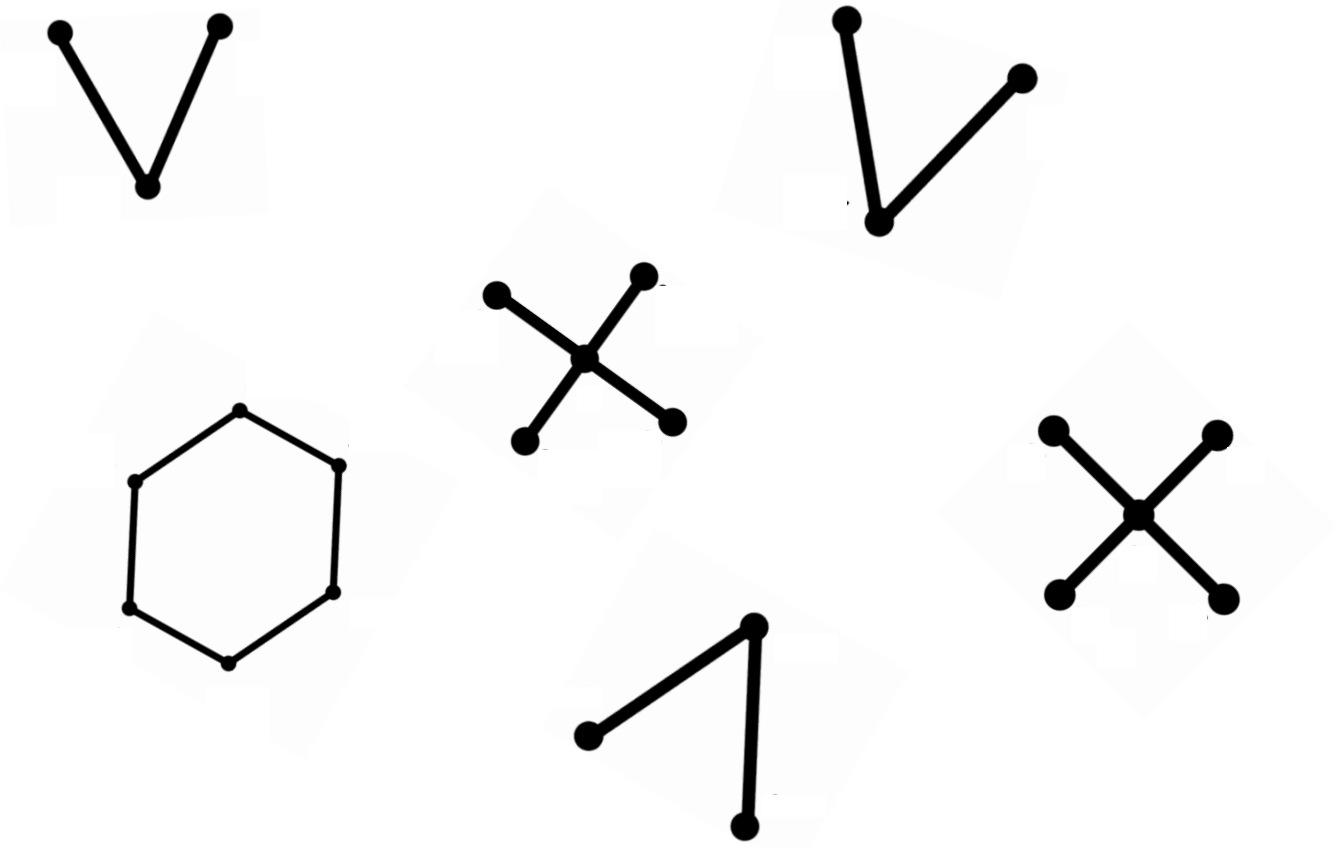
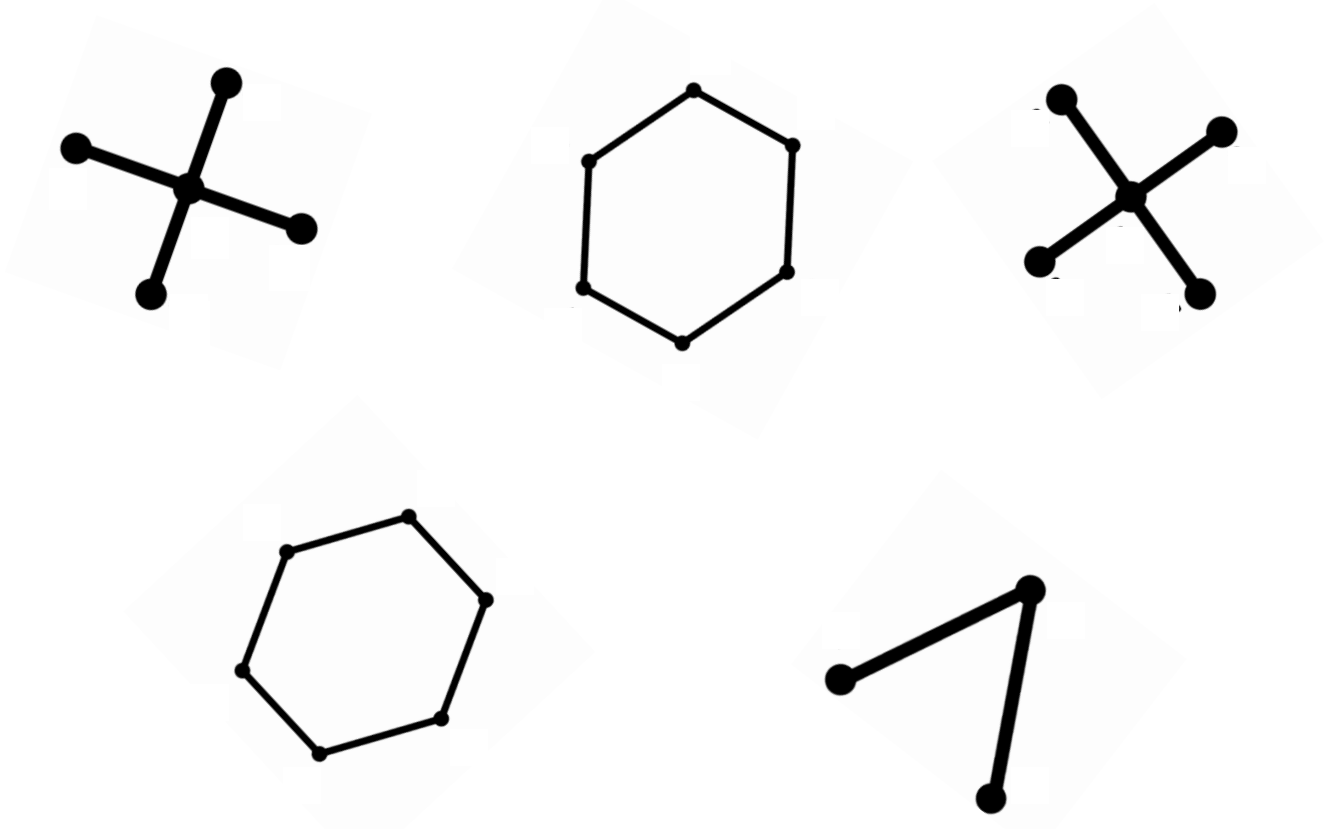


Determining Minimal SWAP Operations for the Qubit-Mapping Problem using Quantum Information Theory

Sacha Szkudlarek



Delft University of Technology

MSc Thesis
Applied Physics

Determining Minimal SWAP Operations for the Qubit-Mapping Problem using Quantum Information Theory

Author:
Sacha Szkudlarek

Supervisors:
Sebastian Feld
Matt Steinberg
Medina Bandic

June 29, 2023

Abstract

This thesis presents a novel formulation to study the qubit-mapping problem (QMP). The presented formulation redefines the problem in terms of density matrices which represent the quantum algorithm and the underlying architecture—allowing the implementation of techniques from quantum information theory to establish a bounded metric space for comparing these density matrices. The main contribution of this thesis is implementing this formulation in an algorithm to determine the minimal bound on the required number of SWAP operations for a pairing of a quantum algorithm to an underlying device where the initial mapping has been provided. Benchmarks have shown a clear dependence on the β-value. Emphasising the need for future investigations of this dependence to enhance the algorithm's effectiveness for more extensive algorithms and architectures. While it is essential to acknowledge that the approach may not currently rival the state of the art.

Contents

List of Figures

1 Introduction

The promise of quantum computers has seen a surge of interest in recent years as it promises to solve many problems intractable for any current classical computer. With quantum computing progress can be made in improving simulations, optimizations, machine learning and cryptography [1].

The fundamental building block of a quantum computer is the qubit, the counterpart to the classical bit. The encoding, storing and manipulation of the current generation of qubits is prone to errors. Moreover, due to the limited number of qubits available in current quantum architectures, it is impossible to implement quantum error correction (QEC) protocols. For these reasons, we are still far from realizing large-scale quantum computers and the exponential speed-up they promise.

Currently, we are limited to architectures that control a few noisy qubits with limited two-qubit gate connectivity. These architectures are called Noisy Intermediate-Scale Quantum (NISQ) devices and comprise of tens of noisy qubits [2]. In addition to the limited two-qubit gate connectivity, NISQ devices have a restricted gate set and shallow circuit depth due to noise [3]. Examples of these architectures are superconducting qubits [4, 5, 6, 7, 8], trapped-ion qubits [9, 10, 11, 12], CMOS silicon spin qubits [13] and photonic qubits [14, 15]. Some of these architectures are accessible through a cloud interface such as IBM's quantum experience [16] and QuTech's quantum inspire [17].

Quantum algorithms have been introduced to utilise these quantum computers. A quantum algorithm describes a procedure that utilises the unique properties of quantum mechanics to solve specific problems more efficiently than classical algorithms. A circuit often describes a quantum algorithm, indicating which operations must be performed on specific qubits to execute the algorithm. The operations in a quantum circuit are often referred to as gates. Since the introduction of quantum algorithms, which have a proven benefit over any classical counterpart, there has been a keen interest in their potential [18]. Peter Shor presented an algorithm that allows for the factorization of natural numbers and computation of discrete logarithms [19], promising an exponential speed-up over the classical counterparts. However, quantum algorithms are hardware-agnostic. They do not take the restriction of NISQ devices into account. Due to the varying limitations of different NISQ devices, it is impossible to run complex quantum algorithms directly. These algorithms would allow us to deal with computationally complex problems. To execute these algorithms, they must be modified to adhere to the limitations of NISQ devices. Therefore, we would initially want to understand how we can modify these algorithms to determine what is possible on the current generation of devices.

To this end, the problem requiring further investigation is that of quantum compilation. Quantum compilation bridges the gap between quantum algorithms and their physical implementation. It transforms quantum algorithms into a sequence of elementary quantum operations that can be executed on the physical quantum computer (currently NISQ devices). The goal is to optimise the quantum circuit's performance and efficiency. The circuit is a description of the quantum algorithm. We must consider the underlying hardware's limitations and constraints during the compilation. Several factors are considered including but not limited to gate restructuring, qubit mapping, reduction of the overall execution time, and mitigation of errors [20].

The specific part of quantum compilation which this thesis will focus on is the qubit-mapping problem (QMP). The QMP refers to assigning the qubits of the quantum algorithm (logical qubits) to those present in existing quantum architectures (physical qubits). This is referred to as the initial mapping process. If the logical qubits have been assigned to the physical qubits such that not all operations specified by the algorithm can be performed, it will be required to change the mapping of the logical qubits on the device. Often times moving a logical qubit to a different physical qubit is done by performing a SWAP operation. Though there are other options to move a qubit depending on the hardware this thesis will only consider SWAP operations. All in all, it is necessary to determine in which way the qubits utilised by the algorithm can initially be mapped to the underlying hardware and in what manner they could be exchanged to ensure that the underlying physical architecture can perform the largest number of operations. This process will allow the quantum algorithms to be executed as efficiently as possible under the constraints of the underlying physical architecture.

In recent years, extensive research has been conducted on the QMP. Initially, brute-force algorithms were employed to address the problem, which proved effective for simple systems. However, quantum architectures have grown and quantum algorithms have demanded more qubits and operations. Heuristic algorithms have emerged as potential solutions to deal with the additional complexities. Another approach involves utilizing a graphical representation, allowing the problem to be reframed using concepts from graph theory. This enables the utilization of existing solutions or heuristics developed for graph-theoretic problems that are equivalent to the QMP.

This thesis aims to go beyond just restating the problem in graph theory. Instead, a new perspective redefining the graphical objects (CG and IG) as quantum mechanical objects represented by density matrices

and using principles from quantum information theory to establish a bounded metric space for comparison of these objects is introduced. This new formulation will allow for the exploration of new techniques to solve the QMP. The main contribution of this thesis is implementing this formulation in an algorithm to determine the minimal bound on the required number of SWAP operations for a CG/IG-pairing where the initial mapping has been provided.

The thesis will appeal to individuals actively involved in the QMP and newcomers aspiring to understand quantum computation better. Specifically, the intricate complexities that arise when executing quantum algorithms on the current generation of devices.

The thesis is structured in the following manner. Initially, an introduction to quantum computation will be presented in Sec. 2. This introduction starts with a comprehensive exploration of the fundamentals of quantum computation. Discussing qubits, their representation, and associated limitations (Sec. 2.1.1). Followed by an examination of quantum operations (Sec. 2.1.2). Lastly, the basics of quantum circuits will be presented (Sec. 2.1.3). Once the reader is familiar with the fundamentals of quantum computation, quantum algorithms will be introduced in detail (Sec. 2.2). After which the implication of the NISQ era will be presented (Sec. 2.3). To conclude Ch. 2 the QMP (Sec. 2.4) will be discussed.

After presenting the necessary foundational information in the previous chapter, Ch. 3 will comprehensively analyse the existing literature on the QMP. This review aims to offer an understanding of the advances made in this field, thereby providing context for the goals of this thesis.

Upon providing the required background to understand the goals of this thesis the methods which will be utilised will be presented in Ch. 4. This chapter will discuss the mathematical prerequisites (Sec. 4.1), the tools from quantum information theory (Sec. 4.2), the additional adaptations to construct the theoretical framework for our contribution (Sec. 4.3) and finally the proposed algorithmic adaptation will be presented (Sec. 4.4).

Once the methods have been presented they will be evaluated in Ch. 5. This chapter contains several worked-out 'toy-model' examples to give an insight into how the proposed algorithm works (Sec. 5.1) and the algorithm will be evaluated on several benchmarks 5.2. The results will then be discussed in Sec. 6. The conclusion (Ch. 7) and future work (Sec. 8) will be the final chapters of the thesis. They will provide an overview of what has been achieved and how it could continue in the future.

2 Quantum Computation

This section will provide the required information on quantum computation relevant to this thesis's scope. It will first give an introduction to the fundamentals of quantum computation. Followed by examples of quantum algorithms which are applications of quantum computation. Moreover, the NISQ-era will be discussed, which refers to the current generation of quantum computer architectures. Lastly, the QMP is briefly introduced. If the reader is already familiar with these topics they are referred to Sec. 3 which provides a literature review of the QMP.

2.1 Fundamentals of quantum computation

In this section, the fundamentals of quantum computation will be discussed for readers who are not yet familiar with the topic. If the fundamentals are known to the reader, they are referred to Sec.2.2, in which quantum algorithms are discussed. For a detailed description of quantum computing the reader is referred to the textbook by Nielsen and Chuang [21]. To start, the qubit forms the basic unit of information for quantum computing. A qubit is a two-level quantum system we may utilise to encode and store quantum information. Combined with quantum operations (quantum gates) that can manipulate the qubit's state, they form the basis of quantum computation and quantum information theory. The representation of the initialization, measurement and manipulation of specific qubits is often denoted with a quantum circuit. Quantum circuits are a representation of quantum algorithms. They show the operations performed on a set of qubits to execute the algorithm successfully. These concepts will be discussed in the remainder of this section.

2.1.1 Qubits, representation and limitations

The fundamental building block of a quantum computer is the qubit. A qubit represents a quantum state and is the counterpart to the classical bit. These quantum states are simple two-level quantum systems that display quantum mechanics' eccentricity. In principle, a quantum system with multiple levels could also form a qubit as long as the system can exist in any quantum superposition of two physically distinguishable quantum states. Examples include the spin state of an electron or the polarization of a single photon. The most promising candidate prototypes are constructed from superconducting qubits [4, 5, 6, 7, 8], trapped-ion qubits [9, 10, 11, 12], CMOS silicon spin qubits [13], photonic qubits [14, 15]. Considering these possibilities, it becomes abundantly clear that as it currently stands, there is not yet a consensus on which technology could best be used to create quantum computers with a larger number of qubits.

Moreover, a classical bit is limited to a value of 0 or 1. The qubit on the other hand can be in a coherent superposition of two basis states typically denoted as $|0\rangle$ and $|1\rangle$. A superposition is a linear combination of the two basis states and is denoted as $\alpha|0\rangle + \beta|1\rangle$, where α and β are normalised complex numbers. It is known that $|\alpha|^2$ and $|\beta|^2$ represent the probability amplitudes of the qubit being in the state $|0\rangle$ or $|1\rangle$ upon performing a measurement operation. Moreover, the complex probability is normalised, meaning $|\alpha|^2 + |\beta|^2 = 1$.

These complex probability amplitudes can be represented by state vectors which live in a two-dimensional Hilbert space, as such a qubit state may also be represented as a vector with the states $|0\rangle$ and $|1\rangle$ as a basis,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\begin{bmatrix}1\\0\end{bmatrix} + \beta\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha\\\beta\end{bmatrix} \tag{1}$$

Here, $|\psi\rangle$ is a two-dimensional, complex and unitary vector, a natural representation of the state of a qubit can be given by displaying the state vector on the Bloch sphere. This representation arises when considering the coefficients α and β in the form $\alpha = \cos\left(\frac{\theta}{2}\right)$ and $\beta = e^{i\phi}\sin\left(\frac{\theta}{2}\right)$, here θ is the colatitude with respect to the z-axis and ϕ is the longitude with respect to the x-axis, this can clearly be seen in Fig. 2a. As mentioned, the state $|\psi\rangle$ can be displayed by a vector on the Bloch sphere named after Felix Bloch [22]. Note that the value in which a classical bit can be in is limited to 0 or 1, which is analogue to the poles of the Bloch sphere. Furthermore, the qubit can be measured with respect to a predetermined basis on the Bloch sphere such as the previously mentioned $|0\rangle$ and $|1\rangle$ basis and before measurement the qubit can be at any point on the surface of the Bloch sphere.

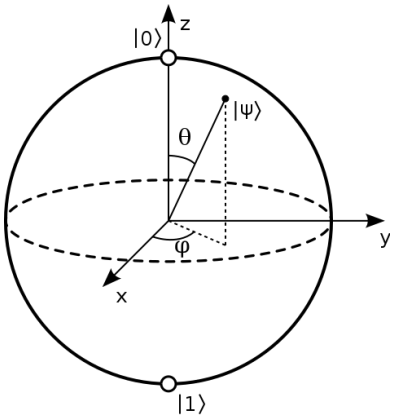


Figure 1: Bloch sphere representing the state vector of the state $|\psi\rangle$ in an intuitive manner.

(a) Fig. obtained from: Wikipedia

Thus far the discussion of the qubit has been limited to the case in which only a single qubit is present. It is however possible to have a combined state of multiple qubits also known as a quantum register. Qubits in a quantum register can exhibit quantum entanglement, a nonlocal property of the set of qubits. It occurs when it is not possible to describe the state of a particle independently of the states of the other particles. The entanglement property is a cornerstone of the power of quantum computation and is exploited in the design of quantum algorithms. The state of a n-qubit quantum register can be denoted as $|\psi\rangle = \alpha_0 |0..0\rangle + \alpha_1 |0..1\rangle + ... + \alpha_{n-1} |1..1\rangle$. A state vector can represent this multi-qubit state in a 2^n dimensional Hilbert space.

One of the current challenges in the development of qubits is the issue of decoherence. This refers to losing a coherent quantum state or a coherent superposition in the computational basis. This occurs due to interactions with the environment, such as thermal noise, magnetic field fluctuations and interactions with nearby atoms or molecules. These interactions limit the time a qubit can be used to perform a quantum computation or store a quantum state. The current generation of qubits is susceptible to decoherence, though the state-of-the-art coherence time of 5500 seconds has been shown by Wang et al. [23] for a $^{171}Yb^{+}$ ion-qubit, there is still much work to be done. Working toward functional quantum computation and application of quantum memories it will be of fundamental importance to ensure stable and coherent qubit registers.

Another challenge is scaling the number of qubits present in the register. The goal in creating quantum computers is to have a quantum register with a large number of qubits so that it will be possible to use the principles of quantum error correction [24]. The idea of quantum error correction is to create a highly entangled state which can protect the quantum information as only parts of the entangled state will interact with the environment at a particular time thus preserving the information present. However, creating such a highly entangled state requires significant overhead in the number of qubits. Furthermore, as the number of qubits is expanded the errors present in the system will propagate and will thus influence the result of a complex computation on a large quantum register will be significant. The result is significantly influenced even when individual operations have a low error rate. Scaling to a larger number of qubits is a complex engineering task which has garnered much attention in recent year [25]. Still, there is yet to be a system which allows for fault-tolerant quantum computation with the help of quantum error correction[26].

2.1.2 Quantum Operations

Qubits are building blocks of quantum computation, but to allow for information processing, additional tools are required, namely quantum operations or quantum gates. These terms are equivalent and may be used interchangeably throughout this thesis. As mentioned in the previous section, an example of a quantum operation is a measurement, which collapses the state of a qubit, $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, to one of the basis states, $|0\rangle$ or $|1\rangle$, with a probability equal to $|\alpha|^2$ or $|\beta|^2$ respectively.

However, it is also possible to alter the qubit's state by applying quantum operations which may act on a single or multiple qubits. This may be done with unitary or non-unitary operations transforming the qubit's state. Non-unitary operations are not trace-preserving and thus do not preserve the inner product of the quantum state. As such these operations are used to model noise and decoherence. Moreover, these operations are not reversible and result in the loss of quantum information. For this reason in the remainder of this thesis, we will be considering unitary operations which are trace-preserving and reversible.

Unitary matrices can represent unitary operations acting on the state vector introduced above. A different

manner of interpreting these quantum operations is as a rotation of the qubit states on the Bloch sphere. Furthermore, it is possible to utilise operations to change the perspective from which the observer is viewing the qubit states on the Bloch sphere.

There are several examples of quantum operations which can be performed on current quantum architectures, such as the Pauli gates (X, Y, Z, $\mathbb{I}$), the Hadamard gate (H), the Controlled-Not (CNOT) gate and the SWAP gate. For a better insight into how these gates manipulate a quantum state, examples of the Hadamard- and the CNOT gate will be discussed in more detail. First, let us note that the Hadamard gate is a single qubit gate and is thus represented by a 2×2 matrix whereas the CNOT gate is a two-qubit gate represented by a $2^2 \times 2^2$ matrix. A quantum operation on n qubits may represent a $2^n \times 2^n$ matrix. Below the matrix representation of these operations is shown:

$$\text{Hadamard (H)} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}; \text{CNOT} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix} \tag{2}$$

Considering the states $|0\rangle$ and $|1\rangle$ from the previous section we see that the Hadamard gate maps the basis state $|0\rangle$ to $(|0\rangle + |1\rangle)/\sqrt{2}$ and $|1\rangle$ to $(|0\rangle - |1\rangle)/\sqrt{2}$. In the case of the CNOT gate, we see that it is applied on two qubits, and the action of CNOT_{pq} will operate on qubit q based on the state of qubit p. If qubit p is in state $|0\rangle$, it does not alter the state of qubit q. When p is in the state $|1\rangle$, it will perform a NOT operation on qubit q. This takes qubit q to the state $|0\rangle$ if it was in the state $|1\rangle$ and vice versa. Moreover, we notice that it works on two qubits. As such we may consider the product state of two qubits to be the tensor product ($\otimes$) of the state vectors of the individual qubits. This may be represented as

$$|p\rangle \otimes |q\rangle = |pq\rangle = \begin{bmatrix}\alpha_p\\ \beta_p\end{bmatrix} \otimes \begin{bmatrix}\alpha_q\\ \beta_q\end{bmatrix} = \begin{bmatrix}\alpha_p\alpha_q\\ \alpha_p\beta_q\\ \beta_p\alpha_q\\ \beta_p\beta_q\end{bmatrix} \tag{3}$$

Lastly, in the scope of this thesis, it will be important to discuss the SWAP operation briefly. The SWAP operation allows for the exchange of the state of a certain qubit to another. The SWAP operation can be decomposed into three consecutive CNOT operations where the middle operation is reversed. This can been represented mathematically in the following manner $\text{SWAP}_{pq} = \text{CNOT}_{pq} \cdot \text{CNOT}_{qp} \cdot \text{CNOT}_{pq}$. Working out the matrices of these operations we find,

$$\text{SWAP} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix} \cdot \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\end{pmatrix} \cdot \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{4}$$

2.1.3 Quantum Circuit

Quantum algorithms consist of qubits and unitary quantum operations performed on specified qubits or qubit pairs. The representation of the operations in an algorithm is often shown as blocks rather than the mathematical representation shown above. In Fig. 3 examples of the operations discussed above are shown.

Figure 3: Example of the block representation of the Hadamard (H) and the Controlled-Not (CNOT) gates.

Quantum algorithms are often represented through the visual abstraction of quantum circuits. An example of a quantum circuit is shown in Fig. 4. In this representation each line represents one qubit and the blocks on those lines represent the quantum operations performed on those qubits. One block represents a single qubit operation whereas a line vertically connecting two qubits represents a two-qubit operation.

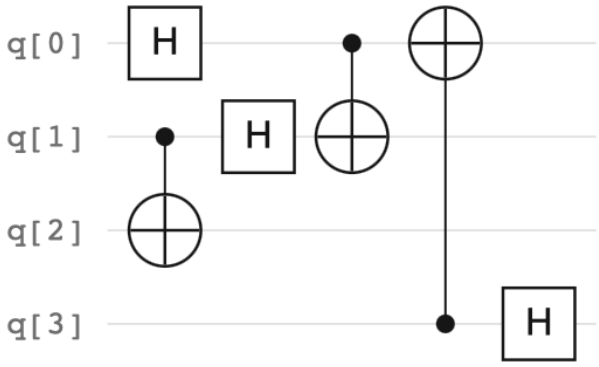


Figure 4: Example of a quantum circuit containing four qubits and several H and CNOT gates.

Though the circuit representation offers a convenient way to visualise quantum algorithms, this is not the preferred method in practical application. Instead, quantum algorithms are often written in plain-text programming languages. Examples of such languages are Scaffold [27], Quipper [28], cQASM [29] and Open QASM [30]. In both the case of the quantum circuits and the text-based programming languages, they are merely representations of the operations which need to be performed by the underlying quantum architecture.

2.2 Quantum Algorithms

Quantum computation is a rapidly growing field that has promised to solve many problems intractable for current and future classical systems. The implementation of quantum computation relies on previously discussed building blocks and has been designed to outperform their classical counterparts. These algorithms have been designed to take advantage of the unique properties of the underlying quantum architectures, such as entanglement and interference. It is important to note that the practical application of these algorithms has not been realised on the scale in which they systematically outperform their classical counterpart due to the limitation of the current quantum architecture. This will be discussed further in Sec. 2.3. Nonetheless, it is relevant to consider the quantum algorithms which have been presented to gain an understanding of the advances that quantum computers could provide. In the following sections, two of these algorithms will be briefly discussed. If the reader wants a more in-depth view of quantum algorithms, a discussion of additional algorithms is presented in the appendix A.

Shor's quantum factoring algorithm

A famous example of the promised power of quantum computing is Shor's algorithm which can determine the prime factors of an integer. Implementing ideas from quantum physics, signal processing, number theory and computer science Peter Shor presented the algorithm in 1994 [19] and promises an exponential speed-up compared to the current state-of-the-art factorisation algorithms. Moreover, it runs in polylogarithmic time, which means that the time it takes to factor an integer N is polynomial in $\log(N)$.

The speed-up results from the fact that a quantum computer has an efficient Fourier transform method, namely the Quantum Fourier Transform (QFT), which allows for efficient period finding and a quantum computer can implement modular exponentiation to create a periodic state. The algorithm functions as follows: first, a random number is generated. This is then used as the base number for modular exponentiation creating a periodic state. At that point, QFT is performed on the periodic state resulting in a frequency sample, which is processed further by classical means. This process may result in a prime factor of N. If this is not the case, the process is repeated. The algorithm terminates when the correct prime factor of N has been determined. For a more thorough explanation including pseudo-code to simulate the algorithm, the reader is referred to [31].

Grover's quantum search algorithm

A different example is Grover's quantum search algorithm which provides a quadratic speed-up for searching through large unstructured databases. Where the classical counterpart to Grover's requires $O(N)$ evaluations the algorithm proposed by Lov Grover in 1996 [32] only requires $O(\sqrt{N})$ evaluations, where N represents the size of the function's domain. The algorithm creates a uniform superposition over all present options, after which repeated destructive interference reduces the probability amplitudes of the states that are not a solution.

This is achieved by changing the amplitude of the solution and then amplifying the difference resulting in a large amplitude for the desired state. Upon measuring the state will collapse and the solution to the desired query will be found with a high probability [33]. The algorithm's applications are not limited to the search of large unstructured databases. It has been shown by Cerf et al. [34] that it can also provide a quadratic speed-up for generic constrained optimisation problems such as the graph colouring problem.

2.3 NISQ-era

The quantum speed-up they promise will require large-scale fault-tolerant quantum computers to implement most of the quantum algorithms discussed in Sec. 2.2. However, significant engineering advances will be required before entering the fault-tolerant quantum computing era. As it currently stands we are limited to architectures which allow for the control of a few noisy qubits, also known as 'Noisy Intermediate Scale Quantum' (NISQ) devices, a term coined by John Preskill [2]. Though there are some promising candidate architectures for the implementation of quantum computing such as superconducting qubits [4, 5, 6, 7, 8], trapped-ion qubits [9, 10, 11, 12], CMOS silicon spin qubits [13], photonic qubits [14, 15]. As mentioned in Sec. 2.1.1 there is no consensus on which architectures would be the best candidate for a scalable quantum computer. Apart from the limited number of qubits, NISQ devices suffer from limited qubit lifetimes, noisy operations and limited two-qubit connectivity.

The limitation of qubit lifetime refers to the fact that current qubits have short coherence times. Therefore, the operations that must be performed to run a specific quantum algorithm must be implemented within the qubit coherence time. This establishes an upper bound on the number of possible operations, limiting the circuit depth of the algorithm which NISQ devices can run effectively [3].

A further limitation of NISQ devices is the fact that the operations which are performed introduce errors. Thus the qubit state obtained after an operation has been performed will not be exactly the state which is desired. The present error rates are reported to be approximately 10^{-3} for single qubit operations and 10^{-2} for measurements and two-qubit operations [35, 36, 37]. For this reason, reducing the number of required operations to perform a quantum algorithm is an important consideration.

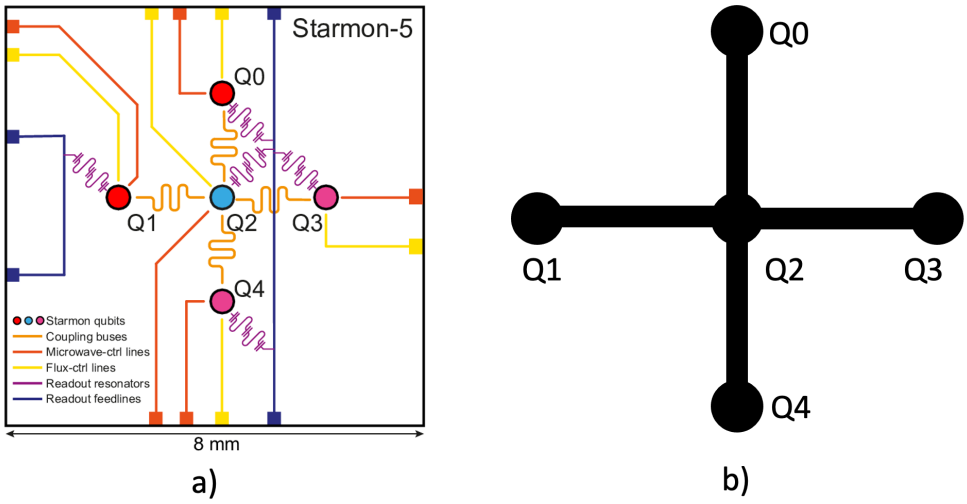


Figure 5: a) This is a description of the Starmon 5 superconducting qubit architecture [38], the orange lines indicate the connections between qubits that can perform two-qubit gates. b) Is the corresponding graphical representation, coupling graph.

Furthermore, NISQ devices often do not have all-to-all connectivity. This means that two-qubit operations can not be performed between arbitrary qubits. Often NISQ devices have a fixed topology which describes between which physical qubits two-qubit operations can be performed. The specific topology of an architecture is also referred to as the coupling graph of the architecture. An example of a physical quantum architecture and its corresponding coupling graph is shown in Fig. 5. Limited two-qubit connectivity is a principal limitation which leads to the consideration of the QMP. The QMP aims to efficiently determine how to implement quantum algorithms on current NISQ devices. This problem will be introduced in Sec. 2.4 and a more detailed description of this problem will be presented in Sec. 3.

It is important to note that algorithms such as Shor's and Grover will most likely not be able to outperform their classical counterparts using the current generation of NISQ devices. Some examples of algorithms could outperform their classical counterparts on near future NISQ devices [39]. Moreover, the experimental demonstrations of quantum key distribution based on the BB84 protocol have been realised [40].

Lastly, there are certain architectures in which two-qubit operations can only be performed in a directed manner, which entails that the topology of the device also fixes the control and target qubit. Dealing with this limitation is beyond the scope of the contribution presented in this thesis.

2.4 The Qubit-Mapping Problem

Quantum compilation is essential to execute a quantum algorithm on a NISQ device. As mentioned in the introduction, quantum compilation considers several factors to allow the successful execution of an algorithm on the underlying hardware. Some factors considered when compiling are gate restructuring, qubit mapping, reduction of the overall execution time, and mitigation of errors [20]. In this thesis, qubit mapping will be considered in detail. Though the other elements are also essential, they are beyond the scope of this thesis.

As mentioned in Sec. 2.3, currently available quantum architectures can not perform all the tasks required to ensure the successful execution of a quantum algorithm. Quantum algorithms have been designed assuming that required operations can be executed on an architecture where qubits can perform gates between arbitrary qubits. However, the current generation of NISQ devices does not meet this requirement. Therefore, it is essential to consider how they can optimally be used to perform the operations required to execute a quantum algorithm.

To utilise the current generation of NISQ devices to their full potential, it is imperative to consider how logical qubits specified by the algorithm should be mapped to the physical qubits of the underlying NISQ architecture. Taking into account the limitations of the architecture for example limited connectivity between physical qubits. This leads to the consideration of the qubit-mapping problem (QMP). The QMP was defined by Li et al. [41] as follows: 'Given an input quantum circuit and the coupling graph of the quantum device, find the **initial mapping** and the intermediate qubit **mapping transition** (by inserting SWAP operations) to satisfy all two-qubit constraints and try to minimise the number of additional gates and circuit depth in the final hardware-compliant circuit.'

It is essential to consider how an algorithm would initially be mapped to an underlying architecture and to determine the minimal number of SWAP operations required to execute it. In the scope of this thesis, we will focus on determining the minimal number of required SWAP operations. This will be done by considering theoretical methods and providing a new approach to this problem. Determining the ideal initial mapping will be beyond the scope of this thesis. The number of SWAP operations will be determined for a given initial mapping which has been provided. The initial mapping defines the assignment of logical qubits to the physical qubits of the underlying architecture. The presented approach will offer an alternative perspective on how to solve the problem compared to the heuristic and brute force searches that have been previously proposed.

It is essential to note that the QMP is NP-complete [42]. This means that a solution can be determined in polynomial time. However, there is no efficient algorithm to solve NP-complete problems in general. Therefore, it remains to be determined how the QMP could be solved in polynomial time. We hope to contribute to this goal. In recent years there has been ample research on the QMP, the literature on this topic will be discussed in depth in Sec. 3. Initially, attempts have been made to solve the problem using brute-force algorithms, which are effective when dealing with simple systems. As the quantum architecture increases in size and the quantum algorithm requires more qubits and operations, heuristic algorithms have been developed as potential solutions to the problem.

A different approach which has been presented is to utilise a graphical representation. This allows us to reframe the problem using concepts from graph theory. In turn, allowing the use of existing solutions or heuristics studied for graph-theoretic problems, equivalent to the QMP. Lastly, the influence of the noisy characteristics of NISQ devices has also been studied.

An example of the QMP will be discussed to give the reader a clear idea of what needs to be considered when aiming to solve the QMP. It is important to note that we simplify the problem as we do not consider the order in which the gates must be implemented in the circuit.

First, the aim is to establish the coupling and interaction graph of the NISQ architecture and the quantum algorithm respectfully, shown in Fig. 6. In this example, the interaction graph is constructed based on the circuit from Fig. 4. The construction of the coupling graph is the same as in Fig. 5.

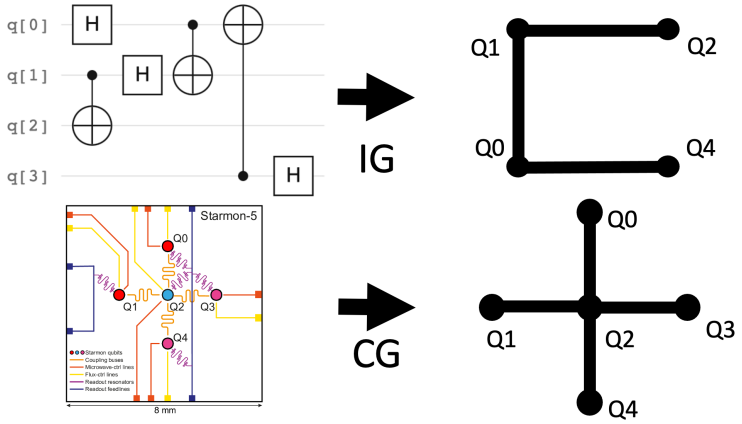


Figure 6: Example of how the coupling and interaction graphs are determined based on the NISQ architecture and the quantum algorithm respectively.

Once the CG and IG have been constructed, it will be possible to map the logical qubits to the physical qubits. The mapping for this example is depicted in Fig. 7 where the colours of the vertices indicate how logical qubits are mapped to the physical qubits. The qubits in the example have been mapped as follows:

$$Q_{0,\text{logical}} \mapsto Q_{2,\text{physical}} \tag{5}$$
$$Q_{1,\text{logical}} \mapsto Q_{0,\text{physical}} \tag{6}$$
$$Q_{2,\text{logical}} \mapsto Q_{4,\text{physical}} \tag{7}$$
$$Q_{3,\text{logical}} \mapsto Q_{3,\text{physical}} \tag{8}$$

The example presented in Fig. 7 is simple and the initial mapping is not difficult to determine. We only need to map a qubit from the IG with two edges to the central qubit ($Q_{2,\text{physical}}$)to obtain the ideal initial mapping. This can be confirmed by considering the number of operations we can execute based on the initial placement. If we map a logical qubit with only one edge to $Q_{2,\text{physical}}$, only one operation can be performed.

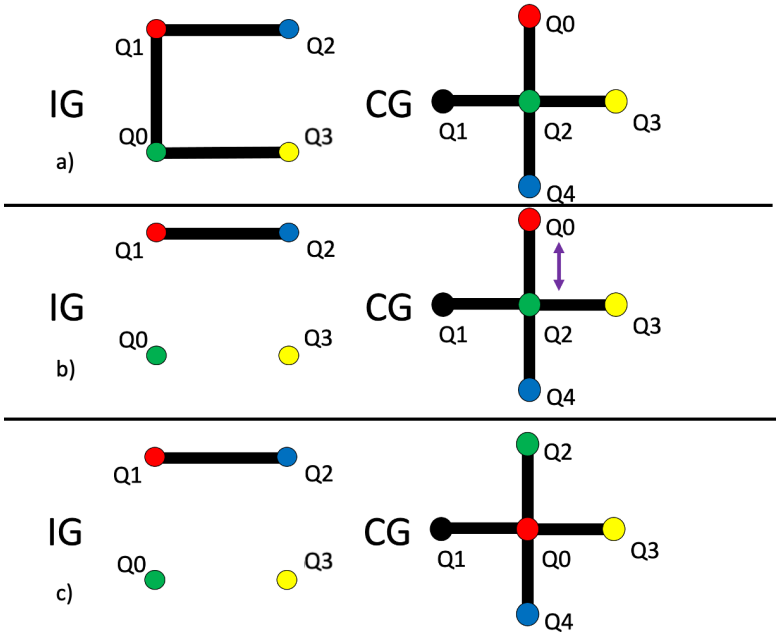


Figure 7: Example of an initial mapping and the SWAP operation, which is performed to allow all gates of the algorithm to be executed.

Still, not all operations the algorithm specifies can be performed after the initial mapping. Therefore SWAP operation will be required to modify the mapping. In Fig. 7 it can be seen that we still need to operate between

the blue and red qubit in the interaction graph. For that reason, the mapping of logical qubits to physical qubits is changed to:

$$Q_{0,\text{logical}} \mapsto Q_{0,\text{physical}} \tag{9}$$
$$Q_{1,\text{logical}} \mapsto Q_{2,\text{physical}} \tag{10}$$
$$Q_{2,\text{logical}} \mapsto Q_{4,\text{physical}} \tag{11}$$
$$Q_{3,\text{logical}} \mapsto Q_{3,\text{physical}} \tag{12}$$

In Fig. 7 the naming of the vertices in the CG and IG are not changed, but their initial mapping is. It would also be possible to change the names of the vertices in the graph to represent the change in mapping, shown in Fig. 8. In that case, the assignment of logical qubits to physical qubits still adheres to Eq. 5-8. Changing the mapping or the naming of the qubits will not influence the required SWAP operations. Therefore, we are free to choose and when introducing and evaluating the proposed approach we chose to change the naming of the qubits.

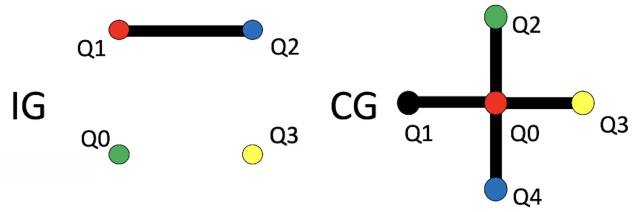


Figure 8: Example showing the difference between changing the initial mapping and naming qubits.

This thesis aims to contribute by reformulating the QMP using tools from quantum information and complex network theory. The primary objective is establishing a lower theoretical bound on the number of SWAP operations required to execute a quantum algorithm for a specified initial mapping.

Having provided an example of how to solve the QMP in Sec. 3 overview of the relevant literature concerning the QMP will be presented.

3 Literature review: Qubit-Mapping Problem

This section will provide an overview of the current literature for the reader to gain insight into the current standing of the Qubit-Mapping Problem (QMP). To get an overview of advances made in recent years, this section will summarise relevant publications on the QMP to give the reader an insight into the history of the problem and the proposed solutions.

3.1 The Qubit-Mapping Problem

The quantum mapping problem has been considered for systems with one-dimensional architectures, such as the Linear Nearest Neighbour (LNN) architecture. In this physical architecture, the qubits are arranged in a line. Examples of LNN architectures are trapped ions [43] and liquid NMR [44]. In the work of Shafaei et al. [45] they aim to explore an efficient realisation of a quantum algorithm on a 1D physical architecture. They focused on reducing the interaction distance of the qubit pairs involved in two-qubit operations, effectively achieving nearest-neighbour compliance. The interaction distance is the distance between the qubits on the line and equals zero when the qubits are nearest neighbours. The total interaction distance is the sum of the interaction distances of all qubit pairs involved in two-qubit operations. The problem has been restated as the Minimum Linear Arrangement (MinLA) problem in graph theory [46] to reduce total interaction distance. A trivial mapping of logical qubits onto the underlying 1D physical architecture often does not yield the minimal total interaction distance. Therefore, it is relevant to consider determining a mapping that reduces the total interaction distance. Moreover, even after this mapping has been obtained, some two-qubit operations from the quantum algorithm may still be impossible to execute since the qubits involved in these operations are not nearest neighbours at that point. Additional SWAP operations are added to make these qubits nearest neighbours.

The proposed approach considers the quantum algorithm an interaction graph (IG) in which the vertices represent the qubits and edges in the graph refer to two-qubit operations. This approach is adopted because minimising the interaction distance when mapping the interaction graph to the coupling graph (CG) of physical architecture is equivalent to solving the MinLA problem. Though the MinLA problem is NP-hard in general [46] there are polynomial time algorithms which can compute exact solutions for certain graphs. Furthermore, approximation algorithms have been proposed to solve the MinLA [47]. Based on the maximum number of two-qubit operations performed on a specific qubit in the quantum algorithm, it is possible to indicate the required number of SWAP operations. If a logical qubit involves more than three two-qubit operations, at least one SWAP operation will be required. Since in a 1D architecture, a qubit only has two nearest neighbours, a SWAP will be required for the third gate to be performed.

The proposed method partitions the circuit into multiple subcircuits. Within each subcircuit, the MinLA procedure is applied to determine the mapping that minimises the total interaction distance of that specific subcircuit. Upon completing this step, SWAP operations are inserted to enable the execution of all two-qubit operations within the subcircuit. Once all two-qubit operations within the subcircuit can be performed, additional SWAP operations are implemented to adjust the mapping and align it with the mapping requirements of the subsequent subcircuit. A look-ahead scheme is introduced to optimise this process. This scheme determines subcircuit boundaries and minimises the number of SWAP operations.

The proposed method has been evaluated by comparing the experimental results based on the benchmarks from [47] to work conducted by Saaedi et al. [48]. The comparison shows that the proposed method achieved an average reduction of 28% in the number of SWAP operations required.

The approach mentioned above was explicitly designed to realise quantum algorithms on 1D architectures. Implementing SWAP operations allowed the mapping to be adjusted to comply with the quantum algorithm. The introduction of SWAP operations significantly affects the overall cost of the resulting circuit. For that reason, extensive research has been conducted to determine efficient mapping methods and SWAP operation insertions for present physical architectures [45, 48, 49, 50, 51, 52].

The required SWAP operations in multidimensional architectures have been addressed as the field progressed. An example of a multidimensional architecture is a physical architecture with qubits arranged in a two-dimensional structure. With the emergence of multidimensional quantum architectures [53, 54, 55, 56, 57, 58], it was crucial to develop methods to construct nearest-neighbour-compliant quantum circuits for these architectures. These circuits will be referred to as multidimensional quantum circuits. Previous work on this topic utilised a heuristic approach [59, 60], but an exact scheme for constructing multidimensional quantum circuits still needed to be introduced. Lye et al. [61] were the first to propose an exact scheme for nearest neighbour optimisation in multidimensional quantum circuits.

The introduction of multidimensional quantum architectures gave rise to additional complexities in determining the minimum number of SWAP operations required compared to one-dimensional architectures. Lye et al. observed that achieving nearest-neighbour compliance in multidimensional architectures involves tackling complex sub-problems. Their proposed solution comprises three steps, each evaluating a complex sub-problem. The proposed method can determine the exact result for circuits of up to six qubits. These exact results can, in turn, be compared to the results of the heuristic approaches [59, 60]

The three complex sub-problems which are evaluated are the following. First, they determine the precise configuration of the qubits of the underlying physical architecture. We refer to this configuration as the coupling graph. Secondly, they provide a cost function for the minimal number of SWAP operations required to exchange the position of any pair of qubits in the coupling graph, and lastly, they determine the minimal number of SWAP insertions based on the configuration and the cost function. Having established the procedure, they turn to the implementation in which they aim to determine the minimal number of SWAP operations while also keeping the number of garbage (unused) qubit positions minimal.

To establish the required SWAP operations to exchange the position of any pair of qubits in the coupling graph, they formulate the problem as an adjacent transposition graph [50]. An adjacent transposition graph represents the various permutation of the qubits in a coupling graph which can be realised by implementing SWAP operations between nearest neighbours. Each node represents a distinct mapping of logical qubits to the coupling graph, while the edges indicate which mappings may be achieved by implementing nearest neighbour SWAP operations. This graph provides a representation of possible mappings and indicates the transition between the mappings. In doing so, the minimal SWAP operations can be considered by determining a minimal path from the starting node to the node representing the desired mapping. Determining the minimal path can be formulated as a Pseudo-Boolean Optimisation problem (PBO) by assigning Boolean variables to nodes and edges, indicating whether they are part of the optimal path. This formulation is then passed to a state-of-the-art PBO solver [62] to obtain the desired minimal path.

Lastly, they consider all possible permutations of qubit positions before each gate specified by the circuit and the costs to create these particular permutations. To solve this, they again employ a PBO formulation which is solved to determine the minimal number of SWAP operations required. Overall they have provided an exact solution to the minimal number of SWAP operations to be inserted to make a generic quantum circuit nearest neighbour-compliant.

While Lye et al. [61] only made the circuits nearest neighbour-compliant, quantum computing became available to the broader public the following year when IBM launched the IBM Q project. The public availability of quantum computing sparked the question of how to most efficiently utilise their architectures to provide the desired functionality when a particular quantum circuit was to be executed. To that end, Zulehner et al. [63] proposed an efficient and automatic mapper taking into account the constraints of the architecture while, in addition, attempting to minimise the additional quantum gates. The work differs from that presented by Lye et al. [61] since Zulehner et al. focus on mapping a quantum algorithm to a specific architecture, namely the IBM QX architectures. Their work considers the limitations imposed by the underlying architecture's physical constraints. Furthermore, the quantum algorithm must be decomposed into elementary operations which can be performed on the architecture. This decomposition is required because quantum operations outlined by the algorithm often cannot be performed directly.

An initial mapping of logical qubits to the underlying IBM QX architecture will likely only meet some physical constraints. Therefore, it will be necessary to perform a SWAP operation to move the logical qubit to the desired location on the architecture to perform the next operation. It is relevant to minimise these SWAP operations since they affect the reliability of the circuit and the execution time of the algorithm. Zulehner et al. [63] have provided an approach based on depth-based partitioning, an A* search algorithm, a look-ahead scheme and a dedicated mapping initialisation. Their proposed approach has been shown to outperform IBM's mapping solution and is fully integrated into IBM's SDK.

They start by decomposing the quantum algorithm to a representation in terms of elementary gates, after which they ensure that the CNOT-constraints of the architecture are satisfied. CNOT-constraints arise because the IBM QX architecture does not have all to all connectivity. Thus, certain physical qubits can only perform gates in a specific direction and with specific qubits. The decomposition of a quantum algorithm into elementary gates has been intensely studied, and thus the tools to perform this operation are available [64, 65, 66, 67, 68, 69, 70]. To fulfil the necessary CNOT-constraints, it is often necessary to incorporate additional Hadamard (H) and SWAP operations. As mentioned before, introducing additional gates reduces the overall reliability of the performed operation.

Thus they propose a method to efficiently map a given quantum circuit (which has been decomposed into elementary gates) to the IBM QX architectures, with the primary objective of adhering to the CNOT-constraints

while minimising the overall number of additional gates introduced, making the circuit CNOT-compliant. CNOT-compliance is similar to making the circuit nearest-neighbour-compliant, but CNOT-compliance also considers the architecture's limitations since not all nearest neighbours can interact. Two steps are taken to make the circuit CNOT-compliant. They partition the circuit into layers consisting of operations that can be applied concurrently and determine which SWAP operations are required to go from one layer to the next. The layers consist of concurrent operations, making it possible to determine a mapping for a layer such that the CNOT-constraints are satisfied. The second step is required because the mapping of logical qubits for different layers will differ. Finding the minimal number of SWAP operations needed to transition from one layer to the next is a problem with an exponential level of complexity regarding the number of physical and logical qubits. An A* search algorithm is applied to deal with this complexity. Only using the A* algorithm based on the previous layer would not ensure that the overall execution is optimal. A look-ahead scheme is proposed to deal with this, incorporating information about the following layer to the cost function of the A* algorithm. This approach, including a look-ahead scheme combined with an initial CNOT-compliant mapping for the first layer, can efficiently map quantum algorithms to existing quantum hardware. The efficiency is validated by experimental evaluations, which show that the proposed approach outperforms the solution provided by IBM at the time.

The work which has previously been discussed does not take into account the error rates which are present in NISQ devices. The introduction of Quantum Error Correction could mitigate the influence of these error rates. However, implementing QEC requires significant overhead in the number of physical qubits needed to run a quantum algorithm. Thus this will not be possible on NISQ devices where the number of qubits is limited. Nonetheless, the NISQ devices could still be valuable without implementing QEC. Tannu et al. [71] have studied the problem of Qubit-Allocation (initial qubit mapping) and Qubit-Movement (inserting SWAP operations) to deal with the variation in the error rates of different single- and two-qubit operations possible on the quantum architectures available at the time. To quantify variation in error rates, they analyse the error rates of the IBM-Q20 architecture for both single- and two-qubit operations. The data they studied was obtained over 52 days. The errors which have been considered are retention errors and operational errors. Retention errors occur due to relaxation of the qubit to the ground state or phase errors resulting from interaction with the environment. Operational errors occur when performing operations, such as the SWAP operation, which can affect the qubit's state. Though the error rates of both single- and two-qubit operations have been considered, the work focuses on the operational errors due to two-qubit gates since these significantly impact the device's reliability.

They define metrics to quantify the reliability of the NISQ device. The defined metrics are the 'Mean Instructions Before Failure' (MIBF), which indicates the number of instructions performed before the first error occurs and the 'Probability of Successful Trial' (PST), which indicates the probability of running a quantum algorithm successfully without an error. The MIBF metric is helpful for quantum algorithms with a large circuit depth, whereas the PST metric is valuable for quantum algorithms that can successfully run on current NISQ devices. Having defined reliability metrics, they show that the device error rate significantly impacts the overall device reliability.

Furthermore, they introduce the notion of Variation-Aware Qubit Movement (VQM) and Variation-Aware Qubit Allocation (VQA), which consider the additional constraints of the error rates of the underlying architecture. The VQM is similar to determining the required number of SWAP operations, and the VQA is similar to mapping the logical qubits to the underlying physical architecture. The differentiating factor is that VQA and VQM also consider the error rates. In practice, this evades the weaker single- and two-qubit operations. This approach significantly improves efficiency when running quantum algorithms on the IBM-Q20 architecture. The approach presented by Tannu et al. [71] can assist in gaining more understanding of problems such as resource sharing and partition problems on NISQ devices. For example, the approach can give insight into whether it would be better to run two copies of a quantum algorithm in parallel or to run one copy on the part of the system which maximises the device's reliability.

The examples discussed utilised a circuit-oriented gate-level description for quantum algorithms. Murali et al. [72] have considered quantum algorithm interpretations expressed in logic operations and quantum functions in a high-level programming environment. Their approach is introduced to allow the design of tool flows which efficiently use the available hardware without sacrificing the high-level programming environment. They achieve this by designing a compiler which takes as input the quantum algorithm represented in the Scaffold language, which is an extension of C with quantum types [73, 27]. The output is a near-optimal spatiotemporal mapping used to produce target code in the OpenQASM language. The output code can be executed directly on the hardware, such as the 16-qubit architecture from the IBM Q project.

They construct a constrained optimisation program considering both the algorithm characteristics and the architecture constraints. When constructing the constrained optimisation, they consider the initial qubit mapping, the gate scheduling and the CNOT routing. The mapping of qubits from the algorithm to the underlying architecture results in the constraint that no two logical qubits can be mapped to a physical qubit. When considering the scheduling of gates, both the start time and the gate duration are considered. For single-qubit operations, the duration depends solely on the execution time. The duration of CNOT gates includes the time required for the SWAP sequence, which ensures the logical qubits involved in the CNOT gate are assigned to physical qubits that can perform the CNOT operation. The scheduling of gates introduces the constraint that a gate should only be executed once the gates it depends on have been completed. Lastly, they consider the implementation of CNOT routing to prevent conflicts between SWAP- and CNOT gates. This results in the constraint that if two CNOT gates overlap in time, the paths of their SWAP sequences on the architecture should not overlap.

The optimisation is performed by introducing a dummy gate that depends on every gate in the quantum algorithm. In addition, the optimisation objective is to minimise the start time of the dummy gate. The optimisation of the objective function was initially attempted using the Optimisation Modulo Theory (OMT) solver in Z3. OMT is an extension of SMT which enables the discovery of exact models optimised for the specific objectives [74]. However, it is slow in practice, so the choice has been made to consider a near-optimal solution. The near-optimal solution is at most a factor of 1.1 from the optimal solution. The procedure of determining the near-optimal solution is referred to as the OPT algorithm.

The proposed compiler uses a Z3 Satisfiability Modulo Theory (SMT) solver to map the quantum algorithm to the hardware while performing the optimisation considering the constraints and optimising the overall execution time. Furthermore, SMT optimisation simultaneously considers the constraints and the dependencies they have on each other. Simultaneously considering these constraints causes a bottleneck for the scalability of the solver. It has been proposed to separate the compilation into two phases to deal with this issue. First, the qubits are mapped, minimising the number of required SWAP operations, and then the gates are scheduled and routed.

Moreover, benchmarks have shown that generating near-optimal code for underlying architectures with a small qubit count and limited coherence time is possible. In the case of architecture with a larger qubit count, a heuristic SMT optimisation can generate a target code guaranteed to finish within the window of coherence of the architecture. It is demonstrated that it is possible to determine near-optimal compilations using the proposed method for current and near-term NISQ devices. In the case in which larger systems have been considered, the heuristic method is demonstrated to be scaleable and can determine coherence compliant schedules.

Examples have also been presented which consider not only the constraints based on the connectivity of the architecture and the available elementary gates but also the constraints resulting from the shared classical control electronics. These constraints can be restrictive, especially when considering the scaling of the mapping to suit larger architectures. The last constraint is that classical electronics are required to control and operate qubits. The classical control electronics are shared among several qubits. An example is qubits measured through the same feedline [75, 76]. The shared control limits the parallelism of quantum operations.

An example that considers the constraints due to shared classical control is the introduction of a mapper called Qmap [77]. This mapper was developed to make a quantum algorithm executable on a scaleable superconducting quantum processor such as the Surface-17. The Surface-17 is a scaleable processor with a surface code architecture [78, 79]. The surface code architecture is a scaleable system capable of performing fault-tolerant quantum computation based on the surface code architecture [80]. Several parameters have been considered in creating the mapper, namely the architecture's elementary gate set, the qubit connectivity constraints and the restrictions imposed by the shared classical control.

They aim to find an initial mapping based on minimising the number of qubit movements with the help of the Integer Linear Programming (ILP) algorithm presented by [81]. The problem of determining an initial mapping has been formulated as a quadratic assignment problem. In this formulation, the communication overhead between qubits is represented by their distance minus 1. Moreover, since it is unlikely that all of the architecture's constraints are satisfied upon performing the initial mapping, they present a heuristic algorithm to perform the routing by inserting SWAP operations. After implementing the initial mapping, this algorithm finds all two-qubit gates where the logical qubits are not nearest neighbours. Then it inserts the required SWAP operations to make these logical qubits adjacent. The optimisation objective is to achieve the shortest circuit latency, which they claim leads to the highest instruction-level parallelism. The latency represents the execution time of the algorithm when considering actual gate durations. The scheduling of gates is also performed with the help of a resource constraint scheduler which reschedules the routed circuit based on the same latency optimisation objective to minimise the required lifetime and the decoherence error of each qubit. Furthermore,

decomposition is performed, decomposing the circuit into the architecture's elementary gates, maintaining the constraints discussed earlier. Lastly, optimisation is implemented to reduce the required operations, as gates that cancel each other out are eliminated. Both the decomposition and optimisation modules can be performed at any point during the mapping procedure. In the case of Qmap, these modules are performed before and after the routing.

In their evaluation, they ran 56 benchmarks from both RevLib [82] and QLib [83] on both the Surface-17 processor [79] as well as the IBM Q Tokyo processor [16]. The parameter they considered in the evaluation is circuit depth which represents the length of the circuit, and circuit latency, which again represents the execution time considering real gate durations. Three routers have been evaluated, showing that the MinExtendRC router [77] yielded the lowest overhead in both circuit latency/depth and number of gates. Compared to a trivial router, this choice of a router shows that the resulting overhead of the number of gates is reduced by 80% Elements which stand out in the proposed mapper (Qmap) are the introduction of the MOVE operation, which can be used instead of a SWAP operation on an uninitialised qubit, which is shown to reduce the resulting overhead substantially. They have found that higher connectivity helps decrease the number of required inserted operations. Rather than optimising to find either the minimal circuit depth or the least number of additional gates [71, 72, 84, 85, 86, 87] proposed to consider the reliability as the optimisation metric, analysing the impact which the mapping has on the success rate of the algorithm. Qmap has been embedded in the OpenQL compiler [88] to support several quantum processors. The constraints of the processors are described in a configuration file. Lastly, they note the importance of a flexible mapper for making quantum circuits executable on different real quantum processors in the future.

As has become clear, it is rarely possible to directly execute quantum algorithms on NISQ-era devices due to the present hardware constraints, such as the limited connections for two-qubit gates. The solutions discussed thus far have some limitations. Namely, they have a high level of complexity, the quality of the initial mappings is often poor, and they have limited flexibility. To address these limitations, Li et al. [41] propose a SWAP-based BidiRectional heuristic search algorithm (SABRE), which applies to NISQ devices with arbitrary connections between qubits. This extension is relevant since the work discussed so far can handle arbitrary coupling graphs. However, their proposed method is limited due to significant runtime [63]. Moreover, the presented mapping strategies lack the ability for global optimisation. Lastly, these strategies cannot optimise for the varying characteristics of NISQ devices, limiting the possible uses of the proposed solutions.

SABRE has been proposed to tackle the limitation of previous solutions. The proposed algorithm consists of an optimised SWAP-based heuristic search scheme and a reverse traversal search technique and introduces a 'decay' effect on the cost function. Noting that the mapping transition should start based on the qubits involved in two-qubit interactions, the heuristic search scheme significantly reduces the search space. The optimised SWAP-based heuristic search can reduce the complexity from $O(exp(N))$ to at most $O(N^{2.5})$ where N represents the number of physical qubits—providing an exponential speed-up. The reverse traversal search technique generates a high-quality initial mapping and deals with the complex initial mapping problem, which is done by considering the reverse traversal of the quantum algorithm. A random initial mapping is generated, and based on this mapping, the SWAP-based heuristic search is performed on the original circuit. The mapping obtained is used as the initial mapping to repeat the process on the reversed circuit. The reversed circuit is created by reversing the sequence of gates in the original circuit. This method gives more weight to the gates present at the start of the algorithm while still considering all the gates present in the rest of the algorithm. Lastly, the introduced 'decay' effect slightly increases the heuristic cost function by adding weight to recently SWAPed qubits. Simulating the algorithm to choose SWAP operations involving different qubits increases the parallelism of added SWAP operations. Moreover, it enables generating of different hardware-compliant circuits with a trade-off between circuit depth and the number of gates. This trade-off can be tuned by modifying the weight of the 'decay' effect.

Furthermore, the algorithm can optimise for different objectives. The objectives are flexibility, fidelity, parallelism and scalability. Flexibility ensures the algorithm can deal with irregular coupling designs and their evolution over time. The fidelity objective deals with the fact that NISQ devices are prone to errors, and as such, it is relevant to minimise the number of required operations, such as the two-qubit interactions. In addition, parallelism aims at allowing SWAP operations to be performed in parallel such that the depth of the circuit is limited. Lastly, scalability aims to keep the algorithm operational as the number of physical qubits increases. The formulated optimisation objectives are evaluated using several benchmarks of different sizes and compared to the existing algorithms in which Zulehner et al.[63] is considered the best algorithm at the time. These benchmarks have been selected from previous works [42, 89] as well as programs from QISKit [90], functions from RevLib [82] and algorithms compiled from both Quipper [28] and ScaffCC [73]. The underlying architecture considered is the Q 20 Tokyo chip from IBM [35]. The metrics used in the evaluation are the

total number of gates and the circuit depth of the final generated circuit, which complies with the underlying architecture's hardware constraints. The SABRE algorithm is shown to reduce the number of additional gates by 91% in smaller circuits and 10% in the case when larger circuits are considered. Furthermore, the speed-up in runtime is significant even though SABRE has been written in Python compared to Zulehner et al. [63], who constructed their algorithm in C++. Lastly, it is shown that it is much more scaleable when compared to Zulehner et al. [63].

SABRE is a promising step in the right direction as it allows studying larger quantum algorithms effectively. Nonetheless, there are still certain limitations to consider. Namely, the implementation of single-qubit gates has not been considered, and only the insertion of SWAP operations has been considered without adapting the gates in the original circuit.

3.2 Graph theory approach to the Qubit-Mapping Problem

As hinted at the start of the previous section, using a graph-theoretic perspective can be relevant when addressing the QMP. [45] achieved the reduction of total interaction distance by restating the problem as the Minimum Linear Arrangement (MinLA) problem in graph theory [46]. Various structures can be represented as graphs, and graph theory can be used to model and solve various problems. Due to the many applications, graph theory has attracted considerable attention in various fields [91]. An example of a problem considered in graph theory is determining the similarities between different graphs. The study of this problem is relevant to this thesis, as it will be incorporated into the algorithm. This section will discuss works that invoke the use of graph theory in proposing solutions for the QMP.

An example of a proposed solution which adopted graph theory to solve the QMP has been presented by Siraichi et al. [92]. The model presented is a combination of finding subgraph isomorphism and token swapping. The subgraph isomorphism problem involves comparing two graphs, $G = (V, E)$ and $H = (V, E)$, where V represents the vertices and E represents the edges of the graph. The objective is to determine whether G contains a subgraph that is isomorphic to H. An isomorphism is a bijection $f : V(G) \to V(H)$ between the vertex sets of G and H such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H. The subgraph isomorphism problem is NP-complete [93], though some instances of the problem can be solved in polynomial time [94, 95]. Subgraph isomorphism is identified as a potential metaphor for the initial mapping problem. However, this analogy only holds if no additional SWAP operations are required after the mapping since all required operations can be executed directly. Furthermore, the token swapping problem [96] refers to placing tokens on vertices of a graph and finding the shortest sequence of swaps such that a token is placed on a different vertex within the graph. In the case of the QMP, the token can be seen as one of the qubits involved in a two-qubit operation. The goal is to find the shortest path such that this qubit is placed on a vertex that shares an edge with the other qubit involved in the operation, which is the same problem as finding the required SWAP operations to make the quantum algorithm CNOT-compliant. Single qubit gates have not been considered. In this work, they rely on an approximate solution to the Token Swapping problem [97]. Combining these two approaches offers a model that completely describes the QMP.

Therefore, Siraichi et al. [92] propose a parameterised polynomial-time algorithm which breaks down the QMP into a combination of token swapping and subgraph isomorphism. Restating the problem simplifies finding a solution since it is possible to use well-established heuristics and approximations developed for token swapping and subgraph isomorphism [98, 99, 100], which requires developing an algorithmic approach based on a combination of solutions developed for token swapping and subgraph isomorphism. First, the mapping problem is split into multiple partial instances for which the subgraph isomorphism problem is solved. A greedy search is implemented to determine solutions to individual instances. In the case of larger circuits, many partial instances could be considered. To deal with this, the partial instances to be solved are bounded by parameters which control the size of the solution space, which may not lead to the optimal solution but offers flexibility to balance optimality and runtime. Once the partial instances have been solved, token swapping is implemented to find the SWAP operations required to transition from one partial instance to the next. The approximation presented by Miltzow et al. [97] is used to solve the token swapping problem.

The proposed algorithm has been compared to the works which have been discussed in the previous section [63, 41, 101] as well as IBM's Python SDK (QISKit) and Zulehner and Wille [102]. It is important to note that none of these algorithms guarantee an optimal solution. Nonetheless, they have shown that the proposed approach can generate more efficient solutions than those presented. The evaluation of the algorithm has been performed on the benchmarks also used by Zulehner et al. [89]. The quality of the mapping has been evaluated based on three parameters, namely, weighted cost, total number of gates and depth of the resulting circuit. Here the weighted cost represents the combined cost of all the gates used in the resulting circuit. This differs from

the total number of gates since the weight of single-qubit and two-qubit gates are 1 and 10, respectively. In addition, the efficiency has been considered based on memory consumption and the required time for processing a quantum algorithm. For an in-depth comparison of the algorithms, the reader is referred to Sec. 4 of Siraichi et al. [92].

The subgraph isomorphism gained traction and led to a new search algorithm incorporating it alongside a filtered depth-limited search. This algorithm was presented by Li et al. [103]. Similar to the work previously discussed, the aim is to construct a graph based on the quantum algorithm, which is isomorphic to a subgraph of a NISQ architecture. This isomorphism determines the initial mapping. All gates from the quantum algorithm that can be executed based on the initial mapping are added to a 'physical' circuit. The 'physical' circuit represents the circuit which can be implemented on the NISQ device. SWAP operations must adapt the mapping to execute the remaining gates if there are still unexecuted gates. A good subsequent mapping could be found by exhaustively searching all possible combinations of SWAP operations. To determine which combination maximises the two-qubit gates that can be executed after the SWAP operations have been implemented. It is proposed to set up a fixed limit $k > 0$, ensuring that only combinations of k SWAP operations are considered to mitigate the high cost of an exhaustive source. Further speeding up the proposed algorithm, they introduce filters. These filters selectively eliminate SWAP operations that do not decrease the distance between the qubits involved in pending two-qubit operations. The filters improve the algorithm's runtime but may filter out the optimal solutions [103].

Therefore, this approach does not outperform previously discussed algorithms [41, 92] when considering a small number of two-qubit operations. Nonetheless, their approach is relevant since it scales polynomially in the number of two-qubit operations. The algorithm significantly reduces the required SWAP operations for many two-qubit operations. In comparing their algorithm to the previously discussed SABRE algorithm [41], they show that for quantum algorithms with over 200 CNOT gates, the improvement in the other required number of SWAP operations is at least 50%. In further comparison, they note that the improvement of the proposed algorithm is particularly true when dealing with quantum algorithms with more than 1000 CNOT gates.

Jiang et al. [104] have proposed an algorithm combining subgraph isomorphism and Tabu search-based adjustments (TSA). Tabu search is implemented to adjust the mapping dynamically. It has the advantage of circumventing local optima and avoiding SWAP operations on recently exchanged qubits. Which enhances the parallelism of gates.

Their approach consists of three steps: preprocessing, isomorphism, completion, and adjustment. The shortest distance between each pair of nodes is determined during preprocessing. In addition, they decompose the circuit into layers containing the two-qubit operations that can be performed in parallel. In the isomorphism and completion step, a subgraph isomorphism algorithm [105] is used to determine the initial mapping. Once these steps have been completed, the adjustments are carried out, inserting SWAP operations to ensure that the mapping of the logical qubits is CNOT-compliant for each layer. A Tabu search-based adjustment algorithm has been introduced to deal with the complexity of finding the optimal SWAP operations. This heuristic algorithm uses a tabu list to prevent redundant solution space exploration. This method aims to adjust large circuits and should produce an output circuit close to the optimal solution.

Their experimental results have shown that the proposed method generates high-quality initial mapping and reduces the number of inserted SWAP operations compared to the previous algorithms [63, 41, 103]. They show a 27.14% reduction in the number of additional gates and a 10.74% reduction in the circuit depth compared to Zulehner et al. [63]. Additionally, compared to Li et al. [103], the reduction amounts to 22.43% As mentioned above, numerous algorithms have been proposed to address the QMP. However, there is still a need to assess the problem while considering the noise characteristics of NISQ devices. A noise-aware heuristic mapping algorithm has been introduced by Steinberg et al. [106] to address this concern. This algorithm considers the initial mapping of a quantum algorithm. Compared to the previous work, this algorithm considers the error-rate statistics of two-qubit, single-qubit and measurement operations of the underlying architecture. An upper and lower bound has been established to evaluate the algorithm's performance. These bounds are determined by comparing the proposed algorithm with two reference algorithms, a brute force and a trivial mapping algorithm. The brute force algorithm exhaustively searches all possible mappings to determine the optimal solution. On the other hand, the trivial algorithm maps the logical qubits to the physical qubits based on their corresponding indices. In this case, logical qubit 1 is mapped to the physical qubit 1, logical qubit 2 is mapped to physical qubit 2, and this goes on until all the logical qubits have been mapped. It has been shown that the proposed mapping algorithm outperforms the trivial mapper and, in some instances, approaches the optimal results obtained by the brute-force algorithm.

The impact of the number of edges of the interaction graph and their distribution are evaluated using the

success rate metric. This metric is tied to the fidelity of the final state obtained by mapping and executing all gates on a specific architecture. The success rate evaluation varies based on the characteristics of the interaction graph mentioned above.

Lastly, the scaling behaviour of the heuristic mapping has been investigated to find that the greedy heuristic scales well compared to the brute-force approach. Moreover, the algorithm works well compared to the trivial mapper when 75% of the physical qubits are utilised. As the number of logical qubits increases, the effectiveness of the proposed algorithm starts to decline.

To conclude this section, we briefly consider the work done to find a sub-graph isomorphism in larger graphs. In the future, the mapping problem will have to scale with the growing number of qubits in quantum architectures, such as IBM's recent chip, which packs 433 qubits [107]. The coupling graphs also increase as the functional number of qubits available increases. The problem of finding sub-graph similarities in large graphs has been studied, and methods have been proposed to deal with the problem efficiently.

A proposed method uses graphlet kernels to determine the distance between graphs. A sub-graph similarity matching algorithm takes the coupling graph as the large target graph and the interaction graph as a query graph. The algorithm then determines an induced sub-graph of the coupling graph, which is most similar to the interaction graph with respect to the graphlet kernel value. Graphlet kernels are known to perform better than others for the specific task at hand [108]. The algorithm first computes vertex labels for the coupling and the interaction graphs. These labels consider the local neighbourhood structure of vertices present in either graph. This labelling opens the possibility to define notions of topological similarity of neighbourhoods corresponding to particular vertices of the graph being considered. The computation of these labels for both graphs and determining the nearest neighbour data structure is part of the preprocessing phase of the algorithm. The vertices of both graphs, which are most similar, are determined using queries on this information.

The proposed algorithm has been tested on social media and search engine platform graphs. These are large graphs with thousands of vertices and millions of edges. They are far more extensive than the coupling graphs of near-future quantum devices. The algorithm finds highly similar matches when queried in these networks. [109].

4 Methods

Current research has utilised classical techniques to consider the mapping problem. This thesis will implement tools from quantum information theory to introduce a new perspective on solving the problem. Previous work has provided an interpretation of the mapping problem from the standpoint of fundamental physics and mathematics [110]. Providing an interpretation which utilises prior work in Quantum Physics, Complex Network Theory and Graph theory. To summarise, the physical architecture represented as the coupling graph and the desired quantum algorithm represented as the interaction graph has been elevated to quantum objects, namely density matrices. This formalism enables the utilisation of established techniques from quantum information theory to quantify the information capacity of the interaction graph (IG) and the coupling graph (CG). The information capacity is represented by the 'von Neumann entropy'. Moreover, by representing the IG and CG as density matrices, their difference can be quantified. This is achieved using the 'quantum Jensen-Shannon divergence' metric, initially introduced by Biamonte et al. [111] 2016 to study complex networks.

As mentioned, the interaction and coupling graphs have been represented as density matrices, which are used in quantum mechanics to encode probability distributions. This representation enables the utilisation of the quantum operation formalism to map the coupling graph to the interaction graph. The mapping operation can be represented as a superoperator, which describes a quantum channel between two quantum states. A quantum channel is the most general description for transforming a quantum state. In the context of this thesis, the quantum states correspond to the density matrices of the IG and CG, while the quantum channel is introduced to represent the mapping between these states. A superoperator can be written down in the Kraus operator representation. We will use this representation to study the mapping problem. Determining the minimal number of required SWAP operations is possible by studying the Kraus operators, which map the coupling graph to the interaction graph. The Kraus operators are related to the SWAP operations possible for a given CG. This is achieved by introducing additional constraints for the Kraus operators. Calculations will be provided for certain example graphs to explore the mathematical framework. Moreover, an algorithm will be proposed to compute the Kraus operators for a CG/IG-pairing where the initial mapping is provided. The algorithm will then be evaluated by considering toy-model examples and several CG/IG-pairings benchmarks. The evaluation will be presented in Sec. 5.

4.1 Mathematical prerequisites

As has been discussed this thesis will aim to redefine the graphical representations of physical quantum architectures and quantum circuits to quantum objects, to address the problem of determining minimal number of SWAP operations required to map the quantum circuits to physical quantum architectures. In this section, the proposed mathematical formulation will be presented. This formulation represents the quantum algorithms and the quantum architectures as graphical objects. Upon establishing these objects they will form the basis which will allow us to elevate them as quantum theoretic objects, namely density matrices. By describing the quantum algorithms and the quantum architectures as density matrices it will be possible to utilise tools from quantum information theory, such as the quantum channel representation, Von-Neumann entropy and Quantum Jensen-Shannon graph divergence.

4.1.1 Graph theory

We will commence with the introduction of the graph theoretic approach. First, the basic definition of a graph will be presented. Several relevant concepts from graph theory which will be required will be discussed. Lastly, it is explained how quantum algorithms and quantum architectures are represented as graphs.

A **simple graph** $G = (V, E)$ is made up of a non-empty finite set $V(G)$ of the element which we call **vertices**, they are also referred to as **nodes**, in combination with a finite set $E(G)$ which contains distinct unordered pairs of distinct elements of $V(G)$ which we call **edges** of the graph G. An edge $\{p, q\}$ connects the vertices p and q in the graph G. The example graph shown in Fig. 9 has vertex set $V(G) = \{1, 2, 3\}$ and an edge set $E(G) = \{\{1, 2\}, \{2, 3\}, \{1, 3\}\}$ [112]. Moreover, a graph is said to be on n vertices in the case that $|V(G)| = n$, in other words, the set $V(G)$ contains n distinct nodes.

Additional concepts from graph theory are required namely the concept of the adjacency matrix A and the degree matrix D. The **adjacency matrix** A is defined as:

$$A(G)_{i,j} = \begin{cases} 1, & \text{if } \{v_i, v_j\} \in E(G); \\ 0, & \text{if } \{v_i, v_j\} \notin E(G). \end{cases} \tag{13}$$

This definition shows that vertices v_i and v_j are adjacent in case an edge connects v_i and v_j. Furthermore, the

degree of a vertex $d(v_i)$ is the number of vertices which are adjacent to v_i, which leads to the following definition of the **degree matrix** D:

$$D(G)_{i,j} = \begin{cases} d(v_i), & \text{if } i = j; \\ 0, & \text{if } i \neq j. \end{cases} \tag{14}$$

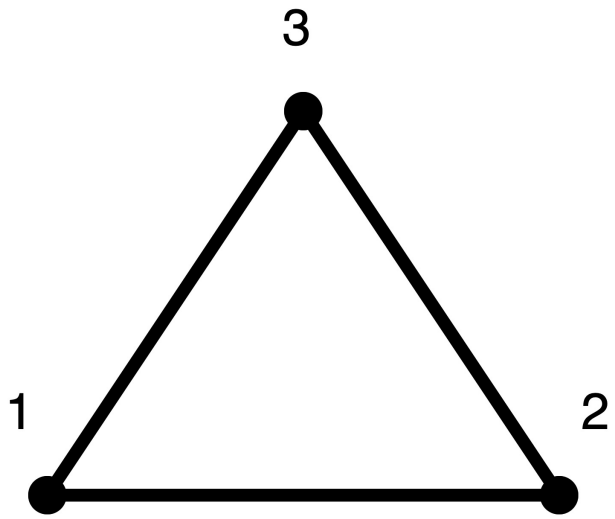


Figure 9: Example graph with 3 vertices and 3 edges.

Thus for the example shown in Fig. 9 we find the following adjacency and degree matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}; D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{15}$$

Lastly we introduce the **graph Laplacian** L which is defined as:

$$L = D - A \tag{16}$$

Here D and A are the degree and adjacency matrix of the graph. In the case that we again consider the graph example in Fig. 9 we find the following graph Laplacian:

$$L = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \tag{17}$$

4.1.2 Graphical representation of quantum algorithms and architectures

In this thesis, the graph of the quantum algorithm will be referred to as the **interaction graph (IG)**, in which the qubits of the circuit which describe the algorithm, are represented as the vertices of the interaction graph $V(IG)$. two-qubit operations are represented as edges $E(IG)$, which are unweighted and undirected indicating that a two-qubit operation can be performed in either direction. Moreover, it is crucial that at this point there is no interpretation provided by the formulation for single qubit operations performed on the qubits.

The physical architecture will be defined in an analogue manner and referred to as the **coupling graph (CG)**. The CG again consists of a vertex set $V(CG)$ which consists of the qubits in the underlying physical architecture. The edge represents the two-qubit interactions allowed based on the topology constraints of the architecture set $E(CG)$. Note that the description of the architecture does not take into account the errors which may take place as two-qubit operations are performed on the underlying architecture. Lastly, it is essential to note that there are no disconnected vertices or sets of vertices in the CG. The last assumption ensures we can exchange all vertices by implementing SWAP operations.

We assume that both the edge sets $E(IG)$ and $E(CG)$ do not contain loops, which is an edge which connects a vertex to itself $\{v_i, v_i\}$, for an arbitrary vertex from the set $V(IG)$ or $V(CG)$. In addition, it will be assumed

that there is at most one edge connecting a pair of vertices, $\{v_i, v_j\}$.

Having established the graphical interpretation there are some additional mathematical tools are required before it is possible to describe these graphs as quantum mechanical objects. These tools will be introduced in Sec. 4.2.

4.2 Quantum information tools

In this section, the translation of the graphical objects representing quantum algorithms and the underlying physical architectures to quantum mechanical objects will be presented. First, the description of the density matrix based on these graphic objects will be introduced, followed by the Von Neumann entropy, the Quantum Jensen-Shannon graph divergence and finally the quantum channel formulation.

4.2.1 Density Matrix

In quantum mechanics, the density matrix or density operator (ρ) is a $n \times n$ matrix which describes the state of a quantum mechanical system that acts on a n-dimensional Hilbert space. A density matrix has several properties which will be important in the scope of this thesis namely, the matrix of a density operator is Hermitian, positive semidefinite and has a trace equal to one. A Hermitian matrix (H) is a complex square matrix equal to its conjugate transpose, $H = \overline{H^T}$. A positive semidefinite matrix is a Hermitian matrix with nonnegative eigenvalues. The trace of a matrix is the sum of its diagonal elements, $tr(\mathbf{H}) = \sum h_{ii} = 1$ where h_{ii} is the matrix element of the ith row and ith column. A density matrix is a generalisation of the state vector introduced in Sec. 2.1.1 to describe the state of a qubit. An advantage of describing a state as a density matrix instead of using state vectors is the fact that a density matrix may represent a pure state as well as mixed states while a state vector can only represent pure states. To understand the difference between pure and mixed states we consider how they are represented on the Bloch sphere introduced in Sec. 2.1.1. A pure state is represented by a point on the sphere's surface whereas an interior point represents a mixed state. In an analogue description, a pure state can be represented by a state vector $|\phi\rangle$ whereas a mixed state is a statistical ensemble of different state vectors. An example of a mixed state is a system with a 50% probability that the state vector is $|\phi_1\rangle$ and a 50% probability that the state vector is $|\phi_2\rangle$. In the case that $\mathrm{Tr}\{\rho\} = 1$ a density matrix is in a pure quantum state, whereas $\mathrm{Tr}\{\rho\} \leq 1$ represents a mixed state. Furthermore, the formulation is useful for describing unknown ensembles of quantum states. The density matrix of a number of states $|\psi_i\rangle$ with probabilities p_i is defined as [21]:

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \tag{18}$$

We will utilise the definition of the density matrix of a graph introduced by Biamonte et al. [111]. They note that the evolution of the information within a network can be denoted by the vector $\mathbf{p}(\beta)$. At a time $\beta = 0$, the evolution of the information of the network across time can then be described as $\mathbf{p}(\beta) = e^{-\beta \mathbf{L}}\mathbf{p}(0)$, where $e^{-\beta \mathbf{L}}$ is the diffusion propagator at time β and $\mathbf{L}$ is the graph Laplacian. To use this propagator as a density matrix, it will be necessary to normalise it to ensure that the definition of this density matrix adheres to the definitions of a density matrix in the standard quantum mechanical formulation. To that end, the density matrix of the graphical object will be defined as follows:

$$\rho(\beta) = \frac{e^{(-\beta L)}}{\mathrm{Tr}\left(e^{(-\beta L)}\right)} \tag{19}$$

The normalisation of the state is ensured by dividing by the trace of the diffusion propagator. This definition of the density matrix can be compared to the Gibbs state or thermal state of a Hamiltonian H at inverse temperature β which is denoted as [113]:

$$\rho_G(\beta) = \frac{e^{-\beta H}}{\mathrm{Tr}(e^{-\beta H})} \tag{20}$$

This comparison makes it clear that the graph Laplacian can be interpreted as the Hamiltonian describing the system's dynamics. Whereas the β value represents a notion of inverse temperature, often denoted as $\beta = \frac{1}{k_B T}$ [114], here k_B is the Boltzmann constant and T is the temperature. The implication of the β value will be discussed further in Sec. 4.2.3.

4.2.2 Von-Neumann entropy

In this section the Von-Neumann entropy will be introduced. We do so because this entropy measure is utilised in the metric for comparing the density matrices of the IG and CG. Entropy quantifies the information or uncertainty present in a system [21]. In the classical case when we learn the value of a random variable X the Shannon entropy [21] tells us how much information was gained on average. If we do not know the value of X the Shannon entropy is measured for the uncertainty. The entropy of a random variable is a function of the probabilities of the different possible values of the random variable. The Shannon entropy may be associated with a probability distribution $(p_1, ..., p_n)$ in the following manner [21]:

$$\mathcal{H}(X) \equiv \mathcal{H}(p_1, ..., p_n) = -\sum^{n} p_i \log_2 p_i. \tag{21}$$

Here p_i denotes the probability of obtaining a specific value for the random variable. Note that if $p_i = 0$ the $\log_2 p_i$ will be undefined, to circumvent this issue the following convention is agreed upon, $0 \log(0) \equiv 0$.

The quantum mechanical counterpart to the Shannon entropy is the von Neumann entropy, where density matrices have replaced the probability distributions. Once a formulation for the density matrix of a graphical object has been established, it becomes possible to consider the IG and CG as quantum states. The von Neumann entropy of a density matrix can then be defined as [21]:

$$\mathcal{S}(\rho) = -\operatorname{Tr}[\rho \log_2 \rho] = -\sum_{i=1}^{N} \lambda_i \log_2 \lambda_i. \tag{22}$$

In which N is the dimension of the density matrix, λ_i are the eigenvalues and ρ represents the density matrix of a graph as defined in Eq. 19 and we again adhere to the convention that $0 \log(0) \equiv 0$. The von Neumann entropy may be used to characterize the entropy of entanglement. Relevant properties of the von Neumann entropy are listed below [21]:

1. The entropy is non-negative. The entropy is zero if and only if the state is pure.
2. In a n-dimensional Hilbert space the entropy is at most $\log_2(n)$. The entropy is equal to $\log_2(n)$ if and only if the system is in the completely mixed state $\frac{\mathbb{I}}{n}$.
3. Suppose a composite system PQ is in a pure state. Then $S(P) = S(Q)$.
4. Suppose p_i are probabilities, and the states ρ_i have support on orthogonal sub spaces. Then

$$S\left(\sum_i p_i \rho_i\right) = H(p_i) + \sum_i p_i S(\rho_i) \tag{23}$$

5. Joint entropy theorem: Suppose p_i are probabilities, $|i\rangle$ are orthogonal states for a system A, and ρ_i is any set of density operators for another system, B. Then

$$S\left(\sum_i p_i \left|i\right\rangle \left\langle i\right| \otimes \rho_i\right) = H(p_i) + \sum_i p_i S(\rho_i) \tag{24}$$

In the case that two distinct quantum systems P and Q have the joint state $\rho^{PQ} = \rho^P \otimes \rho^Q$ the joint entropy is defined as $S(P, Q) \equiv -\operatorname{Tr}\left(\rho^{PQ} \log_2(\rho^{PQ})\right)$. It can then be shown that the von Neumann entropy also adheres to subadditivity and triangle inequality given below:

$$S(P, Q) \leq S(P) + S(Q) \tag{25}$$

$$S(P, Q) \geq |S(P) - S(Q)| \tag{26}$$

These are the properties of the von Neumann entropy which are relevant in the scope of this thesis, for an in-depth discussion including proof of the properties the reader is referred to chapter 11 of the textbook by Nielson and Chuang [21].

4.2.3 β-value

In Fig. 10 the entropy as defined by Eq. 22 of all 4-vertex graphs is compared for β values between $1 - 10$. We study this figure to get a better understanding of how the beta value will influence our proposed method.

Considering the figure it becomes clear that the entropy for $\beta = 0$ is the same for all graphs. Moreover, as $\beta \mapsto 10$ it becomes clear that the entropy of the different graphs cluster.

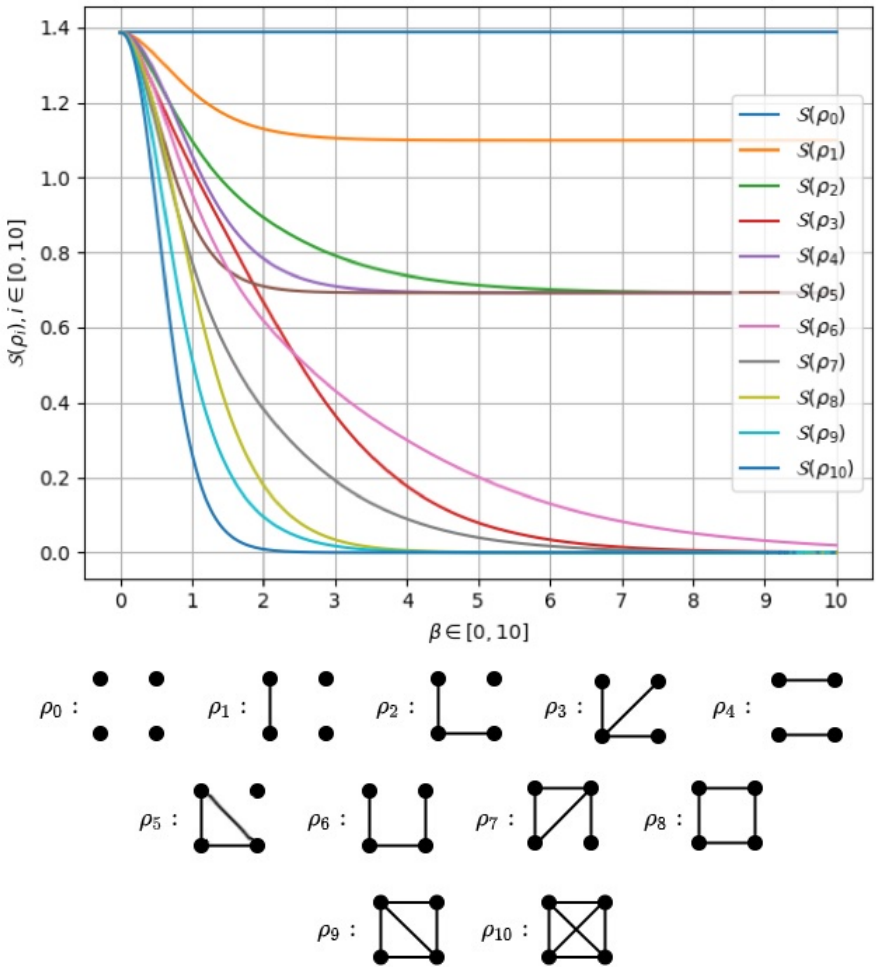


Figure 10: The figure compares the von Neumann entropy of all permutations of 4-vertex graph for several β values. The x-axis runs as $\beta \in [0, 10]$, the y-axis displays the von Neumann entropy of the different 4-vertex graphs.

Upon further inspection of which graphs cluster, it becomes clear that clustering can be related to the number of disjoint vertices of the graph. Disjoint vertices refer to those vertices or vertex pairs which do not have an edge connecting them to the other vertices or vertex pairs. For the empty graph with 4-vertices (ρ_0, the top blue line) logically, the entropy will not be influenced by β, this is because the graph Laplacian of an empty graph is the null matrix with rank corresponding to the number of vertices, thus the density matrix for this graph is independent of β.

If there are disjoint vertices in the graph we see their entropy clusters toward a non-zero value. If there are no disjoint vertices the entropy tends to 0 as $\beta \mapsto 10$. Lastly, it can be seen that the entropy of a more connected graph tends to 0 sooner, this can be seen when comparing ρ_6 (pink line) and ρ_7 (grey line). Considering the graphs for 4 vertices it is possible to distinguish four classes, with no disjoint vertices (ρ_{10}, ρ_9, ρ_8, ρ_7, ρ_6, ρ_3), with two disjoint vertices (ρ_2, ρ_4, ρ_5), with three disjoint vertices (ρ_1) and with four disjoint vertices (ρ_1).

The physical interpretation of the β value has been studied [115]. The interpretation in relation to this work has been a subject of discussion. In addition, the influence of the β value has been observed but it remains unknown how to tune this value such that the minimal number of SWAP operations is guaranteed. The observed influence of the β value will be presented in Sec. 5.1. The interpretation and precise impact of the beta values are important aspects that require further investigation in future work.

4.2.4 Quantum Jensen-Shannon divergence

A central goal of information theory is the quantifying of information present in a probability distribution. Similarly, quantum information theory provides metrics which quantify the difference between quantum states. Examples of such divergences which have been extended to the field of quantum information are the Kullback-Leibler divergence [116, 117] and the Kolmogorov distance [118]. However, these divergences lack some relevant properties such as being bounded and symmetric. For that reason, we will utilise a suitable manner to compare density matrices of graphs namely the quantum Jensen-Shannon divergence ($\mathcal{D}_{\text{QJS}}$) [119] which is defined as:

$$\mathcal{D}_{\text{QJS}}(\phi||\sigma) = \mathcal{S}\left(\frac{\phi+\sigma}{2}\right) - \frac{1}{2}\big(\mathcal{S}(\phi)+\mathcal{S}(\sigma)\big). \tag{27}$$

Here ϕ and σ are density matrices and $\mathcal{S}(\rho)$ is the Von-Neumann entropy defined in Eq. 22. The advantages of the $\mathcal{D}_{\text{QJS}}$ are that it defines an accurate metric bounded between $0 \leq \mathcal{D}_{\text{QJS}} < 2$ [119, 120, 121] and may be used to distinguish between quantum states [122, 123].

4.2.5 Quantum channel

Quantum operations have been introduced in Sec. 2.1.2, in this section the operations are unitary matrices acting on the state vector. There is however a more elegant form to represent quantum operations known as the operator-sum representation [21]. For a quantum operation ϕ acting on a density matrix ρ the operator-sum representation may be written as:

$$\phi(\rho) = \sum_i E_i \rho E_i^\dagger \tag{28}$$

In this equation, the operators E_i are referred to as the Kraus operators, which satisfy the following equation:

$$\sum_i E_i^\dagger E_i = \mathbb{I} \tag{29}$$

The operator-sum representation is analogous to the mathematical description of a completely positive map. Moreover, due to the trace-preserving property that results from Eq. 29 the map is not only completely positive but also trace-preserving. This completely positive trace-preserving (CPTP) map is referred to as a quantum channel in quantum information theory which maps between spaces of operators. This is important because CPTP-maps have many interesting properties, which will be utilised to express the Kraus operators such that they represent SWAP operations possible on a certain architecture.

Lastly, to include the notion of mapping the CG to the IG. We propose that given two quantum states, in our case the density matrices of the CG and IG, we define the quantum channel which maps one state to the other as:

$$\phi(\rho_{\text{CG}}) = \sum_i E_i \rho_{\text{CG}} E_i^\dagger = \rho_{\text{IG}} \tag{30}$$

Where ϕ indicates the quantum channel from the CG to the IG, which adheres to all the aforementioned properties. Since the Kraus operators correspond to the allowed SWAP operations for a specific CG, we chose to map the CG to the IG. Allowing us to determine the minimal number of SWAP operations for a given CG/IG-pairing where the initial mapping is provided. The proposed formulation is more elegant than previous works since we do not require a heuristic or an exhaustive search. Determining the Kraus operators of the quantum channel will provide the desired information. In the next section, we will explain how to reformulate the Kraus operator to represent the allowed SWAP operations of a CG. Following this reformulation, the algorithm determining the minimal number of SWAP operations for a given CG/IG-pairing where the initial mapping is provided will be presented.

4.3 Theoretical model

In this section, we will introduce the model which will be incorporated into the algorithm. The algorithm aims to determine the quantum channel between the state ρ_{CG} and ρ_{IG}, the quantum channel will be found by minimizing the $\mathcal{D}_{\mathrm{QJS}}$. The objective function we would wish to use has the following form:

$$\mathcal{D}_{\mathrm{QJS}}(\phi(\rho_{\mathrm{CG}})||\rho_{\mathrm{IG}}) = \mathcal{S}\left(\frac{\phi(\rho_{\mathrm{CG}}) + \rho_{\mathrm{IG}}}{2}\right) - \frac{1}{2}\Big(\mathcal{S}(\phi(\rho_{\mathrm{CG}})) + \mathcal{S}(\rho_{\mathrm{IG}})\Big). \tag{31}$$

The general operator-sum representation of a quantum channel does not relate the channel to the required SWAP operations which would allow the execution of a quantum algorithm on a quantum architecture. Therefore, we will need to make further adjustments to the proposed formulation. To determine a suitable objective function for the optimisation we must first introduce the concept of a doubly stochastic channel, then the constraints for the operator-sum representation will need to be reevaluated, leading to an adapted formulation of the Kraus operators. The adapted formulation will construct the Kraus operators based on permutation matrices representing the SWAP operations allowed based on the architecture.

4.3.1 Doubly stochastic channel

As discussed in Sec. 4.2.5 the mapping between the density matrices may be represented as a quantum channel which is a completely positive trace preserving (CPTP) map. This mapping operation (ϕ) may also be seen as a doubly stochastic channel since it adheres to the following properties [124]:

1. $\phi(\mathbb{I}) = \mathbb{I}$, where $\mathbb{I}$ is the identity matrix
2. $\mathrm{Tr}(\phi(\rho)) = \mathrm{Tr}(\rho)$ for all ρ

The mixed-unitary channels are a subset of the doubly stochastic channels. Mixed-unitary channels are those channels which can be described by a collection of unitary operators $U_1, ..., U_N \in U(X, Y)$ and a probability distribution $(\nu_1, ..., \nu_N)$. Here $U(X, Y)$ represents the set of unitary mappings from X to Y. The following equation may represent them [125]:

$$\phi(\rho) = \sum_{j=1}^{\mathrm{N}} \nu_j U_j \rho U_j^\dagger \tag{32}$$

In this equation ϕ represents the channel, ρ is a density matrix, $U_1, ..., U_N$ are again unitary operators and $(\nu_1, ..., \nu_N)$ represent a probability distribution. The expression in Eq. 32 is similar to the quantum channel in Eq. 30. If we use the Kraus operator representation, it is important to note that the channel does not uniquely determine the Kraus operators. A quantum operation can be represented by the Kraus operators up to a unitary transformation, $E_i = \sum_j \nu_{ji} U_j$, where ν_{ji} are the elements of a unitary transformation from $E_i \to U_j$ which and $\nu_{ji}^2 = \nu_j$. Thus it is possible to construct a description of the quantum channel equivalent to the Kraus operator formalism which is based on permutation matrices $(P_1, ..., P_{\tilde{\mathrm{N}}})$ and a corresponding probability distribution $(\theta_1, ..., \theta_{\tilde{\mathrm{N}}})$ as follows:

$$\phi(\rho) = \sum_{i=1}^{\tilde{\mathrm{N}}} \theta_i P_i \rho P_i^\dagger \tag{33}$$

Here $\tilde{\mathrm{N}}$ is introduced to show that Eq. 32 and Eq. 33 are different. The reason the description provided above is important will be discussed in Sec. 4.3.3

4.3.2 Adapted operator-sum representation

It can be checked that the representation presented in Eq. 33 adheres to the requirement of Eq. 29 as well as the requirements for a doubly stochastic channel. Since $(\theta_1, ..., \theta_N)$ represents a probability distribution we may note that:

$$\sum_{i=1}^{N} \theta_i = 1 \tag{34}$$

$$0 \leq \theta_i \leq 1 \tag{35}$$

reducing Eq. 33 to,

$$\phi(\rho) = \sum_{i=1}^{N} P_i \rho P_i^\dagger \tag{36}$$

In the case that $\rho = \mathbb{I}$ we find:

$$\phi(\mathbb{I}) = \mathbb{I} \sum_{i=1}^{N} P_i P_i^\dagger = \mathbb{I} \tag{37}$$

Where it is easy to check that $P_i P_i^\dagger = P_i^\dagger P_i = \mathbb{I}$ which means that we adhere to Eq. 29 and also meet the requirements for a doubly stochastic channel, since in our case both ρ_{CG} and ρ_{IG} have trace equal to one by definition. The rewriting of these equations leaves us with a quantum channel which may be described by a finite set of permutation matrices P_i with corresponding probability distribution $(\theta_1, ..., \theta_N)$. The set of permutation matrices is obtained by permitting the rows of an $n \times n$ identity matrix according to some permutation of the numbers 1 to n. Every row and column contains precisely a single 1 and is 0 everywhere else. Every permutation of the identity matrix corresponds to a unique permutation matrix. This will limit the computational requirements for optimisation with respect to the objective function proposed in Eq. 31 since it is possible to determine the θ_i values. The θ_i values are a finite set of independent variables representing the probability that a corresponding permutation matrix will reduce the $\mathcal{D}_{\text{QJS}}$. As it currently stands the number of independent variables will be limited to $n!$, where n represents the number of vertices of the graph under consideration. This limitation is because for a $n \times n$ identity matrix there are $n!$ possible permutation matrices, each with a corresponding θ value.

Nonetheless, further simplification may be done to ensure a connection between the SWAP operations which can be performed on a coupling graph and the formulation of the quantum channel. This may further reduce the computational requirements to allow for the optimisation of required SWAP operations for a given CG/IG-pairing, where the initial placement has been provided. The additional requirements will be introduced in the following section.

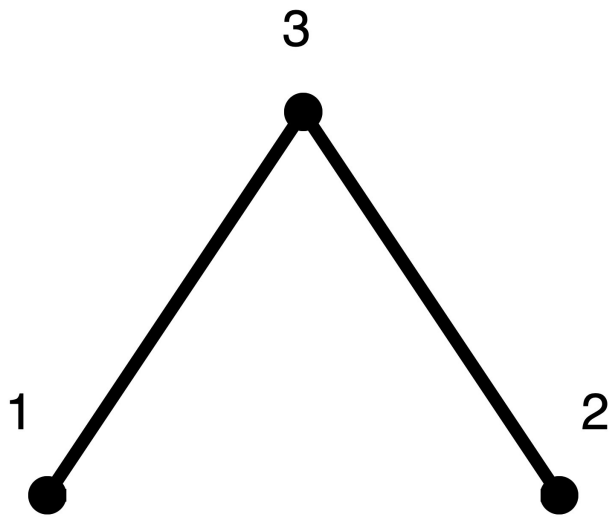


Figure 11: Example graph with 3 vertices and 2 edges.

4.3.3 Permutations based on allowed SWAP operations

As mentioned above the number of permutation matrices will grow as $n!$ in which n represents the rank of the density matrix. In our case, the rank of the density matrix is equivalent to the number of vertices in the original graph representation of either the quantum algorithm or quantum architecture. The number of independent variables we would like to optimise for will also scale as $n!$. Therefore, we will introduce an additional constraint limiting the number of permutation matrices we consider.

To properly introduce this constraint we need to consider a simple coupling graph, which is displayed in Fig. 11. Next, we may consider all the possible permutation matrices for $n = 3$, these are the following:

$$P_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}; P_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}; P_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}; P_5 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}; P_6 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{38}$$

The action of two permutation matrices on the example graph is shown in Fig. 12.

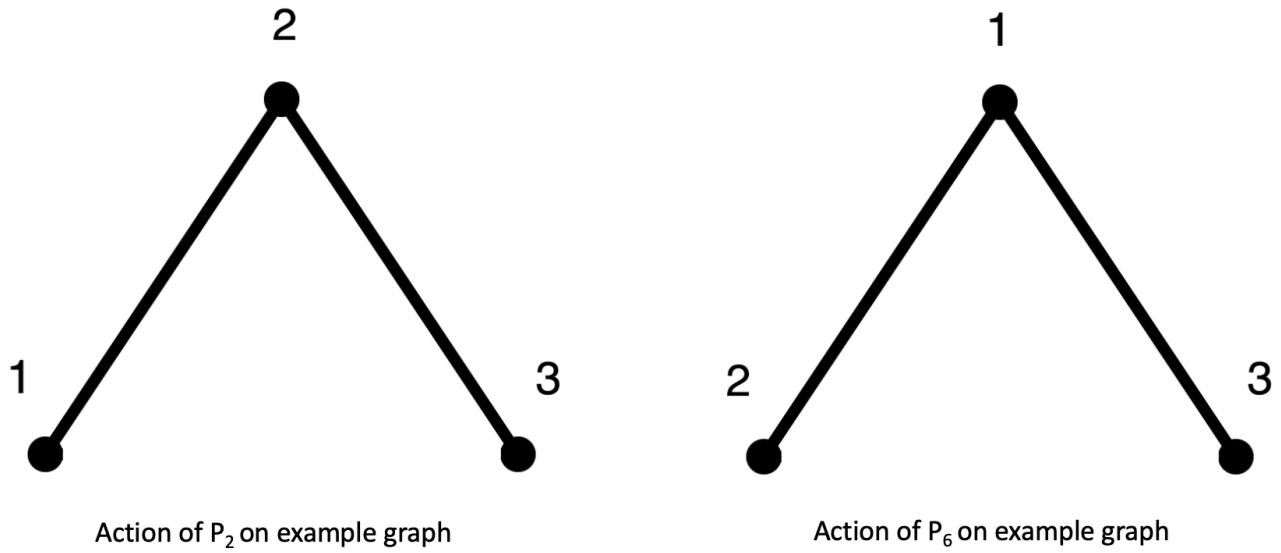


Figure 12: Action of permutation matrix P_2 and P_6 on the example graph.

We consider Fig. 11 and notice that the only SWAP operation which may be possible are those between vertices which are connected by edges in the graph. In the case of the example graph of Fig. 11 these will be $E(G) = \{\{2,3\},\{1,3\}\}$. These edges show that it is only possible to SWAP qubits 2 and 3 or qubits 1 and 3, this corresponds to the action of permutation matrices P_2 and P_3. In Fig. 12 we can see that the action of P_6 does not represent a permutation corresponding to applying a SWAP operation. All other permutation matrices also do not represent allowed SWAP operations. Now we may ask if our current formulation for the quantum channel could be adapted to only represent SWAP operations allowed on the CG for a given initial mapping. To gain a proper understanding we may consider how the graph Laplacian will change if a SWAP operation is performed. Consider the following graph Laplacian of the graph in Fig. 11:

$$L_{CG,1} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 2 \end{bmatrix} \tag{39}$$

Now analyse the case in which the SWAP operation between qubit 2 and qubit 3 has been performed. The resulting graph is shown in Fig. 13, where the red arrow indicates the qubits which have been exchanged. Considering the graph Laplacian of the new graph we find:

$$L_{CG,2} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \tag{40}$$

The result given in Eq. 40 can also be obtained by considering Kraus operator E_i to be equal to the permutation matrix P_2, this results in the following:

$$E_i = E_i^\dagger = P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{41}$$

We may now consider the operation which these Kraus operators represent when applied to the graph Laplacian. To that end the Kraus operators are applied to $L_{CG,1}$ resulting in:

$$\phi(L_{CG,1}) = E_i L_{CG,1} E_i^\dagger = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} = L_{CG,2} \tag{42}$$

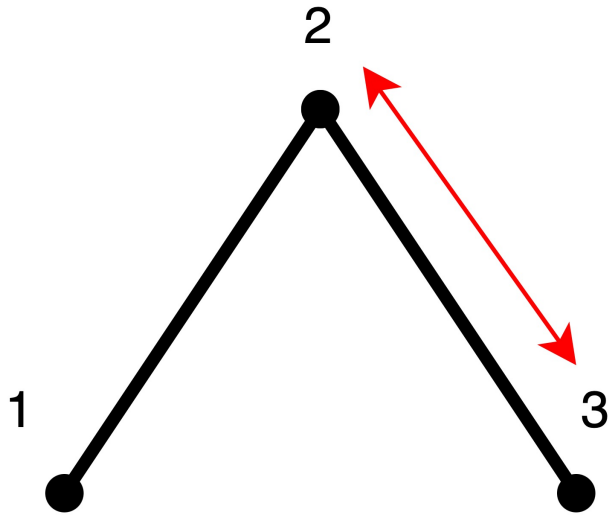


Figure 13: Example graph with 3 vertices and 2 edges, on which a SWAP has been performed.

This clearly shows that SWAP operations on a CG may be described as a quantum channel in which the Kraus operators are the permutation matrices which perform allowed SWAP interactions. In the case of the current example, the initial allowed SWAP operations are determined by considering the vertices in the graph, which are connected by edges. This vertex pair must be in the edge set $E(G)$. For the CG in Fig. 11, the allowed SWAP operations are between the vertex pairs $\{\{2,3\},\{1,3\}\}$ and are represented by the following permutation matrices:

$$P_{\{2,3\}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}; P_{\{1,3\}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{43}$$

In this case $P_{\{2,3\}}$ and $P_{\{1,3\}}$ are equivalent to P_2 and P_3 respectively. The change in notation is adopted to highlight the relationship between permutation matrices and the SWAP operation they represent. The following equation can generalise the permutation matrices which represent allowed SWAP:

$$P_{\{i,j\}} = \begin{cases} p_{\{i,j\}} = p_{\{j,i\}} = 1 & \\ p_{\{x,x\}} = 1 & \forall x \neq i,j \\ \text{All other elements } 0 & \end{cases} \tag{44}$$

Here $P_{\{i,j\}}$ represents the permutation matrix which performs the SWAP operation exchanging qubit i and j and $p_{\{i,j\}}$ indicated the matrix elements of the permutation matrix.

Implementing the constraints discussed above to minimise the $\mathcal{D}_{\text{QJS}}$ as stated in Eq. 31. The additional constraints lead to the following representation of the quantum channel from the CG to the IG:

$$\phi(\rho_{\text{CG}}) = \sum_k E_k \rho_{\text{CG}} E_k^\dagger = \sum_k^M \theta_k P_k \rho_{\text{CG}} P_k \tag{45}$$

Where M is the number of SWAP operations allowed by the coupling graph, moreover, P_k are part of the set $P_{\{i,j\}}$ which consists of the permutation matrices which represent allowed SWAP operations. Notice that P_i is not the same set as P_k, the first indicates the set of all permutation matrices and the latter only those permutations representing allowed SWAP operations. This set of permutation matrices is determined by the edge set $E(G)$ of the coupling graph. The number of independent variables in the objective function is now limited to the number of allowed SWAP operations. In the case of the example CG in Fig. 11 there will be two independent variables, $\theta_{\{1,3\}}$ and $\theta_{\{2,3\}}$. These variables indicate the influence of an allowed SWAP operation on the value of the $\mathcal{D}_{\text{QJS}}$ metric. Minimising this metric with respect to the independent variables will give us an insight into which SWAP operations will reduce the difference between CG and IG for a specific mapping. To determine all SWAP operations required for a given CG/IG-pairing, which SWAP will be required based on the provided initial mapping is first determined. This SWAP operation will then be performed and the resulting graphs will again be compared. This process is repeated until all the required operations described by the IG

can be performed on the CG. This iterative approach will be discussed further in Sec. 4.4.3.

4.4 Algorithmic adaptation

All the required preliminaries and constraints for the algorithm have been discussed in sections 4.1, 4.2 and 4.3. Thus, we can now describe the algorithm that will determine the minimal bound on the number of required SWAP operations based on an initial mapping for a given CG/IG-pairing. The algorithm utilises the SciPy SLSQP-optimiser from the SciPy optimisation package [126]. The following section will provide an outline for the proposed algorithm. The optimiser choice will be discussed briefly and lastly, the iterative approach will be introduced for which an example will be given.

4.4.1 Outline Algorithm

As input the algorithm requires the adjacency matrix of the CG and IG. The adjacency matrices are then converted to their corresponding graph Laplacian. The adjacency matrix of the CG graph is also utilised to determine the permutation matrices representing the SWAP operations allowed for a given mapping. Considering the adjacency matrix of the graph in Fig. 11, we see that the adjacency matrix is:

$$A_{\mathrm{CG}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{46}$$

Since the elements in the adjacency matrix are symmetric when mirrored in the diagonal, it is only necessary to consider those elements above the diagonal. The index of the non-zero entries is added to the set of allowed SWAP operations utilised for constructing the corresponding permutation matrices. In the example, the indices which would be added are $\{1,3\}$ and $\{2,3\}$

Furthermore, the functions have been implemented to execute the equations discussed in Sec. 4.2. This enables us to represent the CG and IG as density matrices. Moreover, the relevant tools from quantum information theory can be implemented to compare the density matrices of the CG and IG. As such the objective function can now be defined:

$$\phi^* = \arg\min_{\phi} \Big[\mathcal{D}_{\mathrm{QJS}}(\phi(\rho_{\mathrm{CG}})||\rho_{\mathrm{IG}})\Big] = \arg\min_{\phi} \Big[\mathcal{D}_{\mathrm{QJS}}\big(\sum_{k}^{M} \theta_k P_k \rho_{\mathrm{CG}} P_k||\rho_{\mathrm{IG}}\big)\Big]. \tag{47}$$

Here ρ_{CG} and ρ_{IG} are the density matrices of CG and IG. P_k represent the permutation matrices constructed according to the procedure discussed in Sec. 4.3.3. Moreover, θ_k represents the independent variables which should minimise the argument in Eq. 47. M is the number of allow SWAP operations and lastly $\mathcal{D}_{\mathrm{QJS}}$ represents the quantum Jensen-Shannon divergence.

The CG is altered based on the highest θ_i value after the optimisation. The SWAP operation corresponding to the highest θ_i value will be performed on the CG. If equal θ values are encountered a random SWAP operation corresponding to these θ values is performed. The IG is also altered. All possible operations based on the initial mapping provided for the CG are removed from the edge set $E(IG)$ of the IG. The optimisation is repeated until $|E(IG)| = 0$, note that by definition. Since in the case that the IG does not have any remaining edges, there are no more operations which need to be performed on the underlying architecture and the algorithm is terminated. The number of iterations the algorithm requires before terminating represents the minimal bound on the required number of SWAP operations for a given CG/IG-pairing with specified initial mapping. This is because the algorithm determines the SWAP operations, reducing the difference between the CG and IG in every iteration. Resulting in those SWAP operations which allow the remaining operations specified by the IG to be performed. Once all these operations are possible the algorithm terminates.

4.4.2 Optimisation method

For the optimisation of the objective function the scipy.optimize.minimize package has been used. The Sequential Least Squares Programming (SLSQP)optimisation method has been chosen due to its ability to deal with constraints effectively [126]. The constraints provided to the optimiser are that all independent variables should sum to one, $\sum_i^M \theta_i = 1$, and that all θ_i are bounded between 0 and 1, $0 \leq \theta_i \leq 1$.

The starting values for optimising the independent variables in the objective function have been randomly generated. This has been done to ensure that the optimisation results are independent of the initial values of the independent variables.

4.4.3 Example iterative approach

To conclude this section an example of the iterative approach will be presented. Further details for this example will be provided in Sec. 5.1. We consider the example graph from Fig. 9 to be the IG and the example graph from 11 to be the CG. We define the mapping as presented in Fig. 14:

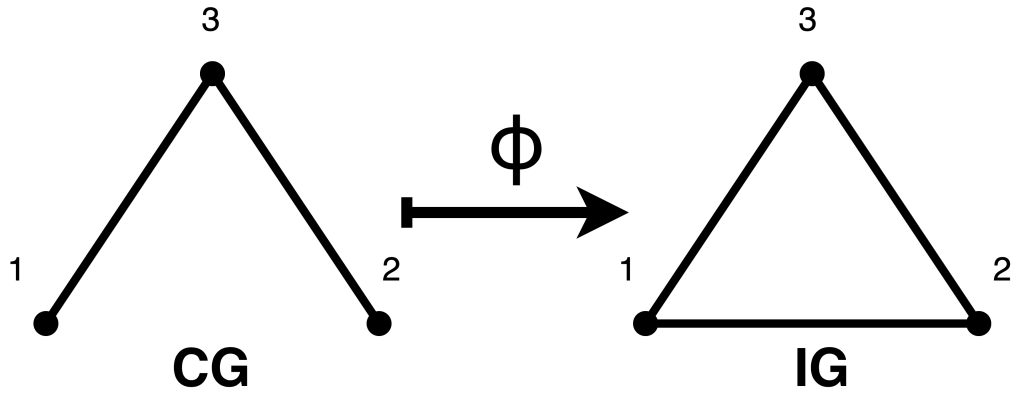


Figure 14: A 3-Vertex Toy-Model, which provides the initial CG/IG-pairing.

Considering this figure, the operations between the pairs $\{2,3\}$ and $\{1,3\}$ can be performed directly for the presented initial mapping. As such they may be removed from the IG. It is also quickly verified that the operation between the qubit pair $\{1,2\}$ will require a SWAP operation. It does not matter which qubit pairs in the CG are exchanged. Both SWAP operations will result in an adapted CG allowing the final operation of the IG between qubit pair $\{1,2\}$ to be executed. Thus the SWAP operation presented in Fig. **??** will be implemented. This results in a new input IG and CG for the algorithm this process is illustrated in Fig. **??**.

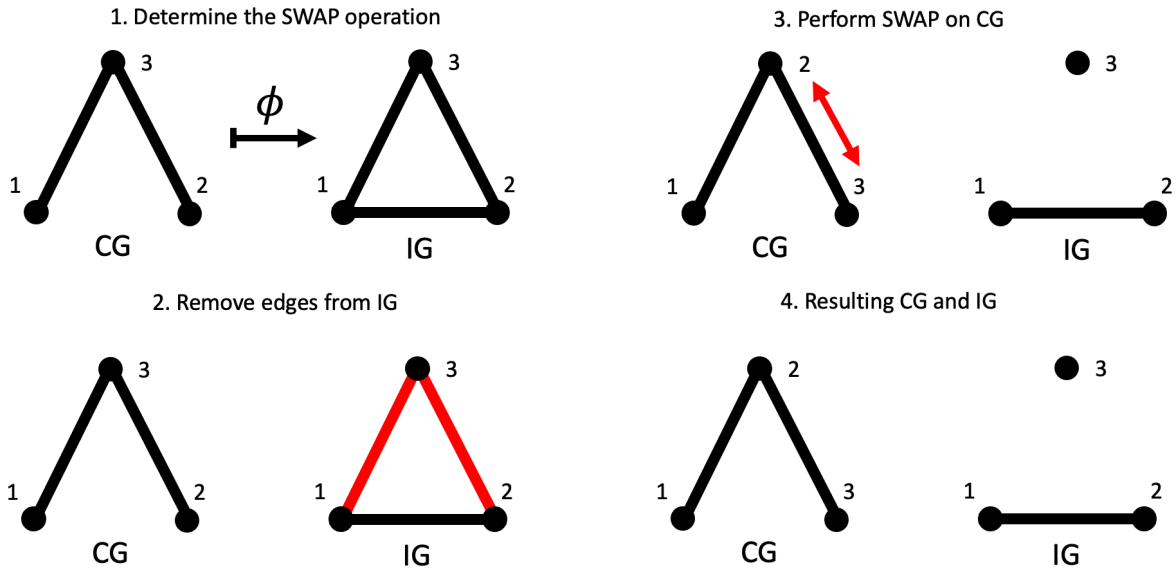


Figure 15: 3-Vertex Toy-Model, in which the first iteration of the algorithm has been performed.

It can be seen in Fig. 19 that the last remaining operation in the IG between qubit pair $\{1,2\}$ can now be performed on the CG. Thus the last edge of the IG may be removed, and the algorithm will terminate. The iteration that results in the termination of the algorithm is illustrated in Fig. 16. We will call the iteration which results in an empty IG the termination iteration. Since only a single iteration of the algorithm was

required to get to the termination iteration, the minimal number of SWAP operations equals 1 for the given CG/IG-pairing with specified initial mapping.

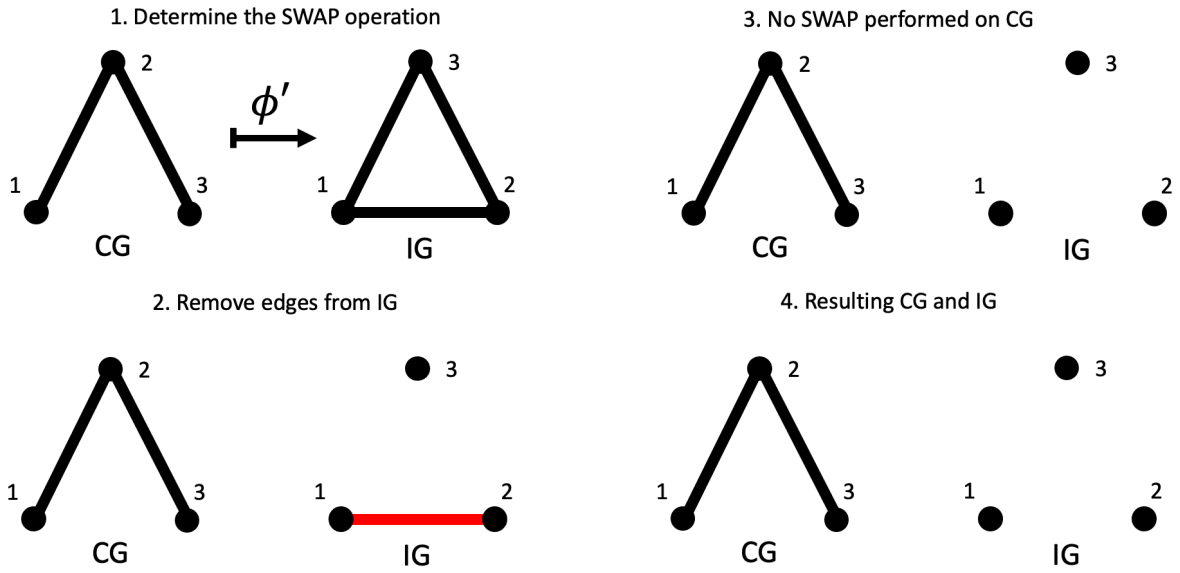


Figure 16: 3-Vertex Toy-Model, in which the termination iteration is obtained.

Algorithm 1 Calculate Number of Required SWAP Operations

Require: Adjacency matrix interaction graph (A_{IG}), adjacency matrix coupling graph (A_{CG}), Initial Placement

Ensure: Determine the number of required SWAP Operations

$L_{CG} \leftarrow A_{CG}$ ▷ Compute graph Laplacian of CG
$L_{IG} \leftarrow A_{IG}$ ▷ Compute graph Laplacian of IG
$\rho_{CG} \leftarrow L_{CG}$ ▷ Compute density matrix ρ_{CG}
$\rho_{IG} \leftarrow L_{IG}$ ▷ Compute density matrix ρ_{IG}
SWAP# $\leftarrow 0$
while $E(IG) \neq \varnothing$ **do**
  $D_{QJS} \leftarrow P_k$, ρ_{CG} and ρ_{IG} ▷ Determines objective function
  $P_k \leftarrow A_{CG}$ ▷ Determine permutation matrices
  $\min_{\theta_i}(D_{QJS})$ ▷ Minimises D_{QJS} with respect to θ_i
  $E(IG) \leftarrow E(IG) - E(CG)$ ▷ Removes edges that can be executed
  $A_{IG} \leftarrow E(IG)$ ▷ Determines the new A_{IG}
  $L_{IG} \leftarrow A(IG)$ ▷ Determines the new L_{IG}
  $\rho_{IG} \leftarrow L(IG)$ ▷ Determines the new ρ_{IG}
  $\rho_{CG} \leftarrow P_i \rho_{CG} P_i$ ▷ Performs SWAP on ρ_{CG}
  SWAP# $\leftarrow$ SWAP# $+ 1$
end while
Output: SWAP#

To give an insight into the structure of the code, pseudo-code is presented above. In this pseudo code, P_k are the permutation matrices corresponding to the SWAP operations the coupling graph allows, and P_i is the permutation matrix corresponding to θ_i, which is determined by the optimisation. SWAP# indicates the number of required SWAP operations. $E(IG) \neq \varnothing$ indicates that the edge set of the interaction graph must be empty.

5 Evaluation

The upcoming section will evaluate the algorithm, which has been implemented in Python. To test the algorithm, several 'toy models' have been evaluated. A 'toy model' is an example of a simple CG/IG-pairing where the minimum bound of the number of SWAP operations can be determined manually. Studying these CG/IG-pairing is relevant to ensure the algorithm performs as expected. After confirming that the algorithm works as expected, a benchmark dataset of numerous CG/IG-pairings will be utilised to evaluate its potential limitations.

5.1 Toy Models

In this section, several 'toy models' will be evaluated. The initial 2- and 3-vertex toy models are trivial since the operation is either not possible or it is easily seen what the required number of SWAP operations should be. Nonetheless, they provide a proof of concept for the algorithm's performance since the expected behaviour can easily be determined, thus it is possible to verify the algorithm's results. The toy model of a 4-vertex graph has been provided to complement the discussion of the β value since it is possible to compare the results of the algorithm for different β values with the plot provided in Fig. 10. The 6-vertex toy model has been provided to illustrate that the β value influences the result of the algorithm.

2-Vertex Model

It can be seen that the 2-vertex model in Fig. 17 is a trivial example since the CG does not have an edge connecting the qubits. For that reason, the operation described by the IG can never be performed. The example is of interest because it is possible to find an analytic solution to both the mapping as well as the inverse mapping (IG $\mapsto$ CG). This analytic solution is provided to illustrate the mathematical steps which the algorithm will implement.

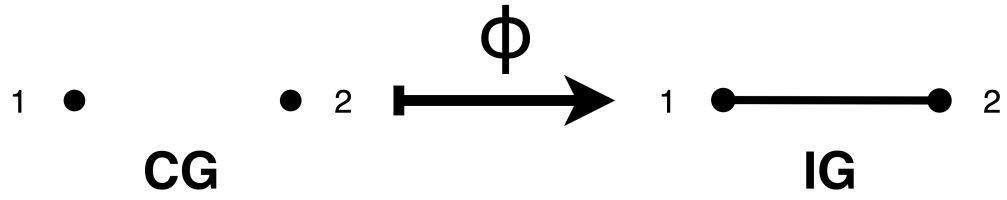


Figure 17: 2 Vertex Toy-Model, not possible to map CG to IG.

Considering these graphs we can determine the graph Laplacian for the IG and CG according to Eq. 16 and we find:

$$L_{IG} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}; L_{CG} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{48}$$

Next, we want to determine the density matrix as described in Eq. 19. First, we consider the ρ_{CG} by plugging L_{CG} into Eq. 19 this results in the following:

$$\rho_{CG} = \frac{e^{(-\beta L_{CG})}}{tr(e^{(-\beta L_{CG})})} = \frac{e^{(-\beta \bar{0})}}{tr(e^{(-\beta \bar{0})})} = \frac{(-\beta \mathbb{I})}{tr(-\beta \mathbb{I})} = \frac{\beta \mathbb{I}}{2\beta} = \frac{\mathbb{I}}{2} \tag{49}$$

Here $\bar{0}$ indicated the null matrix, which only contains zeros. Considering ρ_{IG} the solution is not as straightforward as the previous example. Plugging L_{IG} into Eq. 19 we find:

$$\rho_{IG} = \frac{e^{(-\beta L_{IG})}}{tr(e^{(-\beta L_{IG})})} \tag{50}$$

In the previous case finding the exponential of the matrix was rather straightforward since $e^{\mathbf{0}} = \mathbb{I}$ to determine the exponential $e^{-\beta L_{IG}}$ we will use the Taylor series of the exponential:

$$e^{A} = \sum_{n=0}^{\infty} \frac{A^n}{n!} \tag{51}$$

This gives us the following expression:

$$e^{-\beta L_{IG}} = \sum_{n=0}^{\infty} \frac{(-\beta L_{IG})^n}{n!} \tag{52}$$

We will derive the expression up to the fourth order. This order has been selected to limit the required calculations, while still offering an understanding of the procedure. The initial calculations are determining the matrix powers we will need. Thus the aim is to gain the expression for $(-\beta L_{IG})^4$

$$(-\beta L_{IG}) = \begin{bmatrix} -\beta & \beta \\ \beta & -\beta \end{bmatrix} \tag{53}$$

$$(-\beta L_{IG})^2 = \begin{bmatrix} -\beta & \beta \\ \beta & -\beta \end{bmatrix} \begin{bmatrix} -\beta & \beta \\ \beta & -\beta \end{bmatrix} = \begin{bmatrix} 2\beta^2 & -2\beta^2 \\ -2\beta^2 & 2\beta^2 \end{bmatrix} \tag{54}$$

$$(-\beta L_{IG})^3 = \begin{bmatrix} 2\beta^2 & -2\beta^2 \\ -2\beta^2 & 2\beta^2 \end{bmatrix} \begin{bmatrix} -\beta & \beta \\ \beta & -\beta \end{bmatrix} = \begin{bmatrix} -4\beta^3 & 4\beta^3 \\ 4\beta^3 & -4\beta^3 \end{bmatrix} \tag{55}$$

$$(-\beta L_{IG})^4 = \begin{bmatrix} -4\beta^3 & 4\beta^3 \\ 4\beta^3 & -4\beta^3 \end{bmatrix} \begin{bmatrix} -\beta & \beta \\ \beta & -\beta \end{bmatrix} = \begin{bmatrix} 8\beta^4 & -8\beta^4 \\ -8\beta^4 & 8\beta^4 \end{bmatrix} \tag{56}$$

putting this all together we find:

$$\begin{aligned} e^{-\beta L_{IG}} &= \mathbb{I} + \begin{bmatrix} -\beta & \beta \\ \beta & -\beta \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 2\beta^2 & -2\beta^2 \\ -2\beta^2 & 2\beta^2 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} -4\beta^3 & 4\beta^3 \\ 4\beta^3 & -4\beta^3 \end{bmatrix} + \frac{1}{24}\begin{bmatrix} 8\beta^4 & -8\beta^4 \\ -8\beta^4 & 8\beta^4 \end{bmatrix} \\ &= \begin{bmatrix} 1-\beta+\beta^2-\frac{2}{3}\beta^3+\frac{1}{3}\beta^4 & 1+\beta-\beta^2+\frac{2}{3}\beta^3-\frac{1}{3}\beta^4 \\ 1+\beta-\beta^2+\frac{2}{3}\beta^3-\frac{1}{3}\beta^4 & 1-\beta+\beta^2-\frac{2}{3}\beta^3+\frac{1}{3}\beta^4 \end{bmatrix} \end{aligned} \tag{57}$$

The expression has not been normalised. To normalise, we will need to divide it by the trace. First, we need to determine the straightforward trace.

$$tr(e^{-\beta L_{IG}}) = 2 - 2\beta + 2\beta^2 - \frac{4}{3}\beta^3 + \frac{2}{3}\beta^4 \tag{58}$$

The final expression for ρ_{IG} is then

$$\rho_{IG} = \frac{e^{-\beta L_{IG}}}{tr(e^{-\beta(L_{IG})}} = (2-2\beta+2\beta^2-\frac{4}{3}\beta^3+\frac{2}{3}\beta^4)^{-1}\begin{bmatrix} 1-\beta+\beta^2-\frac{2}{3}\beta^3+\frac{1}{3}\beta^4 & 1+\beta-\beta^2+\frac{2}{3}\beta^3-\frac{1}{3}\beta^4 \\ 1+\beta-\beta^2+\frac{2}{3}\beta^3-\frac{1}{3}\beta^4 & 1-\beta+\beta^2-\frac{2}{3}\beta^3+\frac{1}{3}\beta^4 \end{bmatrix} \tag{59}$$

This is not a clear expression, and for that reason, we chose to write it as:

$$\rho_{IG} = \frac{1}{2}\begin{bmatrix} 1 & \delta \\ \delta & 1 \end{bmatrix} \tag{60}$$

where

$$\delta = \frac{1+\beta-\beta^2+\frac{2}{3}\beta^3-\frac{1}{3}\beta^4}{1-\beta+\beta^2-\frac{2}{3}\beta^3+\frac{1}{3}\beta^4} \tag{61}$$

Considering this expression we recognise that the terms are similar to the Taylor expansion of the exponential function:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + ... \tag{62}$$

However in the case of the expression in Eq. 61, we notice that the sign switches and we have different coefficients. Thus the term in the exponential must contain a constant and a minus sign. It can easily be shown that the expression in Eq. 61 may be written as:

$$\rho_{IG} = \frac{1}{2}\begin{bmatrix} \frac{1+e^{-2\beta}}{1+e^{-2\beta}} & \frac{1-e^{-2\beta}}{1+e^{-2\beta}} \\ \frac{1-e^{-2\beta}}{1+e^{-2\beta}} & \frac{1+e^{-2\beta}}{1+e^{-2\beta}} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & \frac{-1+e^{2\beta}}{(1+e^{2\beta})} \\ \frac{-1+e^{2\beta}}{(1+e^{2\beta})} & 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix} \tag{63}$$

In the second to last step, the numerator and the denominator are multiplied by $e^{2\beta}$. In the last step we introduce $\alpha = \frac{-1+e^{2\beta}}{(1+e^{2\beta})}$ to make the notation more compact. Now we have all the information to consider the

channel between the CG and IG as defined in Eq. 30, this gives us the following equation:

$$\phi(\rho_{\text{CG}}) = \frac{1}{2}\sum_i E_i \mathbb{I} E_i^\dagger = \frac{1}{2}\begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix} \tag{64}$$

Which reduces to:

$$\sum_i E_i E_i^\dagger = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix} \tag{65}$$

Notice that we have an additional requirement from Eq. 29, in order to compare we need to note that $(\sum_i E_i E_i^\dagger)^\dagger = \sum_i E_i^\dagger E_i$, $\rho_{IG}^\dagger = \rho_{IG}$ and $\mathbb{I}^\dagger = \mathbb{I}$. Thus combining the equations we find:

$$\begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix} = \sum_i E_i E_i^\dagger = \sum_i E_i^\dagger E_i = \mathbb{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{66}$$

From this equation it becomes apparent that $\alpha = 0$, considering the expression for α from Eq. 63 we obtain the following:

$$\alpha = \frac{-1 + e^{2\beta}}{(1 + e^{2\beta})} = 0 \tag{67}$$

$$-1 + e^{2\beta} = 0 \tag{68}$$

$$e^{2\beta} = 1 \tag{69}$$

$$2\beta = 0 \tag{70}$$

$$\beta = 0 \tag{71}$$

The solution $\beta = 0$ is a trivial result since for this value of β all density matrices will have the same entropy by definition. This has been mentioned in Sec. 4.2.3. Because we are dealing with a system in which $E(\text{CG}) = \{\varnothing\}$ whereas $E(\text{IG}) = \{\{1, 2\}\}$. The algorithm requires that the edges are removed from the set $E(\text{IG})$ based on the edges present in $E(\text{CG})$. Since in this case, the edge set $E(\text{CG})$ is empty, it will not be possible to perform the algorithm to remove edges from the set $E(\text{IG})$. In addition, it is also impossible to adapt the CG since there are no connected vertices, which means that it is impossible to perform a SWAP operation. The algorithm cannot be executed because it is not possible to remove the edge from the IG and it is also not possible to modify the CG. As a result, the termination iteration can never be obtained. However, this is not a problem as this was a trivial example to illustrate the math.

3-Vertex Model

The example shown in Fig. 18 has already been described in Sec. 4.4.3, here, we will provide a more detailed solution for this model.

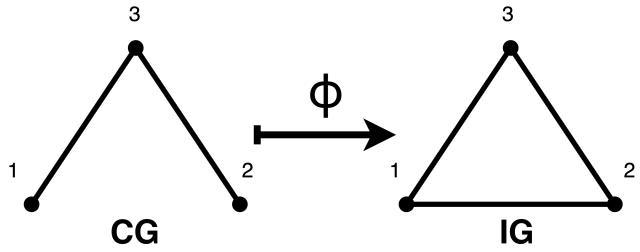


Figure 18: 3 Vertex Toy-Model, which provides the initial mapping of the CG/IG-pairing.

In the 3 vertex model, we find that the Graph Laplacian, L of the interaction graph and the coupling graph to be:

$$L_{IG} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}; L_{CG} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 2 \end{bmatrix} \tag{72}$$

Next, we use Eq. 19 to determine the density matrices of both the IG and the CG. These are found to be:

$$L_{IG} = \begin{bmatrix} 0.333333 & 0.330867 & 0.330867 \\ 0.330867 & 0.333333 & 0.330867 \\ 0.330867 & 0.330867 & 0.333333 \end{bmatrix}; L_{CG} = \begin{bmatrix} 0.35279412 & 0.23385088 & 0.29223324 \\ 0.23385088 & 0.35279412 & 0.29223324 \\ 0.29223324 & 0.29223324 & 0.29441176 \end{bmatrix} \tag{73}$$

Note that in the case of the 3-vertex model, the beta value has been set to 2. Moreover, the choice has been made to denote the numerical values because the exact analytic expressions, which were used in the 2-vertex model, would be too long.

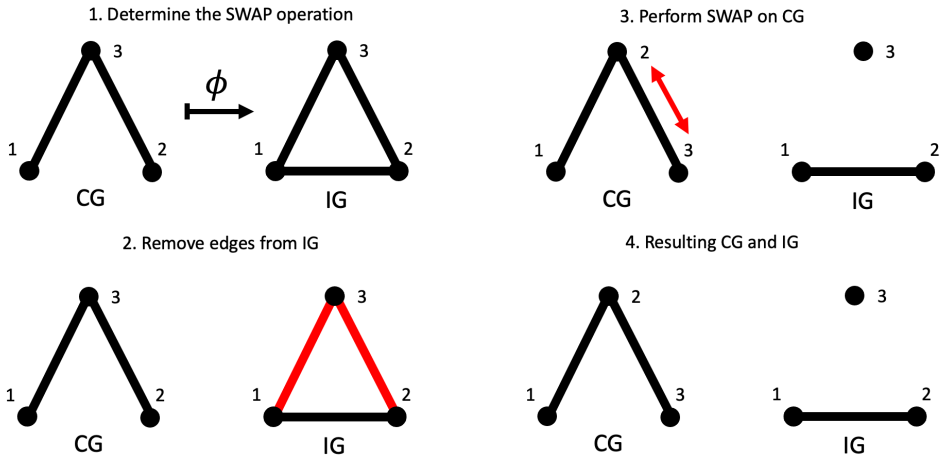


Figure 19: 3-Vertex Toy-Model, in which the first iteration of the algorithm has been performed.

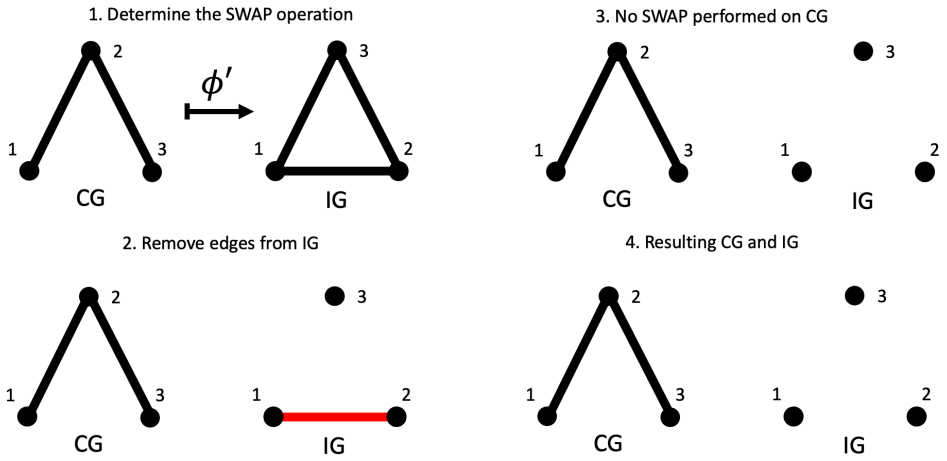


Figure 20: 3-Vertex Toy-Model, in which the termination iteration is obtained.

Using the algorithm to determine the required number of SWAP operations it has been found that for the CG/IG-pairing from Fig. 18, 1 SWAP operation will be required to allow all operations from the IG to be performed. This can be concluded since only one iteration is needed to get to the termination iteration.

4-Vertex Model

The toy model for 4-vertex graphs has been depicted in Fig. 21. These graphs are ρ_6 and ρ_8 from Fig. 10, where the entropy dependence on β has been plotted. We have again chosen $\beta = 2$. This gives the following density matrices:

$$\rho_{IG} = \begin{bmatrix} 0.25 & 0.2410069 & 0.23233729 & 0.2410069 \\ 0.2410069 & 0.25 & 0.2410069 & 0.23233729 \\ 0.23233729 & 0.2410069 & 0.25 & 0.2410069 \\ 0.2410069 & 0.23233729 & 0.2410069 & 0.25 \end{bmatrix} \tag{74}$$

$$L_{CG} = \begin{bmatrix} 0.29106595 & 0.09196759 & 0.1435615 & 0.2256934 \\ 0.09196759 & 0.29106595 & 0.2256934 & 0.1435615 \\ 0.1435615 & 0.2256934 & 0.20893405 & 0.1740995 \\ 0.2256934 & 0.1435615 & 0.1740995 & 0.20893405 \end{bmatrix} \tag{75}$$

Using the algorithm to determine the required number of SWAP operations it has been found that for the CG/IG-pairing from Fig. 21, 2 SWAP operations will be required to allow all operations from the IG to be performed. The SWAP operations and the resulting IG for all iterations are shown in Fig. 22, Fig. 23 and Fig. 24. Notice that the iteration has been depicted for one combination of SWAP operations which meets the minimal bound. In the case of this example, the routing is not unique. The SWAP operations do not have to be performed in a specific order. Interchanging them would not alter the result. Moreover, in the last step instead of performing a SWAP operation on the vertex pair $\{\{2,3\}\}$ it is also possible to SWAP the pair $\{\{1,3\}\}$ since both would result in the qubit 1 and 2 being connected in the final CG.

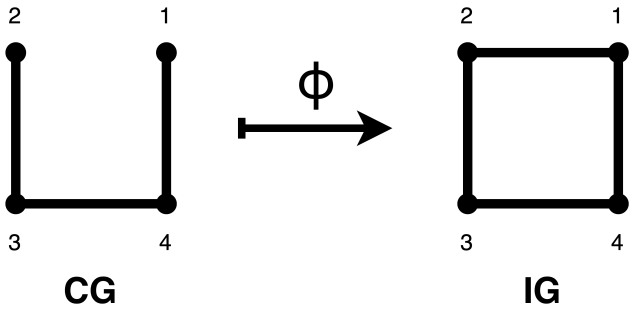


Figure 21: 4 Vertex Toy-Model, which provides the initial mapping of the CG/IG-pairing.

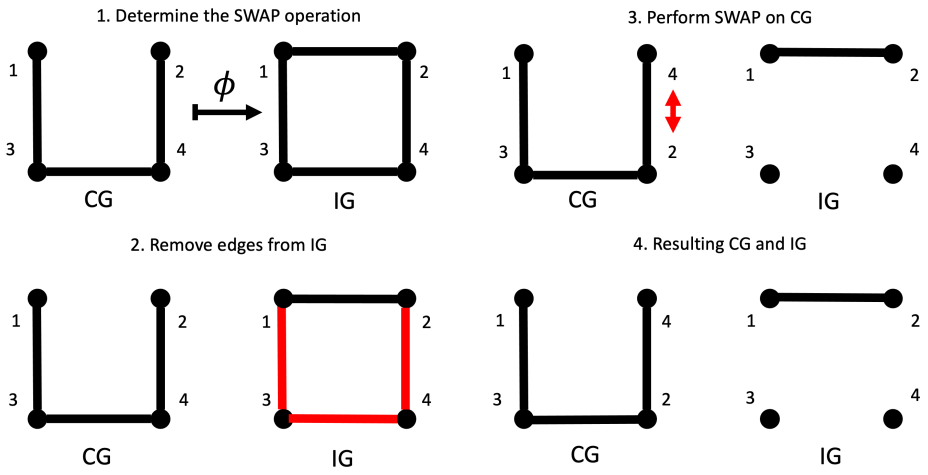


Figure 22: 4 Vertex Toy-Model, in which the first iteration of the algorithm has been performed.

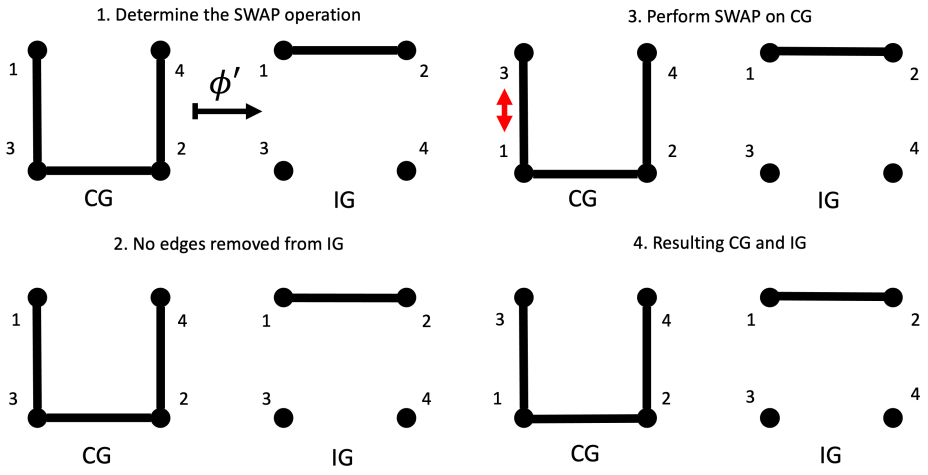


Figure 23: 4 Vertex Toy-Model, in which the second iteration of the algorithm has been performed.

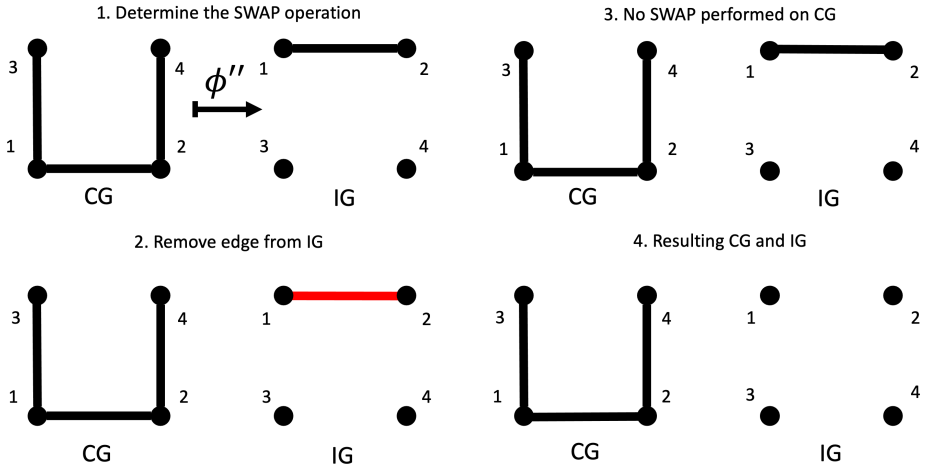


Figure 24: 4-Vertex Toy-Model, in which the termination iteration is obtained.

6-Vertex Model

The toy model for 6-vertex graphs has been depicted in Fig. 25. In Sec. 4.2.3 we noted that the influence of the β value has been observed. This is the first case in which our initial choice for the value of $\beta = 2$ did not provide a minimal number of SWAP operations. Nonetheless, it was possible to obtain the correct result by tweaking the value of β. Since the minimal number of SWAP operations can manually be determined for this example. Different β values were tested until the minimal number of SWAP operations was obtained. For this specific example, $\beta = 0.1$ was used. The other toy models have also been tested for this β value, confirming that they still resulted in minimal SWAP operations. This gives the following density matrices:

$$L_{IG} = \begin{bmatrix} 0.25 & 0.2410069 & 0.23233729 & 0.2410069 \\ 0.2410069 & 0.25 & 0.2410069 & 0.23233729 \\ 0.23233729 & 0.2410069 & 0.25 & 0.2410069 \\ 0.2410069 & 0.23233729 & 0.2410069 & 0.25 \end{bmatrix} \tag{76}$$

$$L_{CG} = \begin{bmatrix} 0.29106595 & 0.09196759 & 0.1435615 & 0.2256934 \\ 0.09196759 & 0.29106595 & 0.2256934 & 0.1435615 \\ 0.1435615 & 0.2256934 & 0.20893405 & 0.1740995 \\ 0.2256934 & 0.1435615 & 0.1740995 & 0.20893405 \end{bmatrix} \tag{77}$$

Using the algorithm to determine the required number of SWAP operations it has been found that for the CG/IG-pairing from Fig. 25, 2 SWAP operations will be required to allow all operations from the IG to be performed. The SWAP operations and the resulting IG for all iterations are shown in Fig. 26, Fig. 27 and Fig. 28. Notice that in this example the choice for which SWAP operation to perform is also not unique. In the iteration shown in Fig. 27 it would also be possible to perform a SWAP operation on the vertex pair $\{\{3,4\}\}$ instead of $\{\{1,3\}\}$ as this would also connect qubit 1 and 4.

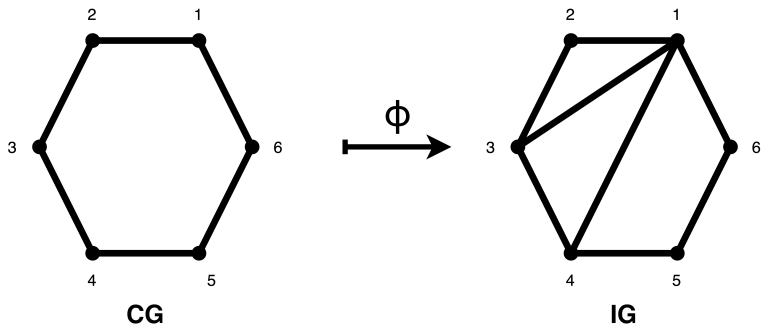


Figure 25: 6 Vertex Toy-Model, which provides the initial mapping of the CG/IG-pairing.

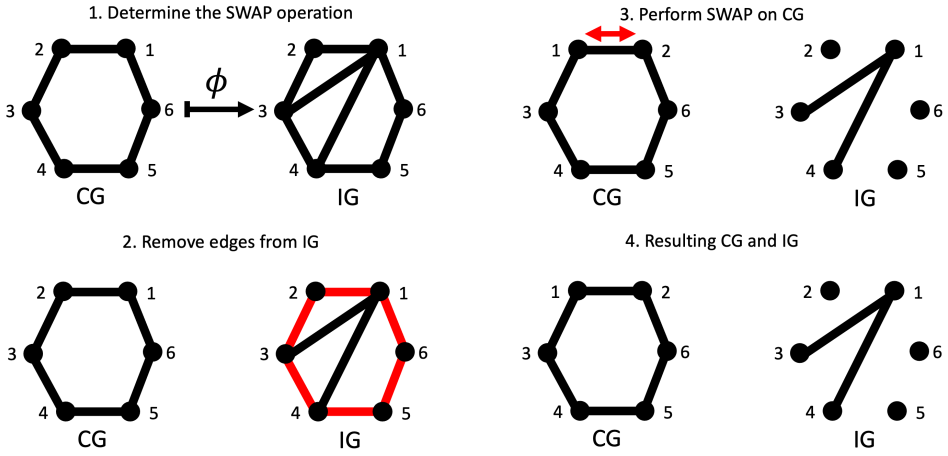


Figure 26: 6 Vertex Toy-Model, in which the first iteration of the algorithm has been performed.

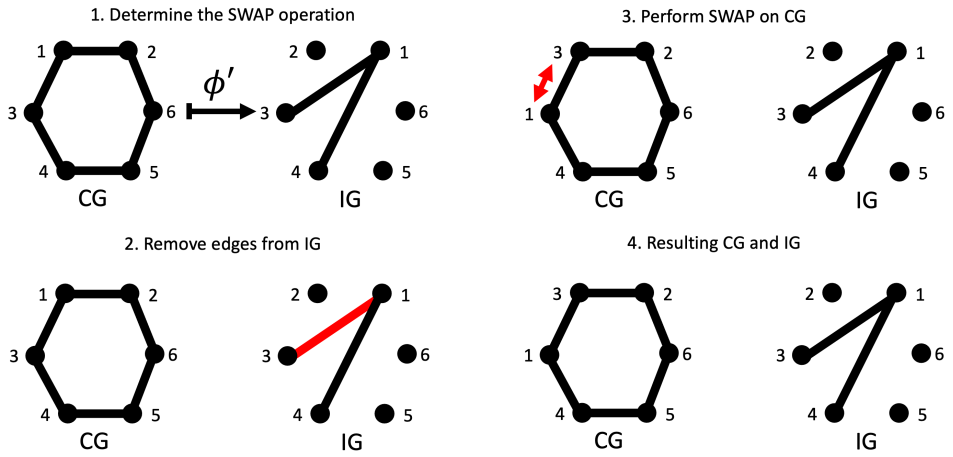


Figure 27: 6 Vertex Toy-Model, in which the second iteration of the algorithm has been performed.

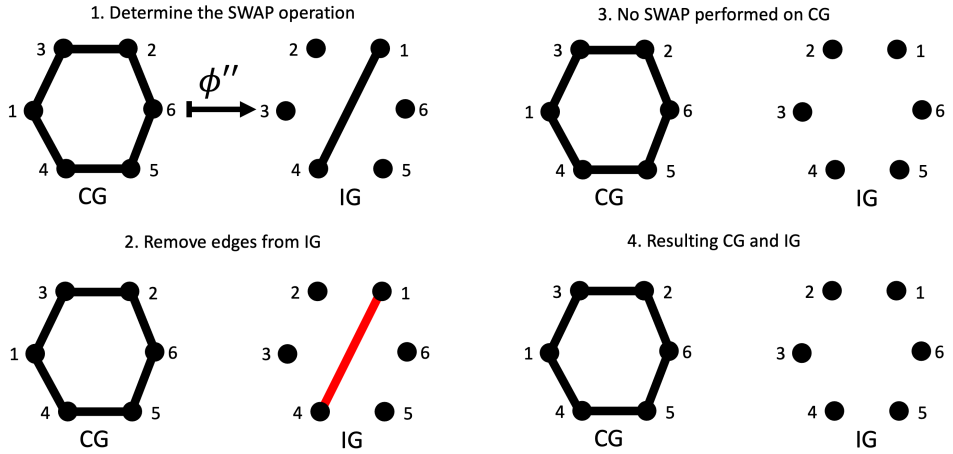


Figure 28: 6-Vertex Toy-Model, in which the termination iteration is obtained.

5.2 Benchmarks

This section will present the benchmarks which have been used to assess the effectiveness of the proposed algorithm. The algorithm is evaluated with CG/IG-pairings, constructed based on the benchmark quantum algorithms and NISQ devices. The quantum algorithms and NISQ devices which are used as benchmarks are presented in the tables below. Appendix B provides the number of SWAP operations achieved for each pairing. Additionally, it is essential to note that the beta value has been modified for all pairings. The β-value was adjusted to gain insight into the β-dependence on the number of SWAP operations. As mentioned in the previous section, the minimal number of SWAP operators obtained depends on the β-value. The β-dependence is discussed further in Sec. 6.2.

The benchmarks have been run on a MacBook Pro (2017) with a 2.3 GHz Dual-Core Intel Core i5, with 8 GB of 2133 MHz LPDDR3 ram. The initial mapping have been provided.

Benchmark name	**Number of logical qubits**
fredkin_n3, grover_n3	3
basis_change_n3, teleportation_n3	3
adder_n4	4
variational_n4, bell_n4	4
cuccaro adder 1b	4
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	4
vbe_adder_1b	4
4gt10-v1_81, q=5_s=2995_2qbf=09_1	5
4gt13_92	5
4gt5_75	5
alu-v1_28	5
alu-v2_31	5
decod24-v1_41	5
error_correctiond3_n5	5
qec_en_n5, qec_sm_n5	5
quantum_volume_n5	5
simon_n6	5
alu-v2_30,q=6_s=2994_2qbf=08_1	6
4gt12-v0_87	6
4gt4-v0_72	6
qaoa 6	6
ex3_229	6
graycode6_47	6
mod5adder_127	6
q=6_s=54_2qbf=022_1	6
sf_274	6
xor5_254	6
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	7

Table 1: This table presents the quantum algorithms used as a benchmark to evaluate the proposed approach.

NISQ device	Number of physical qubits
IBM Athens	5
Starmon-5	5
IBM Yorktown	5
IBM Ourense	5
Surface-7	7
IBM Casablanca	7
Rigetti Agave	8
IBM Melbourne	15
Rigetti Aspen-1	16
Surface-17	17
IBM Singapore	20
IBM Johannesburg	20
IBM Tokyo	20
IBM Paris	27
IBM Rochester	53
Google Bristlecone	72

Table 2: This table presents the NISQ device used as a benchmark to evaluate the proposed approach.

6 Discussion

6.1 Toy Models

The evaluation of the toy models has yielded successful results, demonstrating the algorithm's ability to determine the minimum number of SWAP operations for a given CG/IG-pairing. However, as the size of these pairings increased, it became apparent that the algorithm's results depended on the β-value. For this reason, a more detailed study was conducted by evaluating the algorithm with several benchmarks.

The toy models are specific examples which could have been solved manually. While these examples may not contribute to advancing the state of the art, they illustrate how the proposed approach determines the number of SWAP operations for a given pairing. The evaluation of the toy models underscores the proposed approach's potential to determine the minimal number of SWAP operations.

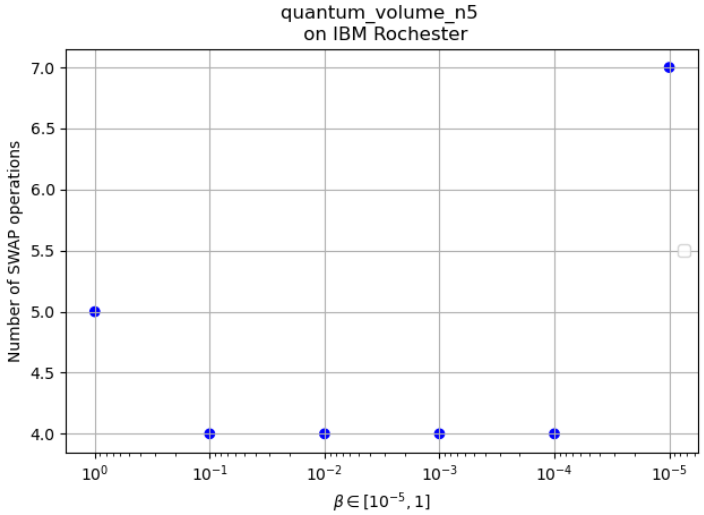
It is worth noting that the algorithm's limitations became evident when considering the 6-vertex model. In that case, the algorithm failed to determine the minimal bound unless the β-value was set correctly. This realisation prompted a more thorough investigation into the impact of the β-value during the benchmark evaluation.

The evaluation of the toy models showed the algorithm's capabilities while eluding to the role of the β-value. This evaluation shows that the algorithm has the potential to determine the minimal bound on the number of SWAP operations.

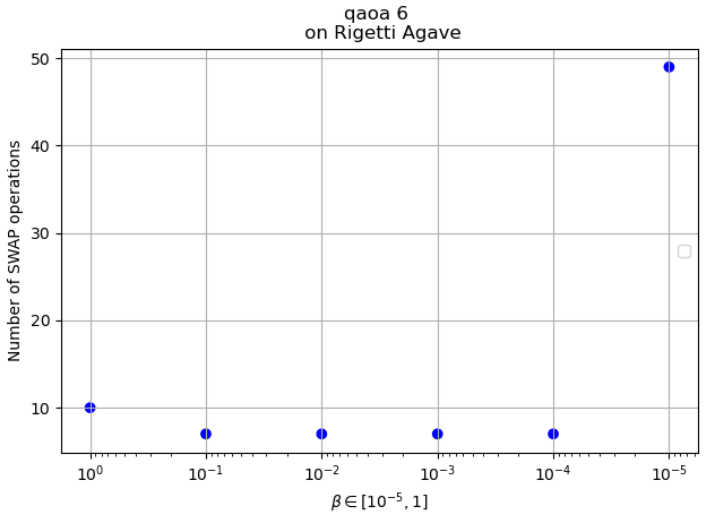
6.2 Benchmark evaluation

When considering the benchmarks, it becomes apparent that the algorithm depends on the β-value. Inspecting the benchmarks it becomes clear that the algorithm has a limited dependence on the β-value for graphs consisting of 4 or less vertices. The β-dependence can be observed as the number of vertices in the graph of the CG/IG-pairing increases. Considering a graph with more vertices means that more possible SWAP operations, increasing the complexity of determining the correct combination.

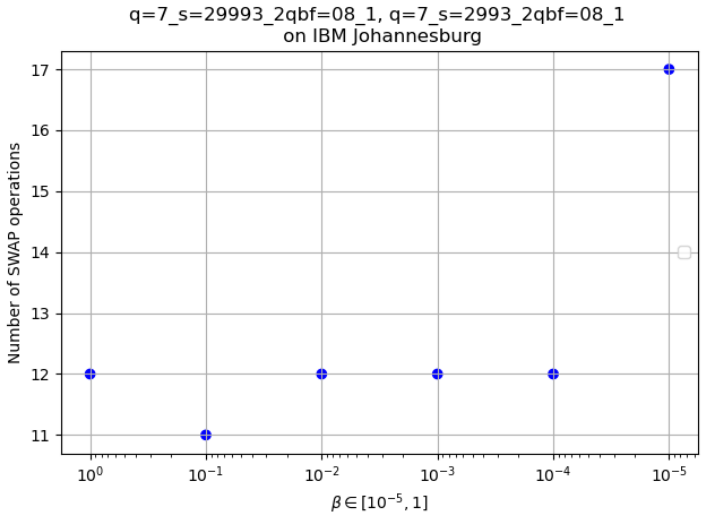
In this section, several benchmarks will be discussed in more detail to show the peculiar β-dependencies. We expect to determine the β-value for a given pairing which determines the minimal bound. When deviating from this value is expected that the number of SWAP operations will increase or stay the same for larger and smaller β-values. However, as we will see later, this is not always the case. The expected result can be seen in the examples in Fig. 29a, Fig. 29b and Fig. 29c.



(a) quantum_volume_n5 benchmark algorithm on IBM Rochester.



(b) qaoa 6 benchmark algorithm on the Rigetti Agave.



(c) q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1 benchmark algorithm on IBM Johannesburg.

Figure 29: Evaluation results of algorithm benchmarks on NISQ devices which exhibit the expected β-dependence.

The example in Fig. 30 has some red points in the plot. These indicate the β-values for which the algorithm could not determine a solution. This example adheres to the expectation that there is a β-value which determines the minimal number of SWAP operations. As the β-value is increase it is clear that the algorithm fails to determine a solution.

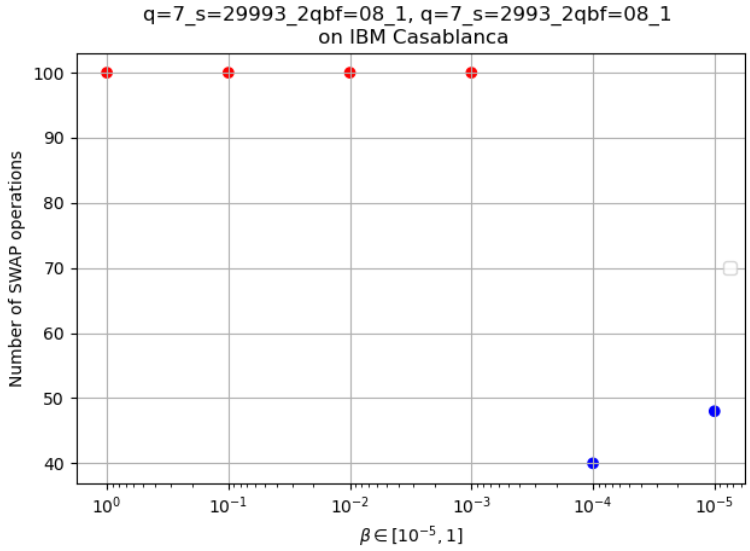


Figure 30: Evaluation results plotted for the q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1 benchmark algorithm on the IBM Casablanca.

There are also benchmark examples where a different β-dependence is observed. In these examples, we can observe 'double'-dips, which refers to evaluating the benchmark for a specific β-value that gives a low SWAP count, then the SWAP count increase as the β-value increases or decreases. However, in these examples, the SWAP count again decreases for an increased/decreased β-value. Example benchmarks that exhibit this β-dependence are shown in Fig. 31 and Fig. 32.

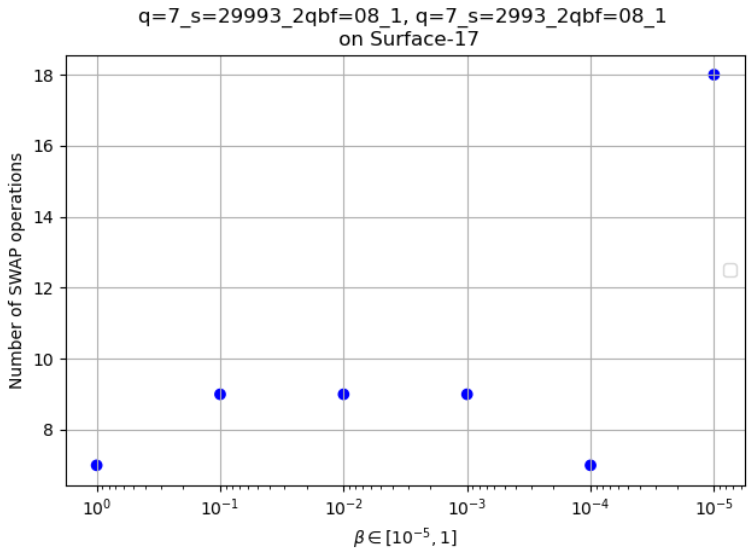


Figure 31: Evaluation results plotted for the q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1 benchmark algorithm on the Surface-17.

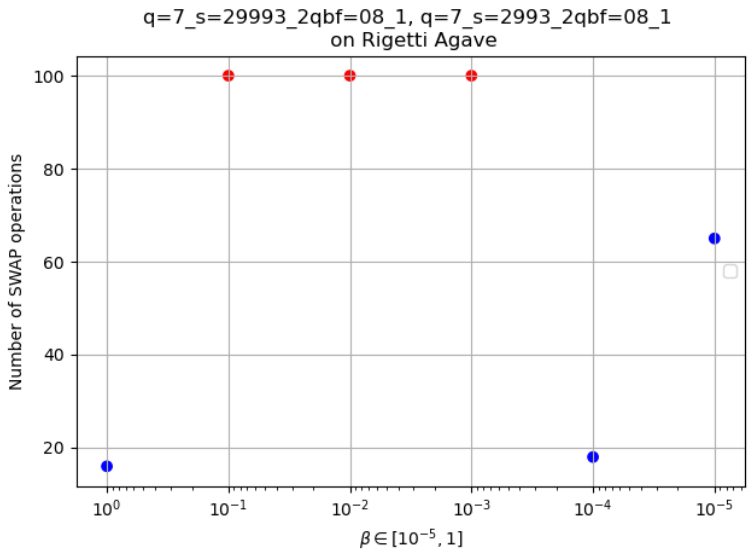


Figure 32: Evaluation results plotted for the q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1 benchmark algorithm on the Rigetti Agave.

Notice that Fig. 32 again contains the red points indicating the algorithm could not determine a solution. Oddly, increasing the β-value allows the algorithm to determine a result lower than the SWAP count, which was obtained for $\beta = 10^{-4}$. The peculiar β-dependence can not be explained and will require further investigation.

The evaluation of the benchmarks has given an insight into the complexity of the β-dependence. The expected β-dependence was observed for most of the benchmarks considered. Nonetheless, several examples have been discussed where the β-dependence deviates from what is expected. This deviation occurs more often as the graph's vertices increase. Determining a systematic approach to establish the correct β-value for a given CG/IG-pairing will be essential. Using the correct β-value will ensure that the algorithm determines the minimal SWAP count.

7 Conclusion

This master thesis aimed to introduce a new perspective to address the Qubit-Mapping Problem (QMP). Using quantum information theoretic descriptions of complex networks to study the relation between graphical objects. The graphical objects studied are based on quantum algorithms (IG) and their underlying physical architectures (CG). Considering the SWAP operations required to perform a quantum algorithm on a NISQ device. The goal is to determine if a minimal bound can be established on the number of SWAP operations for a given CG/IG-pairing. The main contribution of this thesis is the algorithmic implementation. This implementation introduced additional constraints that directly relate free variables in the optimisation to SWAP operations. The algorithm additionally considers SWAP operations which can be performed on the NISQ device based on the constraints of the device.

The introduced perspective redefines architecture's coupling graph (CG) and a quantum algorithm's interaction graph (IG) as density matrices. Elevating these graphical objects to quantum mechanical objects enables the use of processing techniques from quantum information theory. These can then be implemented to determine a bounded metric space for comparing the CG and the IG. This metric space is utilised to determine the objective function for an optimisation algorithm. The techniques that will be implemented to establish the objective function are forming a quantum channel between the CG and IG, establishing an entropic measure (the Von-Neumann entropy) and combining these in the quantum Jensen-Shannon divergence to define the metric space.

A density matrix is a matrix which describes the state of a quantum mechanical system. The quantum channel determines a map from one quantum state (density matrix) to another. In our case, we utilise the quantum channel description based on the Kraus operator formalism. The use of the Kraus operator formalism provides some constraints which are required by definition. However, the algorithmic implementation based only on these constraints is demanding in terms of computational resources. For that reason, additional constraints have been formulated based. These constraints further bound the problem while maintaining the principles of the proposed formulation. The Von-Neumann entropy quantifies the information or uncertainty in the system and is required to define the quantum Jensen-Shannon divergence. The quantum Jensen-Shannon divergence determines the metric space because it is bounded and symmetric. Thus it can quantitatively compare density matrices of graphs on a bounded metric space.

The evaluation of the proposed algorithm starts with evaluating basic toy models, which are trivial when considering the QMP. They are essential in the scope of this thesis since they contribute to understanding the proposed theory. Establishing that the proposed algorithm functions as expected based on the evaluation of the toy models. The evaluation is extended by running benchmarks based on more extensive algorithms and NISQ devices. These benchmarks indicate a clear dependence on the β-value chosen before running the algorithm. To establish the actual minimal bound, it will be required to tune the β-value based on the CG/IG-pairing. Further study of the β-value dependence of the quantum Jensen-Shannon divergence could allow the correct β-value to be determined before the execution of the proposed algorithm. Establishing the correct β-value for a given pairing will minimise the required SWAP operations.

In conclusion, this thesis presents a novel approach that allows further exploration of the QMP. The benchmarks have shown a clear dependence on the β-value. Emphasising the need for future investigations of this dependence to enhance the algorithm's effectiveness for more extensive algorithms and architectures. While it is essential to acknowledge that the approach may not currently rival the state of the art, it does offer a novel framework to study the QMP.

8 Future Work

This section will outline future research directions that can build upon the findings presented in this thesis. The future work can be categorised into two different parts. As mentioned in Sec. 5.1 and Sec. 6 it has become clear that the algorithm depends on the β-value. As it stands, we do not clearly understand why we have seen particular β-dependencies. As mentioned in this algorithm, the initial placement of the logical qubits on the architecture has been provided. It would be relevant to also include the determination of the initial mapping in the algorithm. In addition, the model we used to construct this method is oversimplified. They do not consider the fidelity of the architecture's gates, the number of gates a qubit is involved in and the order in which these gates should be applied. These are elements which should be taken into account to extend the functionality of the algorithm. Enabling a comparison with the state-of-the-art algorithms discussed in Sec. 3. The remainder of this section will discuss the proposed methods for this future work.

β-dependence

The β-dependence can be studied with several different methods. The proposed method would be to investigate the Von-Neumann entropy with respect to the β-value. Extending the work presented in Sec. 4.2.3. Considering the dependence of β on an inverse log scale allows the investigation of this dependence over a larger domain. Plotting the Von-Neumann entropy for β-values over a larger domain may give an insight into what is causing the dependence. The advantage is that we know which graphs are relevant to the study from the benchmarks presented in Sec. 6.2. To gain a detailed understanding, it will be necessary to analyse the Von-Neumann entropy vs. β-value plots for all algorithm iterations. This analysis will help identify the specific points at which the influence of the β-dependence comes into play. Examining these plots will hopefully allow us to establish the relationship between the algorithm's performance and the variation in the β-value. The examination will provide valuable insights into the behaviour of the algorithm.

Extension of the algorithm

Once the β-dependence has been analysed and understood it will be possible to extend the algorithm to include the initial mapping process and deal with additional constraints.

Including the initial mapping of the logical qubits in the algorithm guarantees that it can operate based on an input of the coupling and interaction graphs. Eliminating the need for any process to be required before that algorithm can determine the minimal bound on the number of SWAP operations. Moreover, including the initial mapping will ensure that the algorithm's performance can be compared to other methods proposed to deal with the QMP.

The additional constraints we would like to consider will be the fidelity of the architecture's gates, the number of gates a qubit is involved in and the order in which these gates should be applied. We proposed to add these additions in this order. An overview of the proposed additions is presented in Fig. 33.

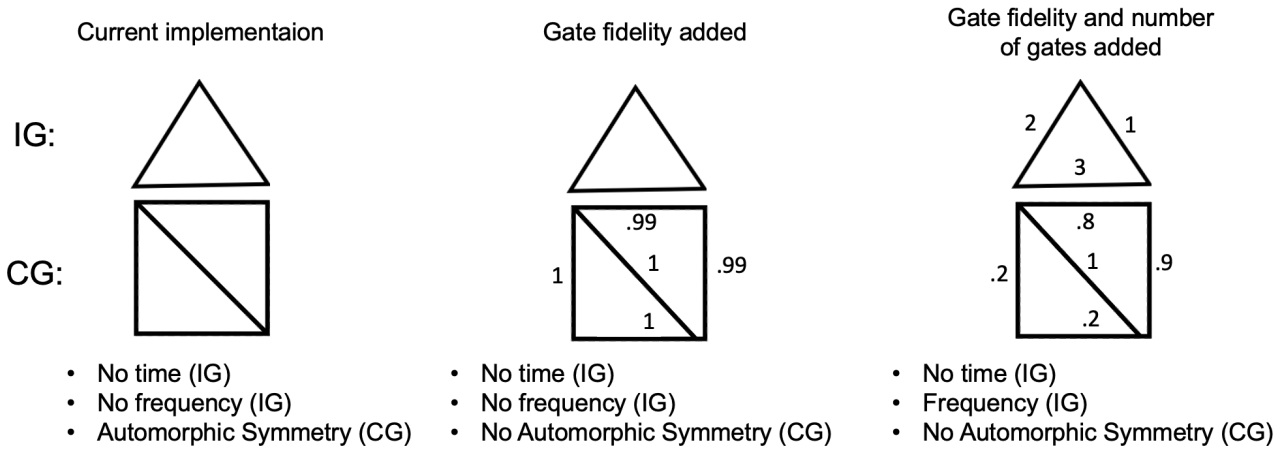


Figure 33: This figure gives an outline of the proposed approach to extend the algorithm's capabilities by considering additional constraints, allowing it to deal with more realistic examples.

The current implementation of the algorithm is depicted in the left example of Fig. 33. This situation is where we do not consider any of the aforementioned constraints. The automorphic symmetry refers to the fact

that the edges of our graphs do not have weights. This indicated that the fidelity of every operation is unity. In other words, there is no error when preforming a SWAP operation. Adding weights to the coupling graph to indicate the fidelity of gates is the first element we proposed adding to the current implementation. The example in the middle of Fig 33 shows a situation where we would like the algorithm to map correctly. This means that the initial mapping would map the logical qubits to the bottom left triangle of the coupling graph, where the weight of the edges is unity. Once this has been implemented, we propose to extend the algorithm further taking into account the number of operations between the logical qubits in the interaction graph, shown in the right example in Fig. 33. In this case, we want the algorithm to map the logical qubit pair with the most interactions to the physical qubit pair with the highest fidelity. In the case of the presented example, the logical qubit pair with three interactions should be mapped to the physical qubit pair connected by an edge with a weight equal to unity. the logical qubit pair with two interactions should be mapped to the physical qubit pair connected by an edge with a weight equal to .9. The last qubit pair will then be mapped to the physical qubit pair connected by an edge with a weight equal to .8.

The constraint we propose to implement is the addition of the order in which these gates should be applied. However, this is difficult, as seen in Sec. 3. It will require the algorithm to be implemented on certain sections of the quantum circuit. As it currently stands, we would recommend considering the techniques which have been presented in Sec. 3 to establish which would be best suited to complement our approach.

References

[1] Jean-François Bobier et al. *What Happens When 'If' Turns to 'When' in Quantum Computing?* Tech. rep. 2021.

[2] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521327X. DOI: 10.22331/q-2018-08-06-79. URL: https://quantum-journal.org/papers/q-2018-08-06-79/.

[3] Sumeet Khatri et al. "Quantum-assisted quantum compiling". In: *Quantum* 3 (July 2018). DOI: 10.22331/q-2019-05-13-140. URL: http://arxiv.org/abs/1807.00800http://dx.doi.org/10.22331/q-2019-05-13-140.

[4] J. Q. You and Franco Nori. "Atomic physics and quantum optics using superconducting circuits". In: (Feb. 2012). DOI: 10.1038/nature10122. URL: http://arxiv.org/abs/1202.1923http://dx.doi.org/10.1038/nature10122.

[5] He Liang Huang et al. *Superconducting quantum computing: a review*. Aug. 2020. DOI: 10.1007/s11432-020-2881-9.

[6] R. Barends et al. "Superconducting quantum circuits at the surface code threshold for fault tolerance". In: *Nature* 508.7497 (2014), pp. 500–503. ISSN: 14764687. DOI: 10.1038/nature13171.

[7] Sergei Slussarenko and Geoff J. Pryde. "Photonic quantum information processing: a concise review". In: (July 2019). DOI: 10.1063/1.5115814. URL: http://arxiv.org/abs/1907.06331http://dx.doi.org/10.1063/1.5115814.

[8] I. Chiorescu et al. "Coherent dynamics of a flux qubit coupled to a harmonic oscillator". In: *Nature* 431.7005 (Sept. 2004), pp. 159–162. ISSN: 00280836. DOI: 10.1038/nature02831.

[9] J I Cirac and P Zoller. *Quantum Computations with Cold Trapped Ions*. Tech. rep. 1995.

[10] Philipp Schindler et al. "A quantum information processor with trapped ions". In: (Aug. 2013). DOI: 10.1088/1367-2630/15/12/123012. URL: http://arxiv.org/abs/1308.3096http://dx.doi.org/10.1088/1367-2630/15/12/123012.

[11] Thomas Monz et al. "14-qubit entanglement: creation and coherence". In: (Sept. 2010). DOI: 10.1103/PhysRevLett.106.130506. URL: http://arxiv.org/abs/1009.6126http://dx.doi.org/10.1103/PhysRevLett.106.130506.

[12] Colin D. Bruzewicz et al. "Trapped-Ion Quantum Computing: Progress and Challenges". In: (Apr. 2019). DOI: 10.1063/1.5088164. URL: http://arxiv.org/abs/1904.04178http://dx.doi.org/10.1063/1.5088164.

[13] R. Maurand et al. "A CMOS silicon spin qubit". In: (May 2016). DOI: 10.1038/ncomms13575. URL: http://arxiv.org/abs/1605.07599http://dx.doi.org/10.1038/ncomms13575.

[14] Sergei Slussarenko and Geoff J. Pryde. "Photonic quantum information processing: a concise review". In: (July 2019). DOI: 10.1063/1.5115814. URL: http://arxiv.org/abs/1907.06331http://dx.doi.org/10.1063/1.5115814.

[15] J. Eli Bourassa et al. "Blueprint for a Scalable Photonic Fault-Tolerant Quantum Computer". In: (Oct. 2020). DOI: 10.22331/q-2021-02-04-392. URL: http://arxiv.org/abs/2010.02905http://dx.doi.org/10.22331/q-2021-02-04-392.

[16] IBM. *Quantum experience*. 2017.

[17] QuTech. *Quantum Inspire*. 2020.

[18] Dave Bacon and Wim Van Dam. *Recent progress in quantum algorithms*. Feb. 2010. DOI: 10.1145/1646353.1646375.

[19] Peter W. Shor. "Algorithms for quantum computation: Discrete logarithms and factoring". In: *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS* (1994), pp. 124–134. ISSN: 02725428. DOI: 10.1109/SFCS.1994.365700.

[20] Medina Bandić, Carmen G. Almudever, and Sebastian Feld. "Interaction graph-based profiling of quantum benchmarks for improving quantum circuit mapping techniques". In: (Dec. 2022).

[21] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2010, p. 676. ISBN: 9781107002173.

[22] F. Bloch. "Nuclear Induction". In: *Physical Review* 70.7-8 (Oct. 1946), pp. 460–474. ISSN: 0031-899X. DOI: 10.1103/PhysRev.70.460.

[23] Pengfei Wang et al. "Single ion qubit with estimated coherence time exceeding one hour". In: *Nature Communications* 12.1 (Dec. 2021). ISSN: 20411723. DOI: 10.1038/s41467-020-20330-w.

[24] Scott Aaronson and Lijie Chen. "Complexity-Theoretic Foundations of Quantum Supremacy Experiments". In: (Dec. 2016).

[25] Daniel Gottesman. "Opportunities and Challenges in Fault-Tolerant Quantum Computation". In: (Oct. 2022).

[26] Yasunari Suzuki et al. "Quantum Error Mitigation as a Universal Error Reduction Technique: Applications from the NISQ to the Fault-Tolerant Quantum Computing Eras". In: *PRX Quantum* 3.1 (Mar. 2022), p. 010345. ISSN: 2691-3399. DOI: 10.1103/PRXQuantum.3.010345.

[27] Ali Javadi Abhari et al. *Scaffold: Quantum Programming Language*. Tech. rep. 2012.

[28] Alexander S. Green et al. "Quipper: A Scalable Quantum Programming Language". In: (Apr. 2013). DOI: 10.1145/2499370.2462177. URL: http://arxiv.org/abs/1304.3390http://dx.doi.org/10.1145/2499370.2462177.

[29] N. Khammassi et al. "cQASM v1.0: Towards a Common Quantum Assembly Language". In: (May 2018).

[30] Andrew W. Cross et al. "Open Quantum Assembly Language". In: (July 2017).

[31] Craig Gidney. *Shor's Quantum Factoring Algorithm*. Aug. 2017.

[32] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*. New York, New York, USA: ACM Press, 1996, pp. 212–219. ISBN: 0897917855. DOI: 10.1145/237814.237866.

[33] Craig Gidney. *Grover's Quantum Search Algorithm*. Mar. 2013.

[34] Nicolas J Cerf, Lov K Grover, and Colin P Williams. *Nested Quantum Search and NP-Hard Problems*. Tech. rep. 2000, pp. 311–338.

[35] IBM: IBM Q Experience Device. *https://quantumexperience.ng.bluemix. net/qx/devices*. 2018.

[36] J. Kelly et al. "State preservation by repetitive error detection in a superconducting quantum circuit". In: (Nov. 2014). DOI: 10.1038/nature14270.

[37] T. Walter et al. "Rapid High-Fidelity Single-Shot Dispersive Readout of Superconducting Qubits". In: *Physical Review Applied* 7.5 (May 2017), p. 054020. ISSN: 2331-7019. DOI: 10.1103/PhysRevApplied.7.054020.

[38] Ahmed Abid Moueddene et al. "A context-aware gate set tomography characterization of superconducting qubits". In: (Mar. 2021).

[39] Andrew J. Daley et al. "Practical quantum advantage in quantum simulation". In: *Nature* 607.7920 (July 2022), pp. 667–676. ISSN: 0028-0836. DOI: 10.1038/s41586-022-04940-6.

[40] Adarsh Jain et al. "Experimental Demonstration of Free Space Quantum Key Distribution System based on the BB84 Protocol". In: *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, July 2020, pp. 1–5. ISBN: 978-1-7281-6851-7. DOI: 10.1109/ICCCNT49239.2020.9225317.

[41] Gushu Li, Yufei Ding, and Yuan Xie. "Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices". In: (Sept. 2018). URL: http://arxiv.org/abs/1809.02573.

[42] Marcos Yukio Siraichi et al. "Qubit allocation". In: *CGO 2018 - Proceedings of the 2018 International Symposium on Code Generation and Optimization*. Vol. 2018-February. Association for Computing Machinery, Inc, Feb. 2018, pp. 113–125. ISBN: 9781450356176. DOI: 10.1145/3168822.

[43] H Häffner et al. "Scalable multiparticle entanglement of trapped ions". In: (2005). DOI: 10.1038/nature04279.

[44] Martin Laforest et al. "Using error correction to determine the noise model". In: *Physical Review A - Atomic, Molecular, and Optical Physics* 75.1 (2007). ISSN: 10941622. DOI: 10.1103/PhysRevA.75.012331.

[45] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. "Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures". In: *Proceedings - Design Automation Conference* (2013). ISSN: 0738100X. DOI: 10.1145/2463209.2488785.

[46] Diplom-Mathematikerin Hanna Seitz and geborene Peters aus Castrop-Rauxel. *Contributions to the Minimum Linear Arrangement Problem*. Tech. rep. 2010.

[47] Jordi Petit. *Experiments on the Minimum Linear Arrangement Problem*. Tech. rep. 2003.

[48] Mehdi Saeedi et al. "Synthesis of quantum circuits for linear nearest neighbor architectures". In: 10 (2011), pp. 355–377. DOI: 10.1007/s11128-010-0201-2.

[49] Yuichi Hirata et al. "An efficient method to convert arbitrary quantum circuits to ones on a linear nearest neighbor architecture". In: *Proceedings of the 3rd International Conference on Quantum, Nano and Micro Technologies, ICQNM 2009*. 2009, pp. 26–33. ISBN: 9780769535241. DOI: 10.1109/ICQNM.2009.25.

[50] Atsushi Matsuo and Shigeru Yamashita. *Changing the Gate Order for Optimal LNN Conversion*. Ed. by Alexis De Vos and Robert Wille. Vol. 7165. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 89–101. ISBN: 978-3-642-29516-4. DOI: 10.1007/978-3-642-29517-1. URL: http://link.springer.com/10.1007/978-3-642-29517-1.

[51] Robert Wille, Aaron Lye, and Rolf Drechsler. *Optimal SWAP Gate Insertion for Nearest Neighbor Quantum Circuits*. Tech. rep. 2014.

[52] Robert Wille, Aaron Lye, and Rolf Drechsler. "Exact reordering of circuit lines for nearest neighbor quantum architectures". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.12 (2014), pp. 1818–1831. ISSN: 02780070. DOI: 10.1109/TCAD.2014.2356463.

[53] L. C.L. Hollenberg et al. "Two-dimensional architectures for donor-based quantum computing". In: *Physical Review B - Condensed Matter and Materials Physics* 74.4 (2006). ISSN: 10980121. DOI: 10.1103/PhysRevB.74.045311.

[54] Naomi H. Nickerson, Ying Li, and Simon C. Benjamin. "Topological quantum computing with a very noisy network and local error rates approaching one percent". In: *Nature Communications* 4 (2013). ISSN: 20411723. DOI: 10.1038/ncomms2773.

[55] M. Saffman, T. G. Walker, and K. Mølmer. "Quantum information with Rydberg atoms". In: *Reviews of Modern Physics* 82.3 (Aug. 2010), pp. 2313–2363. ISSN: 00346861. DOI: 10.1103/RevModPhys.82.2313.

[56] Muir Kumph, Michael Brownnutt, and Rainer Blatt. "Two-dimensional arrays of radio-frequency ion traps with addressable interactions". In: *New Journal of Physics* 13 (July 2011). ISSN: 13672630. DOI: 10.1088/1367-2630/13/7/073043.

[57] J. M. Taylor et al. "Relaxation, dephasing, and quantum control of electron spins in double quantum dots". In: *Physical Review B - Condensed Matter and Materials Physics* 76.3 (July 2007). ISSN: 10980121. DOI: 10.1103/PhysRevB.76.035315.

[58] Alexandre Blais et al. "Quantum-information processing with circuit quantum electrodynamics". In: *Physical Review A - Atomic, Molecular, and Optical Physics* 75.3 (Mar. 2007). ISSN: 10502947. DOI: 10.1103/PhysRevA.75.032329.

[59] Byung Soo Choi and Rodney Van Meter. "A Θ(Mathematical Equation Presented)-depth quantum adder on the 2D NTC quantum computer architecture". In: *ACM Journal on Emerging Technologies in Computing Systems* 8.3 (Aug. 2012). ISSN: 15504832. DOI: 10.1145/2287696.2287707.

[60] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. *Qubit Placement to Minimize Communication Overhead in 2D Quantum Architectures*. Tech. rep. 2014.

[61] Aaron Lye, Robert Wille, and Rolf Drechsler. *Determining the Minimal Number of SWAP Gates for Multi-dimensional Nearest Neighbor Quantum Circuits*. Tech. rep. 2015.

[62] Martin Gebser; Benjamin Kaufmann; Andre´ Neumann; Torsten Schaub. "clasp: A conict-driven answer set solver". In: *Logic Programming and Nonmonotonic Reasoning* (2007), pp. 260–265.

[63] Alwin Zulehner, Alexandru Paler, and Robert Wille. "An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38.7 (Dec. 2018), pp. 1226–1236. ISSN: 02780070. DOI: 10.48550/arxiv.1712.04722. URL: https://arxiv.org/abs/1712.04722v3.

[64] IBM Q. *https://www.research.ibm.com/ibm-q/*. Sept. 2017.

[65] C. Neill et al. "A blueprint for demonstrating quantum supremacy with superconducting qubits". In: (Sept. 2017). DOI: 10.1126/science.aao4309.

[66] R. Courtland. "Google aims for quantum computing supremacy [news]". In: *IEEE Spectrum* 54.6 (2017), pp. 9–10.

[67] D Michael Miller, Robert Wille, and Z Sasanian. *Elementary Quantum Gate Realizations for Multiple-Control Toffoli Gates*. Tech. rep. 2011, pp. 288–293.

[68] Ken Matsumoto and Kazuyuki Amano. "Representation of Quantum Circuits with Clifford and $\pi/8$ Gates". In: (June 2008).

[69] C. Otterstedt R. Wille M. Soeken and R. Drechsler. "Improving the mapping of reversible circuits to quantum circuits using multiple target lines". In: *Asia and South Pacific Design Automation Conf.* 2013, pp. 85–92.

[70] Y. Inoue N. Yasuda R. Wille N. Quetschlich and S. Minato. "Using \pi dds for nearest neighbor optimization of quantum circuits". In: *Int'l Conf. of Reversible Computation* (2016), pp. 181–196.

[71] Swamit S Tannu and Moinuddin K Qureshi. "Not All Qubits Are Created Equal A Case for Variability-Aware Policies for NISQ-Era Quantum Computers". In: (2018).

[72] Prakash Murali et al. "Formal Constraint-based Compilation for Noisy Intermediate-Scale Quantum Systems". In: (Mar. 2019). DOI: 10.1016/j.micpro.2019.02.005. URL: http://arxiv.org/abs/1903.03276http://dx.doi.org/10.1016/j.micpro.2019.02.005.

[73] ScaffCC Compiler. "https://github.com/epiqc/ScaffCC". In: (May 2018).

[74] Roberto Sebastiani and Patrick Trentin. "On Optimization Modulo Theories, MaxSMT and Sorting Networks". In: (Feb. 2017). URL: http://arxiv.org/abs/1702.02385.

[75] David C. McKay et al. "Qiskit Backend Specifications for OpenQASM and OpenPulse Experiments". In: (Sept. 2018).

[76] S. Asaad et al. "Independent, extensible control of same-frequency superconducting qubits by selective broadcasting". In: (Aug. 2015). DOI: 10.1038/npjqi.2016.29.

[77] Lingling Lao et al. "Mapping of quantum circuits onto NISQ superconducting processors". In: *arXiv: Quantum Physics* (2019).

[78] INTEL. *Intel newsroom*. 2019.

[79] R. Versluis et al. "Scalable quantum circuit and control for a superconducting surface code". In: (Dec. 2016). DOI: 10.1103/PhysRevApplied.8.034021. URL: http://arxiv.org/abs/1612.08208http://dx.doi.org/10.1103/PhysRevApplied.8.034021.

[80] Austin G Fowler et al. *Surface codes: Towards practical large-scale quantum computation*. Tech. rep.

[81] L. Lao et al. "Mapping of lattice surgery-based quantum circuits on surface code architectures". In: *Quantum Science and Technology* 4.1 (Jan. 2019). ISSN: 20589565. DOI: 10.1088/2058-9565/aadd1a.

[82] Robert Wille et al. *RevLib: An Online Resource for Reversible Functions and Reversible Circuits*. Tech. rep. 2008. URL: www.revlib.org.

[83] Chia Chun Lin, Amlan Chakrabarti, and Niraj K. Jha. "QLib: Quantum module library". In: *ACM Journal on Emerging Technologies in Computing Systems* 11.1 (2014). ISSN: 15504840. DOI: 10.1145/2629430.

[84] N. M. Linke et al. "Experimental Comparison of Two Quantum Computing Architectures". In: (Feb. 2017). DOI: 10.1073/pnas.1618020114. URL: http://arxiv.org/abs/1702.01852http://dx.doi.org/10.1073/pnas.1618020114.

[85] Will Finigan et al. "Qubit Allocation for Noisy Intermediate-Scale Quantum Computers". In: (Oct. 2018).

[86] Shin Nishio et al. "Extracting Success from IBM's 20-Qubit Machines Using Error-Aware Compilation". In: (Mar. 2019). DOI: 10.1145/3386162.

[87] Prakash Murali et al. "Full-Stack, Real-System Quantum Computer Studies: Architectural Comparisons and Design Insights". In: (May 2019).

[88] N. Khammassi et al. "OpenQL: A Portable Quantum Programming Framework for Quantum Accelerators". In: *ACM Journal on Emerging Technologies in Computing Systems* 18.1 (Jan. 2022), pp. 1–24. ISSN: 1550-4832. DOI: 10.1145/3474222.

[89] Alwin Zulehner, Alexandru Paler, and Robert Wille. "Efficient mapping of quantum circuits to the IBM QX architectures". In: *Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition, DATE 2018*. Vol. 2018-January. Institute of Electrical and Electronics Engineers Inc., Apr. 2018, pp. 1135–1138. ISBN: 9783981926316. DOI: 10.23919/DATE.2018.8342181.

[90] IBM; QISKit Open Source Quantum Information Science Kit. *https: //qiskit.org/*. 2018.

[91] Ferozuddin Riaz and Khidir M. Ali. "Applications of graph theory in computer science". In: *Proceedings - 3rd International Conference on Computational Intelligence, Communication Systems and Networks, CICSyN 2011* (2011), pp. 142–145. DOI: 10.1109/CICSYN.2011.40.

[92] Marcos Yukio Siraichi et al. "Qubit allocation as a combination of subgraph isomorphism and token swapping". In: *Proceedings of the ACM on Programming Languages* 3.OOPSLA (Oct. 2019). ISSN: 24751421. DOI: 10.1145/3360546.

[93] Stephen A Cook. "The Complexity of Theorem-Proving Procedures". In: *STOC. ACM* (1971), pp. 151–158.

[94] David Eppstein. "Subgraph Isomorphism in Planar Graphs and Related Problems". In: *Journal of Graph Algorithms and Applications* 3.3 (1999), pp. 1–27. ISSN: 1526-1719. DOI: 10.7155/jgaa.00014.

[95] Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity*. Vol. 28. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 400–401. ISBN: 978-3-642-27874-7. DOI: 10.1007/978-3-642-27875-4.

[96] Katsuhisa Yamanaka et al. *Swapping Labeled Tokens on Graphs*. Tech. rep. Heidelberg: Springer, 2014, pp. 364–375.

[97] Tillmann Miltzow et al. "Approximation and Hardness for Token Swapping". In: (Feb. 2016). DOI: 10.4230/LIPIcs.ESA.2016.185.

[98] Luigi P Cordella et al. *A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs*. Tech. rep. 10. 2004. URL: http://amalfi.dis.unina.it/graph/.

[99] Wook-Shin Han, Jinsoo Lee, and Jeong-Hoon Lee. "Turboiso: Towards Ultrafast and Robust Subgraph Isomorphism Search in Large Graph Database". In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, June 2013, pp. 337–348. ISBN: 9781450320375. DOI: 10.1145/2463676.2465300.

[100] Peixiang Zhao and Jiawei Han. "On graph query optimization in large networks". In: *Proceedings of the VLDB Endowment* 3.1-2 (Sept. 2010), pp. 340–351. ISSN: 2150-8097. DOI: 10.14778/1920841.1920887.

[101] Marcos Yukio Siraichi et al. "Qubit Allocation". In: 18 (2018). DOI: 10.1145/3168822.

[102] Alwin Zulehner and Robert Wille. "Compiling SU(4) Quantum Circuits to IBM QX Architectures". In: (Aug. 2018).

[103] Sanjiang Li, Xiangzhen Zhou, and Yuan Feng. "Qubit Mapping Based on Subgraph Isomorphism and Filtered Depth-Limited Search". In: *IEEE Transactions on Computers* 70.11 (Apr. 2020), pp. 1777–1788. DOI: 10.1109/TC.2020.3023247. URL: http://arxiv.org/abs/2004.07138http://dx.doi.org/10.1109/TC.2020.3023247.

[104] Hui Jiang, Yuxin Deng, and Ming Xu. "Quantum Circuit Transformation Based on Tabu Search". In: (Apr. 2021). URL: http://arxiv.org/abs/2104.05214.

[105] Shixuan Sun and Qiong Luo. "In-Memory Subgraph Matching: An In-depth Study". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, June 2020, pp. 1083–1098. ISBN: 9781450367356. DOI: 10.1145/3318464.3380581.

[106] Matthew Steinberg et al. "Topological-Graph Dependencies and Scaling Properties of a Heuristic Qubit-Assignment Algorithm". In: (Mar. 2021). DOI: 10.1109/TQE.2022.3160015. URL: http://arxiv.org/abs/2103.15695http://dx.doi.org/10.1109/TQE.2022.3160015.

[107] Michael Brooks. *IBM wants to build a 100,000-qubit quantum computer*. May 2023.

[108] Nino Shervashidze et al. "Efficient graphlet kernels for large graph comparison". In: (2009).

[109] Kanigalpula Samanvi and Naveen Sivadasan. "Subgraph Similarity Search in Large Graphs". In: (2015).

[110] M. Steinberg and M. Bandić and S. B. Szkudlarek and A. Sarkar and S. Feld and C. G. Alumdever. *Formalizing Problems in Quantum Compilation via Quantum Information Theory, Statistical Mechanics, and Graph Theory*. 2023.

[111] Jacob Biamonte, Mauro Faccin, and Manlio De Domenico. "Complex networks from classical to quantum". In: *Communications Physics 2019 2:1* 2.1 (May 2019), pp. 1–10. ISSN: 2399-3650. DOI: 10.1038/s42005-019-0152-6. URL: https://www.nature.com/articles/s42005-019-0152-6.

[112] Robin J. Wilson. *Introduction to Graph Theory Fourth edition*. Tech. rep. 1998.

[113] Martin Kliesch and Arnau Riera. "Properties of thermal quantum states: locality of temperature, decay of correlations, and more". In: (Mar. 2018). DOI: 10.1007/978-3-319-99046-0{_}20.

[114] Robert Sims. *Deriving the Thermal Equilibrium State for a Finite Quantum Spin System*. Tech. rep. 2018.

[115] Carlo Nicolini, Vladimir Vlasov, and Angelo Bifone. "Thermodynamics of network model fitting with spectral entropies". In: *Physical Review E* 98.2 (Aug. 2018), p. 022322. ISSN: 2470-0045. DOI: 10.1103/PhysRevE.98.022322.

[116] Alfred Wehrl. *General properties of entropy*. Tech. rep. 1978.

[117] G6ran Lindblad. *Entropy, Information and Quantum Measurements*. Tech. rep. 1973, pp. 305–322.

[118] Christopher A. Fuchs and Jeroen De Van Graaf. "Cryptographie distinguishability measures for quantum-mechanical states". In: *IEEE Transactions on Information Theory* 45.4 (1999), pp. 1216–1227. ISSN: 00189448. DOI: 10.1109/18.761271.

[119] A. P. Majtey, P. W. Lamberti, and D. P. Prato. "Jensen-Shannon divergence as a measure of distinguishability between mixed quantum states". In: *Physical Review A* 72.5 (Nov. 2005), p. 052310. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.72.052310.

[120] Jop Briët and Peter Harremoës. "Properties of classical and quantum Jensen-Shannon divergence". In: *Physical Review A - Atomic, Molecular, and Optical Physics* 79.5 (May 2009), p. 052311. ISSN: 10502947. DOI: 10.1103/PHYSREVA.79.052311/FIGURES/1/MEDIUM. URL: https://journals.aps.org/pra/abstract/10.1103/PhysRevA.79.052311.

[121] P. W. Lamberti et al. "Metric character of the quantum Jensen-Shannon divergence". In: *Physical Review A - Atomic, Molecular, and Optical Physics* 77.5 (May 2008), p. 052311. ISSN: 10502947. DOI: 10.1103/PHYSREVA.77.052311/FIGURES/1/MEDIUM. URL: https://journals.aps.org/pra/abstract/10.1103/PhysRevA.77.052311.

[122] Manlio De Domenico et al. "Structural reducibility of multilayer networks". In: *Nature Communications* 6.1 (Apr. 2015), p. 6864. ISSN: 2041-1723. DOI: 10.1038/ncomms7864.

[123] Martin Gerlach, Francesc Font-Clos, and Eduardo G. Altmann. "Similarity of Symbol Frequency Distributions with Heavy Tails". In: *Physical Review X* 6.2 (Apr. 2016), p. 021009. ISSN: 2160-3308. DOI: 10.1103/PhysRevX.6.021009.

[124] L. J. Landau and R. F. Streater. "On Birkhoff's theorem for doubly stochastic completely positive maps of matrix algebras". In: *Linear Algebra and its Applications* 193.C (Nov. 1993), pp. 107–127. ISSN: 0024-3795. DOI: 10.1016/0024-3795(93)90274-R.

[125] John Watrous. "Mixing doubly stochastic quantum channels with the completely depolarizing channel". In: (July 2008). URL: http://arxiv.org/abs/0807.2668.

[126] SciPy. *Sequential Least Squares Programming (SLSQP) optimisation method*. 2023.

[127] Nicolas Gisin et al. "Quantum cryptography". In: *Reviews of Modern Physics* 74.1 (Mar. 2002), pp. 145–195. ISSN: 0034-6861. DOI: 10.1103/RevModPhys.74.145.

[128] S. Pirandola et al. "Advances in Quantum Cryptography". In: (June 2019). DOI: 10.1364/AOP.361502.

[129] Charles H. Bennett and Gilles Brassard. "Quantum cryptography: Public key distribution and coin tossing". In: *Theoretical Computer Science* 560.P1 (2014), pp. 7–11. ISSN: 03043975. DOI: 10.1016/j.tcs.2014.05.025.

[130] W. K. Wootters and W. H. Zurek. "A single quantum cannot be cloned". In: *Nature* 299.5886 (Oct. 1982), pp. 802–803. ISSN: 0028-0836. DOI: 10.1038/299802a0.

[131] D. Dieks. "Communication by EPR devices". In: *Physics Letters A* 92.6 (Nov. 1982), pp. 271–272. ISSN: 03759601. DOI: 10.1016/0375-9601(82)90084-6.

[132] Valerio Scarani et al. "The Security of Practical Quantum Key Distribution". In: (Feb. 2008). DOI: 10.1103/RevModPhys.81.1301.

[133] Artur K. Ekert et al. "Eavesdropping on quantum-cryptographical systems". In: *Physical Review A* 50.2 (Aug. 1994), pp. 1047–1056. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.50.1047.

[134] Sam McArdle et al. "Quantum computational chemistry". In: *Reviews of Modern Physics* 92.1 (Mar. 2020), p. 015003. ISSN: 0034-6861. DOI: 10.1103/RevModPhys.92.015003.

[135] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5.1 (July 2014), p. 4213. ISSN: 2041-1723. DOI: 10.1038/ncomms5213.

[136] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. "Progress towards practical quantum variational algorithms". In: *Physical Review A* 92.4 (Oct. 2015), p. 042303. ISSN: 1050-2947. DOI: 10.1103/PhysRevA.92.042303.

[137] Jan-Michael Reiner et al. "Finding the ground state of the Hubbard model by variational methods on a quantum computer with gate errors". In: *Quantum Science and Technology* 4.3 (July 2019), p. 035005. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab1e85.

[138] Jan-Michael Reiner et al. "Effects of gate errors in digital quantum simulations of fermionic systems". In: (Apr. 2018). DOI: 10.1088/2058-9565/aad5ba.

[139] Vedran Dunjko and Hans J Briegel. "Machine learning & artificial intelligence in the quantum domain: a review of recent progress". In: *Reports on Progress in Physics* 81.7 (July 2018), p. 074001. ISSN: 0034-4885. DOI: 10.1088/1361-6633/aab406.

[140] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (Sept. 2017), pp. 195–202. ISSN: 0028-0836. DOI: 10.1038/nature23474.

[141] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum principal component analysis". In: *Nature Physics* 10.9 (Sept. 2014), pp. 631–633. ISSN: 1745-2473. DOI: 10.1038/nphys3029.

[142] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. "Quantum Support Vector Machine for Big Data Classification". In: *Physical Review Letters* 113.13 (Sept. 2014), p. 130503. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.113.130503.

[143] Fernando G. S. L. Brandao and Krysta Svore. "Quantum Speed-ups for Semidefinite Programming". In: (Sept. 2016).

[144] Patrick Rebentrost et al. "Quantum gradient descent and Newton's method for constrained polynomial optimization". In: (Dec. 2016).

Appendices

A Additional quantum algorithms

This appendix is added to provide the reader with additional material in case they are interested in the potential applications of quantum computation.

Quantum cryptography

Possibly the fastest-growing field within quantum information sciences is quantum cryptography. It aims to exploit the principles of quantum mechanics to allow for secure communication [127]. Since it is a vast field, we will only discuss the example of BB84 in this section. There are several more examples and protocols. For a more in-depth review of quantum cryptography, the reader is referred to [127, 128]

BB84 is a quantum key distribution scheme Gilles Brassard and Charles Bennett proposed in 1984 [129]. In this work, they note that when elementary quantum systems can transmit digital information, the uncertainty principle gives rise to novel cryptographic phenomena that can not be achieved with classical devices. The protocol has been proven to be secure based on the no-cloning theorem [130, 131], and the existence of an authenticated public classical channel [132]. An authenticated public channel means an eavesdropper may listen to the information propagated in the channel but can not alter it. The BB84 protocol requires four states and two separate measurement bases, which are required to be maximally conjugate, which means that any pair of states, one from each basis, have the same overlap. Once this has been established, Alice will send individual qubits to Bob, randomly chosen to be in one of the four basis states of the chosen measurement bases. In this step, it is required that they can establish a one-to-one correspondence between the sent and received states. Upon receiving the qubits, Bob will measure them randomly in one of the measurement bases. The measurement may then result in either a perfectly correlated result, in the case that his measurement basis corresponds to the basis chosen by Alice, or an uncorrelated result, in the case that he measures in a different basis. This results in a raw key, a bit string with 25% errors and an error rate too large for standard error correction protocols. At this point in the protocol, the authenticated channel will be necessary. Bob uses this channel to announce his choice of measurement basis to Alice without disclosing the obtained result. In turn, Alice uses the channel to tell Bob whether or not the states he has obtained are correlated or uncorrelated. Those qubits for which the results are uncorrelated are discarded. On average, this results in Bob obtaining 50% of the original bit string, referred to as the sifted key [133]. Alice and Bob can not obtain a specific key with this protocol due to their random choices of measurement basis. However, they can determine the statistics of the key.

The protocol's safety can be explained in the following manner: if an eavesdropper, Eve, intercepts a qubit, Bob will know and can communicate this to Alice, disregarding said qubit. Due to the no-cloning theorem, it is impossible to clone the state Alice has prepared and sent to Bob. Thus, if Eve measures the intercepted qubit, its state will collapse, resulting in Bob receiving one qubit less. Lowering the bit rate between Alice and Bob. However, Eve will not gain any useful information. The protocol clearly shows that the unique principles described by quantum mechanics can be exploited for cryptographic applications.

Quantum chemistry

Another field in which quantum computation shows promise is solving chemistry problems intractable for classical computers and contributing to the understanding of chemical phenomena such as high-temperature superconductivity, solid-state physics, transition metal catalysis and particular biochemical reactions [134]. Increased understanding of such phenomena could lead to developing new materials for both scientific and industrial applications [134]. As mentioned in the introduction to this section, the implementation of such progress is bounded by the limitation of the current quantum architecture. Nonetheless, developments have attempted to tackle the problems stated above with fewer resources available sooner than the desired large fault-tolerant quantum computers. Due to the vast field size, only the variational quantum eigensolver (VQE) example will be discussed. For a more in-depth review of quantum chemistry, the reader is referred to [134].

The variational quantum eigensolver (VQE) first introduced by Peruzzo et al. [135] uses a heuristic approach based on the variational method in quantum mechanics to compute the ground state energy of a Hamiltonian, a central problem to both the field of quantum chemistry as well as condensed matter physics. VQE is a hybrid algorithm using classical and quantum computers to obtain the ground state of a physical system. Quantum computation is utilised for state preparation and measurement subroutine, and classical computation processes the measurement result and updates the quantum computer accordingly.

The approximation done by the VQE performed on a small circuit could be better than classical methods. However, proving if the obtained approximation is a reasonable estimate of the ground state is complex. In principle, a larger circuit would result in a better approximation. Still, the size of the circuits which may be utilised is limited by the error rates present in current devices. Nonetheless, it has been shown that using a variational approach is far less demanding when considering the Hubbard model, a standard model from solid-state physics describing the transition between conducting and insulating systems. This approach may even be practical on small-scale quantum computers, limited by noise [136]. Moreover, the effect of gate errors on the performance of this approach has been considered [137, 138]. Showing promising results to determine the ground state as well as dynamics simulation.

Quantum machine learning

The last mention of applying quantum algorithms will be their implementation to realise a meaningful contribution to machine learning. Quantum systems can produce computationally too complex patterns for classical computers to recreate [139]. It may also be possible to use quantum computation to recognise complex patterns. The realisation of this idea hinges on finding efficient quantum algorithms to perform these tasks. For an in-depth review of the field of quantum machine learning, the reader is referred to [140, 139].

Several algorithms have been proposed to achieve the desired performance. Examples are a quantum principal component analysis algorithm [141], a quantum support vector machine algorithm [142], an algorithm for solving semi-definite programs [143] and a quantum version of Newton's optimisation method [144]. In addition, the example of VQE discussed above is also an implementation of quantum machine learning, underpinning that the field may provide relevant contributions for several applications.

Moreover, it is relevant to consider the reciprocal impact of quantum machine learning and classical machine learning. Exploring the interplay between the two disciplines may offer promising new possibilities [139].

B Benchmark results

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
fredkin_n3, grover_n3	IBM Athens	1	1	1
fredkin_n3, grover_n3	Starmon-5	1	1	1
fredkin_n3, grover_n3	IBM Yorktown	0	0	0
fredkin_n3, grover_n3	IBM Ourense	1	1	1
fredkin_n3, grover_n3	Surface-7	1	1	1
fredkin_n3, grover_n3	IBM Casablanca	1	1	1
fredkin_n3, grover_n3	Rigetti Agave	1	1	1
fredkin_n3, grover_n3	IBM Melbourne	1	1	1
fredkin_n3, grover_n3	Rigetti Aspen-1	1	1	1
fredkin_n3, grover_n3	Surface-17	1	1	1
fredkin_n3, grover_n3	IBM Singapore	1	1	1
fredkin_n3, grover_n3	IBM Johannesburg	1	1	1
fredkin_n3, grover_n3	IBM Tokyo	0	0	0
fredkin_n3, grover_n3	IBM Paris	1	1	1
fredkin_n3, grover_n3	IBM Rochester	1	1	1
fredkin_n3, grover_n3	Google Bristlecone	1	1	1

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
fredkin_n3, grover_n3	IBM Athens	1	1	1
fredkin_n3, grover_n3	Starmon-5	1	1	1
fredkin_n3, grover_n3	IBM Yorktown	0	0	0
fredkin_n3, grover_n3	IBM Ourense	1	1	1
fredkin_n3, grover_n3	Surface-7	1	1	1
fredkin_n3, grover_n3	IBM Casablanca	1	1	1
fredkin_n3, grover_n3	Rigetti Agave	1	1	1
fredkin_n3, grover_n3	IBM Melbourne	1	1	1
fredkin_n3, grover_n3	Rigetti Aspen-1	1	1	1
fredkin_n3, grover_n3	Surface-17	1	1	1
fredkin_n3, grover_n3	IBM Singapore	1	1	1
fredkin_n3, grover_n3	IBM Johannesburg	1	1	1
fredkin_n3, grover_n3	IBM Tokyo	0	0	0
fredkin_n3, grover_n3	IBM Paris	1	1	1
fredkin_n3, grover_n3	IBM Rochester	1	1	1
fredkin_n3, grover_n3	Google Bristlecone	1	1	1

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
basis_change_n3, teleportation_n3	IBM Athens	0	0	0
basis_change_n3, teleportation_n3	Starmon-5	0	0	0
basis_change_n3, teleportation_n3	IBM Yorktown	0	0	0
basis_change_n3, teleportation_n3	IBM Ourense	0	0	0
basis_change_n3, teleportation_n3	Surface-7	0	0	0
basis_change_n3, teleportation_n3	IBM Casablanca	0	0	0
basis_change_n3, teleportation_n3	Rigetti Agave	0	0	0
basis_change_n3, teleportation_n3	IBM Melbourne	0	0	0
basis_change_n3, teleportation_n3	Rigetti Aspen-1	0	0	0
basis_change_n3, teleportation_n3	Surface-17	0	0	0
basis_change_n3, teleportation_n3	IBM Singapore	0	0	0
basis_change_n3, teleportation_n3	IBM Johannesburg	0	0	0
basis_change_n3, teleportation_n3	IBM Tokyo	0	0	0
basis_change_n3, teleportation_n3	IBM Paris	0	0	0
basis_change_n3, teleportation_n3	IBM Rochester	0	0	0
basis_change_n3, teleportation_n3	Google Bristlecone	0	0	0

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
basis_change_n3, teleportation_n3	IBM Athens	0	0	0
basis_change_n3, teleportation_n3	Starmon-5	0	0	0
basis_change_n3, teleportation_n3	IBM Yorktown	0	0	0
basis_change_n3, teleportation_n3	IBM Ourense	0	0	0
basis_change_n3, teleportation_n3	Surface-7	0	0	0
basis_change_n3, teleportation_n3	IBM Casablanca	0	0	0
basis_change_n3, teleportation_n3	Rigetti Agave	0	0	0
basis_change_n3, teleportation_n3	IBM Melbourne	0	0	0
basis_change_n3, teleportation_n3	Rigetti Aspen-1	0	0	0
basis_change_n3, teleportation_n3	Surface-17	0	0	0
basis_change_n3, teleportation_n3	IBM Singapore	0	0	0
basis_change_n3, teleportation_n3	IBM Johannesburg	0	0	0
basis_change_n3, teleportation_n3	IBM Tokyo	0	0	0
basis_change_n3, teleportation_n3	IBM Paris	0	0	0
basis_change_n3, teleportation_n3	IBM Rochester	0	0	0
basis_change_n3, teleportation_n3	Google Bristlecone	0	0	0

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
adder_n4	IBM Athens	2	2	2
adder_n4	Starmon-5	1	1	1
adder_n4	IBM Yorktown	1	1	1
adder_n4	IBM Ourense	2	2	2
adder_n4	Surface-7	0	0	0
adder_n4	IBM Casablanca	2	2	2
adder_n4	Rigetti Agave	2	2	2
adder_n4	IBM Melbourne	0	0	0
adder_n4	Rigetti Aspen-1	0	0	0
adder_n4	Surface-17	0	0	0
adder_n4	IBM Singapore	2	2	2
adder_n4	IBM Johannesburg	2	2	2
adder_n4	IBM Tokyo	0	0	0
adder_n4	IBM Paris	1	4	4
adder_n4	IBM Rochester	2	2	2

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
adder_n4	IBM Athens	2	2	10
adder_n4	Starmon-5	1	1	2
adder_n4	IBM Yorktown	1	1	1
adder_n4	IBM Ourense	2	2	2
adder_n4	Surface-7	0	0	0
adder_n4	IBM Casablanca	2	2	4
adder_n4	Rigetti Agave	2	2	2
adder_n4	IBM Melbourne	0	0	0
adder_n4	Rigetti Aspen-1	0	0	0
adder_n4	Surface-17	0	0	0
adder_n4	IBM Singapore	2	2	7
adder_n4	IBM Johannesburg	2	2	4
adder_n4	IBM Tokyo	0	0	0
adder_n4	IBM Paris	4	4	3
adder_n4	IBM Rochester	2	2	2
adder_n4	Google Bristlecone	0	0	0

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
variational_n4, bell_n4	IBM Athens	0	0	0
variational_n4, bell_n4	Starmon-5	2	1	2
variational_n4, bell_n4	IBM Yorktown	0	0	0
variational_n4, bell_n4	IBM Ourense	0	0	0
variational_n4, bell_n4	Surface-7	0	0	0
variational_n4, bell_n4	IBM Casablanca	0	0	0
variational_n4, bell_n4	Rigetti Agave	0	0	0
variational_n4, bell_n4	IBM Melbourne	0	0	0
variational_n4, bell_n4	Rigetti Aspen-1	0	0	0
variational_n4, bell_n4	Surface-17	0	0	0
variational_n4, bell_n4	IBM Singapore	0	0	0
variational_n4, bell_n4	IBM Johannesburg	0	0	0
variational_n4, bell_n4	IBM Tokyo	0	0	0
variational_n4, bell_n4	IBM Paris	0	0	0
variational_n4, bell_n4	IBM Rochester	0	0	0
variational_n4, bell_n4	Google Bristlecone	0	0	0

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
variational_n4, bell_n4	IBM Athens	0	0	0
variational_n4, bell_n4	Starmon-5	2	2	1
variational_n4, bell_n4	IBM Yorktown	0	0	0
variational_n4, bell_n4	IBM Ourense	0	0	0
variational_n4, bell_n4	Surface-7	0	0	0
variational_n4, bell_n4	IBM Casablanca	0	0	0
variational_n4, bell_n4	Rigetti Agave	0	0	0
variational_n4, bell_n4	IBM Melbourne	0	0	0
variational_n4, bell_n4	Rigetti Aspen-1	0	0	0
variational_n4, bell_n4	Surface-17	0	0	0
variational_n4, bell_n4	IBM Singapore	0	0	0
variational_n4, bell_n4	IBM Johannesburg	0	0	0
variational_n4, bell_n4	IBM Tokyo	0	0	0
variational_n4, bell_n4	IBM Paris	0	0	0
variational_n4, bell_n4	IBM Rochester	0	0	0
variational_n4, bell_n4	Google Bristlecone	0	0	0

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
cuccaro adder 1b	IBM Athens	1	1	1
cuccaro adder 1b	Starmon-5	1	1	1
cuccaro adder 1b	IBM Yorktown	0	0	0
cuccaro adder 1b	IBM Ourense	1	1	1
cuccaro adder 1b	Surface-7	1	1	1
cuccaro adder 1b	IBM Casablanca	1	1	1
cuccaro adder 1b	Rigetti Agave	1	1	1
cuccaro adder 1b	IBM Melbourne	1	1	1
cuccaro adder 1b	Rigetti Aspen-1	1	1	1
cuccaro adder 1b	Surface-17	1	1	1
cuccaro adder 1b	IBM Singapore	1	1	1
cuccaro adder 1b	IBM Johannesburg	1	1	1
cuccaro adder 1b	IBM Tokyo	0	0	0
cuccaro adder 1b	IBM Paris	1	1	1
cuccaro adder 1b	IBM Rochester	1	1	1
cuccaro adder 1b	Google Bristlecone	1	1	1

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
cuccaro adder 1b	IBM Athens	1	1	1
cuccaro adder 1b	Starmon-5	1	1	2
cuccaro adder 1b	IBM Yorktown	0	0	0
cuccaro adder 1b	IBM Ourense	1	1	3
cuccaro adder 1b	Surface-7	1	1	1
cuccaro adder 1b	IBM Casablanca	1	1	1
cuccaro adder 1b	Rigetti Agave	1	1	4
cuccaro adder 1b	IBM Melbourne	1	1	2
cuccaro adder 1b	Rigetti Aspen-1	1	1	1
cuccaro adder 1b	Surface-17	1	1	3
cuccaro adder 1b	IBM Singapore	1	1	1
cuccaro adder 1b	IBM Johannesburg	1	1	2
cuccaro adder 1b	IBM Tokyo	0	0	0
cuccaro adder 1b	IBM Paris	1	1	3
cuccaro adder 1b	IBM Rochester	1	1	1
cuccaro adder 1b	Google Bristlecone	1	1	2

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Athens	3	3	3
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Starmon-5	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Yorktown	1	1	1
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Ourense	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Surface-7	1	1	1
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Casablanca	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Rigetti Agave	3	3	3
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Melbourne	1	1	1
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Rigetti Aspen-1	1	1	1
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Surface-17	1	1	1
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Singapore	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Johannesburg	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Tokyo	0	0	0
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Paris	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Rochester	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Google Bristlecone	1	1	1

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Athens	3	4	3
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Starmon-5	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Yorktown	1	1	1
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Ourense	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Surface-7	1	1	1
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Casablanca	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Rigetti Agave	3	3	4
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Melbourne	1	1	1
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Rigetti Aspen-1	1	1	1
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Surface-17	1	1	1
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Singapore	2	2	3
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Johannesburg	2	2	3
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Tokyo	0	0	0
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Paris	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	IBM Rochester	2	2	2
q=4_s=19996_2qbf=02_1, q=4_s=2996_2qbf=08_1	Google Bristlecone	1	1	1

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
vbe_adder_1b	IBM Athens	1	1	1
vbe_adder_1b	Starmon-5	2	2	2
vbe_adder_1b	IBM Yorktown	1	2	2
vbe_adder_1b	IBM Ourense	1	1	1
vbe_adder_1b	Surface-7	1	1	1
vbe_adder_1b	IBM Casablanca	1	1	1
vbe_adder_1b	Rigetti Agave	1	1	1
vbe_adder_1b	IBM Melbourne	1	1	1
vbe_adder_1b	Rigetti Aspen-1	1	1	1
vbe_adder_1b	Surface-17	1	1	1
vbe_adder_1b	IBM Singapore	1	1	1
vbe_adder_1b	IBM Johannesburg	1	1	1
vbe_adder_1b	IBM Tokyo	0	0	0
vbe_adder_1b	IBM Paris	1	1	1
vbe_adder_1b	IBM Rochester	1	1	1
vbe_adder_1b	Google Bristlecone	1	1	1

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
vbe_adder_1b	IBM Athens	1	1	10
vbe_adder_1b	Starmon-5	2	2	2
vbe_adder_1b	IBM Yorktown	2	2	1
vbe_adder_1b	IBM Ourense	1	1	3
vbe_adder_1b	Surface-7	1	1	1
vbe_adder_1b	IBM Casablanca	1	1	2
vbe_adder_1b	Rigetti Agave	1	1	5
vbe_adder_1b	IBM Melbourne	1	1	1
vbe_adder_1b	Rigetti Aspen-1	1	1	1
vbe_adder_1b	Surface-17	1	1	1
vbe_adder_1b	IBM Singapore	1	1	3
vbe_adder_1b	IBM Johannesburg	1	1	2
vbe_adder_1b	IBM Tokyo	0	0	0
vbe_adder_1b	IBM Paris	1	1	4
vbe_adder_1b	IBM Rochester	1	1	1
vbe_adder_1b	Google Bristlecone	1	1	1

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Athens	6	6	7
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Starmon-5	3	3	3
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Yorktown	2	2	2
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Ourense	5	4	4
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Surface-7	3	3	3
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Casablanca	5	4	5
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Rigetti Agave	6	6	6
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Melbourne	3	3	3
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Rigetti Aspen-1	3	3	3
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Surface-17	3	3	3
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Singapore	5	4	4
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Johannesburg	5	4	4
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Tokyo	1	1	1
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Paris	5	4	4
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Rochester	5	4	4
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Google Bristlecone	3	3	3

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Athens	7	6	46
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Starmon-5	3	3	3
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Yorktown	2	2	2
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Ourense	4	4	8
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Surface-7	3	3	9
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Casablanca	5	5	31
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Rigetti Agave	6	7	9
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Melbourne	3	3	5
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Rigetti Aspen-1	3	3	11
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Surface-17	3	3	3
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Singapore	4	5	8
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Johannesburg	4	5	7
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Tokyo	1	1	5
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Paris	4	5	8
4gt10-v1_81, q=5_s=2995_2qbf=09_1	IBM Rochester	4	4	6
4gt10-v1_81, q=5_s=2995_2qbf=09_1	Google Bristlecone	3	3	23

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
4gt13_92	IBM Athens	2	2	2
4gt13_92	Starmon-5	2	2	2
4gt13_92	IBM Yorktown	2	2	2
4gt13_92	IBM Ourense	2	2	2
4gt13_92	Surface-7	2	2	2
4gt13_92	IBM Casablanca	2	2	2
4gt13_92	Rigetti Agave	2	2	2
4gt13_92	IBM Melbourne	3	3	3
4gt13_92	Rigetti Aspen-1	3	3	3
4gt13_92	Surface-17	2	2	2
4gt13_92	IBM Singapore	3	3	3
4gt13_92	IBM Johannesburg	3	3	3
4gt13_92	IBM Tokyo	0	0	0
4gt13_92	IBM Paris	4	4	4
4gt13_92	IBM Rochester	3	3	3
4gt13_92	Google Bristlecone	2	2	2

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
4gt13_92	IBM Athens	2	2	5
4gt13_92	Starmon-5	2	2	6
4gt13_92	IBM Yorktown	2	2	1
4gt13_92	IBM Ourense	2	2	33
4gt13_92	Surface-7	2	2	4
4gt13_92	IBM Casablanca	2	2	11
4gt13_92	Rigetti Agave	2	2	7
4gt13_92	IBM Melbourne	3	3	5
4gt13_92	Rigetti Aspen-1	3	3	2
4gt13_92	Surface-17	2	2	3
4gt13_92	IBM Singapore	3	3	3
4gt13_92	IBM Johannesburg	3	3	3
4gt13_92	IBM Tokyo	0	0	0
4gt13_92	IBM Paris	4	4	8
4gt13_92	IBM Rochester	3	3	2
4gt13_92	Google Bristlecone	2	2	5

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
4gt5_75	IBM Athens	4	4	6
4gt5_75	Starmon-5	3	3	3
4gt5_75	IBM Yorktown	2	3	3
4gt5_75	IBM Ourense	3	3	3
4gt5_75	Surface-7	2	2	2
4gt5_75	IBM Casablanca	3	3	3
4gt5_75	Rigetti Agave	4	4	6
4gt5_75	IBM Melbourne	2	2	2
4gt5_75	Rigetti Aspen-1	2	2	2
4gt5_75	Surface-17	2	2	2
4gt5_75	IBM Singapore	3	3	3
4gt5_75	IBM Johannesburg	3	4	4
4gt5_75	IBM Tokyo	2	3	1
4gt5_75	IBM Paris	3	3	3
4gt5_75	IBM Rochester	3	3	3
4gt5_75	Google Bristlecone	2	2	2

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
4gt5_75	IBM Athens	6	6	13
4gt5_75	Starmon-5	3	3	4
4gt5_75	IBM Yorktown	3	3	4
4gt5_75	IBM Ourense	3	3	11
4gt5_75	Surface-7	2	3	6
4gt5_75	IBM Casablanca	3	4	10
4gt5_75	Rigetti Agave	6	4	12
4gt5_75	IBM Melbourne	2	3	8
4gt5_75	Rigetti Aspen-1	2	2	2
4gt5_75	Surface-17	2	2	2
4gt5_75	IBM Singapore	3	3	6
4gt5_75	IBM Johannesburg	4	4	9
4gt5_75	IBM Tokyo	1	3	5
4gt5_75	IBM Paris	3	4	6
4gt5_75	IBM Rochester	3	4	5
4gt5_75	Google Bristlecone	2	2	2

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
4gt5_75	IBM Athens	6	6	13
4gt5_75	Starmon-5	3	3	4
4gt5_75	IBM Yorktown	3	3	4
4gt5_75	IBM Ourense	3	3	11
4gt5_75	Surface-7	2	3	6
4gt5_75	IBM Casablanca	3	4	10
4gt5_75	Rigetti Agave	6	4	12
4gt5_75	IBM Melbourne	2	3	8
4gt5_75	Rigetti Aspen-1	2	2	2
4gt5_75	Surface-17	2	2	2
4gt5_75	IBM Singapore	3	3	6
4gt5_75	IBM Johannesburg	4	4	9
4gt5_75	IBM Tokyo	1	3	5
4gt5_75	IBM Paris	3	4	6
4gt5_75	IBM Rochester	3	4	5
4gt5_75	Google Bristlecone	2	2	2

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
alu-v1_28	IBM Athens	3	3	3
alu-v1_28	Starmon-5	2	2	2
alu-v1_28	IBM Yorktown	3	3	3
alu-v1_28	IBM Ourense	2	2	2
alu-v1_28	Surface-7	2	2	2
alu-v1_28	IBM Casablanca	2	2	2
alu-v1_28	Rigetti Agave	3	3	3
alu-v1_28	IBM Melbourne	1	3	3
alu-v1_28	Rigetti Aspen-1	1	3	3
alu-v1_28	Surface-17	2	2	2
alu-v1_28	IBM Singapore	4	4	4
alu-v1_28	IBM Johannesburg	4	4	4
alu-v1_28	IBM Tokyo	1	1	1
alu-v1_28	IBM Paris	4	4	4
alu-v1_28	IBM Rochester	3	3	3
alu-v1_28	Google Bristlecone	1	3	4

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
alu-v1_28	IBM Athens	3	7	14
alu-v1_28	Starmon-5	2	2	5
alu-v1_28	IBM Yorktown	3	2	3
alu-v1_28	IBM Ourense	2	2	16
alu-v1_28	Surface-7	2	2	3
alu-v1_28	IBM Casablanca	2	2	6
alu-v1_28	Rigetti Agave	3	3	3
alu-v1_28	IBM Melbourne	3	1	1
alu-v1_28	Rigetti Aspen-1	3	3	2
alu-v1_28	Surface-17	2	2	3
alu-v1_28	IBM Singapore	4	4	6
alu-v1_28	IBM Johannesburg	4	4	6
alu-v1_28	IBM Tokyo	1	1	34
alu-v1_28	IBM Paris	4	4	16
alu-v1_28	IBM Rochester	3	3	4
alu-v1_28	Google Bristlecone	4	3	3

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
alu-v2_31	IBM Athens	6	6	6
alu-v2_31	Starmon-5	2	2	2
alu-v2_31	IBM Yorktown	3	4	4
alu-v2_31	IBM Ourense	5	4	4
alu-v2_31	Surface-7	3	3	3
alu-v2_31	IBM Casablanca	5	4	4
alu-v2_31	Rigetti Agave	6	6	6
alu-v2_31	IBM Melbourne	3	3	3
alu-v2_31	Rigetti Aspen-1	3	3	3
alu-v2_31	Surface-17	3	3	3
alu-v2_31	IBM Singapore	3	3	3
alu-v2_31	IBM Johannesburg	3	3	3
alu-v2_31	IBM Tokyo	1	1	1
alu-v2_31	IBM Paris	3	4	4
alu-v2_31	IBM Rochester	3	3	3
alu-v2_31	Google Bristlecone	3	3	3

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
alu-v2_31	IBM Athens	6	7	17
alu-v2_31	Starmon-5	2	2	4
alu-v2_31	IBM Yorktown	4	3	9
alu-v2_31	IBM Ourense	4	4	7
alu-v2_31	Surface-7	3	3	15
alu-v2_31	IBM Casablanca	4	4	5
alu-v2_31	Rigetti Agave	6	6	14
alu-v2_31	IBM Melbourne	3	3	15
alu-v2_31	Rigetti Aspen-1	3	3	5
alu-v2_31	Surface-17	3	3	7
alu-v2_31	IBM Singapore	3	4	5
alu-v2_31	IBM Johannesburg	3	4	8
alu-v2_31	IBM Tokyo	1	1	3
alu-v2_31	IBM Paris	4	4	12
alu-v2_31	IBM Rochester	3	3	7
alu-v2_31	Google Bristlecone	3	3	6

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
decod24-v1_41	IBM Athens	4	4	4
decod24-v1_41	Starmon-5	3	3	3
decod24-v1_41	IBM Yorktown	3	3	3
decod24-v1_41	IBM Ourense	3	3	3
decod24-v1_41	Surface-7	2	2	2
decod24-v1_41	IBM Casablanca	3	3	3
decod24-v1_41	Rigetti Agave	4	4	4
decod24-v1_41	IBM Melbourne	1	1	1
decod24-v1_41	Rigetti Aspen-1	1	1	1
decod24-v1_41	Surface-17	2	2	2
decod24-v1_41	IBM Singapore	3	3	3
decod24-v1_41	IBM Johannesburg	4	4	4
decod24-v1_41	IBM Tokyo	1	1	1
decod24-v1_41	IBM Paris	4	4	4
decod24-v1_41	IBM Rochester	3	3	3
decod24-v1_41	Google Bristlecone	1	1	1

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
decod24-v1_41	IBM Athens	4	3	8
decod24-v1_41	Starmon-5	3	3	4
decod24-v1_41	IBM Yorktown	3	3	9
decod24-v1_41	IBM Ourense	3	3	8
decod24-v1_41	Surface-7	2	2	3
decod24-v1_41	IBM Casablanca	3	3	5
decod24-v1_41	Rigetti Agave	4	6	16
decod24-v1_41	IBM Melbourne	1	3	6
decod24-v1_41	Rigetti Aspen-1	1	2	3
decod24-v1_41	Surface-17	2	2	3
decod24-v1_41	IBM Singapore	3	3	8
decod24-v1_41	IBM Johannesburg	4	3	11
decod24-v1_41	IBM Tokyo	1	1	2
decod24-v1_41	IBM Paris	4	4	7
decod24-v1_41	IBM Rochester	3	2	9
decod24-v1_41	Google Bristlecone	1	3	1

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
error_correctiond3_n5	IBM Athens	3	2	2
error_correctiond3_n5	Starmon-5	1	1	1
error_correctiond3_n5	IBM Yorktown	0	0	0
error_correctiond3_n5	IBM Ourense	2	2	2
error_correctiond3_n5	Surface-7	1	1	1
error_correctiond3_n5	IBM Casablanca	2	2	2
error_correctiond3_n5	Rigetti Agave	3	2	2
error_correctiond3_n5	IBM Melbourne	1	1	1
error_correctiond3_n5	Rigetti Aspen-1	1	1	1
error_correctiond3_n5	Surface-17	1	1	1
error_correctiond3_n5	IBM Singapore	2	1	1
error_correctiond3_n5	IBM Johannesburg	1	1	1
error_correctiond3_n5	IBM Tokyo	0	0	0
error_correctiond3_n5	IBM Paris	2	2	2
error_correctiond3_n5	IBM Rochester	1	1	1
error_correctiond3_n5	Google Bristlecone	1	1	1

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
error_correctiond3_n5	IBM Athens	2	2	3
error_correctiond3_n5	Starmon-5	1	1	1
error_correctiond3_n5	IBM Yorktown	0	0	0
error_correctiond3_n5	IBM Ourense	2	2	3
error_correctiond3_n5	Surface-7	1	1	2
error_correctiond3_n5	IBM Casablanca	2	2	5
error_correctiond3_n5	Rigetti Agave	2	2	2
error_correctiond3_n5	IBM Melbourne	1	1	1
error_correctiond3_n5	Rigetti Aspen-1	1	1	7
error_correctiond3_n5	Surface-17	1	1	5
error_correctiond3_n5	IBM Singapore	1	1	10
error_correctiond3_n5	IBM Johannesburg	1	1	3
error_correctiond3_n5	IBM Tokyo	0	0	0
error_correctiond3_n5	IBM Paris	2	2	2
error_correctiond3_n5	IBM Rochester	1	1	1
error_correctiond3_n5	Google Bristlecone	1	1	1

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
qec_en_n5, qec_sm_n5	IBM Athens	2	3	2
qec_en_n5, qec_sm_n5	Starmon-5	0	0	0
qec_en_n5, qec_sm_n5	IBM Yorktown	1	1	1
qec_en_n5, qec_sm_n5	IBM Ourense	1	1	1
qec_en_n5, qec_sm_n5	Surface-7	0	0	0
qec_en_n5, qec_sm_n5	IBM Casablanca	1	1	1
qec_en_n5, qec_sm_n5	Rigetti Agave	2	3	3
qec_en_n5, qec_sm_n5	IBM Melbourne	1	1	1
qec_en_n5, qec_sm_n5	Rigetti Aspen-1	1	1	1
qec_en_n5, qec_sm_n5	Surface-17	0	0	0
qec_en_n5, qec_sm_n5	IBM Singapore	1	1	1
qec_en_n5, qec_sm_n5	IBM Johannesburg	2	2	2
qec_en_n5, qec_sm_n5	IBM Tokyo	1	1	1
qec_en_n5, qec_sm_n5	IBM Paris	1	1	1
qec_en_n5, qec_sm_n5	IBM Rochester	1	1	1
qec_en_n5, qec_sm_n5	Google Bristlecone	0	0	0

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
qec_en_n5, qec_sm_n5	IBM Athens	2	2	14
qec_en_n5, qec_sm_n5	Starmon-5	0	0	0
qec_en_n5, qec_sm_n5	IBM Yorktown	1	1	4
qec_en_n5, qec_sm_n5	IBM Ourense	1	1	1
qec_en_n5, qec_sm_n5	Surface-7	0	0	0
qec_en_n5, qec_sm_n5	IBM Casablanca	1	1	8
qec_en_n5, qec_sm_n5	Rigetti Agave	3	3	2
qec_en_n5, qec_sm_n5	IBM Melbourne	1	1	1
qec_en_n5, qec_sm_n5	Rigetti Aspen-1	1	1	1
qec_en_n5, qec_sm_n5	Surface-17	0	0	0
qec_en_n5, qec_sm_n5	IBM Singapore	1	1	3
qec_en_n5, qec_sm_n5	IBM Johannesburg	2	2	4
qec_en_n5, qec_sm_n5	IBM Tokyo	1	1	2
qec_en_n5, qec_sm_n5	IBM Paris	1	1	1
qec_en_n5, qec_sm_n5	IBM Rochester	1	1	9
qec_en_n5, qec_sm_n5	Google Bristlecone	0	0	0

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
quantum_volume_n5	IBM Athens	6	6	6
quantum_volume_n5	Starmon-5	2	2	2
quantum_volume_n5	IBM Yorktown	2	2	2
quantum_volume_n5	IBM Ourense	3	4	4
quantum_volume_n5	Surface-7	3	4	4
quantum_volume_n5	IBM Casablanca	3	4	4
quantum_volume_n5	Rigetti Agave	6	6	6
quantum_volume_n5	IBM Melbourne	3	3	3
quantum_volume_n5	Rigetti Aspen-1	3	3	3
quantum_volume_n5	Surface-17	3	4	4
quantum_volume_n5	IBM Singapore	5	4	5
quantum_volume_n5	IBM Johannesburg	4	4	4
quantum_volume_n5	IBM Tokyo	1	1	1
quantum_volume_n5	IBM Paris	2	2	2
quantum_volume_n5	IBM Rochester	5	4	4
quantum_volume_n5	Google Bristlecone	3	3	3

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
quantum_volume_n5	IBM Athens	6	6	12
quantum_volume_n5	Starmon-5	2	2	2
quantum_volume_n5	IBM Yorktown	2	2	5
quantum_volume_n5	IBM Ourense	4	4	7
quantum_volume_n5	Surface-7	4	2	3
quantum_volume_n5	IBM Casablanca	4	4	5
quantum_volume_n5	Rigetti Agave	6	6	29
quantum_volume_n5	IBM Melbourne	3	3	3
quantum_volume_n5	Rigetti Aspen-1	3	4	4
quantum_volume_n5	Surface-17	4	3	3
quantum_volume_n5	IBM Singapore	5	5	5
quantum_volume_n5	IBM Johannesburg	4	4	11
quantum_volume_n5	IBM Tokyo	1	1	6
quantum_volume_n5	IBM Paris	2	2	5
quantum_volume_n5	IBM Rochester	4	4	7
quantum_volume_n5	Google Bristlecone	3	3	5

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
simon_n6	IBM Athens	1	1	1
simon_n6	Starmon-5	2	2	2
simon_n6	IBM Yorktown	2	2	2
simon_n6	IBM Ourense	1	1	1
simon_n6	Surface-7	3	3	3
simon_n6	IBM Casablanca	1	1	1
simon_n6	Rigetti Agave	1	1	1
simon_n6	IBM Melbourne	3	3	3
simon_n6	Rigetti Aspen-1	3	3	3
simon_n6	Surface-17	3	3	3
simon_n6	IBM Singapore	3	3	3
simon_n6	IBM Johannesburg	3	3	3
simon_n6	IBM Tokyo	0	0	0
simon_n6	IBM Paris	3	4	4
simon_n6	IBM Rochester	3	3	3
simon_n6	Google Bristlecone	3	3	3

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
simon_n6	IBM Athens	1	1	20
simon_n6	Starmon-5	2	2	2
simon_n6	IBM Yorktown	2	2	4
simon_n6	IBM Ourense	1	1	1
simon_n6	Surface-7	3	3	5
simon_n6	IBM Casablanca	1	1	2
simon_n6	Rigetti Agave	1	1	1
simon_n6	IBM Melbourne	3	2	4
simon_n6	Rigetti Aspen-1	3	3	2
simon_n6	Surface-17	3	3	3
simon_n6	IBM Singapore	3	3	6
simon_n6	IBM Johannesburg	3	2	2
simon_n6	IBM Tokyo	0	0	0
simon_n6	IBM Paris	4	4	6
simon_n6	IBM Rochester	3	3	10
simon_n6	Google Bristlecone	3	3	5

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
alu-v2_30,q=6_s=2994_2qbf=08_1	Surface-7	4	4	4
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Casablanca	9	9	9
alu-v2_30,q=6_s=2994_2qbf=08_1	Rigetti Agave	11	11	11
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Melbourne	4	4	4
alu-v2_30,q=6_s=2994_2qbf=08_1	Rigetti Aspen-1	6	6	6
alu-v2_30,q=6_s=2994_2qbf=08_1	Surface-17	4	4	5
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Singapore	5	5	5
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Johannesburg	5	5	5
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Tokyo	4	2	2
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Paris	7	7	7
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Rochester	6	7	7
alu-v2_30,q=6_s=2994_2qbf=08_1	Google Bristlecone	4	5	6

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
alu-v2_30,q=6_s=2994_2qbf=08_1	Surface-7	4	12	13
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Casablanca	9	9	46
alu-v2_30,q=6_s=2994_2qbf=08_1	Rigetti Agave	11	11	35
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Melbourne	4	6	11
alu-v2_30,q=6_s=2994_2qbf=08_1	Rigetti Aspen-1	6	6	14
alu-v2_30,q=6_s=2994_2qbf=08_1	Surface-17	5	4	14
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Singapore	5	11	12
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Johannesburg	5	7	12
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Tokyo	2	5	7
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Paris	7	7	22
alu-v2_30,q=6_s=2994_2qbf=08_1	IBM Rochester	7	7	25
alu-v2_30,q=6_s=2994_2qbf=08_1	Google Bristlecone	6	6	8

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
4gt12-v0_87	Surface-7	3	3	3
4gt12-v0_87	IBM Casablanca	5	5	7
4gt12-v0_87	Rigetti Agave	7	10	10
4gt12-v0_87	IBM Melbourne	2	2	2
4gt12-v0_87	Rigetti Aspen-1	3	3	3
4gt12-v0_87	Surface-17	3	3	3
4gt12-v0_87	IBM Singapore	5	5	5
4gt12-v0_87	IBM Johannesburg	5	5	5
4gt12-v0_87	IBM Tokyo	2	3	3
4gt12-v0_87	IBM Paris	7	6	6
4gt12-v0_87	IBM Rochester	6	6	6
4gt12-v0_87	Google Bristlecone	3	4	4

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
4gt12-v0_87	Surface-7	3	3	7
4gt12-v0_87	IBM Casablanca	7	5	18
4gt12-v0_87	Rigetti Agave	10	10	21
4gt12-v0_87	IBM Melbourne	2	5	16
4gt12-v0_87	Rigetti Aspen-1	3	7	9
4gt12-v0_87	Surface-17	3	7	5
4gt12-v0_87	IBM Singapore	5	5	13
4gt12-v0_87	IBM Johannesburg	5	7	4
4gt12-v0_87	IBM Tokyo	3	2	21
4gt12-v0_87	IBM Paris	6	7	27
4gt12-v0_87	IBM Rochester	6	6	18
4gt12-v0_87	Google Bristlecone	4	6	14

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
4gt4-v0_72	Surface-7	4	6	6
4gt4-v0_72	IBM Casablanca	7	8	9
4gt4-v0_72	Rigetti Agave	8	10	10
4gt4-v0_72	IBM Melbourne	4	4	4
4gt4-v0_72	Rigetti Aspen-1	6	7	7
4gt4-v0_72	Surface-17	3	3	4
4gt4-v0_72	IBM Singapore	5	5	5
4gt4-v0_72	IBM Johannesburg	5	5	5
4gt4-v0_72	IBM Tokyo	4	2	2
4gt4-v0_72	IBM Paris	7	7	7
4gt4-v0_72	IBM Rochester	8	7	7
4gt4-v0_72	Google Bristlecone	2	2	2

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
4gt4-v0_72	Surface-7	6	5	5
4gt4-v0_72	IBM Casablanca	9	7	52
4gt4-v0_72	Rigetti Agave	10	16	10
4gt4-v0_72	IBM Melbourne	4	4	10
4gt4-v0_72	Rigetti Aspen-1	7	6	12
4gt4-v0_72	Surface-17	4	3	4
4gt4-v0_72	IBM Singapore	5	5	10
4gt4-v0_72	IBM Johannesburg	5	5	29
4gt4-v0_72	IBM Tokyo	2	2	12
4gt4-v0_72	IBM Paris	7	7	10
4gt4-v0_72	IBM Rochester	7	7	20
4gt4-v0_72	Google Bristlecone	2	2	10

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
qaoa 6	Surface-7	4	7	7
qaoa 6	IBM Casablanca	7	8	8
qaoa 6	Rigetti Agave	10	7	7
qaoa 6	IBM Melbourne	3	4	5
qaoa 6	Rigetti Aspen-1	4	5	5
qaoa 6	Surface-17	2	3	3
qaoa 6	IBM Singapore	5	4	4
qaoa 6	IBM Johannesburg	5	4	4
qaoa 6	IBM Tokyo	3	4	4
qaoa 6	IBM Paris	6	7	7
qaoa 6	IBM Rochester	6	7	7
qaoa 6	Google Bristlecone	3	4	5

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
qaoa 6	Surface-7	7	6	10
qaoa 6	IBM Casablanca	8	8	14
qaoa 6	Rigetti Agave	7	7	49
qaoa 6	IBM Melbourne	5	5	13
qaoa 6	Rigetti Aspen-1	5	5	33
qaoa 6	Surface-17	3	5	4
qaoa 6	IBM Singapore	4	4	7
qaoa 6	IBM Johannesburg	4	5	6
qaoa 6	IBM Tokyo	4	4	10
qaoa 6	IBM Paris	7	7	8
qaoa 6	IBM Rochester	7	7	35
qaoa 6	Google Bristlecone	5	5	6

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
ex3_229	Surface-7	4	4	4
ex3_229	IBM Casablanca	7	8	8
ex3_229	Rigetti Agave	9	11	11
ex3_229	IBM Melbourne	2	2	2
ex3_229	Rigetti Aspen-1	5	5	5
ex3_229	Surface-17	2	5	6
ex3_229	IBM Singapore	3	3	3
ex3_229	IBM Johannesburg	3	3	3
ex3_229	IBM Tokyo	2	2	2
ex3_229	IBM Paris	4	4	4
ex3_229	IBM Rochester	6	6	6
ex3_229	Google Bristlecone	2	2	2

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
ex3_229	Surface-7	4	4	7
ex3_229	IBM Casablanca	8	6	23
ex3_229	Rigetti Agave	11	11	37
ex3_229	IBM Melbourne	2	2	3
ex3_229	Rigetti Aspen-1	5	5	10
ex3_229	Surface-17	6	4	3
ex3_229	IBM Singapore	3	3	17
ex3_229	IBM Johannesburg	3	3	25
ex3_229	IBM Tokyo	2	2	2
ex3_229	IBM Paris	4	4	10
ex3_229	IBM Rochester	6	6	16
ex3_229	Google Bristlecone	2	2	7

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
graycode6_47	Surface-7	1	1	1
graycode6_47	IBM Casablanca	1	1	1
graycode6_47	Rigetti Agave	0	0	0
graycode6_47	IBM Melbourne	0	0	0
graycode6_47	Rigetti Aspen-1	0	0	0
graycode6_47	Surface-17	0	0	0
graycode6_47	IBM Singapore	0	0	0
graycode6_47	IBM Johannesburg	0	0	0
graycode6_47	IBM Tokyo	0	0	0
graycode6_47	IBM Paris	0	0	0
graycode6_47	IBM Rochester	0	0	0
graycode6_47	Google Bristlecone	0	0	0

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
graycode6_47	Surface-7	1	1	3
graycode6_47	IBM Casablanca	1	1	3
graycode6_47	Rigetti Agave	0	0	0
graycode6_47	IBM Melbourne	0	0	0
graycode6_47	Rigetti Aspen-1	0	0	0
graycode6_47	Surface-17	0	0	0
graycode6_47	IBM Singapore	0	0	0
graycode6_47	IBM Johannesburg	0	0	0
graycode6_47	IBM Tokyo	0	0	0
graycode6_47	IBM Paris	0	0	0
graycode6_47	IBM Rochester	0	0	0
graycode6_47	Google Bristlecone	0	0	0

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
mod5adder_127	Surface-7	4	6	6
mod5adder_127	IBM Casablanca	7	8	8
mod5adder_127	Rigetti Agave	8	13	14
mod5adder_127	IBM Melbourne	4	5	5
mod5adder_127	Rigetti Aspen-1	5	7	7
mod5adder_127	Surface-17	4	5	6
mod5adder_127	IBM Singapore	5	5	5
mod5adder_127	IBM Johannesburg	5	5	5
mod5adder_127	IBM Tokyo	4	2	2
mod5adder_127	IBM Paris	7	8	8
mod5adder_127	IBM Rochester	8	7	7
mod5adder_127	Google Bristlecone	3	3	4

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
mod5adder_127	Surface-7	6	4	16
mod5adder_127	IBM Casablanca	8	9	43
mod5adder_127	Rigetti Agave	14	13	44
mod5adder_127	IBM Melbourne	5	6	9
mod5adder_127	Rigetti Aspen-1	7	6	14
mod5adder_127	Surface-17	6	4	13
mod5adder_127	IBM Singapore	5	5	11
mod5adder_127	IBM Johannesburg	5	7	11
mod5adder_127	IBM Tokyo	2	3	19
mod5adder_127	IBM Paris	8	8	20
mod5adder_127	IBM Rochester	7	7	26
mod5adder_127	Google Bristlecone	4	4	11

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
q=6_s=54_2qbf=022_1	Surface-7	4	4	4
q=6_s=54_2qbf=022_1	IBM Casablanca	7	7	8
q=6_s=54_2qbf=022_1	Rigetti Agave	5	9	9
q=6_s=54_2qbf=022_1	IBM Melbourne	2	2	2
q=6_s=54_2qbf=022_1	Rigetti Aspen-1	3	5	5
q=6_s=54_2qbf=022_1	Surface-17	3	3	3
q=6_s=54_2qbf=022_1	IBM Singapore	3	3	4
q=6_s=54_2qbf=022_1	IBM Johannesburg	3	3	3
q=6_s=54_2qbf=022_1	IBM Tokyo	4	4	4
q=6_s=54_2qbf=022_1	IBM Paris	4	8	7
q=6_s=54_2qbf=022_1	IBM Rochester	5	5	6
q=6_s=54_2qbf=022_1	Google Bristlecone	2	2	2

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
q=6_s=54_2qbf=022_1	Surface-7	4	6	20
q=6_s=54_2qbf=022_1	IBM Casablanca	8	7	17
q=6_s=54_2qbf=022_1	Rigetti Agave	9	9	12
q=6_s=54_2qbf=022_1	IBM Melbourne	2	2	7
q=6_s=54_2qbf=022_1	Rigetti Aspen-1	5	5	6
q=6_s=54_2qbf=022_1	Surface-17	3	4	7
q=6_s=54_2qbf=022_1	IBM Singapore	4	3	9
q=6_s=54_2qbf=022_1	IBM Johannesburg	3	3	3
q=6_s=54_2qbf=022_1	IBM Tokyo	4	4	16
q=6_s=54_2qbf=022_1	IBM Paris	7	8	42
q=6_s=54_2qbf=022_1	IBM Rochester	6	5	16
q=6_s=54_2qbf=022_1	Google Bristlecone	2	2	3

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
sf_274	Surface-7	4	4	4
sf_274	IBM Casablanca	6	6	6
sf_274	Rigetti Agave	4	4	4
sf_274	IBM Melbourne	3	3	3
sf_274	Rigetti Aspen-1	2	3	3
sf_274	Surface-17	3	3	3
sf_274	IBM Singapore	5	5	5
sf_274	IBM Johannesburg	5	5	5
sf_274	IBM Tokyo	3	3	3
sf_274	IBM Paris	7	12	8

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
sf_274	Surface-7	4	4	4
sf_274	IBM Casablanca	6	6	12
sf_274	Rigetti Agave	4	4	12
sf_274	IBM Melbourne	3	3	6
sf_274	Rigetti Aspen-1	3	3	2
sf_274	Surface-17	3	3	6
sf_274	IBM Singapore	5	6	9
sf_274	IBM Johannesburg	5	5	31
sf_274	IBM Tokyo	3	3	42
sf_274	IBM Paris	8	8	14

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
xor5_254	Surface-7	3	-	25
xor5_254	IBM Casablanca	3	4	4
xor5_254	Rigetti Agave	4	4	4
xor5_254	IBM Melbourne	3	3	3
xor5_254	Rigetti Aspen-1	3	3	3
xor5_254	Surface-17	2	2	2
xor5_254	IBM Singapore	2	2	2
xor5_254	IBM Johannesburg	2	2	2
xor5_254	IBM Tokyo	1	1	1
xor5_254	IBM Paris	4	3	3
xor5_254	IBM Rochester	3	3	3
xor5_254	Google Bristlecone	2	2	2

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
xor5_254	Surface-7	25	7	7
xor5_254	IBM Casablanca	4	4	22
xor5_254	Rigetti Agave	4	4	15
xor5_254	IBM Melbourne	3	3	6
xor5_254	Rigetti Aspen-1	3	3	14
xor5_254	Surface-17	2	2	23
xor5_254	IBM Singapore	2	2	7
xor5_254	IBM Johannesburg	2	2	4
xor5_254	IBM Tokyo	1	1	15
xor5_254	IBM Paris	3	3	11
xor5_254	IBM Rochester	3	3	7
xor5_254	Google Bristlecone	2	2	3

Benchmark name	**Device name**	β=10^{0}	β=10^{-1}	β=10^{-2}
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	Surface-7	-	-	8
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Casablanca	-	-	-
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	Rigetti Agave	16	-	-
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Melbourne	-	7	7
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	Rigetti Aspen-1	11	10	10
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	Surface-17	7	9	9
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Singapore	12	11	13
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Johannesburg	12	11	12
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Tokyo	7	6	5
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Rochester	17	-	14

Benchmark name	**Device name**	β=10^{-3}	β=10^{-4}	β=10^{-5}
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	Surface-7	8	54	22
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Casablanca	-	40	48
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	Rigetti Agave	-	18	65
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Melbourne	7	7	18
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	Rigetti Aspen-1	10	9	25
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	Surface-17	9	7	18
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Singapore	13	10	34
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Johannesburg	12	12	17
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Tokyo	5	4	10
q=7_s=29993_2qbf=08_1, q=7_s=2993_2qbf=08_1	IBM Rochester	14	16	38