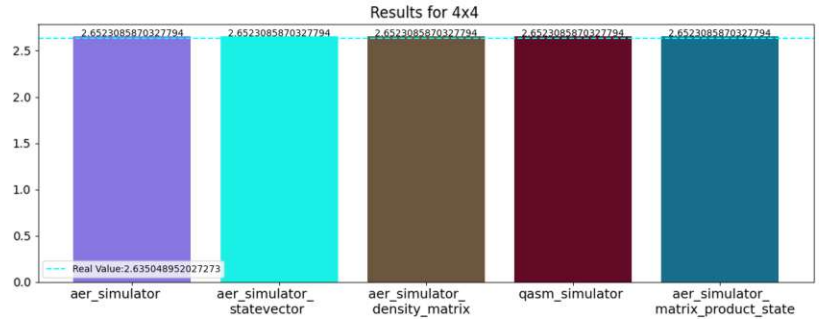
To select the appropriate backend for the given problem instances, it is crucial to consider the computational time required by each backend for solving a problem instance. Since we have already demonstrated that the accuracy of different backend simulators does not differ, an additional criterion is needed to determine the right backend. For this purpose, we performed the same calculations and used the same problem instances, but this time, we measured the computation duration of the HHL algorithm. The results are illustrated in Figure 6.19. It is evident that the *aer_simulator* backend results in relatively poor execution times, especially for dimensions 2×2 and 4×4 when compared to the other backend simulators. While for 2×2 instances, the *aer_simulator_statevector* backend delivers the fastest results. Regarding a 4×4 instance, the *qasm_simulator* backend is the fastest in terms of computation time. It is important to note that these differences are minimal, differing only by fractions of a second. For problem instances of size 8×8, the execution times of the individual simulator backends converge again, with the *aer_simulator_statevector* backend having the fastest computation time by a narrow margin. The choice of the corresponding simulator backend is therefore not critical for individual executions. However, for frequent executions, the choice of backend can make a difference, as computation times accumulate in such cases.

The detailed distribution of time for computing the solution for various input matrices of different dimensions using the *aer_simulator_statevector* backend is highlighted in Figure 6.20. For linear equation systems of dimensions 16×16 or larger, the computation time significantly increases, requiring more than one and a half minutes to find a corresponding solution.

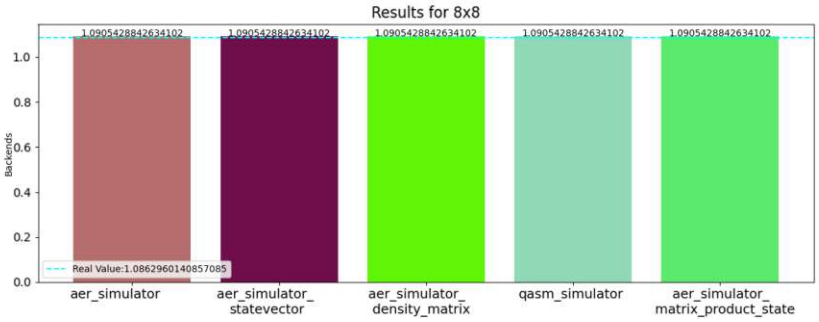
HHL is an algorithm that relies on complex quantum circuits in the background to compute a result. This implies the utilization of a relatively large number of quantum gates, resulting in a corresponding circuit depth. Each backend simulator operates in slightly different



(a) Linear equation dimension 2×2



(b) Linear equation dimension 4×4

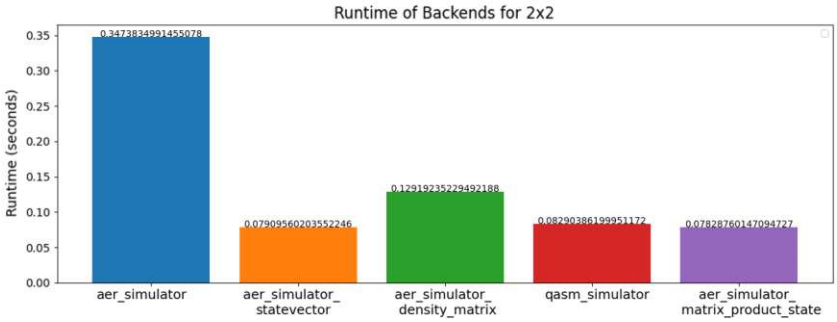


(c) Linear equation dimension 8×8

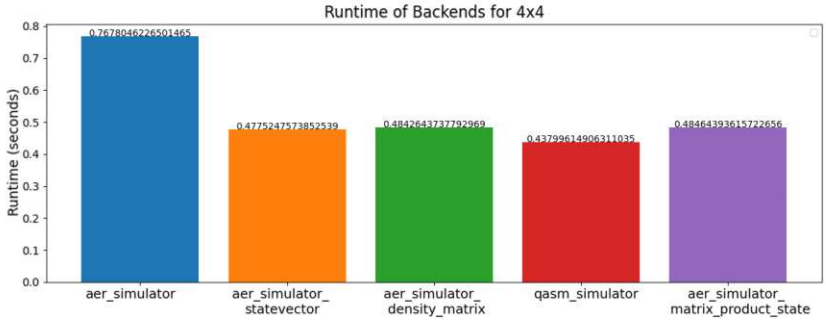
Figure 6.18: Accuracy of different simulator backends for various input dimension

ways, leading to variations in the circuits generated in the background. A comparison of the individual simulator backends based on their circuit parameters is shown in Figure 6.21a. From the diagram, it can be observed that the *aer_simulator_density_matrix* and *aer_simulator_matrix_product_state* backends employ 58.3̇% more gates than the other backends. This increase contributes to a higher circuit depth.

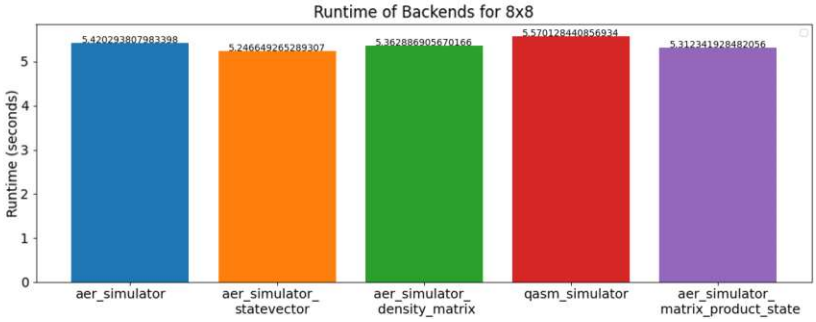
Qiskit enables the optimization of quantum circuits and their encoding for a specific quantum target device. In our case, we aim to optimize the quantum circuit for the *aer_simulator* backend. For this purpose, the *transpile* function offers various optimization



(a) Linear equation dimension 2×2



(b) Linear equation dimension 4×4



(c) Linear equation dimension 8×8

Figure 6.19: Comparison of runtime per simulator backend for different input dimension

levels from 0 to 3. As seen in Figure 6.21b, applying an optimization level of 1 reduces the circuit depth and gate size from 102 and 81 to 82 and 71. For the other optimization levels, there is no further impact, and the values remain unchanged. In this case, an optimal result can already be achieved with optimization level 1. It is essential to consider the effects on computation time for different optimization levels. Optimization level 3 significantly increases the time required to produce a result compared to the other levels. Figure 6.22 illustrates the temporal distribution of the various transpile levels in different sizes for linear equation systems. It is evident that the input dimensions of the linear

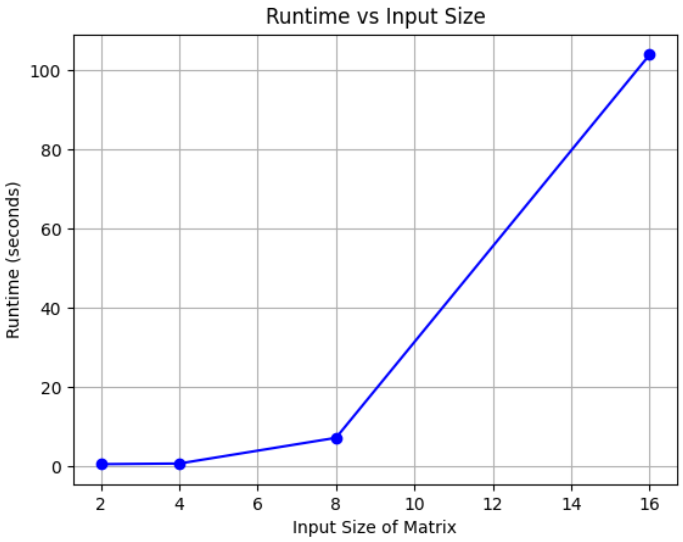
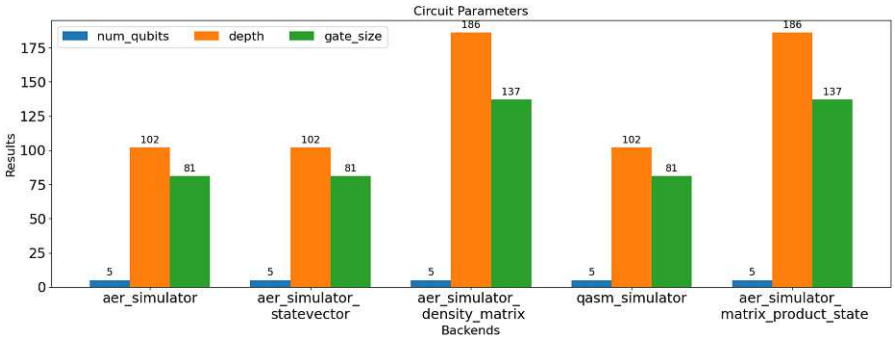
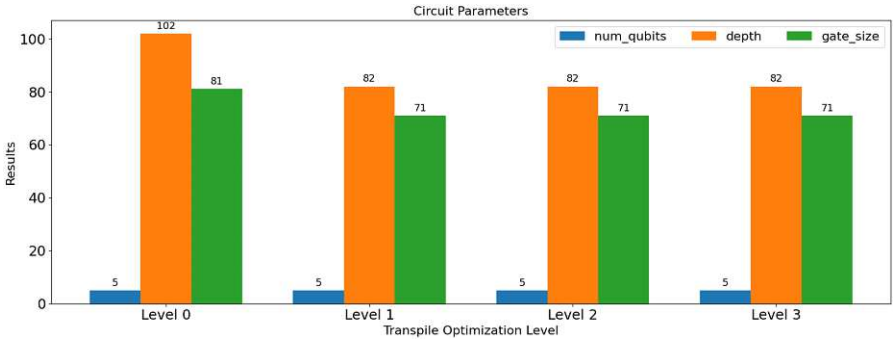


Figure 6.20: Computational time for different input dimension for HHL algorithm

equation system significantly influence the transpile time of the circuit. While 2×2 linear equation systems can be computed relatively quickly, this process becomes notably slower for 16×16 equation systems, with level 3 taking several minutes.
Finally, we investigated whether there are changes in circuit parameters (num_qubits, depth (longest path in the circuit), size (number of gates)) beyond a certain circuit size. As previously seen in Figure 6.21b, only optimization level 1 produced a change from the original result for a 2×2 input. We extended the experiment up to 8 problem instances and compare how the *transpile* function affects different circuit parameters. The results from that experiment are shown in Figure 6.23. Once again, a difference is observed only when transitioning from level 0 to level 1 since changes are only noticeable in the number of used gates and circuit depth. For levels 2 and 3, the values for the number of qubits, circuit depth, and size remain unchanged, identical to level 1.
The execution times of HHL on the edge quantum simulator and the local computer reveal significant differences. As the input size increases, a noticeable increase in computation time can be observed for both local and remote devices. The contrast becomes especially noticeable when considering the HHL algorithm's performance on the remote quantum simulator. The change in computation time from a dimension of 8×8 to 16×16 is significant. These effects are illustrated in Figure 6.24.

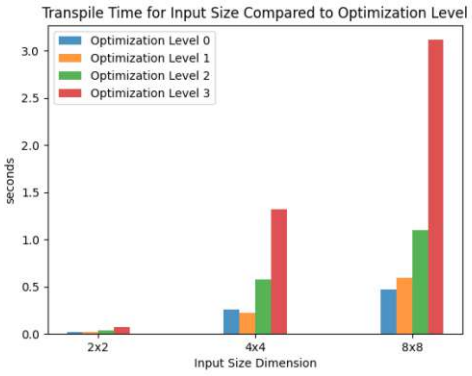


(a) Circuit Parameters for differnt simulator backends

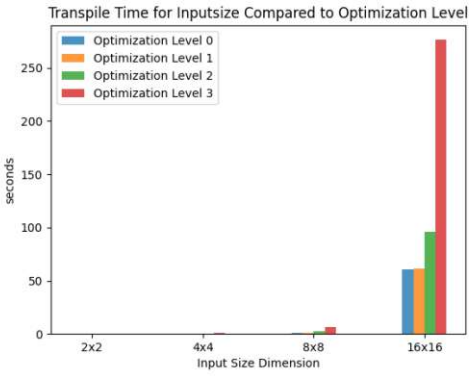


(b) Circuit parameters for *aer_simulator* backend with different optimization levels

Figure 6.21: Transpile HHL circuit observations



(a) Transpile times for up to 8×8 input size



(b) Transpile times for up to 16×16 input size

Figure 6.22: Transpile times for different input sizes in comparison

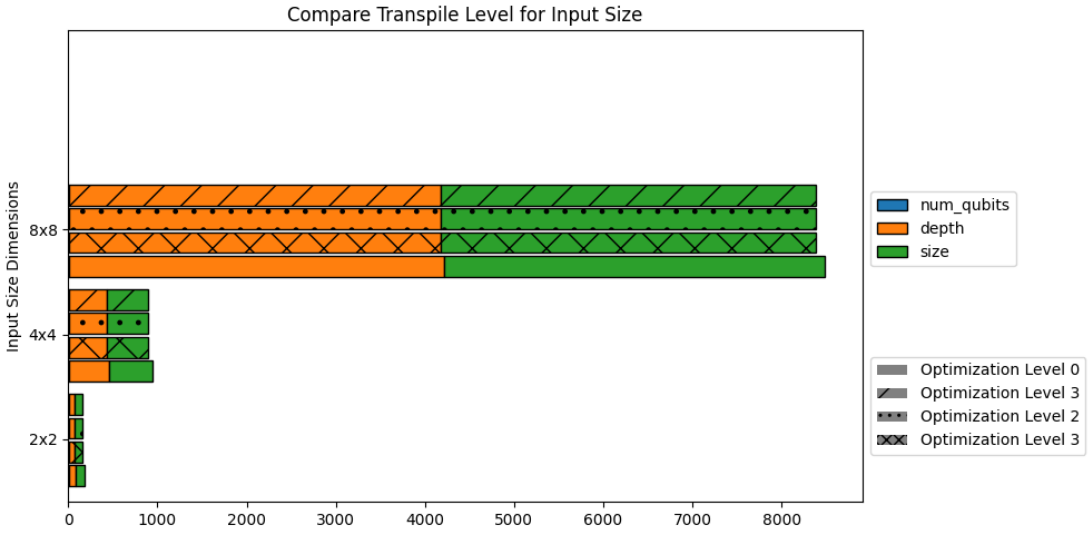


Figure 6.23: Transpile circuit parameters for different input dimensions of problem instance

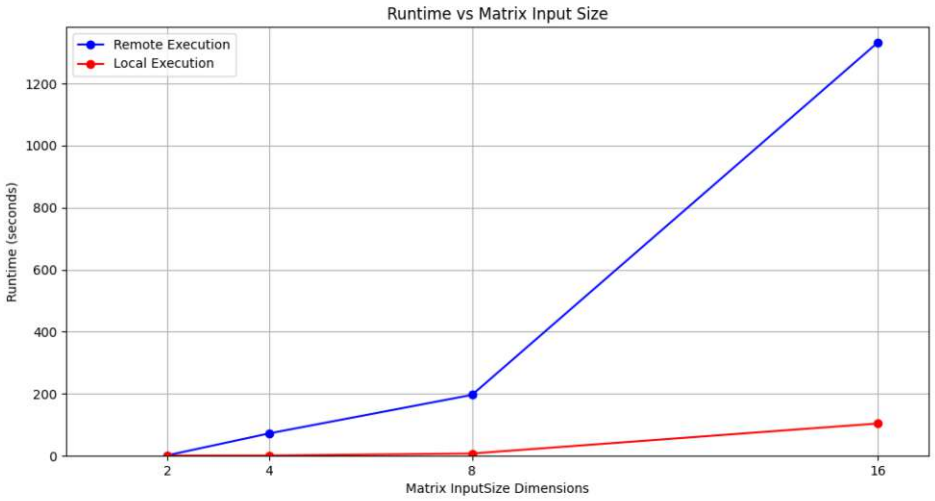


Figure 6.24: Computational times compared for local machine and quantum edge simulator

6.5 Discussion

The experimental scenarios conducted and analyzed in this chapter have shown the limitations of current quantum edge simulators. Furthermore, these experiments have provided valuable insights into the reliability of the algorithms under various conditions. Within the scope of the evaluation, significant differences in computation times between local and remote devices were observed. This phenomenon is distinctly evident in the cases of VQE, QAOA, and HHL, where the problem instance plays a crucial role in determining the computational duration on different devices. It can be concluded that the hardware of the devices is a critical factor influencing the computation time. In the case of QAOA, the results reveal that for problem instances with 19 or more nodes in the inputgraph, brute-force computation takes longer than obtaining results from a quantum edge simulator. Additionally, the HHL algorithm consistently exhibits the longest computation times, even for smaller problem instances, when compared to the computation times of VQE and QAOA. The execution times for different problem instance sizes for the three algorithms are shown in Figure 6.25. Furthermore, it is important to mention that alongside the temporal aspect, the inaccuracy or deviation from the exact result increases as the input size grows. This phenomena is particularly noticeable in the case of the VQE algorithm, where deviations for larger problem instances can reach integer values.

In general, the results identical that current quantum edge simulators are capable of performing timely calculations. However, careful consideration is necessary when considering the deployment of simulators at the network edge. For critical applications where ensuring 100% accuracy is mandatory, such deployments are not advisable. Conversely, when the objective is to compute approximations or work with approximate values, the use of quantum edge simulators can be a viable option. Additionally, aside from accuracy, significant attention must be paid to the resource scaling of the utilized simulators to obtain results within the expected timeframe.

In the context of specific quantum algorithms, the choice of optimizer and ansatz method in VQE was found to influence result accuracy, with a decreasing accuracy trend observed with larger matrix dimensions. The RealAmplitude ansatz most of the time performed best for both speed and accuracy. QAOA consistently provided accurate solutions for Max-Cut problems, with the COBYLA optimizer showing promising performance across different backend simulators. HHL showed consistent accuracy across backend simulators, with execution times increasing notably for larger input dimensions. These findings emphasize the importance of selecting the appropriate quantum algorithm based on specific problem requirements and the available computational resources. The observed trade-off between accuracy and computation time underscores the need for a nuanced approach in algorithm selection, particularly in scenarios where both speed and precision are critical factors.

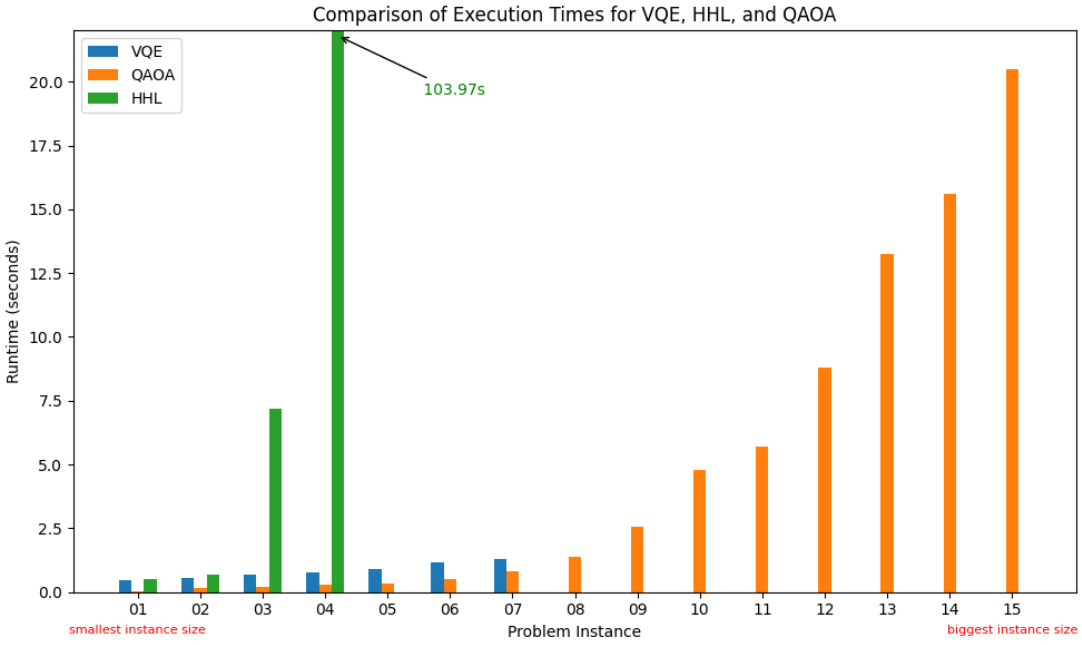


Figure 6.25: Computational times compared for VQE, HHL and QAOA for different problem instances sizes. Problem instances on the x-axis are visualized in ascending order.

CHAPTER 7

Conclusion

7.1 Summary

Quantum computing refers to a new way of solving computational problems. The hardware based on this concept fundamentally differs from conventional computers in our daily lives. As it is anticipated that more computational power will be required in the future, it may be worthwhile to offload complex tasks that can be effectively solved using quantum mechanical principles to these devices. Even today, edge computing is gaining significant interest in offloading certain computation tasks to usually more powerful node. It is precisely in this context that we have identified a new potential benefit and opportunity for efficient computation on quantum edge simulators at the network's edge.

The objective of this thesis was to implement the algorithms VQE, QAOA, and HHL using Qiskit and analyze their behavior in greater detail. Numerous benchmark tests were conducted to make assessments regarding execution time and accuracy based on the usage of different hyperparameter configurations. Additionally, these experiments provided recommendations for the most optimal hyperparameter settings for the algorithms.

The individual results have shown that quantum edge simulators could play a crucial role for solving problem instances in the future. However, some performance improvements are still needed to provide near-real-time results. Our thesis results have demonstrated that for large input sizes, the computation time increases significantly for all three tested algorithms. Comparatively, there are substantial differences in computation time when compared to a more powerful local device. This suggests that resource-rich edge devices are required to realize current quantum edge simulators.

Furthermore, it has become evident that the accuracy of the returned results is not always achieved. Instead, in some cases, the results are an approximation of the exact

solution, with a certain degree of deviation, particularly noticeable in VQE. On the other hand, the HHL algorithm is the most resource-intensive, thus requiring the longest time to produce results. QAOA, however, falls in between the other two algorithms concerning accuracy and execution time.

The experiments conducted provide essential insights into which hyperparameter configurations should be used for the individual algorithms to achieve the best possible results for the problem instances used in this work. The choice of the corresponding simulator backend, optimizer or ansatz is often a crucial factor in the algorithm's performance.

7.2 Future Work

The research conducted in this thesis provides valuable insights into the feasibility and performance of deploying quantum simulators at the network edge for offloading computational tasks. However, there are several areas for future work that can expand upon and enhance the findings presented here.

Scaling to Larger Problems While this thesis primarily focused on small-sized quantum problems, future research could investigate the scalability of quantum simulators at the network edge. Exploring their ability to address larger and more complex quantum problems would be instrumental in understanding the practical limits of edge devices. This involves not only evaluating the computational power of edge devices but also optimizing algorithms and methodologies to handle more extensive problem instances efficiently.

Integration with Real Quantum Hardware As quantum computing technology advances, the integration of real quantum hardware into the benchmarking and offloading process becomes increasingly important. Future work could involve incorporating access to real quantum computers for benchmarking and evaluating their performance in comparison to quantum simulators and edge devices. This integration would require addressing the challenges of orchestrating tasks between quantum simulators and real quantum hardware, taking into account the varying capabilities and limitations of these platforms.

Exploring Additional Quantum Algorithms In addition to the VQE, QAOA, and HHL algorithms, future research could expand the scope to include other quantum algorithms. Assessing the suitability of various quantum algorithms for offloading to edge devices would provide a more comprehensive understanding of edge computing's potential in the quantum domain. This exploration may uncover novel algorithms or hybrid approaches that are well-suited for edge quantum simulators.

Exploring Diverse Problem Instances Throughout this thesis, our focus has primarily been on running individual quantum algorithms with specific problem instances.

However, there is significant potential for expanding our research to encompass a broader range of problem instances, particularly within the context of QAOA. While our current results are based exclusively on instances of the Max-Cut problem, further exploration can involve applying QAOA to solve different problem instances. By diversifying the problem domains we address, we can gain a more comprehensive understanding of the algorithm's performance and applicability in various scenarios.

Optimizing Edge Device Performance Efforts to optimize the performance of quantum edge simulators yield valuable results. This optimization may involve hardware enhancements, software optimizations, or exploring different edge computing architectures to achieve improved computational capabilities. Future work should look into the characteristics of edge device optimization, considering factors such as power efficiency, memory management, and parallel processing capabilities.

Real-World Applications Extending the research to evaluate the practical applicability of offloading quantum computations to edge devices in real-world scenarios is crucial. Investigating use cases in industries such as finance, logistics, and materials science could provide valuable insights into the benefits of edge quantum computing. Future studies could involve collaboration with domain experts to identify specific challenges and opportunities for quantum-powered edge solutions.

Noise Models and Quantum Error Mitigation All experiments were primarily conducted on ideal quantum simulators that do not employ noise models. Consequently, the quantum edge simulators were run without environmental noise, which can influence real-world calculations. It is, therefore, reasonable to apply and analyze noise models and error mitigation strategies to all individual algorithms, similar to the briefly mentioned implementation for VQE with noise models and error mitigation in Chapter 5.2.3. Various fake backends can be utilized for this purpose to test their diverse effects on environmental noise and the final outcome. Subsequently, errors arising from noise can be reduced using different error mitigation strategies. This work presents just one approach to noise reduction, but there are many different approaches available to improve the final result. Furthermore, exploring techniques such as error-correcting codes and adaptive error mitigation algorithms tailored to the constraints of edge computing environments could be valuable.

Machine Learning Integration The integration of machine learning techniques for optimizing the offloading process and decision-making regarding task allocation between edge devices and quantum resources presents an intriguing area for future research. Exploring the synergy between quantum computing and machine learning can yield innovative solutions. Future work should consider the development of adaptive algorithms that use machine learning to dynamically allocate computational tasks based on the capabilities and workloads of edge and quantum devices.

By addressing these areas in future research, we can continue to build upon the findings of this thesis and contribute to a deeper understanding of the potential of edge quantum simulators in the evolving landscape of quantum computing.

List of Figures

List of Tables

List of Code Samples

Acronyms

DFT Density-Functional Theory. 17

DMFT Dynamical Mean-Field Theory. 17

HHL Harrow-Hassidim-Lloyd. 14, 25, 26, 28, 39–41, 59, 62, 65, 67, 68

HNN Hybrid Neural Network. 18

MAE Mean Absolute Error. 30

MAPE Mean Absolute Percentage Error. 30

NISQ Noisy Intermediate-Scale Quantum. 14, 15, 23

QAOA Quantum Approximate Optimization Algorithm. 13, 14, 19, 20, 22, 23, 25–28, 37–39, 41, 52–56, 65, 67–69

QEC Quantum Error Correction. 15

QFT Quantum Fourier Transform. 14

QNN Quantum Neural Network. 23

QPE Quantum Phase Estimation. 14

QPU Quantum Processing Unit. 18

QSE Quantum Software Engineering. 18

RSA Rivest–Shamir–Adleman. 19

TCP Transmission Control Protocol. 26, 40, 43

UML Unified Modeling Language. 18

VQA Variational Quantum Algorithm. 1, 3, 4, 12, 13, 17, 21–23, 25, 33, 35, 37

VQE Variational Quantum Eigensolver. 12, 13, 19, 20, 22, 23, 25–28, 35–39, 41, 44–46, 48, 65, 67–69

VRP Vehicle Routing Problem. 22

Bibliography

[AASG19] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. Analysis of quantum approximate optimization algorithm under realistic noise in superconducting qubits, 2019.

[ADA] Adam - qiskit 0.45.0 documentation. `https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.ADAM.html#qiskit.algorithms.optimizers.ADAM`. (Accessed on 11/19/2023).

[AF23] Muhammad Alsaiyari and Muhamad Felemban. Variational quantum algorithms for solving vehicle routing problem. In *2023 International Conference on Smart Computing and Application (ICSCA)*, pages 1–4, 2023.

[AFG+09] M Armbrust, A Fox, R Griffith, AD Joseph, R Katz, A Konwinski, G Lee, D Patterson, A Rabkin, I Stoica, et al. Above the clouds: A berkeley view of cloud computing. eecs dept. *Univ. California, Berkeley, No. UCB/EECS-2009-28 [Online]. Available: http://radlab. cs. berkeley. edu*, 2009.

[AGM06] Antonio Acín, Nicolas Gisin, and Lluis Masanes. From bell's theorem to secure quantum key distribution. *Phys. Rev. Lett.*, 97:120405, Sep 2006.

[AL97] Daniel S. Abrams and Seth Lloyd. Simulation of many-body fermi systems on a universal quantum computer. *Physical Review Letters*, 79(13):2586–2589, sep 1997.

[AMR+22] David Amaro, Carlo Modica, Matthias Rosenkranz, Mattia Fiorentini, Marcello Benedetti, and Michael Lubasch. Filtering variational quantum algorithms for combinatorial optimization. *Quantum Science and Technology*, 7(1):015021, feb 2022.

[AS11] R Ainsworth and J K Slingerland. Topological qubit design and leakage. *New Journal of Physics*, 13(6):065030, jun 2011.

[aZW23] Hector Jose Morrell Jr au2, Anika Zaman, and Hiu Yung Wong. Step-by-step hhl algorithm walkthrough to enhance the understanding of critical quantum computing concepts, 2023.

[BAAM20] J.-H. Bae, Paul M. Alsing, Doyeol Ahn, and Warner A. Miller. Quantum circuit optimization using quantum karnaugh map. *Scientific Reports*, 10(1):15651, Sep 2020.

[Bac13] Dave Bacon. *Introduction to quantum error correction*, page 46–77. Cambridge University Press, 2013.

[BFL22] Eric Bourreau, Gérard Fleury, and Philippe Lacomme. Mixer hamiltonian with qaoa for max k-coloring : numerical evaluations, 2022.

[BIS+18] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595–600, Jun 2018.

[BKGN+13] Robin Blume-Kohout, John King Gamble, Erik Nielsen, Jonathan Mizrahi, Jonathan D. Sterk, and Peter Maunz. Robust, self-consistent, closed-form tomography of quantum logic gates on a trapped ion qubit, 2013.

[BWM+16] Bela Bauer, Dave Wecker, Andrew J. Millis, Matthew B. Hastings, and Matthias Troyer. Hybrid quantum-classical approach to correlated materials. *Physical Review X*, 6(3), sep 2016.

[CAB+21] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, Sep 2021.

[CBAK13] Shantanav Chakraborty, Subhashish Banerjee, Satyabrata Adhikari, and Atul Kumar. Entanglement in the grover's search algorithm, 2013.

[CDMB+22] Sandeep Suresh Cranganore, Vincenzo De Maio, Ivona Brandic, Tu Mai Anh Do, and Ewa Deelman. Molecular dynamics workflow decomposition for hybrid classic/quantum systems. In *2022 IEEE 18th International Conference on e-Science (e-Science)*, pages 346–356, 2022.

[Cir] Circuit sampling in an algorithm — qiskit runtime ibm client 0.12.0 documentation. `https://qiskit.org/ecosystem/ibm-runtime/migrate/migrate-sampler.html`. (Accessed on 09/11/2023).

[CK74] V. Cerf and R. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, 22(5):637–648, 1974.

[COB] Cobyla - qiskit 0.45.0 documentation. `https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.COBYLA.html#qiskit.algorithms.optimizers.COBYLA`. (Accessed on 11/19/2023).

[Com09] Clayton W. Commander. *Maximum cut problem, MAX-CUTMaximum Cut Problem, MAX-CUT*, pages 1991–1999. Springer US, Boston, MA, 2009.

[Con] Configure error mitigation — qiskit runtime ibm client 0.12.0 documentation. `https://qiskit.org/ecosystem/ibm-runtime/how_to/error-mitigation.html`. (Accessed on 09/09/2023).

[Coo22] Crispin H. V. Cooper. Exploring potential applications of quantum computing in transportation modelling. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14712–14720, 2022.

[CWY+23] Bingren Chen, Hanqing Wu, Haomu Yuan, Lei Wu, and Xin Li. Quantum portfolio optimization: Binary encoding of discrete variables for qaoa with hard constraint, 2023.

[dLMPL19] Franklin de Lima Marquezino, Renato Portugal, and Carlile Lavor. *Bits and Qubits*, pages 7–34. Springer International Publishing, Cham, 2019.

[dW17] Ronald de Wolf. The potential impact of quantum computers on society. *Ethics and Information Technology*, 19(4):271–276, Dec 2017.

[EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47:777–780, May 1935.

[EW06] J. Eisert and M. M. Wolf. *Quantum Computing*, pages 253–286. Springer US, Boston, MA, 2006.

[fak] fake_provider | ibm quantum documentation. `https://docs.quantum-computing.ibm.com/api/qiskit/providers_fake_provider`. (Accessed on 09/09/2023).

[FGG14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[Get] Getting started - qiskit 0.44.0 documentation. https://qiskit.org/documentation/getting_started.html. (Accessed on 08/13/2023).

[Git] Github - janlahmann/rasqberry: The rasqberry project: Exploring quantum computing and qiskit with a raspberry pi and a 3d printer. https://github.com/JanLahmann/RasQberry. (Accessed on 08/13/2023).

[GKS+20] Sukhpal Singh Gill, Adarsh Kumar, Harvinder Singh, Manmeet Singh, Kamalpreet Kaur, Muhammad Usman, and Rajkumar Buyya. Quantum computing: A taxonomy, systematic review and future directions. *CoRR*, abs/2010.15559, 2020.

[GKS+22] Sukhpal Singh Gill, Adarsh Kumar, Harvinder Singh, Manmeet Singh, Kamalpreet Kaur, Muhammad Usman, and Rajkumar Buyya. Quantum computing: A taxonomy, systematic review and future directions. *Software: Practice and Experience*, 52, 01 2022.

[Got97] Daniel Gottesman. Stabilizer codes and quantum error correction, 1997.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996.

[GS21] Paul Griffin and Ritesh Sampat. Quantum computing for supply chain finance. In *2021 IEEE International Conference on Services Computing (SCC)*, pages 456–459, 2021.

[HHL09a] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.

[HHL09b] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), oct 2009.

[Hid19] Jack D. Hidary. *Qubits, Operators and Measurement*, pages 17–36. Springer International Publishing, Cham, 2019.

[HLL+22] Jiaqi Hu, Junning Li, Yanling Lin, Hanlin Long, Xu-Sheng Xu, Zhaofeng Su, Wengang Zhang, Yikang Zhu, and Man-Hong Yung. Benchmarking variational quantum eigensolvers for quantum chemistry, 2022.

[Hom22a] Matthias Homeister. *Fehler korrigieren*, pages 235–249. Springer Fachmedien Wiesbaden, Wiesbaden, 2022.

[Hom22b] Matthias Homeister. *Vom Bit zum Quantenregister*, pages 9–66. Springer Fachmedien Wiesbaden, Wiesbaden, 2022.

[Hom22c] Matthias Homeister. *Vom Quantenregister zum Quantenschaltkreis*, pages 67–98. Springer Fachmedien Wiesbaden, Wiesbaden, 2022.

[JBTS19] Farzan Jazaeri, Arnout Beckers, Armin Tajalli, and Jean-Michel Sallese. A review on quantum computing: Qubits, cryogenic electronics and cryogenic mosfet physics, 2019.

[JJ12] Jonathan A. Jones and Dieter Jaksch. *Quantum Information, Computation and Communication*. Cambridge University Press, 2012.

[Kas21a] Venkateswaran Kasirajan. *Quantum Circuits and DiVincenzo Criteria*, pages 149–233. Springer International Publishing, Cham, 2021.

[Kas21b] Venkateswaran Kasirajan. *Quantum Error Correction*, pages 375–430. Springer International Publishing, Cham, 2021.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[Kit95] A. Yu. Kitaev. Quantum measurements and the abelian stabilizer problem, 1995.

[Kom20] Bhagvan Kommadi. *Quantum Computing Solutions: Solving Real-World Problems Using Quantum Computing and Algorithms*. Apress Berkeley, CA, 12 2020.

[Kra88] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988.

[KW22] Ioannis Kolotouros and Petros Wallden. Evolving objective function for improved variational quantum optimization. *Phys. Rev. Res.*, 4:023225, Jun 2022.

[LB13] Ching-Yi Lai and Todd A. Brun. Entanglement increases the error-correcting ability of quantum error-correcting codes. *Phys. Rev. A*, 88:012320, Jul 2013.

[LBF] L_bfgs_b - qiskit 0.45.0 documentation. `https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.L_BFGS_B.html#qiskit.algorithms.optimizers.L_BFGS_B`. (Accessed on 11/19/2023).

[LCS+23] Huan-Yu Liu, Zhao-Yun Chen, Tai-Ping Sun, Cheng Xue, Yu-Chun Wu, and Guo-Ping Guo. Can variational quantum algorithms demonstrate quantum advantages? time really matters, 2023.

[LLSK22] Jonathan Wei Zhong Lau, Kian Hwee Lim, Harshank Shrotriya, and Leong Chuan Kwek. Nisq computing: where are we and where do we go? *AAPPS Bulletin*, 32(1):27, Sep 2022.

[LWW+21] Hai-Ling Liu, Yu-Sen Wu, Lin-Chun Wan, Shi-Jie Pan, Su-Juan Qin, Fei Gao, and Qiao-Yan Wen. Variational quantum algorithm for the poisson equation. *Phys. Rev. A*, 104:022418, Aug 2021.

[mat] Matplotlib documentation — matplotlib 3.7.1 documentation. `https://matplotlib.org/stable/index.html`. (Accessed on 04/14/2023).

[MBMRMRAE23] Vicente Moret-Bonillo, Eduardo Mosqueira-Rey, Samuel Magaz-Romero, and Diego Alvarez-Estevez. Hybrid classic-quantum computing for staging of invasive ductal carcinoma of breast, 2023.

[MEAG+20] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. *Rev. Mod. Phys.*, 92:015003, Mar 2020.

[MGJ+12] Easwar Magesan, Jay M. Gambetta, B. R. Johnson, Colm A. Ryan, Jerry M. Chow, Seth T. Merkel, Marcus P. da Silva, George A. Keefe, Mary B. Rothwell, Thomas A. Ohki, Mark B. Ketchen, and M. Steffen. Efficient measurement of quantum gate error by interleaved randomized benchmarking. *Phys. Rev. Lett.*, 109:080505, Aug 2012.

[MH21] Ryutaroh Matsumoto and Manabu Hagiwara. A survey of quantum error correction. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 104-A:1654–1664, 2021.

[MK13] C. Monroe and J. Kim. Scaling the ion trap quantum processor. *Science*, 339(6124):1164–1169, 2013.

[MM12] Dan C. Marinescu and Gabriela M. Marinescu. Chapter 1 - preliminaries. In Dan C. Marinescu and Gabriela M. Marinescu, editors, *Classical and Quantum Information*, pages 1–131. Academic Press, Boston, 2012.

[MYZ+17] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358, 2017.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[NEL] Nelder_mead - qiskit 0.45.0 documentation. https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.NELDER_MEAD.html. (Accessed on 11/19/2023).

[Net] Networkx — networkx documentation. https://networkx.org/. (Accessed on 08/13/2023).

[NM65] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 01 1965.

[Noi] Noisy simulators in qiskit runtime — qiskit runtime ibm client 0.12.0 documentation. https://qiskit.org/ecosystem/ibm-runtime/how_to/noisy_simulators.html. (Accessed on 09/09/2023).

[Num] Numpy. https://numpy.org/. (Accessed on 11/20/2023).

[pan] pandas - python data analysis library. https://pandas.pydata.org/. (Accessed on 11/20/2023).

[PCP22] Ricardo Pérez-Castillo and Mario Piattini. Design of classical-quantum systems with uml. *Computing*, 104(11):2375–2403, Nov 2022.

[PMS+14] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1), jul 2014.

[Pos21] Gesche Pospiech. *Was ist Quanteninformatik?*, pages 3–18. Springer Fachmedien Wiesbaden, Wiesbaden, 2021.

[Pow07] Michael JD Powell. A view of algorithms for optimization without derivatives. *Mathematics Today-Bulletin of the Institute of Mathematics and its Applications*, 43(5):170–174, 2007.

[Pre18] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, August 2018.

[pyt] Welcome to python.org. https://www.python.org/. (Accessed on 04/14/2023).

[qisa] https://qiskit.org/documentation/. https://qiskit.org/documentation/. (Accessed on 04/14/2023).

[Qisb] Qiskit. https://qiskit.org/. (Accessed on 08/13/2023).

[qisc] qiskit_nature.testing.random - qiskit nature 0.7.0. https://qiskit.org/ecosystem/nature/_modules/qiskit_nature/testing/random.html. (Accessed on 11/19/2023).

[QKBW23] Nils Quetschlich, Vincent Koch, Lukas Burgholzer, and Robert Wille. A hybrid classical quantum computing approach to the satellite mission planning problem, 2023.

[Rasa] Raspberry pi 4 model b specifications – raspberry pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. (Accessed on 08/13/2023).

[rasb] Rasqberry | the rasqberry project: Exploring quantum computing and qiskit with a raspberry pi and a 3d printer. https://rasqberry.org/. (Accessed on 04/14/2023).

[RSGC21] Gokul Subramanian Ravi, Kaitlin N. Smith, Pranav Gokhale, and Frederic T. Chong. Quantum computing in the cloud: Analyzing job and machine characteristics. In *2021 IEEE International Symposium on Workload Characterization (IISWC)*, pages 39–50, 2021.

[Sch35] E. Schrödinger. Die gegenwärtige Situation in der Quantenmechanik. *Naturwissenschaften*, 23(48):807–812, Nov 1935.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, oct 1997.

[Sim] Simulators — qiskit 0.43.1 dokumentation. https://qiskit.org/documentation/locale/de_DE/tutorials/simulators/1_aer_provider.html. (Accessed on 08/13/2023).

[SLM+21] Samuel A Stein, Ryan L'Abbate, Wenrui Mu, Yue Liu, Betis Baheri, Ying Mao, Guan Qiang, Ang Li, and Bo Fang. A hybrid system for learning classical data in quantum states. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 1–7, 2021.

[SLS] Slsqp - qiskit 0.45.0 documentation. https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.SLSQP.html#qiskit.algorithms.optimizers.SLSQP. (Accessed on 11/19/2023).

[SMM23] Harshdeep Singh, Sonjoy Majumder, and Sabyashachi Mishra. Benchmarking of different optimizers in the variational quantum algorithms for applications in quantum chemistry. *The Journal of Chemical Physics*, 159(4), jul 2023.

[SMS23] Pooja Srivastava, Anushtup Mishra, and Yogesh K. Srivastava. *From Quantum Mechanics to Quantum Computing*, pages 15–30. Springer Nature Singapore, Singapore, 2023.

[Soc] Socket programming howto — python 3.11.4 documentation. `https://docs.python.org/3/howto/sockets.html`. (Accessed on 08/13/2023).

[Spa98] James C. Spall. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins Apl Technical Digest*, 19(4):482–492, 1998.

[SPS] Spsa - qiskit 0.45.0 documentation. `https://qiskit.org/documentation/stubs/qiskit.algorithms.optimizers.SPSA.html#qiskit.algorithms.optimizers.SPSA`. (Accessed on 11/19/2023).

[Str16] Gilbert Strang. *Introduction to linear algebra*. Wellesley-Cambridge Press, Wellesley, fifth edition edition, 2016.

[SWA21] Waheeda Saib, Petros Wallden, and Ismail Akhalwaya. The effect of noise on the performance of variational algorithms for quantum chemistry. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 42–53, 2021.

[TBG17] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. Error mitigation for short-depth quantum circuits. *Phys. Rev. Lett.*, 119:180509, Nov 2017.

[TCC+22] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. The variational quantum eigensolver: A review of methods and best practices. *Physics Reports*, 986:1–128, 2022. The Variational Quantum Eigensolver: a review of methods and best practices.

[tcp] tcp-latency · pypi. `https://pypi.org/project/tcp-latency/`. (Accessed on 08/30/2023).

[Tra] Transpiler (qiskit.transpiler) - qiskit 0.44.0 documentation. `https://qiskit.org/documentation/apidoc/transpiler.html`. (Accessed on 08/13/2023).

[TSR+21] Yevhenii Trochun, Sergii Stirenko, Oleksandr Rokovyi, Oleg Alienin, Evgen Pavlov, and Yuri Gordienko. Hybrid classic-quantum neural networks for image classification. In *2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing*

Systems: Technology and Applications (IDAACS), volume 2, pages 968–972, 2021.

[UFN+22] Paramita Basak Upama, Md Jobair Hossain Faruk, Mohammad Nazim, Mohammad Masum, Hossain Shahriar, Gias Uddin, Shabir Barzanjeh, Sheikh Iqbal Ahamed, and Akond Rahman. Evolution of quantum computing: A systematic survey on the use of quantum computing tools. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 520–529, 2022.

[VROVR+23] Sebastián V. Romero, Eneko Osaba, Esther Villar-Rodriguez, Izaskun Oregi, and Yue Ban. Hybrid approach for solving real-world bin packing problem instances using quantum annealers. *Scientific Reports*, 13(1):11777, Jul 2023.

[Wei09] Stefan Weigert. No-cloning theorem. In *Compendium of quantum physics*, pages 404–405. Springer, 2009.

[WGS22] Junchao Wang, Guoping Guo, and Zheng Shan. Sok: Benchmarking the performance of a quantum computer. *Entropy*, 24(10), 2022.

[Wil11] Colin P. Williams. *Introduction*, pages 3–49. Springer London, London, 2011.

[Wit14] Peter Wittek. 4 - quantum computing. In Peter Wittek, editor, *Quantum Machine Learning*, pages 41–53. Academic Press, Boston, 2014.

[WL21] Jonathan Wurtz and Peter Love. Maxcut quantum approximate optimization algorithm performance guarantees for p>1. *Phys. Rev. A*, 103:042612, Apr 2021.

[WWJ+20] Madita Willsch, Dennis Willsch, Fengping Jin, Hans De Raedt, and Kristel Michielsen. Benchmarking the quantum approximate optimization algorithm. *Quantum Information Processing*, 19(7):197, Jun 2020.

[WZ82] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, Oct 1982.

[YM08] Noson S. Yanofsky and Mirco A. Mannucci. *Introduction*, page 1–6. Cambridge University Press, 2008.

[YZJ23] Zebo Yang, Maede Zolanvari, and Raj Jain. A survey of important issues in quantum computing and communications. *IEEE Communications Surveys & Tutorials*, 25(2):1059–1094, 2023.

[ZBLN97] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, dec 1997.

[ZuRJ+22] Sultan M. Zangi, Atta ur Rahman, Zhao-Xo Ji, Hazrat Ali, and Huan-Guo Zhang. Decoherence effects in a three-level system under gaussian process. *Symmetry*, 14(12), 2022.

[Ștl21] Gheorghe M. Ștefan. Let's consider moore's law in its entirety. In *2021 International Semiconductor Conference (CAS)*, pages 3–10, 2021.