Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

The Quantum Approximate Optimization Algorithm for the Shortest Vector Problem

Author:
Max Al-Hasso

Supervisor:
Dr. Roberto Bondesan

Submitted in partial fulfillment of the requirements for the MSc degree in MSc Computing (Security and Reliablity) of Imperial College London

September 2023

Abstract

A sufficiently large quantum computer is capable of breaking the security of the most commonly used cryptographic systems. This has motivated the need to design new quantum-resistant cryptographic primitives in order to maintain the security of digital communication. Lattice-based cryptography is the most promising candidate for a secure, post-quantum cryptographic system, and the security of these systems is underpinned by lattice problems. The shortest vector problem (SVP) is among the most prominent of these lattice problems.

The current NISQ (noisy intermediate-scale quantum) era of quantum computing is characterized by quantum computers which are noisy and have limited qubits, meaning only small circuits can be implemented. Variational quantum algorithms are hybrid classical-quantum algorithms which make use of shallow, parameterized quantum circuits and classical optimization of the circuit parameters to solve optimization problems. Two leading variational quantum algorithms are the Variational Quantum Eigensolver (VQE) and the Quantum Approximate Optimization Algorithm (QAOA).

In this work, we explore how VQE and QAOA can be used to solve SVP using NISQ devices. In particular, introduce a novel encoding of SVP into QAOA which reduces the search space of the optimization problem compared to a previous encoding. We also present our experimental evaluation of this QAOA encoding and compare it to the performance of VQE.

Acknowledgments

I would like to express my gratitude to Dr. Roberto Bondesan for their invaluable mentorship throughout the course of this research. Their guidance, expertise, and support have been instrumental in shaping this project.

Contents

Chapter 1

Introduction

1.1 Post-Quantum Cryptography

Cryptographic systems have been used to provide secure communication for thousands of years and they continue to be of critical importance today as they serve as the backbone for the infrastructure of the internet, from online banking to text messaging. Modern cryptographic systems derive their security from utilising algorithms that exist at the limits of what is possible to compute efficiently, meaning only the intended recipient is able to accurately decrypt the information sent to them. Since the invention of public-key cryptography by Diffe, Hellman (1), and Merkle (2) it has been possible to establish secure channels of communication in the presence of eavesdroppers, tamperers, and without having to have first exchanged information. These channels allow for the internet to exist today as any given packet of data transmitted from one user to another will invariably be pass through many untrusted routers.

The security of the most popular public-key cryptograhic systems used today are dependent on the computational hardness assumptions of one of three underlying mathematical problem. These problems are integer factorization, the discrete logarithm problem, and the elliptic-curve discrete logarithm problem. RSA public-key encryption (3), ElGamel encryption (4), and the elliptic curve digital signature algorithm are based on each of these problems respectively. The issue with these systems is that the problems which they are based on are only intractable on classical computers. There are quantum algorithms which are able to efficiently solve each of these problems on a sufficiently large quantum computer. Post-quantum cryptography refers to cryptographic systems which are secure in the presence of both classical and large-scale quantum computers.

1.1.1 The Quantum Threat to Classical Cryptography

Shor's algorithm is a milestone quantum algorithm that efficiently factors large numbers and therefore poses a threat to the popular cryptographic systems in use today (5, 6). Shor also proposed a similar algorithm which solves the discrete logarithm problem. Shor's algorithm outperforms the best known classical algorithms for integer factorisation and operates in polylogarithmic time by exploiting quantum parallelism. Despite its groundbreaking potential, a practical implementation of Shor's algorithm on integers that are cryptographically relevant is still a considerable challenge due to the large number of qubits required. Additionally, these qubits must be perfectly fault tolerant as otherwise the factorisation fails (7).

While not as threatening as Shor's algorithm, Grover's algorithm provides a quadratic speedup for searching an unsorted database (8). This can be utilised to search through a list of possible encryption keys and therefore weaken symmetric cryptographic systems. Classical algorithms require N iterations to search N keys, whereas Grover's algorithm requires only $\sqrt{N}$, this halves the security offered by a given key length.

Despite the fact that quantum computers are not yet large enough to run these algorithms, there is already an urgent need to develop quantum-safe algorithms. Retrospective decryption, otherwise known as a 'harvest-now, decrypt-later' attack, is an approach to accessing sensitive data that is not temporally sensitive by storing encrypted data until a quantum computer capable of breaking the encryption is available.

1.1.2 Promising Approaches to Post-Quantum Cryptography

Due to the threat that quantum computers pose to classical cryptography, the study of post-quantum cryptography has emerged as an active area of research to ensure the security of future digital communication. Several promising approaches have been proposed in recent years (9) each basing their security in different hard mathematical problems, the most prominent of these being lattice-based cryptography, code-based cryptography, and hash-based cryptography. The U.S National Institute of Standards and Technology (NIST) has played a key role in coordinating research efforts and standardizing post-quantum cryptographic algorithms.

Lattice-based cryptography stands out as a leading candidate among the post-quantum cryptographic approaches. The primitives of lattice-based cryptographic systems are constructed from mathematical problems based on geometric structures called lattices, an $n-$dimensional grid of points that infinitely expand in every dimension. Of the encryption and digital signature schemes being evaluated by NIST, the majority are lattice-based due in part to the versatility of the primitives that can be created and used for encryption, key exchange, and digital signatures.

Code-based cryptography is a well-understood, quantum-resistant alternative approach to lattice-based systems. The main algorithm in consideration is the McEliece

cryptosystem (10), first proposed in 1978, in which the algorithmic primitive is derived from an error correcting code. Even in the presence of a sufficiently large quantum computer, there are no known threats to this system (11). The primary drawback, however, is the large size of the public key produced making this system comparatively impractical.

Hash-based cryptography is another promising approach to developing post-quantum cryptographic systems which use primitives based on cryptographic hash functions for their security. These functions take inputs of arbitrary length and generate fixed-length strings of bytes. The key properties of a hash function are collision resistance, pre-image resistance, and second pre-image resistance. This means that it is infeasible for two inputs to generate the same output, that an attacker cannot learn the input which generates a given output, and an attacker cannot find an input which generates the same output as another given input. These functions can then be used to make cryptographic primitives for protocols such as digital signatures. (12)

NIST continues to play a critical role in the process of consolidating and standardizing post-quantum cryptographic algorithms to ensure sufficient testing of these schemes and eventual wide-spread adoption. In 2016, the organisation began the process of developing standards for post-quantum cryptography and hosted a competition to solicit candidate algorithms and schemes. Last year in 2022, four algorithms were selected as part of NIST's post-quantum cryptographic standard and recently the first draft standards for three of the four (9). For general encryption, NIST selected the lattice-based CRYSTALS-Kyber (13) algorithm and for digital signatures CRYSTALS-Dilithium (13), FALCON (14), and SPHINCS+ (15) were selected. CRYSTALS-Dilithium is intended to be the primary algorithm, with FALCON providing signatures smaller than Dilithium is capable of. Lastly, SPHINCS+ was also selected as, critically, the other three algorithms are all lattice-based, whereas SPHINCS+ is hash-based. There are four additional algorithms for general encryption which are still under consideration for the purpose of alleviating the homogeneity of the current selections (16).

1.2 Research Aims and Objectives

1.2.1 Aims

In this work, we aim to investigate quantum attacks against lattice-based cryptographic systems. Specifically, we focus on quantum algorithms which solve the shortest vector problem (SVP), a core lattice problem which underpins the security of lattice-based cryptography.

Currently, we are in the NISQ (noisy intermediate scale quantum) era of quantum computing. NISQ devices are noisy and have a limited number of qubits, making the implementation of many quantum algorithms impractical. Variational quantum algorithms are hybrid classical-quantum algorithms which leverage these devices to

solve optimization problems by using small parameterized quantum circuits. The two primary variational quantum algorithms are the Variational Quantum Eigensolver (VQE) (17) and the Quantum Approximate Optimization Algorithm (QAOA) (18).

1.2.2 Contributions

In this work, we:

- Propose a novel encoding of the shortest vector problem for qudit-based QAOA. This reduces the quantum resources required to run the algorithm compared to an existing encoding (19) for qubit QAOA as the search space is encoded more compactly, with each solution being represented by exactly one quantum state.
- Develop a custom quantum simulator for the quantum subroutine of qudit-based QAOA for SVP. Creating a simulator specific to this problem allows for several optimizations to be made which allows for more efficient simulation of the preparation of the trial state. This improved performance allows for the simulation of problem instances in higher dimensions than could be achieved otherwise.
- Reproduce the results of Albrecht et al. (19) for instances of lattices in small dimensions, implementing the VQE algorithm using Qiskit (20).
- Conduct experimental evaluation of qudit-based QAOA for SVP.

1.3 Ethical Considerations

This project is concerned with the simulation of quantum algorithms which solve lattice problems. As such, the following ethical considerations have been taken into account.

1. **Ethical Use of Cryptography:** The techniques under investigation should be used solely for research purposes and not for any malicious or unethical activities.
2. **Legal and Licensing Concerns:** All libraries used are open-source, freely available, and correctly cited.
3. **Environmental Concerns:** This project is primarily concerned with simulating quantum algorithms and as such the energy consumed by the compute resource which ran these simulations should be acknowledged.
4. **Transparency and Reproducibility:** All methodologies, tools, and algorithms used in this investigation are documented. The research findings are presented

in a manner that allows for independent verification. The code developed during the course of this project is freely available.[1]

[1] `https://gitlab.doc.ic.ac.uk/moa418/qaoa-for-svp`

Chapter 2

Preliminaries

2.1 Lattice-Based Cryptography

A lattice is an infinite set of periodic points in $n-$dimensional space, conceptually similar to a vector space except that each of the vectors may only contain discrete elements. An example of a 2-dimensional lattice can be seen in figure 2.1. Lattices give rise to a class of optimization problems called lattice problems which are conjectured to be hard, even for a quantum computer. These problems serve as the foundation for a diverse variety of lattice-based cryptographic primitives.

The most famous of these problems, and the focus of this project, is the shortest vector problem (SVP). Simply put, given some lattice $\mathcal{L}$, find the shortest non-zero vector. A formal definition of the problem and supporting concepts are found in section 2.1.1, based on the work of Li et al. (21) and Micciancio and Regev (11).

2.1.1 Fundamental Lattice Theory

Definition 2.1.1 (Lattice) *A lattice is generated by the linear combinations of a set of linearly independent vectors. That is, given a set of n linearly independent vectors in $m-$dimensional space, $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^m$*

$$\mathcal{L}(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \{a_1\mathbf{v}_1 + \ldots + a_n\mathbf{v}_n \mid a_1, \ldots, a_n \in \mathbb{Z}\}$$

In the case that $m = n$, the rank and dimension of the lattice are equal and the lattice is referred to as a full-rank lattice. In cryptography we are primarily concerned with full-rank lattices and as such we will be dealing with full-rank lattices from here on.

Definition 2.1.2 (Basis) *A set of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ which span a lattice $\mathcal{L}(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ are a basis of the lattice. That is,*

$$\mathcal{L}(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \{z_1\mathbf{b}_1 + \ldots + z_n\mathbf{b}_n \mid z_1, \ldots, z_n \in \mathbb{Z}\}$$

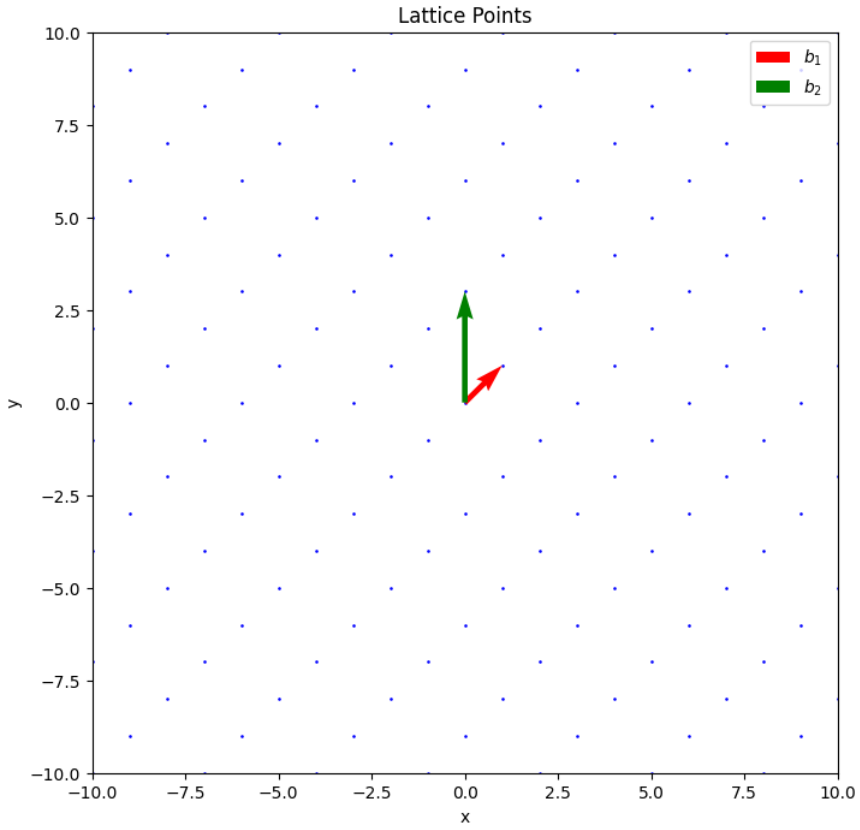

Figure 2.1: A 2-dimensional lattice with generated by the basis vectors $\{b_1, b_2\}$

For example, figure 2.1 shows a 2-dimensional lattice spanned by the vectors $\{\mathbf{b}_1, \mathbf{b}_2\}$. It is also useful to consider a matrix representation of the basis. Given a basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$ of some matrix $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, we write $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{n \times n}$ to be the matrix $\mathbf{B}$ with the basis vectors as columns. The lattice $\mathcal{L}(\mathbf{B})$ is then given by,

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} \mid \mathbf{x} \in \mathbb{Z}^n\} \tag{2.1}$$

Definition 2.1.3 (Unimodular matrix) *A matrix $\mathbf{U} \in \mathbb{Z}^{n \times n}$ is unimodular if and only if $\det(\mathbf{U}) = \pm 1$.*

Remark 2.1.4 *Similar to vector spaces, there are multiple bases that generate the same lattice, that is, $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$ (11). It has been shown that $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$ if and only if there exists some unimodular matrix $\mathbf{U}$ such that $\mathbf{B}' = \mathbf{UB}$ (21).*

The process of finding a basis $\mathbf{B}'$ with short, nearly orthogonal vectors that generates the lattice $\mathcal{L}(\mathbf{B})$, where $\mathbf{B}$ is some arbitrary basis, is known as lattice reduction and an important concept for solving lattice problems.

Definition 2.1.5 (Fundamental Domain) *The fundamental domain is the region bounded by the basis vectors of a lattice. That is,*

$$\mathcal{F}(\mathbf{B}) = \{a_1\mathbf{b}_1 + \ldots + a_n\mathbf{b}_n \mid a_1, \ldots, a_n \in [0, 1)\}$$

Remark 2.1.6 *The n-dimensional volume of the fundamental domain is given by the size of the determinant of the basis matrix,* $\text{vol}(\mathcal{F}(\mathbf{B})) = |\det(\mathbf{B})|$. *(21)*

Definition 2.1.7 (Determinant) *The determinant of a lattice $\mathcal{L}(\mathbf{B})$ is defined as the $n-$dimensional volume of $\mathcal{F}(\mathbf{B})$.*

$$\det(\mathcal{L}(\mathbf{B})) = \text{vol}(\mathcal{F}(\mathbf{B})) = |\det(\mathbf{B})|$$

Corollary 2.1.8 *The determinant of a lattice, and therefore the $n-$dimensional volume of the fundamental domain, is an invariant property of the lattice meaning it is equal for different bases of the same lattice.*

$$\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})| = |\det(\mathbf{U}\mathbf{B}')| = |\det(\mathbf{U})| \cdot |\det(\mathbf{B}')| = |\det(\mathbf{B}')| = \det(\mathcal{L}(\mathbf{B}'))$$

as $\det(\mathbf{U}) = \pm 1$ *from definition 2.1.3.*

Example 2.1.9 *Let the lattice $\mathcal{L}(\mathbf{B})$ be generated by the basis*

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Then we have that $\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})| = 1$. *Now, consider the lattice $\mathcal{L}(\mathbf{B}')$ be generated by the basis*

$$\mathbf{B}' = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

As $\det(\mathcal{L}(\mathbf{B}')) = |\det(\mathbf{B}')| = 1$ *as well, we determine that $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$.*

Definition 2.1.10 (Shortest Vector) *Given a lattice $\mathcal{L}(\mathbf{B})$, the shortest vector is the minimum distance between two points in $\mathcal{L}(\mathbf{B})$. The length of the shortest vector, denoted λ_1, is*

$$\lambda_1(\mathcal{L}(\mathbf{B})) = \min\{||\mathbf{v}|| \mid \mathbf{v} \in \mathcal{L}(\mathbf{B}) \setminus \mathbf{0}\}$$

Related to the shortest vector is the concept of the ith successive minima. Consider a closed n-dimensional ball centered on at the origin of a lattice with radius r. The ith successive minima is the minimum value of r such that i linearly independent vectors are contained within the intersection of the ball and the lattice. The ith successive minima is denoted λ_i, noting that the shortest vector is a special case where $i = 1$.

2.1.2 Lattice Problems

The most well known lattice problems are the shortest vector problem (SVP), the closest vector problem (CVP), and the shortest independent vectors problem (SIVP). While the primary focus of this project is on algorithms for SVP, all three of these problems will be introduced here along with their approximate variants. As in the previous section, the definitions and supporting work presented here are based on the work of Li et al. (21) and Micciancio and Regev (11).

Definition 2.1.11 (The Shortest Vector Problem) *For a given lattice basis* $\mathbf{B}$*, find a vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *such that* $||\mathbf{v}|| = \lambda_1(\mathcal{L}(\mathbf{B}))$*. Equivalently, find the shortest non-zero vector in* $\mathcal{L}(\mathbf{B})$*.*

Definition 2.1.12 (The Closest Vector Problem) *For a given lattice basis* $\mathbf{B} \in \mathbb{R}^{n \times n}$ *and a target vector* $\mathbf{t} \in \mathbb{R}^n$*, find a vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *such that for all* $\mathbf{w} \in \mathcal{L}(\mathbf{B})$ $||\mathbf{v} - \mathbf{t}|| \leq ||\mathbf{w} - \mathbf{t}||$*. Equivalently, find a vector in* $\mathcal{L}(\mathbf{B})$ *that is closest to* $\mathbf{t}$*.*

Definition 2.1.13 (The Shortest Independent Vectors Problem) *For a given lattice basis* $\mathbf{B}$*, find a set of* n *linearly independent vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathcal{L}(\mathbf{B})$ *such that*

$$\max_{1 \leq i \leq n} ||\mathbf{v}_i|| = \lambda_n(\mathcal{L}(\mathbf{B}))$$

The approximate versions of SVP and SIVP, and the associated promise problem for SVP, are of particular importance due to their use in reductions to prove the security of lattice-based cryptographic systems (22). Approximate versions of lattice problems intuitively relax the constraints in the original problem definitions.

Definition 2.1.14 (SVP$_\gamma$) *For a given lattice basis* $\mathbf{B}$ *and* $\gamma \geq 1$*, find a vector* $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ *such that* $||\mathbf{v}|| \leq \gamma \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$*.*

Definition 2.1.15 (SIVP$_\gamma$) *For a given lattice basis* $\mathbf{B}$ *and* $\gamma \geq 1$*, find a set of* n *linearly independent vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathcal{L}(\mathbf{B})$ *such that*

$$\max_{1 \leq i \leq n} ||\mathbf{v}_i|| = \gamma \cdot \lambda_n(\mathcal{L}(\mathbf{B}))$$

Promise problems, such as GAPSVP$_\gamma$ (defined below), are generalized decision problems in which the algorithm must catogorize its inputs into either *yes* instances or *no* instances. However, unlike decision problems, the union of the *yes* and *no* instances is not the set of all possible inputs. In the case that the input belongs to neither instance, there is no restriction on the behaviour of the algorithm.

Definition 2.1.16 (GAPSVP$_\gamma$) *For a given lattice basis* $\mathbf{B}$*,* $d > 0$*, and* $\gamma \geq 1$*, the instance (*$\mathbf{B}$*,* d*) is a yes instance if:*

$$\lambda_1(\mathcal{L}(\mathbf{B})) \leq d$$

And is a no instance if:

$$\lambda_1(\mathcal{L}(\mathbf{B})) \geq \gamma \cdot d$$

Gap problems and approximation problems are used in lattice-based cryptography to prove the security of lattice-based cryptographic systems. The hardness of the two problems is connected in that if for some γ the gap problem is proven to be hard, then the γ-approximation problem is also hard, as shown by Li et al (21). This is helpful as γ-approximation problems are useful tools for reductions used to show the hardness of lattice-based cryptographic systems, for example (22).

2.1.3 Worst-Case Hardness

Lattice-based cryptography relies on the worst-case hardness of the underlying lattice-problems, setting it apart from traditional cryptographic approaches that usually rely on average-case hardness. For a given problem, worst-case hardness means that the problem is only hard for some of the problem instances. Compared to this average-case hardness is a stronger assumption as it means that the problem is hard for most problem instances. Historically, cryptographic systems have been based on problems with average-case hardness as without a way to generate specifically hard instances of a problem, worst-case hardness does not offer any security guarantees.

Cryptographic systems based on lattices differ from traditional systems with average-case hardness as their security are able to be based on worst-case hardness. This was first made possible by the seminal work of Ajtai (23) who showed how to generate hard instances of lattice problems and introduced one-way functions with the security grantee that, if the attacker can invert the function they can provably solve any instance of SVP_γ. This unique property of lattice-based cryptography is a large part for its promise as providing post-quantum security as a quantum computer has to be able to efficiently solve the hardest instances of a problem, not only the average instances as is the case with integer factorization, for example.

Learning with errors (LWE) is a lattice-based problem first introduced by Regev (22) which seeks to recover some secret information from a set of linear equations distorted by noise. This problem is presumed to be hard as Regev showed a reduction from SVP_γ and SIVP_γ to LWE (22). This lattice problem was then used to construct a public-key encryption system with a worst-case security guarantee, meaning that an attacker capable of solving LWE is also capable of efficiently approximating SVP.

The current leading post-quantum cryptographic systems are primarily lattice-based, as discussed in section 1.1.2. CRYSTALS-kyber (13) is NIST's selection for general encryption purposes and is based on a variant of LWE called module learning with errors. The shortest vector problem remains at the core of the security guarantees of lattice-based cryptographic schemes.

2.2 Classical Algorithms for SVP

2.2.1 Exact SVP Solutions

There are two classes of classical algorithm which are used to solve SVP exactly (11). The first class of algorithm offers both exponential time- and space-complexity. The AKS sieve algorithm belongs to this class but is largely considered impractical due to it's exponential space-complexity. However, Nguyen presented results showing that AKS sieve algorithms are, in fact, tractable for dimensions up to $n = 50$ (24). Unfortunately, the dimensions of lattice this algorithm can efficiently handle are too small to be cryptographically relevant as dimensions of $n \approx 400$ are required.

The second class of algorithm, offers improved space-complexity, $poly(n)$, but super-exponential time-complexity. Enumeration methods are the best performing type of algorithm which belongs to this class, achieving a run time of $\tilde{\mathcal{O}}(n^{n/2e})$ (25).

2.2.2 Approximate SVP$_\gamma$ Solutions

The LLL algorithm, proposed by Lenstra, Lenstra, and Lovász, has polynomial time-complexity and can find approximate solutions to SVP$_\gamma$ (26). The LLL algorithm translates the basis of an input lattice into a basis with short, nearly orthogonal vectors. When small values of γ are considered, an alternative approximation algorithm is the Block Korkine-Zolotarev algorithm (27). This algorithm trades improved accuracy in the approximation of the solution for a poorer time-complexity.

2.3 Quantum Algorithms for Optimization

Quantum algorithms have the potential to more efficiently find solutions to SVP. Quantum search algorithms such as Grover's algorithm (8) provides quadratic speedup over classical search algorithms, which could be applied to SVP. However, this is not a particularly significant speedup and the circuit depth of Grover's algorithm is prohibitive with current NISQ devices. Adiabatic style algorithms could also be applied to SVP, but encounter a similar issue as the noise present in NISQ devices prevents the algorithms from successfully finding the shortest vector (28).

Variational quantum algorithms are a promising approach in quantum computing that are capable of utilising NISQ devices to solve a variety of problems. These algorithms revolve around the concept of optimizing a parameterized quantum circuit in order to minimize the objective function which the circuit is encoding. For a given set of circuit parameters, the quantum circuit is executed and the final state is measured many times in order to sufficiently sample the wavefunction. The measurements of the quantum execution are then passed to a classical optimizer which is used to generate a new set of parameters. This process is repeated until the system converges on a solution.

2.3.1 Adiabatic Quantum Computation

Adiabatic quantum computing (AQC) is an alternative approach to the conventional quantum circuit model for quantum computation. AQC utilises the adiabatic theorem in order to perform calculations.

Theorem 2.3.1 (The Adiabatic Theorem) *A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum. (29)*

Adiabatic computation is therefore achieved by first determining some target Hamiltonian, the ground state of which is a solution to the problem to be solved. Then,

the ground state of a system with a simple Hamiltonian is prepared after which the simple Hamiltonian is slowly evolved to the target Hamiltonian. As the system was prepared in the ground state, by the adiabatic theorem the system remains in its instantaneous state, that is the ground state of the target Hamiltonian which represents a solution to the given problem.

The evolution of the system is governed by the time-dependent Schrodinger's equation (30):

$$i\hbar\frac{\partial}{\partial t}\left|\psi(t)\right\rangle = \hat{H}\left|\psi(t)\right\rangle \tag{2.2}$$

The spectral gap of a Hamiltonian is the difference in energy between the ground state and the first excited state. The spectral gap plays a crucial role in AQC. As the system is initially in the ground state, it cannot reach a state which is less excited and by also evolving the Hamiltonian so slowly that it is never excited enough to reach the first excited state, the system will indefinitely remain in the ground state.

Consider the quantum system with the initial Hamiltonian H_0, the final Hamiltonian H_1. The objective of AQC is to evolve from H_0 at $t = t_0$ to H_1 at $t = t_1$. Finally, let the delay factor $\tau = t_1 - t_0$. We can then create a time-dependent Hamiltonian by defining $H(t)$ as a linear combination of the initial and target Hamiltonian (31):

$$H(t) = \left(1 - \frac{t}{\tau}\right) H_0 + \frac{t}{\tau} H_1 \tag{2.3}$$

A perfectly adiabatic evolution would require $\tau \to \infty$, however Fahri showed that for typical eigenvalues of $H(t)$ that the size of τ is $\mathcal{O}(g_{\min}^{-2})$, where $g_{\min}$ is the spectral gap. This allows us to perform useful, although not necessarily efficient, quantum computation in finite time.

2.3.2 Variational Quantum Eigensolvers

Variational quantum eigensolvers (17) (VQE) are a class of quantum algorithm designed to find the ground state of quantum systems. They are hybrid quantum algorithms meaning they make use of both classical and quantum subroutines. VQEs have been utilised in a variety of applications including quantum chemistry (32), condensed matter physics (33), and optimization (19).

This algorithm is based on the variational method of quantum mechanics which provides an upper bound for the ground state energy of a system.

Definition 2.3.2 (The Variational Principle) *For a Hamiltonian $\hat{\mathcal{H}}$ with ground state energy E_0, for any given state $|\psi\rangle$ (34)*

$$E_0 \leq \left\langle\psi\right| \hat{\mathcal{H}} \left|\psi\right\rangle$$

That is the expectation of the energy of a state $|\psi\rangle$, which is not the ground state, will always be an overestimate of the energy of the ground state. Therefore, VQE seeks to find the value of $|\psi\rangle$ which corresponds to the lowest expectation of $\hat{\mathcal{H}}$, ideally E_0.

To achieve this, the trial wavefunction $|\psi\rangle$ is generated by a parameterized quantum circuit, known as an ansatz. Using the conventional gate model (opposed to the adiabatic model described in section 2.3.1), we can implement this ansatz as a series of unitary parameterized gates, collectively denoted $U(\boldsymbol{\theta})$ with $\boldsymbol{\theta}$ being a set of parameters in $(-\pi, \pi]^N$. Then, an initial state is prepared, typically an even superposition among all possible quantum states

$$|\psi_0\rangle = \frac{1}{2^{N/2}} \sum_{\mathbf{z}} |z\rangle \tag{2.4}$$

where N is the number of qubits. By applying the parameterized unitary $U(\boldsymbol{\theta})$ to the initial state we are able to write the VQE optimization problem (35)

$$E_{\text{VQE}} = \min_{\boldsymbol{\theta}} \langle\psi_0| U^\dagger(\boldsymbol{\theta}) \hat{\mathcal{H}} U(\boldsymbol{\theta}) |\psi_0\rangle \tag{2.5}$$

This equation highlights the hybrid nature of VQE. The trial state is prepared on a quantum device, first by generating the initial state and then by applying the quantum circuit. This trial state prepared and measured multiple times in order to accurately sample the wave function. This sampled wavefunction is then passed to a classical computer to calculate the expectation. In order to improve the accuracy of the expectation, the parameters are updated using classical optimization to ideally find the minimum expectation and ground state of the Hamiltonian.

2.3.3 The Quantum Approximate Optimization Algorithm

The quantum approximate optimization algorithm (QAOA) (18) is a quantum algorithm designed to solve combinatorial optimization problems, a class of problems for which finding the exact solution is computationally challenging. Like VQE, QAOA is a hybrid algorithm making use of both classical and quantum subroutines and is one of the most promising algorithms for NISQ devices. It operates by first generating an initial state which is a superposition of candidate solutions and then applies a parameterized quantum circuit, which is iteratively updated to increase the probability of measuring a good solution.

QAOA is inspired by adiabatic quantum computing, and approximates adiabatic evolution by using the Suzuki-Trotter decomposition (36), a method for approximating the evolution operator $e^{-i\mathcal{H}t}$. The Hamiltonian $\mathcal{H}$ is separated into two main parts, $\mathcal{H} = \mathcal{H}_C + \mathcal{H}_M$, where $\mathcal{H}_C$ is the diagonal cost Hamiltonian whose ground state corresponds to the solution of the optimization problem, and $\mathcal{H}_M$ is the mixing Hamiltonian which allows the evolution to explore the full Hilbert space. The cost Hamiltonian and mixing Hamiltonian can then be used in sequence as the operators $e^{-\mathcal{H}_C t/L}$ and $e^{-\mathcal{H}_M t}$ in order to approximate the evolution operator $e^{-i\mathcal{H}t}$.

The objective of QAOA is to find an approximate solution to a combinatorial optimization problem, that is finding a state z_0 which minimizes the cost function $C(\mathbf{z})$ of the problem. Here, the core structure of QAOA will be outlined (18, 37).

We consider a 2^n dimensional Hilbert space with standard computation basis

$$|z\rangle, z \in \{0, 1\} \tag{2.6}$$

QAOA begins with the preparation of the initial state $|\psi_0\rangle$. This must be a simple, easily prepared quantum state as for NISQ devices we are limited by circuit depth and do not want to detract from the main computation. Typically, a uniform superposition of the computational basis states is used.

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{z}} |\mathbf{z}\rangle \tag{2.7}$$

where N is the number of qubits used.

The quantum state $|\psi_0\rangle$ is then evolved to the trial state by applying layers of repeating unitary gates, which correspond to the problem Hamiltonian H_C and the mixing Hamiltonian H_M. The problem Hamiltonian H_C is diagonal in the computational basis and encodes the cost function $C(\mathbf{z})$.

$$\mathcal{H}_C |\mathbf{z}\rangle = C(\mathbf{z}) |\mathbf{z}\rangle \tag{2.8}$$

Therefore we define the problem Hamiltonian $\mathcal{H}_C$ as

Definition 2.3.3 (Problem Hamiltonian) *(37)*

$$\mathcal{H}_C = \sum_{\mathbf{z}} C(\mathbf{z}) |\mathbf{z}\rangle \langle \mathbf{z}|$$

The mixing Hamiltonian is typically the sum of Pauli-X operators over all qubits.

Definition 2.3.4 (Mixing Hamiltonian) *(18)*

$$\mathcal{H}_M = \sum_{j=1}^{n} X_j$$

It is worth noting that other mixing Hamiltonian do exist in the literature (38). The problem and mixing Hamiltonians $\mathcal{H}_C, \mathcal{H}_M$ can then be used to generate parameterized unitary gates, the phase separation gate $U_C(\gamma)$ and the mixing gate $U_M(\beta)$ respectively.

$$U_C(\gamma) = \exp(-i\gamma\mathcal{H}_C) \tag{2.9}$$

$$U_M(\beta) = \exp(-i\beta\mathcal{H}_M) \tag{2.10}$$

A central aspect of QAOA is the layered structure in which the unitary gates are applied. The parameter $p \geq 1$ is the number of layers of gates applied in the preparation of the trial state. An iteration of QAOA is therefore parameterized by p, $\boldsymbol{\gamma} = \gamma_0, \ldots, \gamma_n$, and $\boldsymbol{\beta} = \beta_0, \ldots, \beta_n$. The tail state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ is then given by

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U_M(\beta_p)U_C(\gamma_p) \cdots U_M(\beta_1)U_C(\gamma_1) |\psi_0\rangle \tag{2.11}$$

The parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ then need to be optimized in order to produce the trial state with the lowest possible expectation. In order to do so, we first need to perform measurements on the trial state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ and then an estimation of the trial state's expectation value can be calculated classically.

$E_{\boldsymbol{\gamma},\boldsymbol{\beta}}$, the expectation of $\mathcal{H}_C$ in the trial state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$, is calculated according to

$$E_{\boldsymbol{\gamma},\boldsymbol{\beta}} = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}| \mathcal{H}_C |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle \tag{2.12}$$

However, as the measurement of the trial quantum state collapses to a single value it is not possible to accurately calculate the expectation from a single preparation of the trial state. Instead the trial state is prepared and measured many times, with the number of measurements being called the number of shots. By having a large number of shots, we are able to more accurately sample the wavefunction of the trial state. A quantum state collapses to a specific $\mathbf{z} = (z_1, \ldots, z_n)$ with probability $\mathcal{P}(\mathbf{z}) = |\langle \mathbf{z}|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle|^2$. The expectation of $\mathcal{H}_C$, $E_{\boldsymbol{\gamma},\boldsymbol{\beta}}$, can therefore be approximated by

$$E_{\boldsymbol{\gamma},\boldsymbol{\beta}} \approx \sum_{\text{samples } \mathbf{z}} \mathcal{P}(\mathbf{z})C(\mathbf{z}) \tag{2.13}$$

where $C(\mathbf{z})$ is the original cost function to be minimized.

The above process is repeated for different values of $\boldsymbol{\gamma}, \boldsymbol{\beta}$, until optimal parameters $\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*$ are found. This can be achieved by using classical optimization to determine the values of $\boldsymbol{\gamma}, \boldsymbol{\beta}$ to trial. Once $\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*$ are found, the process can be repeated again to produce the state $|\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*\rangle$, this can then be sampled to find solutions to the original optimization problem. The ideal output would be a superposition of just the global minima, but this is usually not the case and other states may be included in the superposition of $|\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*\rangle$ also. A small amount of classical post processing must be applied given a number of samples of $|\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*\rangle$, in which the most probable solutions can be taken as candidate solutions for the optimization problem.

2.4 Qudit Systems

Conventionally, in quantum computing the basic unit of information is the quantum bit or qubit. Qudits are a generalization of this concept and is the basic unit of information in higher dimension quantum computing. The key difference between

the two is that where a qubit can exist as the superposition of two states, say $|0\rangle$ and $|1\rangle$, a qudit can exist as a superposition of d states, that is $|0\rangle, \ldots, |d-1\rangle$.

Qudit systems offer increased information density over qubits as a single d-level qudit can encode $\log_2(d)$ bits of classical information compared to a single bit in a qubit. While the mapping from d-level systems to qubits can be achieved efficiently, the realisation of qudit systems remains an active area of research due to the potential of more efficient hardware based on physical, multilevel systems. (37).

A Hilbert space is a vector space which contains all possible states of a quantum system. Generalizing from qubits to qudits increases the dimension of the Hilbert space as each computational unit can be in d different states. Typically, for an n qubit system the Hilbert space is $(\mathbb{C}^2)^{\otimes n}$ and has dimension 2^n, whereas for an n qudit system the Hilbert space is $(\mathbb{C}^d)^{\otimes n}$ and has dimension d^n.

In order to use the circuit model of quantum computing in a qudit system, generalized versions of the unitary gates are required. For the purposes of this project that is limited to the generalized Pauli gates, and they are discussed in the Related Work, chapter 3.

Some physical systems naturally have multiple distinct energy levels and can be used to encode qudits as well as qubits. Qudit systems are an area of significant research and have been realised on many physical systems such as photons (39), trapped ions (40), and nuclear magnetic resonance (41).

Chapter 3

Related Work

3.1 Variational Algorithms for SVP

The recent work of Albrecht et al. (19) explores the efficiency with which NISQ devices can be used to solve SVP. The authors introduce a new mapping from SVP to a quadratic unconstrained binary optimization problem (QUBO) which bounds the search space. Additionally, the authors propose two methods for excluding the zero vector from the search space. This work discusses both VQE and QAOA as the method for determining the ground state of the Hamiltonian, but focuses primarily on the former and produces experimental results only for VQE.

3.1.1 Mapping SVP to a QUBO Formulation

One approach to forming a Hamiltonian suitable for variational quantum algorithms such as VQE and QAOA is to map the problem in question to a QUBO representation and then generate an Ising Hamiltonian from the QUBO representation. An Ising Hamiltonian is a Hamiltonian which is comprised of only Pauli-Z operators and interactions between, at most, pairs of qubits. The cost function of a QUBO formulation has the general form

$$C(s_1 s_2 \dots s_n) = c + \sum_i c_{ii} s_i + \sum_{i \neq j} c_{i,j} s_i s_j \tag{3.1}$$

where each s_i in $s_1 s_2 \dots s_n$ is a binary variable, and $c, \{c_{i,j}\}_{1 \leq i,j \leq n}$ are coefficients.

The Pauli-Z operator has eigenvalues ± 1 but as we are mapping from bit strings, we need an operator with eigenvalues $\{0, 1\}$. This can be achieved by mapping each binary variable is mapped to $\frac{I-Z}{2}$, where I is the identity operator and Z is the Pauli-Z gate. Therefore, a full mapping from the QUBO representation of the problem to an Ising Hamiltonian is

$$\mathcal{H} = c + \sum_i c_{ii} \frac{I_i - Z_i}{2} + \sum_{i \neq j} c_{i,j} \frac{I_i - Z_i}{2} \frac{I_j - Z_j}{2} \tag{3.2}$$

where I_i is the identity operator and Z_i is the Pauli-Z operator, each acting on the ith qubit (19).

Using the above, the authors introduce their mapping of SVP from a quadratic constrained integer optimization problem to a QUBO formulation, and then from a QUBO formulation to an Ising Hamiltonian. Recalling the definition of SVP (2.1.11), we have

$$\lambda_1^2 = \min_{\mathbf{y} \in \mathcal{L}(\mathbf{B}) \setminus \{0\}} ||\mathbf{y}||^2 \tag{3.3a}$$

$$= \min_{\mathbf{x} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}} \sum_{i=1}^{n} x_i^2 \cdot \mathbf{G}_{ii} + 2 \sum_{1 \leq i < j \leq n} x_i \cdot x_j \cdot \mathbf{G}_{i,j} \tag{3.3b}$$

where $\mathbf{G}$ is a Gram matrix of the lattice basis $\mathbf{B}$, that is $\mathbf{G} = \mathbf{B} \cdot \mathbf{B}^T$ (19). Equation 3.3b is the constrained integer optimization problem, the authors then introduce a mapping from this to a QUBO by introducing a bound on each integer such that $|x_i| \leq a$.

$$x_i = -a + \sum_{j=0}^{\lfloor \log(2a) \rfloor - 1} \left(2^j \tilde{x}_{i,j}\right) + (2a + 1 - 2^{\lfloor \log(2a) \rfloor}) \cdot \tilde{x}_{i,\lfloor \log(2a) \rfloor} \tag{3.4}$$

By substituting equation 3.4 into equation 3.3b, we acquire a QUBO formulation of SVP with a bounded search space. However, in performing this substitution we also ignore the $\mathbf{x} \neq 0$ constraint. The solution to the resulting QUBO is therefore the zero vector. The authors discuss approaches to dealing with this in the classical optimization step. (19)

$$\begin{matrix} & & \min \\ \tilde{x}_{1,0} & \cdots & \tilde{x}_{1,\lfloor \log 2a_1 \rfloor} \\ \vdots & \ddots & \vdots \\ \tilde{x}_{n,0} & \cdots & \tilde{x}_{n,\lfloor \log 2a_n \rfloor} \end{matrix} \quad c + \sum_{\tilde{x}_{i,j}} c_{i,j} \cdot \tilde{x}_{i,j} + \sum_{\tilde{x}_{i,j}, \tilde{x}_{k,l}} d_{i,j,k,l} \cdot \tilde{x}_{i,j} \cdot \tilde{x}_{k,l} \tag{3.5}$$

This formulation allows us to convert the bounded SVP problem, that is the problem where each integer in the vector of coefficients is within a bounded range, to an Ising Hamiltonian for use with either VQE or QAOA.

This mapping is powerful as it limits the search space to a ball of radius A, which the algorithm must optimize over to find the shortest vector and therefore limits the number of qubits required. Crucially, however, the ball must encode the shortest vector. The probability for encoding the shortest vector as an eigenstate of the Hamiltonian is shown in figure 3.1. Three strategies are compared experimentally over 1024 lattices, where n is the lattice dimension and m is the number of qubits available. Each strategy assigns a different distribution of qubits to the elements of the coefficient vector x. The number of qubits assigned to a coefficient x_i dictates the bound a on x_i. (19)

1. Each x_i is assigned $\lfloor \frac{m}{n} \rfloor$ qubits, uniformly distributing the qubits.

2. Each x_i is assigned $\lfloor \frac{m}{n} \rfloor$ qubits. Then $m - \left(n \cdot \lfloor \frac{m}{n} \rfloor\right)$ randomly chosen coefficients are allocated an additional qubit. This uniformly distributes the qubits, and randomly allocates the remaining qubits.

3. Define the Gaussian Heuristic as (42),

$$gh(\mathcal{L}) = \sqrt{\frac{n}{2\pi e}} \cdot \text{Vol}(\mathcal{L})^{\frac{1}{n}} \tag{3.6}$$

   The qubits are assigned according to radius $A = gh(\mathcal{L})$, which is then scaled to assign at most m qubits.

As seen in figure 3.1, with a small number of qubits available, as with NISQ devices, the probability of encoding the shortest vector as an eigenstate of the problem Hamiltonian decreases as the dimension of the lattice increases, until eventually reaching zero. Also of note, this probability only indicates how likely the shortest vector is included in the search space, and does not indicate how likely the shortest vector would be found in an execution of the algorithm.

Returning to the constraint that the shortest vector is non-zero, which is ignored in equation 3.5, the authors discuss multiple strategies to find the shortest non-zero vector. The first approach, is to simply ignore the constraint and instead find the trail state which is optimized to find the ground state corresponding to the zero vector. Then when performing measurements of the final trial state, output the shortest non-zero vector. This approach works for lattices in small dimensions (43), but due to the increasingly complex optimization corresponding to larger lattices, this approach is unlikely scale.

The second approach is to find a new Hamiltonian penalises the zero vector. However, the authors show that the QUBO associated with this Hamiltonian requires an an additional $n-2$ qubits in order to encode the constraint using auxiliary variables, where n is the rank of the lattice (19).

The final approach is to introduce a penalty for measurements of the zero vector at the optimization step. The authors claim this approach is only possible for VQE as the modification occurs during classical post processing whereas for QAOA the cost function is defined strictly by the optimization strategy[1] (19). The author's approach is to introduce a multiplicative penalty to the sample mean of the cost function. Let N be the number of shots conducted for a trial state, and $\tilde{N}$ be the number of measurements which were non-zero. The modified cost function used by the authors is then,

$$C'(\theta) = \left(\frac{N}{\tilde{N}}\right) \frac{1}{N} \sum_{i=1}^{N} C(\mathbf{m}_i) \tag{3.7}$$

where $C(\mathbf{z})$ is the original, classical cost function and $\mathbf{m}_i$ is a measured bit string.

[1]Contrary to this claim, modifying the cost function of QAOA at the point of optimization is a strategy to encode constraints seen in other work (37).

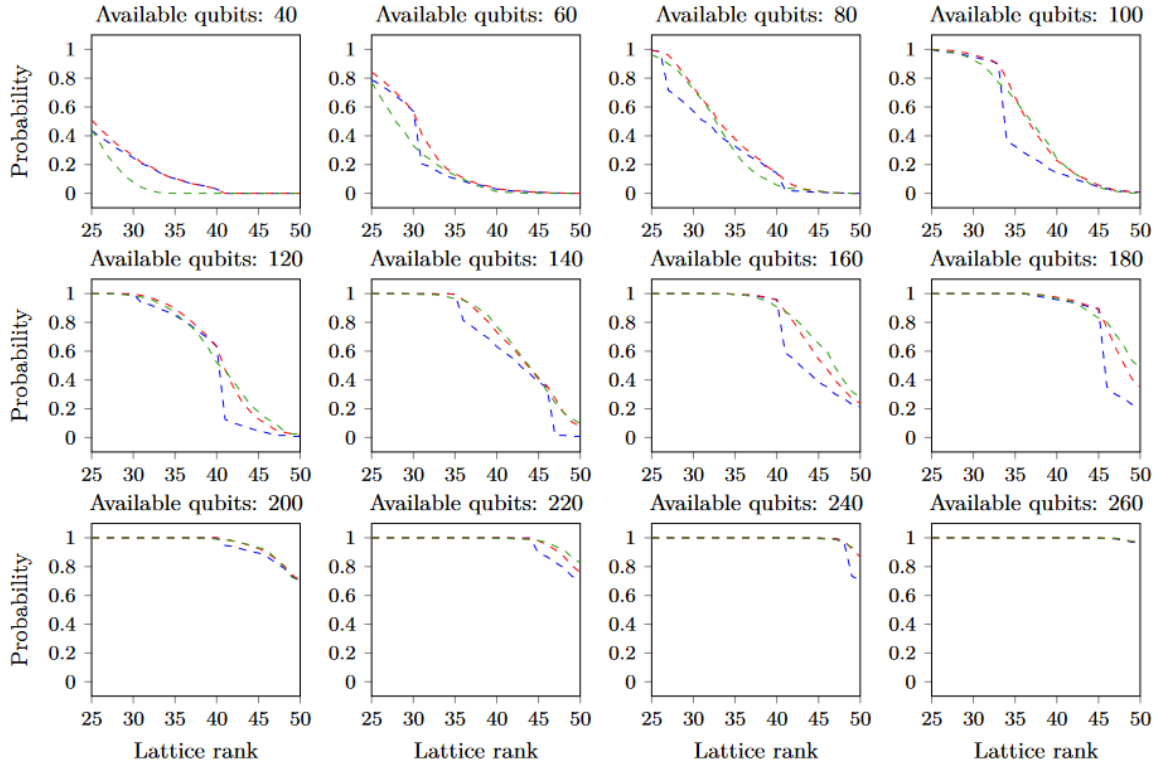

Figure 3.1: Probability of encoding the shortest lattice vector as an eigenvector of the problem Hamiltonian using strategies described in 3.1.1 in blue, red and green respectively (19)

3.2 CVaR Optimisation

A recent work by Barkoutsos et al. (44) explores the application of Conditional Value-at-Risk (CVaR) to the classical optimization step of variational quantum algorithms. A CVaR-version of the equation for calculating the expectation of a quantum state is presented by the authors.

Definition 3.2.1 (CVaR$_\alpha$) *For a random variable X, the cumulative density function of X, F_X, and a confidence level $\alpha \in (0, 1]$*

$$\mathit{CVaR}_\alpha(X) = \mathbb{E}[X | X \leq F_X^{-1}(\alpha)]$$

That is, CVaR$_\alpha$ is the expectation of only the lower α-tail of a distribution.

CVaR can be utilised in the classical optimization of a variational quantum algorithm by modifying the calculation of the value of the expectation. Recall equation 2.12 is expectation of a quantum state, and that the expectation may be approximated by equation 2.13. Instead of calculating the expectation as an average of the value of all shots taken, the CVaR-version instead considers only the α-tail of the measurements. For K shots, consider (without loss of generality) that the set of measurements $\{m_1, \ldots, m_K\}$ is in nondecreasing order with respect to the cost

associated with each measurement, $C(m_i)$. The objective function is given by (44)

$$\frac{1}{\lceil \alpha K \rceil} \sum_{k=0}^{\lceil \alpha K \rceil} C(m_i) \tag{3.8}$$

This introduces α as an additional parameter to tune within the variational quantum algorithm. $\alpha = 1$ corresponds to a conventional sample mean, whereas $\lim_{\alpha \to 0}$ corresponds to taking the single lowest measured value as the value of the objective function.

Using $\alpha < 1$ smooths the optimization landscape, which helps alleviate one of the major problems of QAOA and VQE, that being optimization landscapes with many local minima different to the global minima. Barkoutsos et al. showed theoretically and through empirical trials, both simulated and on real quantum devices, that using a CVaR-version of the objective function improves the performance of QAOA and VQE algorithms.

3.3 QAOA for Qudit Systems

A recent work by Deller et al. (37) explored how using qudit quantum systems for integer optimization problems is often more resource efficient. The authors present a version of QAOA suitable for qudit systems and a variety of methods to implement constraints present in integer optimization problems.

As discussed above, in section 2.4, a qudit is the generalization of the conventional qubit, with each qudit representing d possible states. The standard computational basis for a qudit system is denoted $|z\rangle$ with $z \in \{0, \ldots, d-1\}$. In order to perform meaningful operations on a given qudit, we must first define a set of generalized quantum gates that can be applied to a d-level system.

Definition 3.3.1 (Generalized Pauli-Z) *(45) Let ω be the dth root of unity,*

$$\omega = e^{2\pi i/d}$$

then

$$Z = \sum_{z=0}^{d-1} \omega^z \, |z\rangle \langle z|$$

Definition 3.3.2 (Generalized Pauli-X) *(45)*

$$X = \sum_{z=0}^{d-1} |(z+1) \mod d\rangle \langle z|$$

The application of these gates gives us

$$Z \, |z\rangle = \omega^z \, |z\rangle \tag{3.9a}$$

$$X|z\rangle = |(z+1) \mod d\rangle \tag{3.9b}$$

The application of a gate to a single qudit in an N qudit follows the same notation as is used with qubits, that is

$$X_j = I \otimes \ldots \otimes I \otimes X \otimes I \ldots \otimes I \tag{3.10}$$

where I is the identity operator and the only qudit which is acted on by the gate is the jth qudit.

3.3.1 Hamiltonians for Qudit QAOA

Utilising these two generalized gates, Deller et al. (37) introduce methods for generating problem Hamiltonians and Mixing Hamiltonians for qudit systems.

One approach to finding a suitable problem Hamiltonian for QAOA is by representing it in the form of an Ising Hamiltonian, that is a Hamiltonian represented by only generalized Pauli-Z operators and at most pairs of interactions between qudits. The authors present a way of representing the problem Hamiltonian, as defined above 2.3.3, as a product of generalized Pauli-Z gates using the Fourier transform.

$$\mathcal{H}_C = \frac{1}{d^N} \sum_{\mathbf{a}} \hat{C}(\mathbf{a}) \prod_{j=1}^{N} Z_j^{a_j} \tag{3.11}$$

where N is the number of qudits, $\mathbf{a} \in \mathbb{Z}_d^N$, and Fourier transform of the cost function $\hat{C}(\mathbf{a})$ is given by

$$\hat{C}(\mathbf{a}) = \sum_{\mathbf{z}} C(\mathbf{z}) e^{-2\pi i \mathbf{a}\cdot\mathbf{z}/d} \tag{3.12}$$

Equation 3.11 shows that it is possible to represent the diagonal Hamiltonian of an arbitrary cost function as a product of generalized Pauli-Z operators.

In order to produce a operator representation of the mixing Hamiltonian, as defined above 2.3.4, Deller et al. make use of angular momentum operators. The representation of the mixing Hamiltonian is given as

$$\mathcal{H}_M = \sum_{i=1}^{N} L_{x,i} \tag{3.13}$$

where the angular momentum operator (in the x direction) is given by

$$L_x = \frac{1}{2}(L_+ + L_-) \tag{3.14a}$$

$$L_+ |\ell, m\rangle = \sqrt{(\ell - m)(\ell + m + 1)}\, |\ell, m+1\rangle \tag{3.14b}$$

$$L_- |\ell, m\rangle = \sqrt{(\ell + m)(\ell - m + 1)}\, |\ell, m-1\rangle \tag{3.14c}$$

where $\ell = (d-1)/2$, $m = z - (d-1)/2$, and $|z\rangle = |\ell, m\rangle$. Using angular momentum operators in the x direction, the mixing Hamiltonian is able to sufficiently explore the set of possible states.

However, as noted by the authors, different constructions of the mixing Hamiltonian are equivalent provided they sufficiently explore the set of possible states. Another representation of the mixing Hamiltonian can be constructed using generalized Pauli-X gates.

First a single qudit mixing Hamiltonian is defined as (38)

$$h_M(r) = \sum_{i=1}^{r} \left(X^i + (X^\dagger)^i\right) \tag{3.15}$$

where $r \in \{1, \ldots, d-1\}$ and $X^\dagger$ is the inverse generalized Pauli-X gate. The parameter r dictates the connectivity of the single qudit mixing Hamiltonian. The case $r = 1$ is a special case called the 'single-qudit ring mixer' (38). In this case, $h_M(1)$ is connected to the two neighbouring computational basis vectors, that is

$$h_M(1)\,|z\rangle = (X + X^\dagger)\,|z\rangle = |z+1 \mod d\rangle + |z-1 \mod d\rangle \tag{3.16}$$

Using $r = 1$, a full mixing Hamiltonian can then be defined as (46)

$$\mathcal{H}_M = \sum_{i=0}^{N-1} h_M(1)_i \tag{3.17}$$

where N is the number of qudits and $h_M(1)_i$ is $h_M(1)$ applied to the ith qudit.

3.3.2 Encoding Constraints for Qudit QAOA

One of the core problems of applying variational quantum algorithms to the shortest vector problem is handling handling the case of the zero vector solution. Naturally, the solution to the unconstrained minimization problem would be the vector which carries zero weight for each basis vector. However, as is represented in the problem definition 2.1.11, we seek the shortest non-zero vector.

Previously, in the work by Albrecht et al. (19) the authors introduce a strategy for encoding the constraint for use with VQEs by modifying the cost function at the point of classical optimization. The authors claim that this approach works for VQE but cannot work for QAOA as the cost function is defined by the optimization strategy itself (19). As an alternative approach for enforcing the constraint, the authors suggest encoding the constraint in the QUBO at the cost of $n - 2$ qubits, which is acknowledged as being impractical.

The claim that the QAOA cost function cannot be modified at the point of classical optimization is contradicted by (37), in which a straightforward approach to

achieving this modification for both qubit and qudit systems is presented. In classical optimization problems, constraints can be enforced by adding penalty functions to the unconstrained minimization problem

$$\tilde{C}(\mathbf{z}) = C(\mathbf{z}) + \sum_{m=1}^{M} \lambda_m P_m[g_m(\mathbf{z})] \tag{3.18}$$

where M is the number of constraints, λ is a penalty coefficient, P is the penalty function, and g is the function representing the constraint (37, 47). Treating the quantum subroutine of QAOA as a black box, a penalty function can then be applied to the classical optimization step to enforce constraints. This achieved by modifying the approximation of the expectation (2.13)

$$\tilde{E}_{\boldsymbol{\gamma},\boldsymbol{\beta}} = \sum_{\text{samples } \mathbf{z}} \mathcal{P}(\mathbf{z}) \left(C(\mathbf{z}) + \sum_{m=1}^{M} \lambda_m P_m[g_m(\mathbf{z})] \right) \tag{3.19}$$

This approach allows for the quantum subroutine to remain unchanged, using a Hamiltonian which encodes an unconstrained version of the optimization problem. However, as the quantum subroutine has not changed, the search space remains unchanged meaning that despite being penalized it is possible for infeasible solutions to be included in the trail state's superposition. This can be handled by classical post processing where infeasible solutions are removed from samples of the final trial state.

An alternative approach to encoding the constraints of an optimization problem is to directly encode the constraint in the problem Hamiltonian (37). The authors present a method for creating a Hamiltonian from the modified cost function defined in equation 3.18

$$H_{\tilde{C}} = \sum_{\mathbf{z}} \tilde{C}(\mathbf{z}) \left|\mathbf{z}\right\rangle \left\langle\mathbf{z}\right| \tag{3.20}$$

This modifies the quantum subroutine of QAOA and can have impacts on the hardware requirements. Depending on the optimization problem and the set of constraints it may be preferable either to encode the penalty at the point of classical optimization to avoid modifications to the quantum hardware, or to modify the Hamiltonian to leave the classical optimization step unchanged.

Chapter 4

VQE for SVP

In this chapter we introduce the technical work conducted for this project. The primary objective of this part of the project is to replicate the results of Albrecht et al. (19). The authors use VQE to solve SVP in small dimensions using the approach described in section 3.1.

4.1 Implementation

As we are initially aiming to reproduce the results of Albrecht et al., the design for the implementation of this variational quantum algorithm comes from their paper (19). The original results make use of *Xacc* and *QUEST* for their implementations of variational quantum algorithms and quantum emulation respectively (19). For our implementation, we aim to reproduce these results using the *Qiskit* (20) python library to simulate the quantum subroutine.

4.1.1 The Problem Hamiltonian

In order to run VQE we must first construct a problem Hamiltonian that encodes the shortest vector problem. Recall the QUBO formulation of the problem (equation 3.5) which encodes the unconstrained version of SVP in order to map the problem to an Ising Hamiltonian. Therefore, for a given lattice $\mathcal{L}(\mathbf{B})$ we seek a representation of SVP of the form

$$c + \sum_{\tilde{x}_{i,j}} c_{i,j} \cdot \tilde{x}_{i,j} + \sum_{\tilde{x}_{i,j}, \tilde{x}_{k,l}} d_{i,j,k,l} \cdot \tilde{x}_{i,j} \cdot \tilde{x}_{k,l} \tag{4.1}$$

Here, the constant coefficients c, $c_{i,j}$, and $d_{i,j,k,l}$, as well as the number of binary variables i, j are dependent on the lattice basis $\mathbf{B}$.

As we will see in section 4.1.4, the bases taken as input to this algorithm do not have to be square, a lattice basis matrix $\mathbf{B} \in \mathbb{Z}^{m \times n}$ generates an n-dimensional lattice

despite being in m-dimensional space. Therefore for a given bound $a \geq |x_i|$ and lattice basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$, with $m > n$, we need to be able to generate the coefficients c, $c_{i,j}$, and $d_{i,j,k,l}$. To do so, we want to find a polynomial expression of the λ^2, the square of the length of the vector generated by $\mathbf{x} \in \{-a, \ldots, a\}^n$, in terms of the variables $\tilde{x}_{i,j} \in \{0, 1\}$.

$$\lambda^2 = \mathbf{x}^T \cdot \mathbf{B}^T \cdot \mathbf{B} \cdot \mathbf{x} \tag{4.2a}$$

$$= \mathbf{x}^T \cdot \mathbf{G} \cdot \mathbf{x} \tag{4.2b}$$

We want to write $\mathbf{x}$ in terms of $\tilde{x}_{i,j}$. Recall equation 3.4,

$$x_i = -a + \sum_{j=0}^{\lfloor \log(2a) \rfloor - 1} \left(2^j \tilde{x}_{i,j}\right) + (2a + 1 - 2^{\lfloor \log(2a) \rfloor}) \cdot \tilde{x}_{i,\lfloor \log(2a) \rfloor} \tag{4.3}$$

Therefore we have

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} -a + \sum_{j=0}^{\lfloor \log(2a) \rfloor - 1} (2^j \tilde{x}_{0,j}) + (2a + 1 - 2^{\lfloor \log(2a) \rfloor}) \cdot \tilde{x}_{0,\lfloor \log(2a) \rfloor} \\ \vdots \\ -a + \sum_{j=0}^{\lfloor \log(2a) \rfloor - 1} (2^j \tilde{x}_{n,j}) + (2a + 1 - 2^{\lfloor \log(2a) \rfloor}) \cdot \tilde{x}_{n,\lfloor \log(2a) \rfloor} \end{bmatrix} \tag{4.4}$$

Let

$$\tilde{\mathbf{X}} = \begin{bmatrix} \tilde{x}_{1,0} & \ldots & \tilde{x}_{1,\lfloor \log(2a) \rfloor} \\ \vdots & \ddots & \vdots \\ \tilde{x}_{n,0} & \ldots & \tilde{x}_{n,\lfloor \log(2a) \rfloor} \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 2^0 \\ \vdots \\ 2^{\lfloor \log(2a) \rfloor - 1} \\ 2a + 1 - 2^{\lfloor \log(2a) \rfloor} \end{bmatrix}, \quad \mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{4.5}$$

where $\dim(\tilde{\mathbf{X}}) = n \times (\lfloor \log(2a) \rfloor + 1)$, $\dim(\mathbf{A}) = \lfloor \log(2a) \rfloor + 1$, and $\dim(\mathbf{1}) = n$. We can then write $\mathbf{x}$ as

$$\mathbf{x} = -a \cdot \mathbf{1} + \tilde{\mathbf{X}} \cdot \mathbf{A} \tag{4.6}$$

We then substitute equation 4.6 into equation 4.2b, giving us

$$\lambda^2 = (-a \cdot \mathbf{1} + \tilde{\mathbf{X}} \cdot \mathbf{A})^T \cdot \mathbf{G} \cdot (-a \cdot \mathbf{1} + \tilde{\mathbf{X}} \cdot \mathbf{A}) \tag{4.7a}$$

$$= \left((-a \cdot \mathbf{1})^T + (\tilde{\mathbf{X}} \cdot \mathbf{A})^T\right) \cdot \mathbf{G} \cdot (-a \cdot \mathbf{1} + \tilde{\mathbf{X}} \cdot \mathbf{A}) \tag{4.7b}$$

which provides a straightforward matrix equation suitable for any valid lattice basis that generates a polynomial in terms of $\tilde{x}_{i,j}$ for the length of the vector generated by $\mathbf{x}$. The coefficients and constant of this polynomial can then be stored giving the values of $c_{i,j}$,$d_{i,j,k,l}$, and c respectively. Of note, the polynomial generated is quadratic and is simplified to a linear function as $\tilde{x}_{i,j} \in \{0, 1\}$ and $\tilde{x}_{i,j}^2 = \tilde{x}_{i,j}$.

Using the coefficients generated above, it is then straightforward to map from the QUBO (equation 4.1) to the problem Hamiltonian. Recall equation 3.2,

$$\mathcal{H} = c + \sum_i c_{ii} \frac{I_i - Z_i}{2} + \sum_{i \neq j} c_{i,j} \frac{I_i - Z_i}{2} \frac{I_j - Z_j}{2} \tag{4.8}$$

Example 4.1.1 *Let* $\mathbf{B}$ *be a basis matrix for a 2-dimensional lattice where*

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \tag{4.9}$$

The Gram matrix $\mathbf{G}$ *is then given by*

$$\mathbf{G} = \mathbf{B}^T \cdot \mathbf{B} = \begin{bmatrix} 1 & 1 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 3 & 9 \end{bmatrix} \tag{4.10}$$

Suppose we choose the bound $a = 1$*, then* $\lfloor \log(2a) \rfloor = 1$*.* $\tilde{\mathbf{X}}$*,* $\mathbf{A}$*, and* $\mathbf{1}$ *are defined as*

$$\tilde{\mathbf{X}} = \begin{bmatrix} \tilde{x}_{1,0} & \tilde{x}_{1,1} \\ \tilde{x}_{2,0} & \tilde{x}_{1,1} \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \mathbf{1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{4.11}$$

The polynomial λ^2 *is then generated from the following matrix multiplication*

$$\lambda^2 = (-a \cdot \mathbf{1} + \tilde{\mathbf{X}} \cdot \mathbf{A})^T \cdot \mathbf{G} \cdot (-a \cdot \mathbf{1} + \tilde{\mathbf{X}} \cdot \mathbf{A}) \tag{4.12a}$$

$$= \left(\begin{bmatrix} -1 \\ -1 \end{bmatrix} + \begin{bmatrix} \tilde{x}_{1,0} & \tilde{x}_{1,1} \\ \tilde{x}_{2,0} & \tilde{x}_{2,1} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)^T \begin{bmatrix} 2 & 3 \\ 3 & 9 \end{bmatrix} \left(\begin{bmatrix} -1 \\ -1 \end{bmatrix} + \begin{bmatrix} \tilde{x}_{1,0} & \tilde{x}_{1,1} \\ \tilde{x}_{2,0} & \tilde{x}_{1,1} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \tag{4.12b}$$

$$\begin{aligned} = & 2\tilde{x}_{1,0}^2 + 4\tilde{x}_{1,0}\tilde{x}_{1,1} + 6\tilde{x}_{1,0}\tilde{x}_{2,0} + 6\tilde{x}_{1,0}\tilde{x}_{2,1} - 10\tilde{x}_{1,0} + 2\tilde{x}_{1,1}^2 + 6\tilde{x}_{1,1}\tilde{x}_{2,0} \\ & + 6\tilde{x}_{1,1}\tilde{x}_{2,1} - 10\tilde{x}_{1,1} + 9\tilde{x}_{2,0}^2 + 18\tilde{x}_{2,0}\tilde{x}_{2,1} - 24\tilde{x}_{2,0} + 9\tilde{x}_{2,1}^2 - 24\tilde{x}_{2,1} + 17 \end{aligned} \tag{4.12c}$$

From this polynomial we are able to extract the coefficients required to produce the Hamiltonian. Let the coefficients be represented by the diagonal Hamiltonian $\mathbf{C}$*, and* c *= 17*

$$\mathbf{C} = \begin{bmatrix} -8 & 4 & 6 & 6 \\ 0 & -8 & 6 & 6 \\ 0 & 0 & -15 & 18 \\ 0 & 0 & 0 & -15 \end{bmatrix} \tag{4.13}$$

Substituting c *and* $\mathbf{C}$ *into equation 4.8 we arrive at the desired problem Hamiltonian.*

4.1.2 Ansatz Circuit

A crucial component of VQE is the selection of the ansatz circuit used to generate the trial states to be applied to the problem Hamiltonian. For our implementation we use the same choice of circuit as Albrecht et al. (19, 48), seen in figure 4.1. This circuit uses C-NOT gates and rotation gates about the Y axis in order to produce trial states.

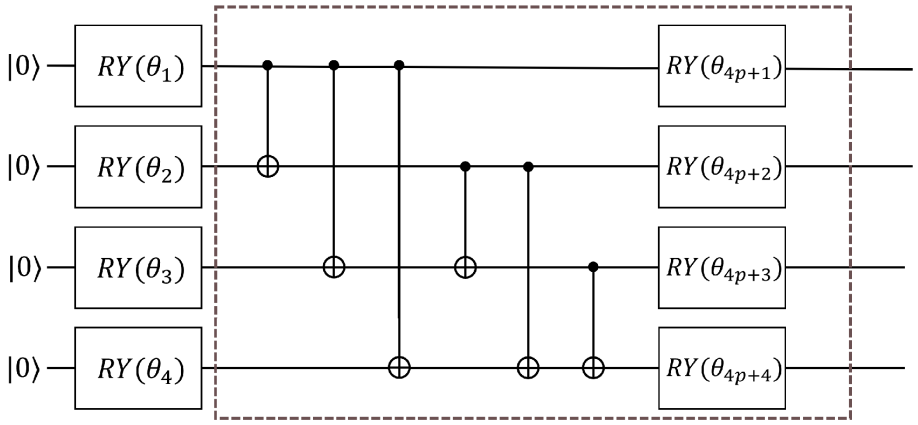

Figure 4.1: A circuit diagram of the ansatz circuit (48)

4.1.3 Classical Optimization

The classical optimization subroutine of this algorithm is comprised of several important components. First, the constraint of the SVP optimization problem is imposed at this point. From equation 3.7, the modified cost function is given by

$$C'(\theta) = \left(\frac{N}{\tilde{N}}\right) \frac{1}{N} \sum_{i=1}^{N} C(\mathbf{m}_i) \tag{4.14}$$

Where $C(\mathbf{z})$ is the original cost function, N is the number of measurements and $\tilde{N}$ is the number of measurements which have non-zero cost. The purpose of this modification is to guide the ansatz circuit away from including the superposition of states which correspond to the zero vector in the trial state.

Continuing to follow the design of Albrecht et al. (19), the cost penalized cost function is further modified to incorporate CVaR$_\alpha$. Recall the definition of the CVaR objective function (equation 3.8),

$$\frac{1}{\lceil \alpha K \rceil} \sum_{k=0}^{\lceil \alpha K \rceil} C(m_i) \tag{4.15}$$

Applying CVaR to the penalized cost function, the resulting modified cost function is

$$C'(\theta) = \left(\frac{N}{\tilde{N}}\right) \frac{1}{\lceil \alpha K \rceil} \sum_{k=0}^{\lceil \alpha K \rceil} C(m_i) \tag{4.16}$$

Empirically, the authors determined that $\alpha = 0.175$ is the most optimal (19).

Finally, the authors selected the constrained optimization by linear approximations (COBYLA) optimization method to perform the classical optimization. This method is commonly used for variational quantum algorithms (49, 50) as it is derivative-free which has been shown to be more efficient compared to gradient-based methods

(51). Additionally, due to the rotation gates used to compose the ansatz circuit, the parameters of variational quantum algorithms are periodic, meaning that we are able to limit the search space by using constrained optimization. Our implementation utilizes the built-in Scipy (52) minimization function, using the COBYLA method.

4.1.4 Generating Instances of Hard Lattices

As seen in section $2.1.3$, the security of a lattice-based cryptographic system is dependent on the worst-case hardness the underlying lattice problem. Therefore, in order to break the system the attacker must be capable of solving the hardest instances of the lattice problem, in our case that problem is SVP. Lattices with short, nearly-orthogonal basis vectors are easier problem instances, whereas lattices with long, nearly-parallel basis vectors are hard problem instances (19).

In the work by Albrecht et al. (19), hard problem instances are generated according to

$$\mathbf{A} = \begin{pmatrix} \mathbf{I}_{d-k} & \tilde{\mathbf{A}} \\ \mathbf{0} & q \cdot \mathbf{I}_k \end{pmatrix} \in \mathbb{Z}^{m \times m} \tag{4.17}$$

where q, k, m are positive integers, $m \geq k$, $\mathbf{I}_x \in \mathbb{Z}^{x \times x}$ is the identity matrix, and $\tilde{\mathbf{A}}$ is randomly sampled from $\mathbb{Z}^{(m-k)\times k}$. The lattice basis $\mathbf{A}$ generates a d-dimensional q-ary lattice. For the experiments conducted in (19), the authors chose the values $q = 65537$, $m = 180$, and $k = 90$.

A reduced lattice basis $\mathbf{A}'$ is generated by running the classical LLL algorithm on the lattice basis $\mathbf{A}$. The input to the VQE algorithm, basis matrix $\mathbf{B} \in \mathbb{Z}^{n \times d}$, is generated by taking the submatrix consisting of the first n rows of $\mathbf{A}'$. The authors state that as LLL solves SVP directly for full rank n-dimensional lattices when n is small, and instead use $n \times d$ bases as input which 'leaves some work for the quantum algorithm to do' (19).

Of note, due to a difference in convention the authors consider the rows of $\mathbf{B}$ as the basis vectors of $\mathcal{L}(\mathbf{B})$, whereas in our implementation we consider the columns $\mathbf{B}$, therefore we take the transpose of $\mathbf{B}$ as the input.

The dataset considered for each dimension $2 \leq n \leq 28$ consists of 128 basis matrices (19). For each matrix, we calculate the length of the shortest vector and the corresponding vector $\mathbf{x}$ via classical enumeration.

4.2 Evaluation

Albrecht et al. (19) consider two primary metrics for measuring the performance of the VQE. The first is the overlap of the final trial state $|\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*\rangle$ with the ground state of problem Hamiltonian. The second is the number of iterations required at the classical optimization step as is the dominant factor when calculating the time-complexity of the algorithm.

Definition 4.2.1 (Overlap) *(48) The overlap of two quantum states is given by the absolute value of their inner product*

$$\textit{overlap}(|\psi\rangle, |\phi\rangle) = |\langle\psi|\phi\rangle|$$

For normalized states, the overlap of a state is $[0, 1]$

The authors consider the overlap between the final trial state and the first excited state of the problem Hamiltonian. However, for any given shortest vector $\mathbf{x}$, the vector $-\mathbf{x}$ will have the same length and exist in the search space, meaning there exists degenerate states. Additionally, due to the representation of $\mathbf{x}$ in terms of $\tilde{x}_{i,j}$ the number of degenerate states is further increased. The number of possible combinations of $\mathbf{x}$ is $(2a+1)^n$ as $|x_i| \leq a$ and $\dim(\mathbf{x}) = n$. The number of possible combinations of $\tilde{\mathbf{X}}$ is $2^{n\cdot(\lfloor\log(2a)\rfloor+1)}$ as $\tilde{x}_{i,j} \in \{0, 1\}$ and $\dim(\tilde{\mathbf{X}}) = n \times (\lfloor\log(2a)\rfloor + 1)$. Therefore, as $(2a+1)^n < 2^{n\cdot(\lfloor\log(2a)\rfloor+1)}$, by the pigeon hole principle, at one excitement level will have degenerate states.

Example 4.2.2 *Consider the system described in example 4.1.1, the possible states are shown in table 4.1. For this system, we can see there are 4 degenerate first excited states corresponding to the shortest vector.*

$\tilde{x}_{1,0}$	$\tilde{x}_{1,1}$	$\tilde{x}_{2,0}$	$\tilde{x}_{2,1}$	x_1	x_2	λ^2
0	0	0	0	-1	-1	17
0	0	0	1	-1	0	2
0	0	1	0	-1	0	2
0	0	1	1	-1	1	5
0	1	0	0	0	-1	9
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	9
1	0	0	0	0	-1	9
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	9
1	1	0	0	1	-1	5
1	1	0	1	1	0	2
1	1	1	0	1	0	2
1	1	1	1	1	1	17

Table 4.1: A table showing the possible states of the system described in example 4.1.1

In the case of degenerate first excited states, the authors calculate the overlap by summing the overlap of the final trial state with each first excited state (48). Let $\mathbf{S}$ be the set of vectors $\mathbf{x}$ such that $\mathbf{B}\cdot\mathbf{x}$ is a shortest vector, then the total overlap of a trial state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ is given by

$$\sum_{\mathbf{x}\in\mathbf{S}} |\langle\boldsymbol{\gamma}, \boldsymbol{\beta}|\mathbf{x}\rangle| \tag{4.18}$$

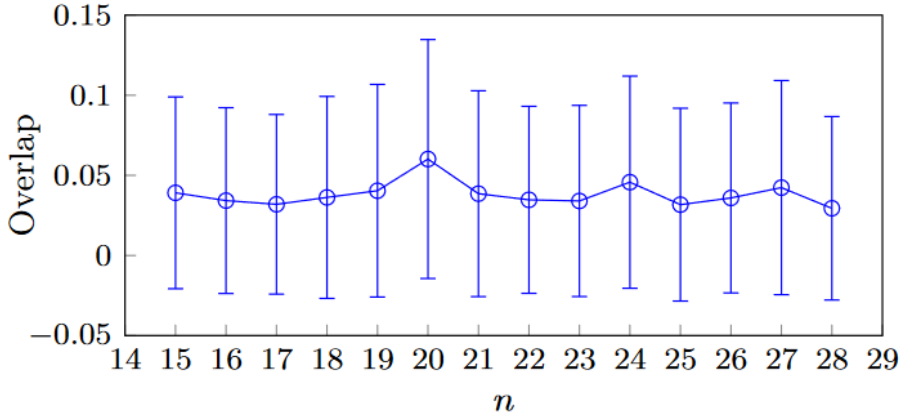

Figure 4.2: The average overlap of the final trial state with the first excited state, for dimensions $15 \leq n \leq 28$ (19)

4.2.1 Experiment 1: Average Overlap

In the first experiment conducted (19), the authors evaluate the mean overlap of the trial state with the set of shortest vectors, shown in figure 4.2. For each dimension, 128 lattice instances are tested using the VQE algorithm. Empirically, the α value found to be most optimal is $\alpha = 0.175$.

We see that as n increases the value of the overlap remains roughly constant at approximately 0.04. This is promising as it shows that the efficacy of this system is stable for $n < 29$. Of note, the authors comment that the average median overlap when $n = 16$ is 0.006. We can see that this median is much less than the reported mean, implying outliers of very high value which skew the mean to be greater. Also, we can see that the lower end of the error bar shows a negative overlap. This is due to a misuse of the central limit theorem, as the overlap cannot be negative. Instead there are likely a few outliers with a high overlap value introducing the wide error range.

We replicated the above experiments using our implementation of the VQE algorithm. We used the same experimental setup, using 128 lattice instances for each dimension and used and α value of 0.175. However, due to the high computational cost of simulating quantum circuits with qiskit and the the fact that the search space grows exponentially in n, we were limited to testing dimensions $n \leq 7$. The results of this experiment are shown in figure 4.3

The algorithm is expected to perform significantly better in a reduced search space and the results shown in figure 4.3 show that to be true. However, the mean overlap appears to increase with the dimension n, which is unexpected. Dimension $n = 7$ in particular achieves a high value for mean overlap. As indicated by the increased standard deviation, this is likely due to a small number of executions which resulted in abnormally high overlaps.

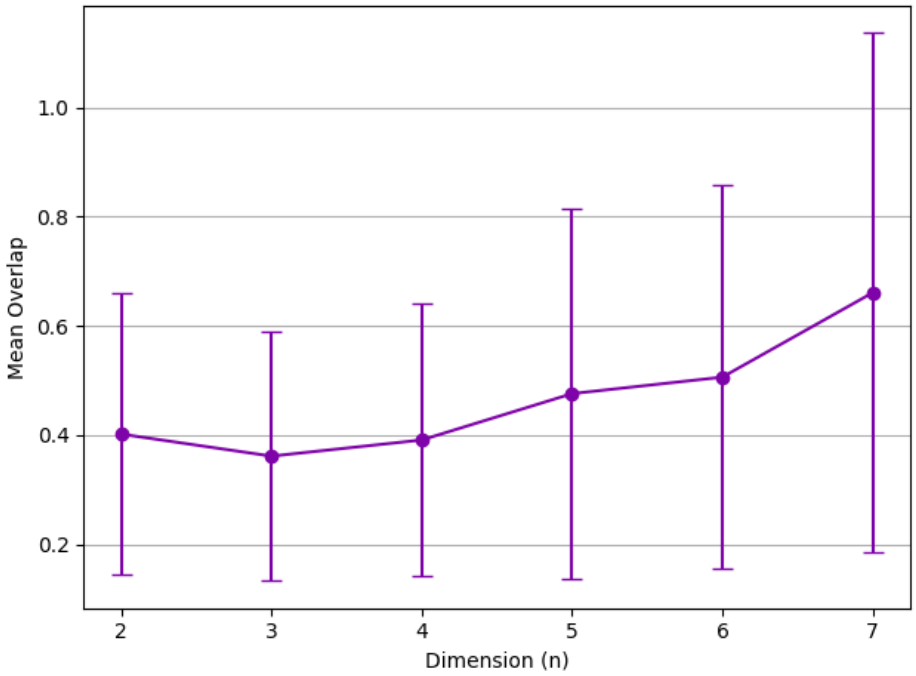

Figure 4.3: The average overlap of the final trial state with the first excited state, for dimensions $2 \leq n \leq 7$, generated using our implementation.

Of note, among these results there are several lattice instances for which the algorithm achieved a mean overlap greater than one. Recalling definition 4.2.1, the overlap of two normalized states should be less that one, indicating that the value calculated in these experiments is not normalized. Potential alternatives to this metric include using a normalized superposition of shortest vectors to calculate the overlap, using a normalized superposition of shortest vectors to calculate the fidelity (square of the overlap), and the approximation ratio (introduced in section 5.3).

4.2.2 Experiment 2: Average Iterations

The second key metric for evaluating the performance of the VQE algorithm is the average number of iteration taken to converge to the final trial state. Using the same experimental setup as before, that is 128 lattice instance, $\alpha = 0.175$, and dimensions $15 \leq n \leq 28$, the results are shown in 4.4. We see that the number of iterations required for convergence increases linearly with n. This does not indicate that finding optimal parameters can be achieved in linear time, only that the the algorithm converges in linear time.

Both results are promising for the use of VQE and variational quantum algorithms for solving SVP, with this algorithm approximately constant overlap and linear time complexity in n. However, the dimensions of lattices for which the algorithm can be simulated classically are much smaller than the dimension of cryptographically relevant lattices $(n \approx 400)$ (19). While we are unable to reasonably extrapolate these results to high dimensions, the do indicate that there is potential promise for using

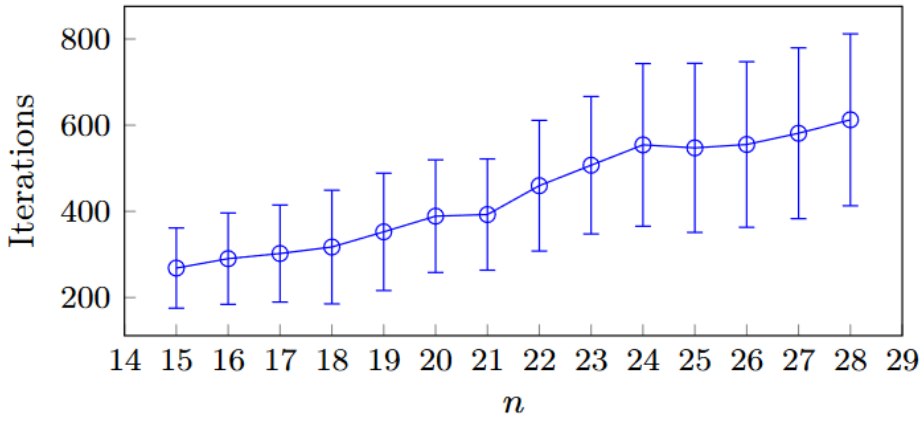

Figure 4.4: The average number of iterations taken to converge, for dimensions $15 \leq n \leq 28$ (19).

variational quantum algorithms to more efficiently solve SVP.

Chapter 5

Qudit-Based QAOA for SVP

In this chapter we continue discussing the technical work conducted for this project. The primary objective is to evaluate QAOA as a potential algorithm for more efficiently solving SVP. In order to achieve this, we first present a novel design for solving SVP using qudit-based QAOA.

5.1 Design

When comparing the use cases of variational quantum algorithms, the VQE algorithm is primarily used to solve problems related to finding the ground state of a quantum system, whereas, QAOA is typically used to solve combinatorial optimization problems. In the first part of the practical work conducted, we saw an application of the VQE algorithm to SVP. In the remaining work discussed, we explore how QAOA may be applied to SVP and compare how each of the variational quantum algorithms perform. Specifically, we introduce and evaluate our qudit-based QAOA algorithm for SVP.

5.1.1 Encoding SVP

The work by Albrecht et al. (19) introduced an encoding of SVP to a QUBO formulation and from the QUBO formulation to an Ising Hamiltonian to be used with either VQE or QAOA. The authors chose to utilise VQE due to the difficulty of encoding the constraint that the solution is non-zero in a form suitable for QAOA. Therefore, in order to use QAOA we introduce a new encoding of the problem.

One key issue identified with the previous encoding is that the QUBO mapping introduces many duplicate states and increases the search space more quickly with respect to n. As seen in section 4.2, for every $(2a+1)^n$ elements in the combinatorial search space $2^{n\cdot(\lfloor\log(2a)\rfloor+1)}$ states are used. In order to limit the additional growth of the search space we adopt a qudit-based system in order to encode the problem using a bijective mapping.

Let $\mathbf{B}$ be a lattice basis which generates lattice $\mathcal{L}(B)$. We want to find a vector $\mathbf{x} \in \mathbb{Z}^n$ such that $\mathbf{B} \cdot \mathbf{x}$ is the shortest vector. Similarly to the previous encoding, we introduce a bound a such that $|x_i| < a$ for each $x_i \in \mathbf{x}$. This allows us to limit the search space to an n-dimensional ball of radius A around origin. Consider a d-level qudit system, where $d = 2a+1$ with n qudits. The encoding of $\mathbf{x} \in \{-a, \ldots, a\}^n$ into $|\mathbf{z}\rangle$ with $\mathbf{z} \in \{0, \ldots, d-1\}^n$ is

$$\mathbf{z} = \mathbf{x} + a \cdot \mathbf{1} \tag{5.1}$$

where $\mathbf{1}$ is the n-dimensional vector with all entries as 1.

Having defined the search space, the QAOA algorithm will then seek to find the state $|\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*\rangle$ that has a high probability of measuring $\mathbf{z}$ where the corresponding $\mathbf{x}$ is the coefficients of the shortest vector. However, the encoding currently ingnores the constraint that $\mathbf{x}$ must be a non-zero vector. Based on the work of Deller et al. (37) (section 3.3.2), we will introduce this constraint directly in to the problem Hamiltonian.

5.1.2 The Problem Hamiltonian

Next, we introduce the problem Hamiltonian. We first define the cost function $C(\mathbf{x})$.

$$C(\mathbf{x}) = \mathbf{x}^T \cdot \mathbf{G} \cdot \mathbf{x} \tag{5.2}$$

where $\mathbf{G}$ is the Gram matrix of $\mathbf{B}$.

To introduce the constraint that the solution must be a non-zero vector, a penalty is introduced to the cost of the zero vector. Recall equation 3.18 (37),

$$\tilde{C}(\mathbf{z}) = C(\mathbf{z}) + \sum_{m=1}^{M} \lambda_m P_m[g_m(\mathbf{z})] \tag{5.3}$$

There is a single constraint to apply so $M = 1$. We must then define P and g, the penalty function and constraint, respectively.

$$g(\mathbf{z}) = \prod_{i=1}^{n} (1 - \delta(z_i, 0)) \tag{5.4}$$

where δ is the Kronecker delta

$$\delta(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases} \tag{5.5}$$

Therefore $g(\mathbf{z}) = 1$ if $\mathbf{z}$ is the zero vector and $g(\mathbf{z}) = 0$ otherwise. Let P_1 be the penalty function for an equality, that is $P_1(x) = |x|$, and the modified cost function is given by,

$$\tilde{C}(\mathbf{z}) = C(\mathbf{z}) + \lambda g(\mathbf{z}) \tag{5.6}$$

Recalling equation 3.18, the problem Hamiltonian $\mathcal{H}_{\tilde{C}}$ given

$$\mathcal{H}_{\tilde{C}} = \sum_{\mathbf{z}} \tilde{C}(\mathbf{z}) \left|\mathbf{z}\right\rangle \left\langle \mathbf{z}\right| \tag{5.7}$$

$H_{\tilde{C}}$ is therefore the diagonal matrix with the values of $\tilde{C}(\mathbf{z})$ on the diagonals. Additionally, we know that this is an Ising Hamiltonian suitable for use with QAOA from equation 3.11 which shows that for an arbitrary cost function $C(\mathbf{z})$ it is possible to represent the corresponding Hamiltonian in terms of only generalized Pauli-Z operators (37).

Example 5.1.1 *Let* $\mathbf{B}$ *be a basis matrix for a 2-dimensional lattice where*

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \tag{5.8}$$

The Gram matrix $\mathbf{G}$ *is then given by*

$$\mathbf{G} = \mathbf{B}^T \cdot \mathbf{B} = \begin{bmatrix} 1 & 1 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 3 & 9 \end{bmatrix} \tag{5.9}$$

Suppose we choose the bound $a = 1$*, we have* $d = 2 \cdot 1 + 1 = 3$ *and therefore use a qudit system with three levels. The encoding of* $\mathbf{x}$ *into* $|\mathbf{z}\rangle$ *is given by*

$$\mathbf{z} = \mathbf{x} + \mathbf{1} \tag{5.10}$$

where $\mathbf{z} \in \{0, 1, 2\}^2$*,* $\mathbf{x} \in \{-1, 0, 1\}^2$*, and* $\mathbf{1} = \binom{1}{1}$*.*

Using the modified cost function (equation 5.6) and the formula for the problem Hamiltonian (equation 5.7), the problem Hamiltonian for the lattice generated by $\mathbf{B}$ *is given by*

$$\mathcal{H}_{\tilde{C}} = \begin{bmatrix} 17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 17 \end{bmatrix} \tag{5.11}$$

where λ *is a parameter to be tuned during the optimization process.*

5.1.3 The Mixing Hamiltonian

The mixing Hamiltonian must allow the algorithm to sufficiently explore the state space in order to find the optimal solution. In section 3.3.1 two mixing Hamiltonians

suitable for qudit systems were introduced. The first is based on angular momentum operators and the other based on generalized Pauli-X operators. In our design, we opted for the later, but as noted by Deller et al. (37) the choice should not have an impact on the performance of the algorithm.

Recall equation 3.17 (37),

$$\mathcal{H}_M = \sum_{i=0}^{N-1} h_M(1)_i \tag{5.12}$$

The Mixing Hamiltonian is given by the sum of partial single-qudit ring mixers over all qudits. Recalling 3.16, these partial mixers are defined as

$$h_M(1)\,|z\rangle = (X + X^\dagger)\,|z\rangle = |z+1 \mod d\rangle + |z-1 \mod d\rangle \tag{5.13}$$

Example 5.1.2 *Consider a qudit system with three levels. The generalized Pauli-X operator, and its inverse, are given by*

$$X = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad X^\dagger = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{5.14}$$

Therefore,

$$h_M(1)\,|z\rangle = (X + X^\dagger)\,|z\rangle = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{5.15}$$

5.2 Implementation

In order to classically simulate the qudit system, we developed a custom simulator for qudit-based qaoa. Qiskit does not provide a way to simulate qudits, and while some qudit simulators do exist (53), we chose to create our own simulator as specific operations in the QAOA algorithm can be implemented more efficiently. For example, the application of the problem Hamiltonian Unitary is reduced the cross product of two vectors instead of the application of many generalized Pauli-Z gates.

5.2.1 The Mixing Hamiltonian

The unitary generated by the mixing Hamiltonian for one qudit is given by

$$U_M(\beta) = \exp\left(-i\beta\left(X + X^\dagger\right)\right) \tag{5.16}$$

This can be implemented with the application of $d-1$ X gates per qudit.

$$U_M(\beta) = \exp\left(-i\beta\left(X + X^\dagger\right)\right) \tag{5.17}$$

$$= \exp\left(-i\beta X\right)\exp\left(-i\beta X^\dagger\right) \tag{5.18}$$

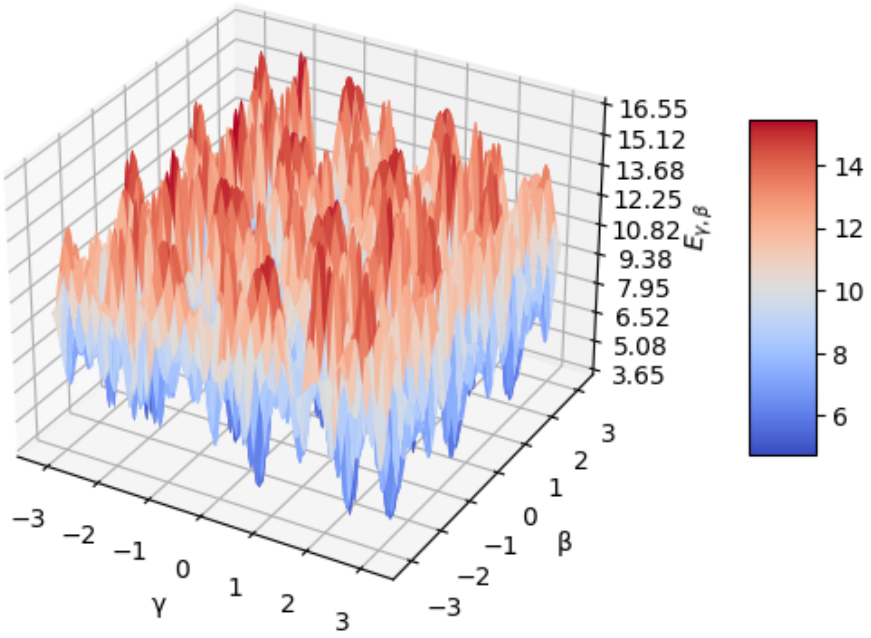

Figure 5.1: The optimization landscape for 2-dimensional lattice with $p = 1$

The Maclaurin series expansion of $U_M(\beta)$ is therefore

$$U_M(\beta) = \left(\sum_{n=0}^{\infty} \frac{(-i\beta X)^n}{n!}\right)\left(\sum_{n=0}^{\infty} \frac{(-i\beta X^{-1})^n}{n!}\right) \tag{5.19}$$

As the generalized Pauli-X operator is $X|z\rangle = |z+1 \mod d\rangle$, we have

$$X^d = I = \left(X^\dagger\right)^d \tag{5.20}$$

Therefore the X operator can be applied at most $d-1$ times and result in a unique state. We can then group the coefficients of like terms from the infinite sums and apply $d-1$ X gates.

5.2.2 Classical Optimization

The classical optimization method chosen for our implementation is COBYLA, for the same reasons described in section 4.1.3. COBYLA is typically used in QAOA algorithms for its balance between rate of convergence and accuracy of results (49).

However, during initial testing of the algorithm we observed that the optimizer would regularly find a local minima different to the global minima. Figure 5.1 shows the optimization landscape of a 2-dimensional lattice with 1 QAOA layer ($p = 1$). For even a simple lattice the optimization landscape is rough with many local optima and minima.

In order to improve the quality of solutions generated by the algorithm, we implemented a multi-start optimization strategy using COBYLA. Using restarts is a good way to improve the accuracy of local optimizers when dealing with complex optimization landscapes as the initial parameters may greatly affect the end result (46). Optimizing using a multi-start strategy has been shown to improve the quality of QAOA results (54).

Another method used for improving the performance of the algorithm is utilizing CVaR when calculating the expectation during the classical optimization step. This is implemented in the same way as described in section 4.1.3.

5.3 Evaluation

In order to evaluate the performance of our qudit-based QAOA algorithm for SVP, we first calculated the same metrics as used by Albrecht et al. (19) for evaluating the VQE algorithm. These are the mean overlap between the final trial state and the set of shortest vectors and the number of iterations required for convergence. In addition, we calculate the approximation ratio of the final trial state. This metric gives us an understanding of how well the solution approximates SVP.

Definition 5.3.1 (Approximation Ratio) *(55) The approximation ratio $\bar{\alpha}$ is given by*

$$\bar{\alpha} = \frac{E_{\boldsymbol{\gamma}^*,\boldsymbol{\beta}^*}}{C_{\min}}$$

where $C_{\min} = \min_{\mathbf{z}} C(\mathbf{z})$ and $\bar{\alpha} \geq 1$

A key structural difference between VQE and QAOA is that QAOA is comprised of many repeating layers, where the number of layers is denoted p. Theoretically, as p approaches infinity, QAOA approaches adiabatic evolution and the final state would be the ground state of the problem Hamiltonian. Practically, as p increases the circuit depth increases which would introduce more noise and errors when being executed on a NISQ device. Additionally, the number of parameters to optimize increases and the classical optimization step becomes harder. Therefore in our experiments we consider a range of values of p.

The lattice inputs considered in these experiments are the same lattices generated for the VQE experiments. Unless otherwise stated, averages are taken over 128 lattice instances.

5.3.1 Experiment 1: Determining α

In order to implement CVaR as a strategy for improving the optimization step, we must first determine a value of α. CVaR provides the most benefit when α is small (44). We therefore tested a range of values of α, with $\alpha \in [0.025, 0.25]$.

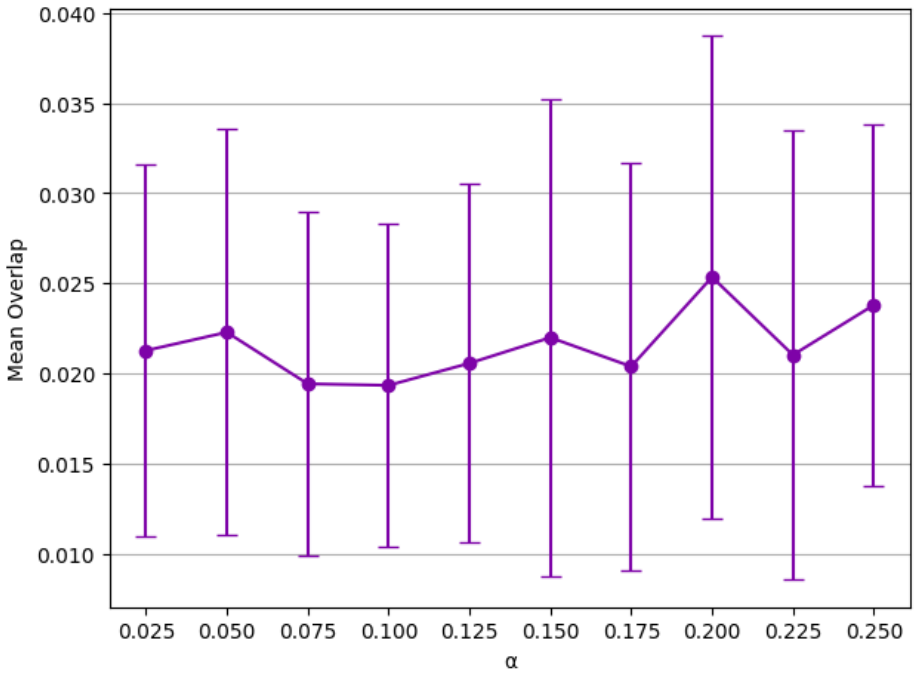

Figure 5.2: The mean overlap with the shortest vector as a function of α, for a set of 32 lattices with dimension $n = 8$

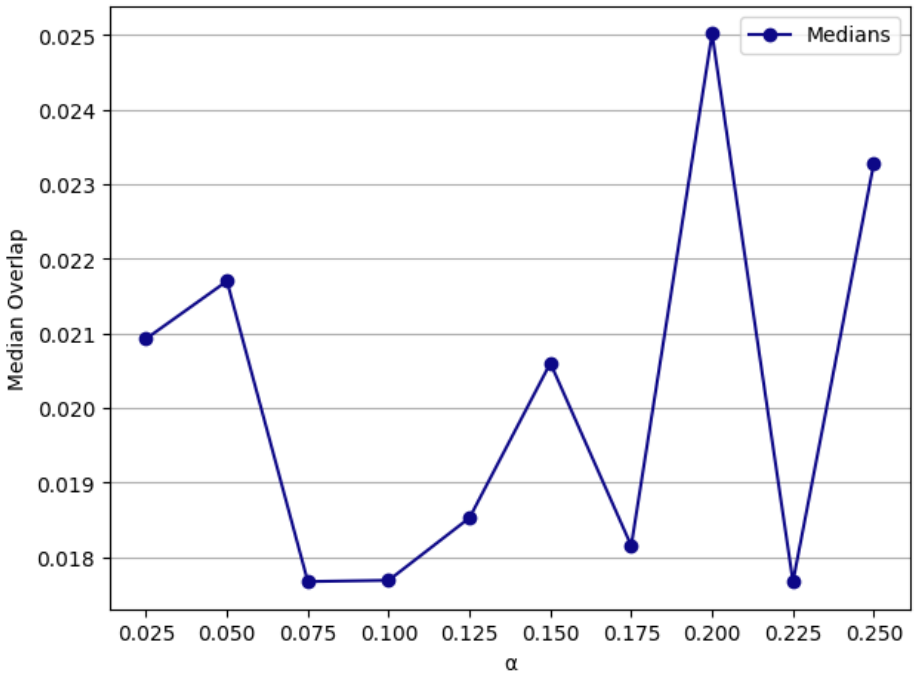

Figure 5.3: The median overlap with the shortest vector as a function of α, for a set of 32 lattices with dimension $n = 8$

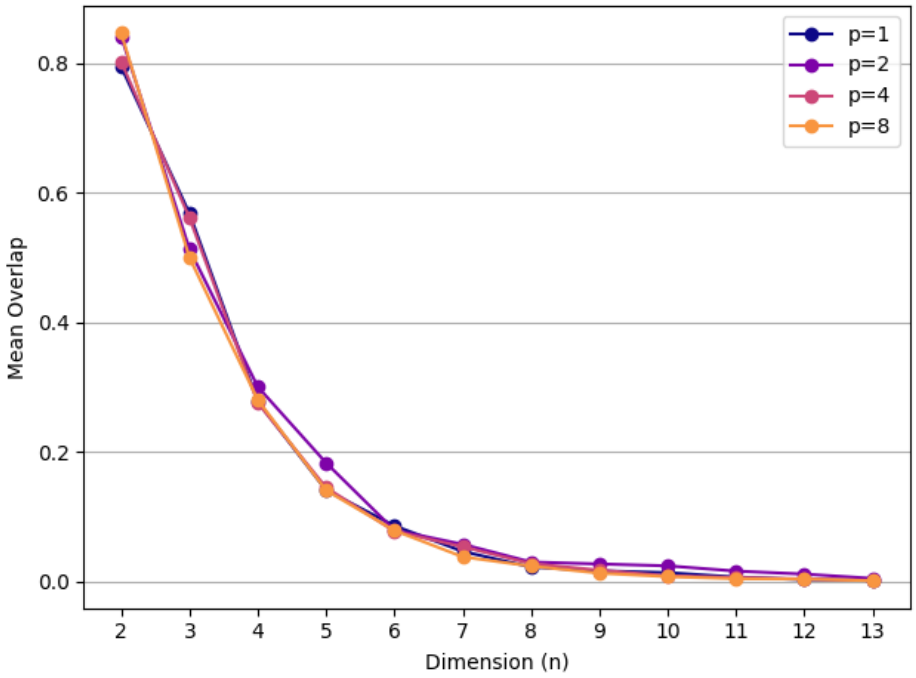

Figure 5.4: The mean overlap of the final state with the set of shortest vectors

Figure 5.2 and figure 5.3 respectively show the mean overlap and median overlap with the shortest vector as a function of α. These values were calculated using a set of 32 lattices with dimension $n = 8$. For both the median and mean we can see that $\alpha = 0.2$ resulted in the greatest overlap. For the remaining experiments, we used $\alpha = 0.2$.

5.3.2 Experiment 2: Average Overlap

Figure 5.4 and 5.5 shows the mean overlap of the final trial state with the set of shortest vectors for $6 \leq n \leq 11$ and for different values of p. We see that as n increases the the value of the mean overlap rapidly decreases. Figure 5.4 indicates that this decay may be asymptotic.

The value of p which performs the best is $p = 2$, which differs from what is expected theoretically. As p increases, the final trial state should increasingly become more similar to the ground state of the problem Hamiltonian and the solution to SVP. However, this does not account for the noise present in NISQ devices nor the increased difficulty of optimizing a greater number of parameters. As our simulator does not artificially introduce noise, this indicates that after a certain point the algorithm is failing to successfully optimize the parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$.

For $8 \leq n \leq 12$, the best performing value of the overlap is between 0.01 and 0.03 which is slightly worse than the mean overlap of the VQE algorithm.

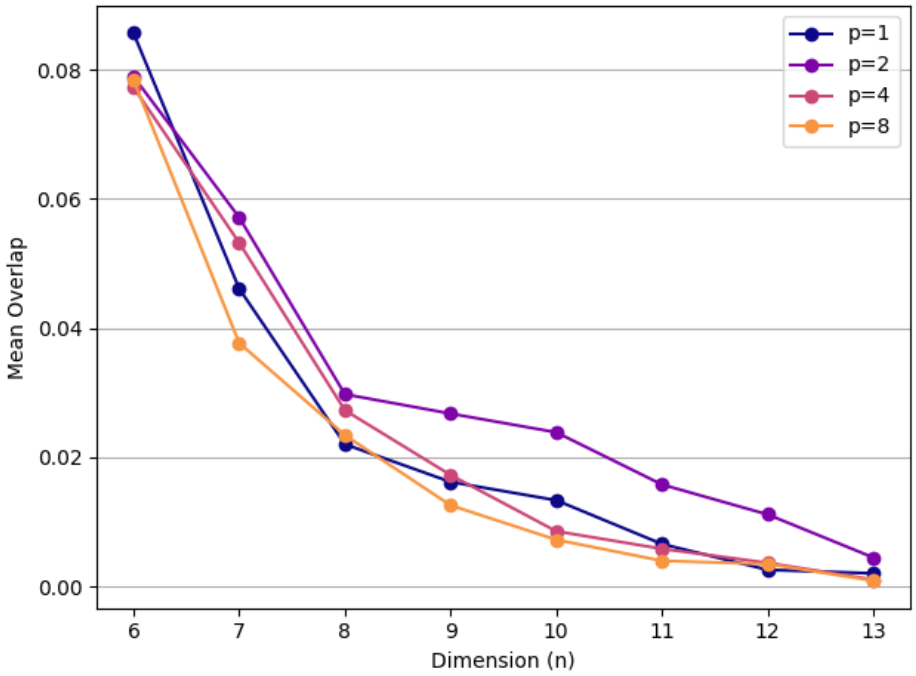

Figure 5.5: The mean overlap of the final state with the set of shortest vectors, for a reduced number of dimensions

5.3.3 Experiment 3: Average Iterations

Figure 5.6 shows the number of iterations required for the QAOA algorithm to converge on a final trial solution. Firstly, as is expected, the number of iterations increases as p increases. This is due to the fact that there are more parameters to optimize. Secondly, the number of iterations remains roughly constant as n increases. We expect that the number of iterations should increase as the problem complexity increases, as seen with VQE in figure 4.4.

Taking into consideration that the mean value of the overlap quickly decays as n increases, and that the number of iterations in the optimization remains constant we conclude that the optimizer is failing to successfully find a global minima. Instead, the optimization gets stuck early and converges on a local minima. Referring back to figure 5.1, we can see that this is because of a rough optimization landscape that only becomes increasingly more difficult as n and p increases.

5.3.4 Experiment 4: Average Approximation Ratio

The final metric used to evaluate the QAOA algorithm is the mean approximation ratio. This measures the factor by which the QAOA output approximates the true shortest vector.

Figure 5.7 shows the mean approximation ratio increasing linearly with n. As before, the algorithm performs best when $p = 2$. However, even in the small dimensions

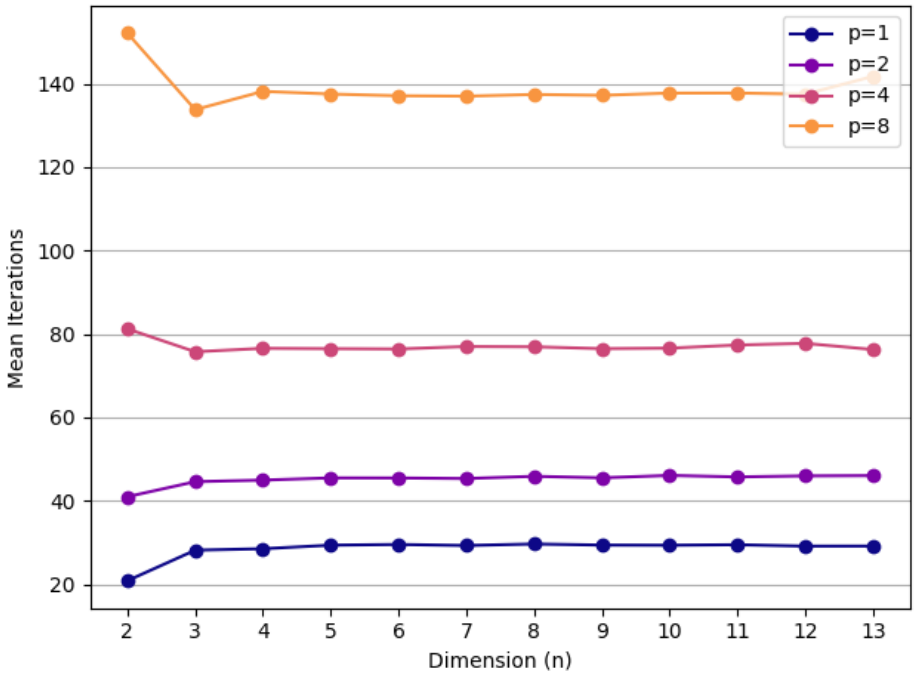

Figure 5.6: The number of iterations required to converge on a final trial state

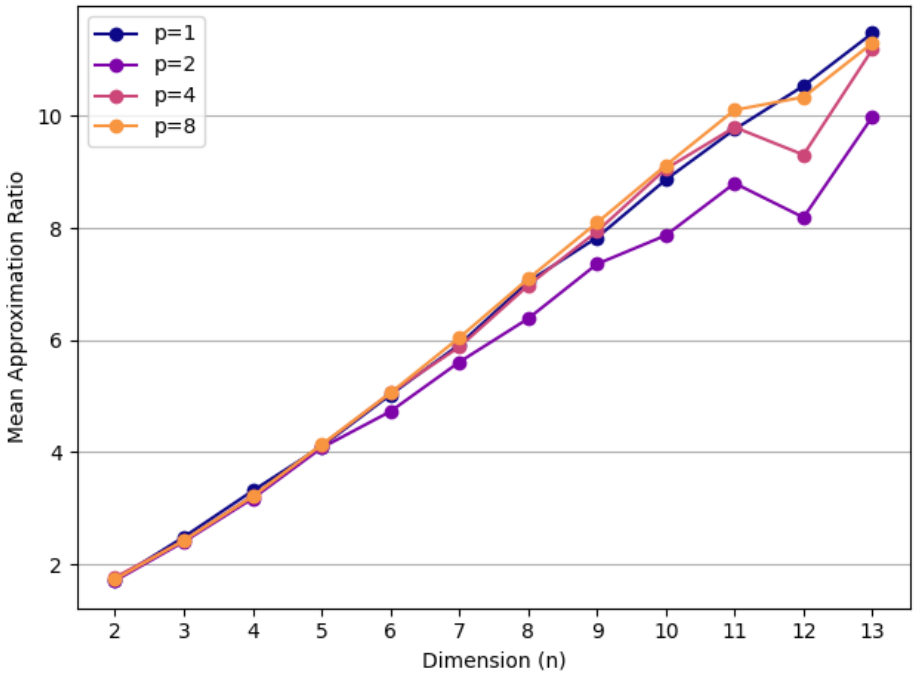

Figure 5.7: The mean approximation ratio of the final state

which were simulated for this experiment, the approximation ratio reaches $\bar{\alpha} > 11$.

The large approximation ratio is characteristic of a 'flat' final state $|\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*\rangle$, meaning that the final state is comprised of the superposition of many different states each with small probability. This indicates that the algorithm is not able to successfully optimize to a final state which collapses to the solution with high probability.

5.3.5 Summary

In summary, several experiments were conducted to evaluate the performance of the qudit-based QAOA for SVP system. Initially, we empirically found the best α value to be $\alpha = 0.2$, which was set for the remaining experiments. Following this, we investigated the overlap between the final state and the set of shortest vectors, the number of iterations required for the optimization to converge, and the approximation ratio.

From these experiments we saw that for very small n, the algorithm performs very well and then performs less well as n increases. The drop off in mean overlap, combined with the constant number of iterations required for convergence and linearly increasing approximation ratio, are symptomatic of the COBYLA optimizer becoming stuck in a local minima and not being able to find the global minima.

Throughout the experiments the algorithm performed best when $p = 2$. This is likely because using two layers strikes a balance between being able to create an ansatz similar enough to the solution and having a small number of parameters to optimize.

Overall, qudit-based QAOA for SVP performed comparably to the VQE algorithm produced by Albrecht et al. (19). For the dimensions tested, the QAOA approach performed slightly worse with regards to average overlap with the set of shortest vectors. With further investigation into optimization strategies, we

Chapter 6

Conclusion and Future Work

6.1 Contributions

In this work, we successfully:

- Proposed a novel encoding of the shortest vector problem for qudit-based QAOA. This reduces the quantum resources required to run the algorithm compared to a QUBO encoding (19) for qubit QAOA as the search space is encoded more compactly, with each solution being represented by exactly one quantum state.

- Developed a custom quantum simulator for the quantum subroutine of qudit-based QAOA for SVP. Creating a simulator specific to this problem allows for several optimizations to be made which allows for more efficient simulation of the preparation of the trial state. This improved performance allows for the simulation of problem instances in higher dimensions than could be achieved otherwise.

- Reproduced the results of Albrecht et al. (19) for instances of lattices in small dimensions, implementing the VQE algorithm using Qiskit (20). Our results confirmed that using a QUBO encoding of SVP and finding the ground state via VQE is a viable approach to solving SVP.

- Conducted experimental evaluation of qudit-based QAOA for SVP. Our findings showed that this approach is promising as compared to VQE for SVP (19) the performance was comparable, but did slightly under-perform. Overall, the experimental results indicated that the optimization step of the algorithm regularly resulted in local minima and struggled to find the global minima.

6.2 Future Work

There are a number of interesting directions in which the work presented here can be continued,

- **Benchmarking Quantum and Classical Algorithms for SVP**
  As research into variational quantum algorithms for SVP continues, we believe there is a need to create a set of benchmarks for both classical and quantum SVP solvers in order to accurately assess the performance and progression of quantum algorithms for SVP.

- **Improved Optimization Strategies**
  As discussed in the evaluation of qudit-based QAOA for SVP, the optimization step of the algorithm fails to find the global minima. An investigation into alternative optimization methods and strategies to improve the performance of the optimizer would likely allow the algorithm see improved performance.

- **Alternative Mixing Hamiltonians**
  As discussed in the evaluation of qudit-based QAOA for SVP, the optimization landscape for this algorithm is very rough and contains many local minima. The mixing Hamiltonian affects the shape of the optimization landscape and so implementing alternative mixing Hamiltonians may yield improved performance

- **Alternative Encoding**
  Consider the cost function $C(\mathbf{x})$ of our encoding of SVP, we observe the costs are symmetric around the zero vector. That is the vector $\mathbf{x}$ and $-\mathbf{x}$ have the same value. Using this fact, it may be possible to dramatically reduce the search space and quantum resources required by considering an encoding which is able to leverage this symmetry.

Bibliography

[1] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976. ISSN 1557-9654. doi: 10.1109/TIT.1976.1055638.

[2] Martin E Hellman. An Overview of Public Key Cryptography. *IEEE COMMUNICATIONS SOCIETY MAGAZINE*, 1978.

[3] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2): 120–126, February 1978. ISSN 0001-0782. doi: 10.1145/359340.359342.

[4] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, Lecture Notes in Computer Science, pages 10–18, Berlin, Heidelberg, 1985. Springer. ISBN 978-3-540-39568-3. doi: 10.1007/3-540-39568-7_2.

[5] P.W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, November 1994. doi: 10.1109/SFCS.1994.365700.

[6] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997. ISSN 0097-5397, 1095-7111. doi: 10.1137/S0097539795293172.

[7] Jin-Yi Cai. Shor's Algorithm Does Not Factor Large Integers in the Presence of Noise, June 2023.

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

[9] Information Technology Laboratory Computer Security Division. Post-Quantum Cryptography — CSRC — CSRC. https://csrc.nist.gov/projects/post-quantum-cryptography, January 2017.

[10] Classic McEliece: Intro. https://classic.mceliece.org/index.html.

[11] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Post-*

Quantum Cryptography. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-540-88701-0 978-3-540-88702-7. doi: 10.1007/978-3-540-88702-7.

[12] Handbook of Applied Cryptography. https://cacr.uwaterloo.ca/hac/.

[13] Peter Schwabe. Kyber. https://pq-crystals.org/kyber/, .

[14] Falcon. https://falcon-sign.info/.

[15] Peter Schwabe. SPHINCS+. https://sphincs.org/, .

[16] NIST Announces First Four Quantum-Resistant Cryptographic Algorithms. https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms, July 2022.

[17] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a quantum processor. *Nature Communications*, 5(1):4213, July 2014. ISSN 2041-1723. doi: 10.1038/ncomms5213.

[18] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm, November 2014.

[19] Martin R. Albrecht, Miloš Prokop, Yixin Shen, and Petros Wallden. Variational quantum solutions to the Shortest Vector Problem. *arXiv.org*, February 2022. doi: 10.22331/q-2023-03-02-933.

[20] Qiskit. https://qiskit.org.

[21] Yang Li, Kee Siong Ng, and Michael Purcell. A Tutorial Introduction to Lattice-based Cryptography and Homomorphic Encryption, September 2022.

[22] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, September 2009. ISSN 0004-5411. doi: 10.1145/1568318.1568324.

[23] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, New York, NY, USA, July 1996. Association for Computing Machinery. ISBN 978-0-89791-785-8. doi: 10.1145/237814.237838.

[24] Phong Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2:181–207, July 2008. doi: 10.1515/JMC.2008.009.

[25] Daniele Micciancio and Michael Walter. Fast Lattice Point Enumeration with Minimal Overhead. In *Proceedings of the 2015 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 276–294. Society for Industrial and Applied Mathematics, December 2014. ISBN 978-1-61197-374-7. doi: 10.1137/1.9781611973730.21.

[26] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982. ISSN 1432-1807. doi: 10.1007/BF01457454.

[27] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66 (1):181–199, August 1994. ISSN 1436-4646. doi: 10.1007/BF01581144.

[28] David Joseph. A suite of quantum algorithms for the shortestvector problem. August 2021. doi: 10.25560/96595.

[29] M. Born and V. Fock. Beweis des Adiabatensatzes. *Zeitschrift für Physik*, 51(3): 165–180, March 1928. ISSN 0044-3328. doi: 10.1007/BF01343193.

[30] Michael A. Nielsen and Isaac L. Chuang. Quantum Computation and Quantum Information: 10th Anniversary Edition. https://www.cambridge.org/highereducation/books/quantum-computation-and-quantum-information/01E10196D0A682A6AEFFEA52D53BE9AE, December 2010.

[31] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum Computation by Adiabatic Evolution, January 2000.

[32] Peter Deglmann, Ansgar Schäfer, and Christian Lennartz. Application of quantum calculations in the chemical industry—An overview. *International Journal of Quantum Chemistry*, 115(3):107–136, 2015. ISSN 1097-461X. doi: 10.1002/qua.24811.

[33] Mucio A. Continentino. *Key Methods and Concepts in Condensed Matter Physics: Green's Functions and Real Space Renormalization Group*. IOP Publishing, April 2021. ISBN 978-0-7503-3395-5.

[34] David J Griffiths. *Introduction to Quantum Mechanics: Pearson New International Edition*. Pearson Education Limited, NOIDA, INDIA, 2013. ISBN 978-1-292-03714-1.

[35] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. The Variational Quantum Eigensolver: A review of methods and best practices. *Physics Reports*, 986:1–128, November 2022. ISSN 03701573. doi: 10.1016/j.physrep.2022.08.003.

[36] Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*, volume 222 of *Graduate Texts in Mathematics*. Springer International Publishing, Cham, 2015. ISBN 978-3-319-13466-6 978-3-319-13467-3. doi: 10.1007/978-3-319-13467-3.

[37] Yannick Deller, Sebastian Schmitt, Maciej Lewenstein, Steve Lenk, Marika Federer, Fred Jendrzejewski, Philipp Hauke, and Valentin Kasper. Quantum approximate optimization algorithm for qudit systems. *Physical Review A*, 107(6):

062410, June 2023. ISSN 2469-9926, 2469-9934. doi: 10.1103/PhysRevA.107.062410.

[38] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms*, 12(2):34, February 2019. ISSN 1999-4893. doi: 10.3390/a12020034.

[39] Hsuan-Hao Lu, Zixuan Hu, Mohammed Saleh Alshaykh, Alexandria Jeanine Moore, Yuchen Wang, Poolad Imany, Andrew Marc Weiner, and Sabre Kais. Quantum Phase Estimation with Time-Frequency Qudits in a Single Photon. *Advanced Quantum Technologies*, 3(2):1900074, 2020. ISSN 2511-9044. doi: 10.1002/qute.201900074.

[40] A. B. Klimov, R. Guzmán, J. C. Retamal, and C. Saavedra. Qutrit quantum computer with trapped ions. *Physical Review A*, 67(6):062313, June 2003. doi: 10.1103/PhysRevA.67.062313.

[41] Shruti Dogra, Arvind, and Kavita Dorai. Determining the parity of a permutation using an experimental NMR qutrit. *Physics Letters A*, 378(46):3452–3456, October 2014. ISSN 0375-9601. doi: 10.1016/j.physleta.2014.10.003.

[42] Léo Ducas. Shortest Vector from Lattice Sieving: A Few Dimensions for Free. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, volume 10820, pages 125–145. Springer International Publishing, Cham, 2018. ISBN 978-3-319-78380-2 978-3-319-78381-9. doi: 10.1007/978-3-319-78381-9_5.

[43] David Joseph, Alexandros Ghionis, Cong Ling, and Florian Mintert. Not-so-adiabatic quantum computation for the shortest vector problem. *Physical Review Research*, 2(1):013361, March 2020. ISSN 2643-1564. doi: 10.1103/PhysRevResearch.2.013361.

[44] Panagiotis Kl Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving Variational Quantum Optimization using CVaR. *Quantum*, 4:256, April 2020. ISSN 2521-327X. doi: 10.22331/q-2020-04-20-256.

[45] Yuchen Wang, Zixuan Hu, Barry C. Sanders, and Sabre Kais. Qudits and High-Dimensional Quantum Computing. *Frontiers in Physics*, 8, 2020. ISSN 2296-424X.

[46] Jordi R. Weggemans, Alexander Urech, Alexander Rausch, Robert Spreeuw, Richard Boucherie, Florian Schreck, Kareljan Schoutens, Jiří Minář, and Florian Speelman. Solving correlation clustering with QAOA and a Rydberg qudit system: A full-stack approach. *Quantum*, 6:687, April 2022. doi: 10.22331/q-2022-04-13-687.

[47] Alice Smith, David Coit, Thomas Bäck, David Fogel, and Zbigniew Michalewicz. Penalty Functions. July 1998.

[48] Miloš Prokop. RE: Inquiry about Variational Quantum Solutions to the Shortest Vector Problem.

[49] Mario Fernández-Pendás, Elías F. Combarro, Sofia Vallecorsa, José Ranilla, and Ignacio F. Rúa. A study of the performance of classical minimizers in the Quantum Approximate Optimization Algorithm. *Journal of Computational and Applied Mathematics*, 404:113388, April 2022. ISSN 0377-0427. doi: 10.1016/j.cam.2021.113388.

[50] M. J. D. Powell. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In Susana Gomez and Jean-Pierre Hennart, editors, *Advances in Optimization and Numerical Analysis*, pages 51–67, Dordrecht, 1994. Springer Netherlands. ISBN 978-90-481-4358-0 978-94-015-8330-5. doi: 10.1007/978-94-015-8330-5_4.

[51] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, February 2016. ISSN 1367-2630. doi: 10.1088/1367-2630/18/2/023023.

[52] SciPy. https://scipy.org/.

[53] Cirq — Google Quantum AI. https://quantumai.google/cirq.

[54] Ruslan Shaydulin, Ilya Safro, and Jeffrey Larson. Multistart Methods for Quantum Approximate optimization. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8, September 2019. doi: 10.1109/HPEC.2019.8916288.

[55] Kostas Blekos, Dean Brand, Andrea Ceschini, Chiao-Hui Chou, Rui-Hao Li, Komal Pandya, and Alessandro Summer. A Review on Quantum Approximate Optimization Algorithm and its Variants, June 2023.