Submitted in accordance with the requirements for the degree of
Doctor of Philosophy

UNIVERSITY OF LEEDS

SCHOOL OF PHYSICS & ASTRONOMY

Application of quantum walks on graph structures to quantum computing

Neil Brian Lovett

March 2011

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Within this thesis, several chapters form the basis of jointly authored publications, which are detailed as follows:

1. Chapter 3 - Universal computation using the discrete time quantum walk. This work has been published in Physical Review A [1]. It was partly completed as a summer project in 2009 in which I took a lead role in supervising and guiding the students. In addition, I then completed the remainder of the work prior to publication.

2. Chapter 5 - Quantum walk search algorithm on non-periodic structures. In this chapter, sec. 5.2 was work completed while assisting several undergraduates with final year BSc projects. It is included in a post conference proceedings from Physics and Computation 2009 where I presented this and other work [2]. The remainder of the chapter is solely my own work.

3. Chapter 7 - Effects of connectivity on the quantum walk search algorithm on regular structures. This work forms the basis of a publication which is currently undergoing peer review [3]. It was completed as part of a summer project in 2010, where I took the lead role in supervising and guiding the student.

4. Chapter 8 - Quantum walk searching on percolation lattices. Whilst the work in this chapter has not yet been submitted for publication, it was completed with an undergraduate student, Rob Heath, as part of his final year MPhys project.

5. Chapter 9 - Generation of topologically useful entangled states. This work was completed with the assistance and guidance of my secondary supervisor, Ben Varcoe, and is currently accepted for publication [4].

Acknowledgements

I would like to begin by thanking Viv Kendon for her supervision during my PhD. She has been a constant source of support and knowledge throughout, and has constantly encouraged me even when I thought I could never finish. She has helped me solve many issues I have encountered, along with providing new ways of looking at problems. I would also like to thank her for both her assistance in securing a post-doctoral position at the University of Calgary, and the amount of travel I have been lucky enough to do.

I extend my thanks to my secondary supervisor, Ben Varcoe, who provided additional support throughout my PhD, specifically during the time we collaborated on the experimental cluster state generation scheme detailed in this thesis. In addition, I thank all the other lecturers within the QI group for their help and support during my time at Leeds, in particular, Jacob Dunningham and Tim Spiller.

I would also like to thank all the undergraduate project students I have helped to supervise over the past few years. This experience has allowed me to not only extend my work in many different areas, but also to increase my communication and interpersonal skills.

Aside from my PhD work, I would like thank Jessica Cooper who I met at the start of my PhD. We have become, and I am sure will continue to be, great friends and I have greatly enjoyed the time we have spent together sharing an office. I am grateful for all the fun times we have had, including many distractions from both work and writing our theses. On a more serious note, Jessica has also helped me with many problems I have encountered during my PhD, both academically and on

a personal level. In addition, I would like to extend my thanks and best wishes to Simon and I look forward to seeing them get married this summer.

Finally, I would like to thank everyone else who has known and supported me during my PhD. In particular, my family, especially my niece and nephew - Arianna and Cavan - have provided constant support and confidence in my abilities. I must also mention all my friends who, without their support and friendship, I don't think I would have been able to complete this thesis. In particular, my thanks go to Glen, Kirk, Nick and my wee Scottish lassie Maya for some amazing times in the past three years.

Abstract

Quantum computation is a new computational paradigm which can provide fundamentally faster computation than in the classical regime. This is dependent on finding efficient quantum algorithms for problems of practical interest. One of the most successful tools in developing new quantum algorithms is the quantum walk. In this thesis, we explore two applications of the discrete time quantum walk. In addition, we introduce an experimental scheme for generating cluster states, a universal resource for quantum computation.

We give an explicit construction which provides a link between the circuit model of quantum computation, and a graph structure on which the discrete time quantum walk traverses, performing the same computation. We implement a universal gate set, proving the discrete time quantum walk is universal for quantum computation, thus confirming any quantum algorithm can be recast as a quantum walk algorithm.

In addition, we study factors affecting the efficiency of the quantum walk search algorithm. Although there is a strong dependence on the spatial dimension of the structure being searched, we find secondary dependencies on other factors including the connectivity and disorder (symmetry). Fairly intuitively, as the connectivity increases, the efficiency of the algorithm increases, as the walker can coalesce on the marked state with higher probability in a quicker time. In addition, we find as disorder in the system increases, the algorithm can maintain the quantum speed up for a certain level of disorder before gradually reverting to the classical run time.

Finally, we give an abstract scheme for generating cluster states. We see a linear scaling, better than many schemes, as doubling the size of the generating grid in

our scheme produces a cluster state which is double the depth. Our scheme is able to create other interesting topologies of entangled states, including the unit cell for topological error correcting schemes.

List of publications

Below are a list of publications that have resulted from my PhD.

1. Universal quantum computation using the discrete time quantum walk, N. B. Lovett, S. Cooper, M. Everitt, M. Trevers and V. Kendon, Physical Review A, 81, 042330, 2010.

2. Spatial search using the discrete time quantum walk, N. B. Lovett, M. Everitt, M. Trevers, D. Mosby, D. Stockton and V. Kendon, Accepted and to appear in Natural Computation, Proceedings of Physics and Computation 2009. ArXiv preprint arXiv:1010.4705.

3. Effect of connectivity on the coined quantum walk search algorithm, N. B. Lovett, M. Everitt and V. Kendon, Submitted to Theory of Computing Systems as part of Proceedings of Computability in Europe 2010.

4. Generation of topologically useful entangled states, N. B. Lovett and B. T. H. Varcoe, Accepted and to appear in International Journal of Unconventional Computing, special issue 'New Worlds of Computation', 2010. arXiv:1101.1220.

5. Analogue computation with microwaves, M. S. Everitt, M. L. Jones, V. Kendon, N. B. Lovett and R. C. Wagner, International Journal of Unconventional Computing, 6, 3–14, 2010.

Contents

List of Figures

List of Tables

Abbreviations

AND	...	Logical AND gate
OR	...	Logical OR gate
NOT	...	Logical NOT gate
NAND	...	Logical NAND gate (comprised of NOT and AND gates)
C-NOT	...	Controlled NOT gate
C-PHASE	...	Controlled PHASE gate
NMR	...	Nuclear magnetic resonance
QFT	...	Quantum Fourier transform
3-SAT	...	Constraint satisfiability problem (specific case with three logical clauses)
QED	...	Quantum electrodynamics
2D	...	2 dimensional
3D	...	3 dimensional
DTRW	...	Discrete time random walk
CTRW	...	Continuous time random walk
DTQW	...	Discrete time quantum walk
CTQW	...	Continuous time quantum walk
DFT	...	Discrete Fourier transform
QMA	...	Quantum Merlin Arthur
BQP-complete	...	Bounded error quantum polynomial time
DES	...	Data encryption standard
MBQC	...	Measurement based quantum computation

List of Symbols

D	...	Spatial dimension
$\|0\rangle$	...	Logical 0 state
$\|1\rangle$	...	Logical 1 state
$\|\psi\rangle$	...	General quantum state (often used as initial state)
$\|+\rangle$	...	Logical + state $(1/\sqrt{2}[\|0\rangle + \|1\rangle])$
$\|-\rangle$	...	Logical - state $(1/\sqrt{2}[\|0\rangle - \|1\rangle])$
N	...	Total size of input (often used as size of dataset)
V	...	Vertices
E	...	Edges
d	...	Valency of vertex (degree)
A	...	Adjacency matrix of a graph G
L	...	Laplacian matrix of a graph G
M	...	Transition matrix
p	...	Probability
γ	...	Hopping rate
i	...	Complex number $= \sqrt{-1}$
$\hbar$	...	Planks constant$/2\pi$
$\dagger$	...	Hermitian conjugate
S	...	Shift operator
t	...	Timesteps
H_c	...	Complex Hadamard
I	...	Identity operator

$G^{(d)}$	...	Grover operator (of dimension d)
$DFT^{(d)}$	...	DFT operator (of dimension d)
$\mathbb{R}$	...	Real numbers
σ_x	...	Sigma X operation
P	...	Phase gate
$G_m^{(d)}$	...	Grover operation for marked state (of dimension d)
δ	...	Bias in coin operator
ϕ	...	Phase of coin operator
$T_{d,t}$	...	Tunnelling operator (of dimension d with tunnelling edges t)
h	...	Height of lattice
w	...	Width of lattice
l	...	Length of lattice
$\langle r \rangle$	...	Spread of quantum walk
s	...	Shortest path distance
p_c	...	Critical percolation probability

Chapter 1

Introduction

1.1 Quantum Information

Although we perceive the world around us to be entirely classical in nature, it is fundamentally quantum mechanical at its physical roots. Ever since the early part of the 20th century, scientists have been able to better explain many physical phenomena using quantum mechanics, as well as show remarkable differences between the classical world we see and the quantum. See any good quantum mechanics textbook for more information, for example Dirac [5]. In fact, many technologies we take for granted are based on quantum mechanical principles, for example the transistor, an integral part of any modern day computer. However, even though on a fundamental level these devices operate using quantum mechanics, we still treat them as being classical in the same way as we treat the idea of information that they process.

In his seminal work, Landauer [6] stated that information is 'physical' as physical systems must be used to process information. As such, any properties or intuition about information must stem from physical processes or laws. As mentioned above, the physical world is inherently quantum mechanical and therefore it seems reasonable that we should define our interpretation of information on this basis. Using this idea of a quantum mechanical definition of information, both Manin [7] and Feynman [8, 9] independently envisaged the idea of a quantum mechanical computer. Initially,

their idea was to use a quantum mechanical system (or computer) to simulate other quantum systems, a problem which is inherently difficult to do on classical computers due to the number of degrees of freedom. It was only later that Deutsch [10] thought of the same idea as a universal quantum computer, i.e. a quantum mechanical version of a classical computer. As well as laying down the framework for quantum computation in general, including the circuit model of quantum computation, Deutsch also showed the first quantum algorithm in his work. Since then, many other quantum algorithms have been found, most notably Shor's factoring algorithm [11, 12] and Grover's algorithm for searching an unsorted dataset [13]. We discuss these and the development of quantum algorithms in more detail in sec. 1.2.

In classical computing, the basic 'unit' of information is the bit which can take a logical value of 0 or 1. Many bits are then manipulated by logic gates, for example AND or NOT gates. In fact, universal classical computation can be obtained using just NAND gates. In the quantum setting, the basic unit is the quantum bit or 'qubit'. A qubit can take not just the values 0 and 1 but a linear combination of the two, i.e. the qubit can be in a superposition of the two logical states. The state of the qubit is represented as a vector but can also be pictorially shown on the Bloch sphere, fig. 1.1, the logical 0 and 1 being represented as $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ respectively. An arbitrary state of a qubit is defined as a linear superposition of the two logical basis states thus,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1.1}$$

where $\alpha = \cos(\theta/2)$, $\beta = e^{i\phi}\sin(\theta/2)$ and $|\alpha|^2 + |\beta|^2 = 1$. Individual qubits are manipulated by quantum gates which are unitary operators on the state. For an individual qubit, an operator is a 2×2 matrix which performs a rotation on the

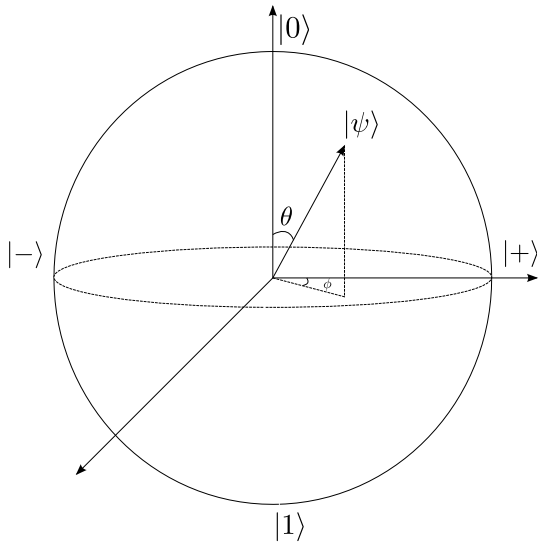

Figure 1.1: Bloch sphere. Pictorial representation of the state of a single qubit. An arbitrary qubit has the state $|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$.

state around the Bloch sphere. For example, the Hadamard operator,

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{1.2}$$

performs a rotation from the $|0\rangle$ state to the $|+\rangle$ state as

$$H|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right), \tag{1.3}$$

producing an equal superposition of the 0 and 1 logical states. Obviously, some form of manipulation of multiple qubits is also needed to allow universality. In the quantum setting, this is accomplished by a controlled unitary operation on the state. The most common of these is the Controlled-NOT (C-NOT),

$$C\text{-NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{1.4}$$

which acts on the state of two qubits,

$$|\psi\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right), \tag{1.5}$$

thus

$$C\text{-NOT}|\psi\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |11\rangle + |10\rangle\right). \tag{1.6}$$

In this case, the logical state of the second qubit is flipped conditioned on the logical state of the first qubit. To allow universal quantum computation, an arbitrary single qubit rotation combined with the C-NOT gate is required. This construction is often called the circuit model of quantum computation and is the quantum analogue of the classical circuit model.

In addition to the circuit model, another paradigm of quantum computation was introduced by Raussendorf and Briegel [14], known as one-way quantum computation or measurement-based quantum computation. In this model, a cluster state (or more generally, a graph state) is created as a resource for the computation. This is an entangled state of qubits which are all entangled using a Controlled-Phase gate (C-Phase),

$$C\text{-PHASE} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{1.7}$$

The computation progresses across the cluster state by making single qubit measurements, gradually destroying the initial resource state, hence the name 'one-way'. The result of each measurement is then 'fed-forward' to allow any errors to be corrected by selecting the correct basis for measurement at a later state. Both models of quantum computation are equivalent and more details can be found in any good textbook on quantum computation, for example Nielsen and Chuang [15].

As in the case of classical computing (biological systems, mechanical analogue systems and transistors), there are various proposals for the implementation of a

quantum computer. These include systems based on trapped ions [16, 17], quantum dots [18, 19], linear optics [20], nuclear magnetic resonance [21, 22] and superconducting qubits [23]. All these proposals have good and bad points and none have yet provided a scaleable architecture. Much experimental research is still going on in the attempt to build a quantum computer and it is yet unknown as to which architecture may provide a working, large-scale quantum computer. In fact, due to the problems encountered thus far experimentally, it may be that an entirely new idea is needed.

One of the biggest problems experimentally is to maintain the coherence of large systems for the required time of the computation. As such, both quantum error correcting schemes [24, 25] and fault tolerant or topological architectures [26–30] have both received much interest from the community as a whole. For more information on how many of these schemes operate see both the references and also Nielsen and Chuang [15]. In this thesis, we are primarily concerned with quantum algorithms and we assume that a fault tolerant, scaleable quantum computer exists on which they could be run.

1.2 Quantum Algorithms

In his seminal work, Deutsch [10] introduced the first quantum algorithm, one which can establish whether a given function, $f : \{0,1\} \rightarrow \{0,1\}$, is balanced or constant in just one quantum query as opposed to two in the classical case. This problem was then generalised to functions of the form $f : \{0,1\}^n \rightarrow \{0,1\}$ by Deutsch and Jozsa [31]. The complexity increases to two queries in the quantum case and to $2^{n-1}+1$ in the deterministic classical case. However (as is the case for many quantum / classical algorithm comparisons), if a probabilistic classical algorithm is used, the solution can be obtained with an error ϵ in $O(\log \frac{1}{\epsilon})$ queries. This was extended again by Bernstein and Vazirani [32] to a certain family of functions where given $a \in \{0,1\}^n$, $b \in \{0,1\}$ then $f_a(x) = a.x \otimes b$. Finally, Simon [33] found an exponential separation in the number of queries needed from the quantum to classical case for

another similar problem. This was the first example of a quantum algorithm with an exponential difference in run time.

Shortly after these initial developments, Shor [11, 12] gave another example of a quantum algorithm with a complexity exponentially smaller than the best known classical algorithm. This algorithm, for factoring, is probably the most well known and important of all quantum algorithms discovered thus far. It led to a surge of interest into quantum algorithms and quantum computing in general due to the fact it could render most current encryption techniques vunerable, as they rely on the problem of factoring a number into its primes as being computationally intractable. Shor introduces the quantum Fourier transform (QFT) and his algorithm can be used to find orders and periods of functions and also discrete logarithms. These are all special cases of the Abelian hidden subgroup problem.

The final early quantum algorithm is due to Grover [13] which is able to find a specific item from an unordered dataset in a time quadratically faster than the classical case. Although not as dramatic a speed up over classical, this is one of the few quantum algorithms which is both optimal and provably faster than any possible classical algorithm [34, 35], including those based on probabilistic techniques. This algorithm uses amplitude amplification and was generalised in [36]. This technique can be used in various other problems including 3-SAT and a generalised version of Simon's algorithm [37].

The continued development of a universal quantum computer is only viable if more quantum algorithms can be found that are of practical use and also fundamentally faster than the equivalent classical algorithms. Since the discovery of the first few quantum algorithms, more recent developments see new algorithms falling (generally) within one of five main categories:

1. Algorithms based on the Quantum Fourier Transform (QFT)

2. Algorithms based on amplitude amplification

3. Quantum simulation algorithms

4. Adiabatic algorithms

5. Quantum walk algorithms

We have already discussed most of the quantum algorithms which fall into the first two categories. The next category, quantum simulation algorithms, is the basis of Feynman's original motivation for quantum computing.

The first simulation algorithms were given by Lloyd [38] and Zalka [39]. The main idea is to simulate a given Hamiltonian by breaking it down into a sum of simpler ones. We will not discuss these in detail here, instead we refer the reader to the references or a recent review article, see Brown et al. [40].

Adiabatic algorithms are a fairly recent advance in quantum algorithms developed by Farhi et al. [41]. The idea is to encode the solution to a hard problem into the ground state of a Hamiltonian which is easy to simulate. If we then let the system evolve slowly, whilst keeping it in the ground state, then eventually the system will end in the state representing the solution to the problem. Obviously, it depends on how slowly is 'slowly enough' as to whether the algorithm is efficient. Adiabatic algorithms have been used to solve 3-SAT, in a fashion which is close to optimal [42, 43]

The final category of algorithms is that of the quantum walk. The quantum walk is the quantum mechanical analogue of the classical random walk or Markov chain and was introduced, with algorithmic application in mind by Aharonov et al. [44] and Ambainis et al. [45]. As the bulk of this thesis deals with quantum walks, in particular several quantum walk algorithms, we discuss these in detail in the subsequent chapters.

For several recent reviews which give much more detail on all of the algorithms we mention above, see Mosca [46], Smith and Mosca [47] or Childs and van Dam [48].

1.3 Numerical Methods

The work in this thesis is primarily numerical in nature. This includes the simulations of the quantum walk search algorithm in Chapters 5-8 and also the work on generating cluster states in Chapter 9. We discuss here details of methods and computational tools used.

The simulations we ran of the quantum walk searching algorithm were primarily written in MATLAB with some additional coding done in Python. All the code written in for use in this thesis was done by myself, there was no inbuilt package used. We attach a piece of sample code for a typical search on a 2D Cartesian lattice in Appendix A. Much of the other code we wrote was much more complex, for example simulating the quantum walk search algorithm on percolation lattices. Although the code we wrote was in MATLAB, many of the inbuilt functions of MATLAB were not used. It was primarily used for the ease of checking the numerical results and their subsequent plotting.

In the work on fractal structures, percolation lattices and certain other structures, the underlying graph we ran the quantum walk search algorithm on also had to be generated. This was primarily done using Python due to its ease of processing graphs in general. This allowed us to visualise the underlying graphs easily using Graphviz. This generation stage for the fractal structures was complex and became computationally intractable after a few generations were computed. For example, on a 2.2GHz MacBook, the generation of upto, and including, 7 generations took just over a week.

Performing the quantum walk search algorithm on the fractal and non-periodic structures required the repetition of the algorithm on each structure where the marked state could be in any position. Using percolation lattices required running the algorithm on many different, randomly generated, percolation lattices. All these simulations again took between 7-10 days to complete.

Finally, the work completed in Chapter 9 on the generation of entangled states was completed using a combination of Python and C. The basic algorithms we wrote

for establishing the structure produced was written in C, with the output processed in Python to give the resulting graph. These simulations only took a non-trivial amount of time in the cases where the generating grid was larger in size or more complex, i.e. 'structures within structures'.

1.4 Thesis Aim

The main aim of this thesis is to investigate applications of the discrete time quantum walk. We primarily focus on the factors which affect the efficiency of the discrete time quantum walk search algorithm. In previous work, only the dependence on the spatial dimension of the structure to be searched has really been investigated. In this thesis, we study not only this, but also how the layout (connectivity) of the structure, disorder and also symmetry affect the efficiency of the search algorithm.

In addition, we also investigate the power of the discrete time quantum walk, showing that it can be considered universal for quantum computation. The aim here was to provide a mapping from the circuit model based interpretation of quantum computation to a graphical model on which the discrete time walk propagates from one side to the other, thus performing a computation.

Finally, by using graph theoretic knowledge obtained from the work completed on quantum walks, we propose an experimental scheme to generate cluster states, using cavity quantum electrodynamics (QED), for quantum computation. In fact, it turns out this scheme can generate a variety of (useful) entangled graph states.

1.5 Thesis Overview

We start in Chapter 2 by giving a detailed overview of the quantum walk model as background theory for the remainder of the thesis. We discuss the classical random walk and its extensions to the quantum regime in both continuous and discrete time. We show an example of quantum walk dynamics on both the infinite line and then on higher dimensional structures. After discussing the applications of the quantum

walk, in particular for algorithmic use, we conclude the chapter with a review of experimental progress in realising a quantum walk thus far.

The power of the discrete time quantum walk is investigated in Chapter 3 where we show that the discrete time quantum walk can be considered universal for quantum computation. This is an extension of work done by Childs [49] which showed universality of the continuous time quantum walk and the 'lazy' discrete time quantum walk which can be used to approximate the continuous time quantum walk. We give a set of graph structures on which the discrete time quantum walk propagates deterministically. This chapter forms the basis of work published in Physical Review A [1].

We introduce the searching problem in Chapter 4 along with the best classical strategy for searching an unordered dataset. We then move to the quantum case and give a description of Grover's algorithm [13] before moving on to the model we use for the remainder of the thesis: the quantum walk search algorithm first introduced by Shenvi et al. [50]. This chapter also gives an overview of previous results known for the quantum walk search algorithm and its efficiency.

The following four chapters, 5-8, show our investigation of the factors affecting the efficiency, and hence the run time, of the quantum walk search algorithm. We start in Chapter 5 by discussing structures on which the quantum walk search algorithm is less efficient or fails completely. We start by investigating the search algorithm on the line and cycle, which can obtain no quantum speed up as simple arguments show, but reveal how the symmetry of the coin is important. We then move to lattices with non-periodic boundary conditions where reflection effects hamper the progress of the quantum walk and thus the efficiency of the search algorithm. This is shown on the Bethe lattice, fractal structures and also the 2D and 3D basic Cartesian lattices.

In Chapters 6 and 7, we introduce a simple form of tunnelling which allow us to interpolate between structures of varying spatial dimension and connectivity. The work in these two chapters forms the basis of two pieces of work [2, 3], the former

of which has been published. Using this simple model of tunnelling, we investigate the effect, on regular lattices, of spatial dimension and connectivity respectively.

Finally, in Chapter 7, we show how the effect of disorder can affect the efficiency of the search algorithm by using percolation lattices as a substrate for the quantum walk. These allow us to gradually vary the level of disorder (and hence reduce the symmetry) in the database arrangement, allowing us to investigate how much disorder, if any, the quantum walk search algorithm can tolerate.

Our last main chapter contains work separate to that of quantum walks. Here, we discuss an abstract scheme to experimentally create cluster states. We envision the scheme to operate using streams of atoms interacting in cavities (or possible regions of electric fields) to mediate an interaction, thus performing a Controlled Phase gate. Using the graph theory knowledge obtained during my PhD, we show how we can extend the scheme to generate not only cluster states but also other useful entangled states. The work contained in this chapter forms the basis of work published in [4].

Finally, we conclude the thesis with an overview of our results in Chapter 10. We also discuss our plans for the future and how the work contained in this thesis could be extended.

Chapter 2

Quantum Walks

2.1 Introduction

Classical random walks have long been used in many areas of science, especially mathematics and physics. In more recent years, they have found new applications in classical computer science. Many new algorithms based on classical random walks have been introduced for key problems which outperform the previously best known algorithms. Many examples exist but perhaps the most notable being an algorithm for the constraint satisfiability problem [51, 52], approximating the permanent of a matrix [53] and estimating the volume of a convex body [54]. Random walks have also been extensively used for random sampling of large state spaces in Monte Carlo methods - see [55–58] and references therein for a good review of these methods and other random walk-based algorithms.

Due to the success of applying classical random walks to algorithm design, a natural place to look for faster quantum algorithms was a quantum version of a classical random walk. The term 'quantum random walk' was first used by Aharonov et al. [59] where a precursor to the two well known models of quantum walks was introduced. In this work, intermediate measurements are made at each time step of the walk. Several others used the concept of the quantum walk in varying forms including Feynman and Hibbs [60] in the form of path integrals, Meyer [61, 62] in the

context of quantum cellular automator and also Watrous [63] in the context of space bounded computation. The first formal treatment and study of quantum walks for algorithmic purposes was pioneered by Ambainis et al. [45] and Aharonov et al. [44] who introduced a discrete time quantum walk. They proved that a quantum walk on the line or cycle spread or mixed, respectively, in a time quadratically faster than a classical random walk. Just as a classical random walk can also be defined in continuous time, Farhi and Gutmann [64] introduced the continuous time quantum walk which also provides the same algorithmic speed up. Although we describe both types of quantum walk in later sections, we concentrate on the discrete time quantum walk for the remainder of the thesis, since it is more convenient for numerical calculations.

After the introduction of the two types of quantum walk, they have been studied extensively. Many new quantum algorithms have been developed, with varying speed ups over the best known classical algorithms. One of the most notable quantum walk based algorithms is the 'glued tress' algorithm introduced by Childs et al. [65]. In this algorithm, a quantum walker is above to traverse the structure, two binary trees linked together randomly at the leaves, in a time exponentially faster than the best known classical algorithm. This was extended from previously work also by Childs et al. [66] where the same exponential speed up was found across a similar structure with pairwise links of the leaves of the binary trees. Shortly after, Shenvi et al. [50] introduced a quantum walk search algorithm which can match the quadratic speed up of Grover's algorithm. We discuss this algorithm at length in chapter 3. Many other quantum walk based algorithms have been discovered which give a polynomial speed up over the best known classical algorithms, many of which are of practical interest. For example, Ambainis [67] and Magniez et al. [68] gave algorithms for element distinctness and the triangle finding problem respectively. In addition, the continuous time quantum walk has recently been shown by Childs [49] to be universal for quantum computation, and as such, a computational primitive. Lovett et al. [1] later showed that the discrete time quantum walk is also universal

for quantum computation, which we discuss in more depth in chapter 4. This means that any quantum algorithm can be recast as a quantum walk algorithm in either continuous or discrete time. As such, the quantum walk has now become a standard tool in algorithm design, for a good review of algorithms based on quantum walks see Ambainis [69].

In order to discuss quantum walks in detail we will need to define some notation from standard graph theory. We first review this before moving on to define the continuous time quantum walk. The discrete time quantum walk is defined next, firstly on the line and then we review how it can operate on higher dimensional structures. We then finish this chapter with a discussion of the applications of quantum walks and current experimental progress thus far.

2.2 Graph theory

In order to discuss quantum walks, on both the line and in higher dimensions, we must use some standard graph theoretical terms. We define all the terms we need for the thesis here as we will use many of these terms in later chapters.

A general graph, G, is defined as an ordered pair formed from a set of vertices, V, and a set E of edges. A member of the set of edges is a subset of vertices, i.e. $(x, y) \in V$, where x and y are elements of the set V. To be precise, if the pair of vertices in an edge is unordered, the graph we describe is then known as undirected. All the graph structures we discuss will be of this type. An example of an undirected graph is shown in fig. 2.1. We define the order of the graph to be the number of vertices, $|V|$, and the size to be the number of edges, $|E|$. In fig. 2.1, we can see that some vertices have more (or less) connections to other vertices than others. The degree d, or valency, of a vertex is defined as the number of edges incident upon it. If all vertices in a graph have the same degree we define it as regular, or k-regular where k is the degree of every vertex. A 2D Cartesian lattice (either infinite or with periodic boundary conditions) is an example of a k-regular graph where $k = 4$, i.e. all vertices have degree $d = 4$. Edges which start and end at the same vertex are

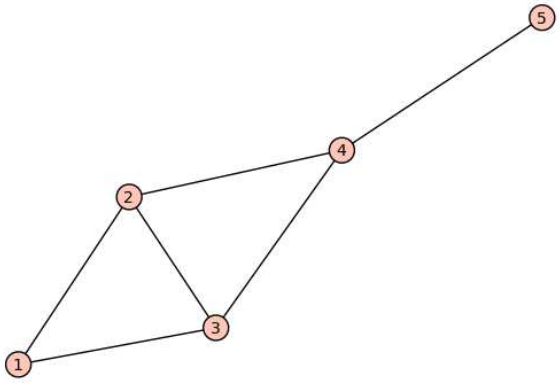

Figure 2.1: An example of a general, undirected graph. It consists of 5 vertices, of varying degree, and 6 edges.

known as self loops. In this thesis, we only consider graphs which have no self loops.

We will be studying graphs of varying topology and structures in later chapters. We must define the structure of the underlying graph we are studying in a succinct way. Most will be a single connected graph, that is there is a path, along edges, from any vertex to any other, i.e. there is one connected component. One way to define the connectivity of a structure is by the adjacency matrix. For a graph, G, of N vertices, this is a $N \times N$ matrix where the entries of the matrix, $A_{a,b}$, is nonzero if an edge exists between vertices a and b. In an undirected graph with no self loops, we define it as

$$A_{a,b} = \begin{cases} 1 & a, b \in G \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

A similar matrix of connectivity is the Laplacian of the graph. This can be defined as

$$L = A - D_d, \tag{2.2}$$

where D_d is the diagonal matrix whose entries $D_{a,a} = d_a$, where d_a is the degree of each vertex. As an example, we show both the adjacency and laplacian matrices of

the graph in fig. 2.1,

$$A_G = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \tag{2.3}$$

$$D_G = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2.4}$$

$$L_G = \begin{pmatrix} -2 & 1 & 1 & 0 & 0 \\ 1 & -3 & 1 & 1 & 0 \\ 1 & 1 & -3 & 1 & 0 \\ 0 & 1 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}. \tag{2.5}$$

2.3 The continuous time quantum walk

The continuous time quantum walk was introduced, with algorithmic applications in mind, by Farhi and Gutmann [64]. Although we do not focus on them here we describe both the continuous time classical random walk and continuous time quantum walk next. In contrast to the discrete time quantum walk which we describe later, the continuous time quantum walk takes place solely on the vertices of the underlying graph and has no 'coin' operation.

2.3.1 Continuous time random walk

A continuous time random walk is a classical Markov process on a general graph, G. We define a matrix, M, which is used to update the probability distribution across

the vertices of the graph at each timestep. The entries of the matrix, $M_{a,b}$, give the probability of moving from vertex a to vertex b. As such, non-zero entries only occur if a and b are connected. In a simple, unbiased walk, these non-zero entries are equal to $1/d_a$, where d_a is the degree of vertex a. If we define p_a^t as the probability of being at vertex a at time t then

$$p_a^{t+1} = \sum_b M_{a,b} p_b^t. \tag{2.6}$$

This gives the probability of being at any vertex after a 'step' of the walk. If $\vec{p}_t$ is the probability distribution over the entire graph, $\vec{p}_t = \left(p_1^t, p_2^t, \ldots, p_{|V|}^t\right)$, then we can write the updated probability distribution as

$$\vec{p}_{t+1} = M\vec{p}_t. \tag{2.7}$$

The transition matrix, M, for this simple, unbiased walk on a cycle with N vertices would be of the following form,

$$M = \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 & 0 & \ldots & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & \ldots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & & \vdots \\ \vdots & & & & \ddots & & & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & \ldots & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.8}$$

Obviously, applying this transition matrix iteratively only gives us updated probability distributions at fixed time so is in effect discrete. In order to turn this into a random walk in continuous time, we instead have a fixed hopping rate, γ, which gives the transition probability between two (connected) vertices per unit time. The

new continuous time transition matrix, M^c, then becomes of the form,

$$M^c_{a,b} = \begin{cases} -\gamma & a \neq b \text{ and } a, b \text{ are connected} \\ 0 & a \neq b \text{ and } a, b \text{ are not connected} \\ d_a\gamma & \text{a = b} \end{cases} \tag{2.9}$$

In continuous time setting, we can write the probability of transition from one vertex to another as we did in eq. 2.6 by the differential equation,

$$\frac{dp_a(t)}{dt} = -\sum_b M^c_{a,b} p_b(t), \tag{2.10}$$

where $p_a(t)$ is the probability of being at vertex a at time t. For the entire probability distribution over all vertices, this becomes

$$\vec{p}(t) = e^{-M_c t}\vec{p}(0) \tag{2.11}$$

as in eq. 2.7. This equation will transform an initial probability distribution according to the matrix M_c in continuous time. The structure of both the initial translation matrix, M, and the continuous time translation matrix, M_c, is very similar. We note this provides a direct link between the discrete time classical random walk and the continuous time classical random walk. In fact, much work in classical Markov chain theory is to show the equivalence of the two models, and also properties of interest, for example mixing and hitting times [70].

An equivalent way of defining the continuous classical random walk is by the structure of the underlying graph, using the adjacency or Laplacian matrices. We note here that the transition matrix M is equivalent to the Laplacian of the graph L up to a constant factor, $-\gamma$. The differential equation to describe the evolution of the walk then becomes

$$\frac{dp_a(t)}{dt} = \gamma \sum_b L_{a,b} p_b(t). \tag{2.12}$$

We note here the Laplacian is a probability conserving process, that is $\sum_a L_{a,b} = 0$. We note that any process that obeys this conservation of probability can be used here. This definition of the random walk is equivalent to eq. (2.10) and the continuous time walk is often defined in this way. We will define the continuous time quantum walk in the same fashion.

2.3.2 Continuous time quantum walk

A continuous time quantum walk is the quantum analogue of the classical continuous time random walk previously defined. It takes places directly on the vertices of a graph in a N dimensional Hilbert space spanned by the basis states $\mid a\rangle$, where $a \in G$. We can then write the N complex amplitudes, $q_a(t)$, at each vertex in terms of the general state of the walker, $\mid \psi(t)\rangle$,

$$q_a(t) = \langle a|\psi(t)\rangle. \tag{2.13}$$

The dynamics of the system can then be described by the Schrödinger equation,

$$i\frac{dq_a(t)}{dt} = \sum_b H_{a,b}q_b(t), \tag{2.14}$$

where we set $\hbar = 1$ for convenience. If we compare eqs. (2.10) and (2.14), we notice they are similar. If we set H in eq. (2.14) to be equal to $-\gamma L$, we end up with just a factor of i difference. In this definition, we set the Hamiltonian of the Schrödinger equation to be the Laplacian of the graph. This does not have to be the case, we just need to ensure that the operator is unitary, $H = H^\dagger$. For example, the adjacency matrix of the graph could be used. If a regular graph, one in which all vertices have the same degree, then the adjacency matrix will give the same dynamics as the Laplacian [64]. This is due to the fact the diagonals just introduce a global phase which is of no consequence.

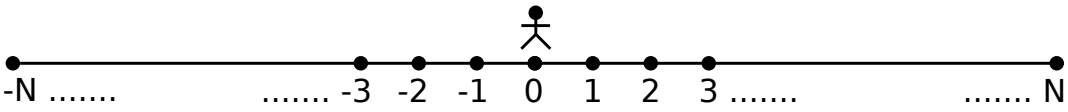

Figure 2.2: Representation of a discrete time random walk on a line. The walker starts at the origin and moves left or right depending on the outcome of an unbiased coin toss.

2.4 The discrete time quantum walk

There has been great interest in the field of quantum walks and their use in algorithm design. We will discuss the plethora of results stemming from quantum walk research in a later section. Firstly however, we turn to the discrete time quantum walk, which we focus upon for the remainder of the thesis. We firstly define the classical discrete time random walk before moving to the quantum case, defining the discrete time quantum walk on both the infinite line and in higher dimensions.

2.4.1 Discrete time random walk

The discrete time classical random walk is probably the most well known random walk that exists. We often think of this as a 'drunkards' walk. Consider a number line from $-N$ to N with a walker placed at the origin as shown in fig. 2.2. The walker tosses an unbiased coin and dependent on the outcome moves either left or right by one position. After repeating this coin toss and movement for t timesteps we will find the walker, on average, back at the origin. In fact, if we perform the random walk a large enough number of times we get a binomial distribution of the walkers final position centered about the origin, fig. 2.5. We note here that the spread of this distribution, quantified by the standard deviation, is equal to the number of timesteps t.

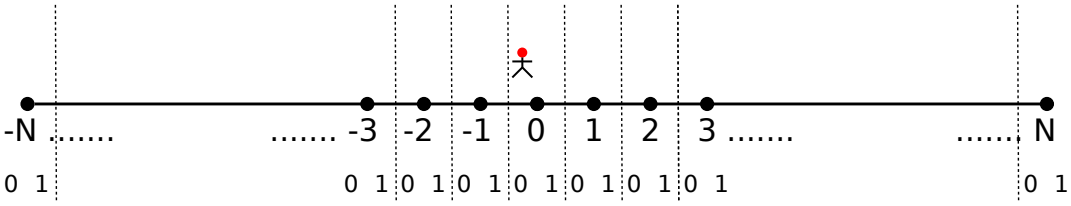

Figure 2.3: Representation of a discrete time quantum walk on a line. The walker starts at the origin, with an internal coin state of 0. The walker is acted upon by a unitary operator, the Hadamard in this case, and then a conditional shift operator at each timestep.

2.4.2 Discrete time quantum walk on the line

A discrete time quantum walk on the line is defined in direct analogy with a classical random walk. In the quantum case, the walker is replaced by a quantum particle carrying a two state quantum system for the coin. In order to maintain quantum dynamics, which must be reversible, the 'coin toss' is effected by a unitary operator. We denote the basis states for the quantum walk as an ordered pair of labels in a 'ket' $|x, c\rangle$, where x is the position and $c \in \{0, 1\}$ is the state of the coin. We place the walker at the origin with an internal coin state of 0 as shown in fig. 2.3. At each timestep we act on the quantum walker with a coin operator followed by a conditional shift operator. The simplest coin operator is the Hadamard H, defined by its action on the basis states as

$$\begin{aligned} H|x,0\rangle &= \frac{1}{\sqrt{2}}(|x,0\rangle + |x,1\rangle) \\ H|x,1\rangle &= \frac{1}{\sqrt{2}}(|x,0\rangle - |x,1\rangle), \end{aligned} \tag{2.15}$$

and the shift operation S acts on the basis states thus

$$\begin{aligned} S|x,0\rangle &= |x-1,0\rangle \\ S|x,1\rangle &= |x+1,1\rangle. \end{aligned} \tag{2.16}$$

The coin operator splits the walker into a superposition of coin states and the conditional shift operator then moves the walker to the correct position based on the

coin state. The first three steps of a discrete time quantum walk starting from the origin, in coin state 0, are shown pictorially in fig. 2.4 and mathematically as

$$\begin{aligned}
(SH)^3|0,0\rangle &= (SH)^2 S\frac{1}{\sqrt{2}}(|0,0\rangle \ + |0,1\rangle) \\
&= (SH)^2\frac{1}{\sqrt{2}}(|-1,0\rangle \ + |1,1\rangle) \\
&= (SH)S\frac{1}{2}(|-1,0\rangle \ + |-1,1\rangle \ + |1,0\rangle \ - |1,1\rangle) \\
&= SH\frac{1}{2}(|-2,0\rangle \ + |0,1\rangle \ + |0,0\rangle \ - |2,1\rangle) \\
&= S\frac{1}{\sqrt{8}}(|-2,0\rangle \ + |-2,1\rangle \ + |0,0\rangle \ - |0,1\rangle \ + |0,0\rangle \ + |0,1\rangle \\
&\quad - |2,0\rangle \ + |2,1\rangle) \\
&= \frac{1}{\sqrt{8}}(|-3,0\rangle \ + |-1,1\rangle \ + 2|-1,0\rangle \ - |1,0\rangle \ + |3,1\rangle). \qquad (2.17)
\end{aligned}$$

As the walk progresses, quantum interference occurs whenever there is more than one possible path of t steps to the position. This can be both constructive and destructive, as shown in eq. (2.17), which causes some probabilities to be amplified or decreased at each timestep. This leads to the different behaviour compared to its classical counterpart: spreading at a rate proportional to t, quadratically faster than the classical random walk. In addition, the centre part of the distribution, in the interval $[-t/\sqrt{2}, t/\sqrt{2}]$, is fairly uniform. This is the opposite of the classical random walk which has an exponential drop in probability after just a few standard deviations from the origin. These properties of the quantum walk on the line were obtained by both Ambainis et al. [45] and Nayak and Vishwanath [71].

As the walker can now be in a superposition of positions on the line, we obtain a probability distribution of the quantum walker after one run of the entire walk. Obviously, this is due to the fact the coin operator is now deterministic. However, if we were to measure the coin after the required number of timesteps, we would get a random output as in the classical case. We show both the classical and quantum probability distributions after 100 steps in fig. 2.5. If the walk is imperfect and some decoherence is allowed, we can see the gradual change from the quantum case back

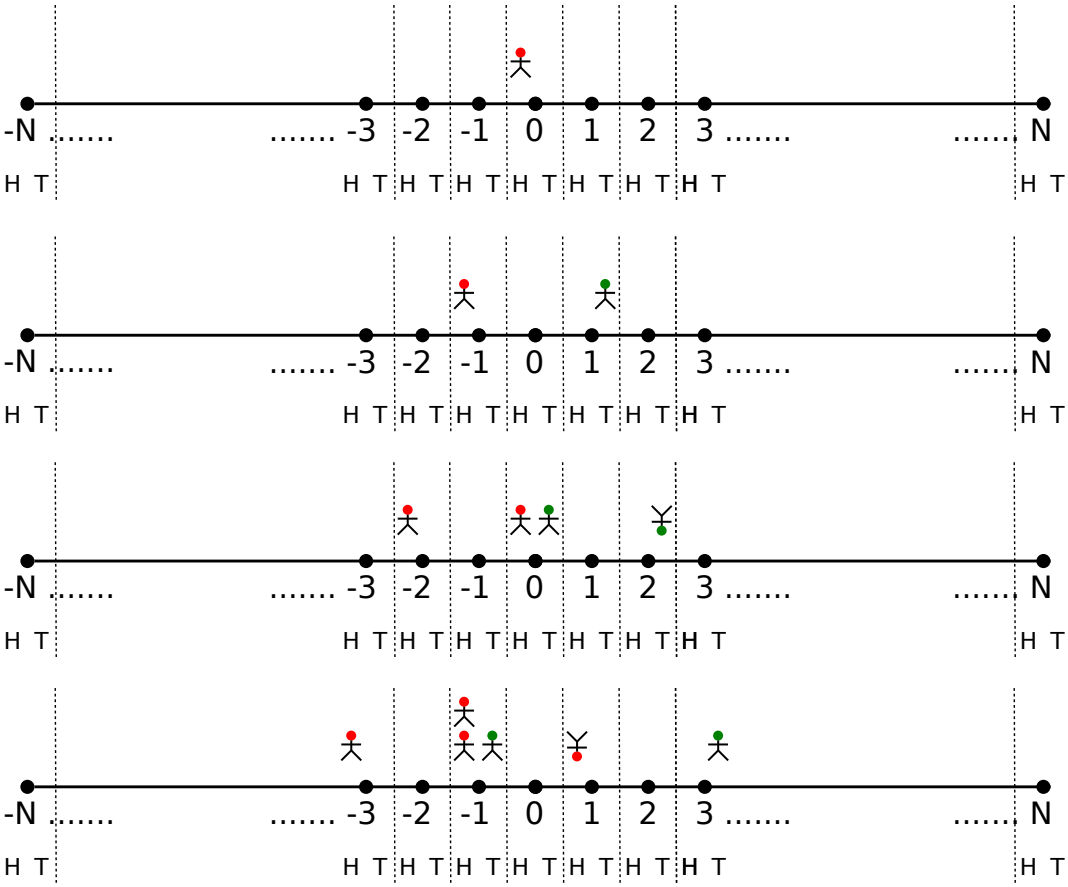

Figure 2.4: Representation of the first three steps of a discrete time quantum walk as in eq. (2.17). The walker is started in an initial state of $|0,0\rangle$. In the figure, red walkers represent the coin state 0, green walkers represent the coin state 1 and walkers with a negative coefficient are shown upside down.

to classical. Kendon and Tregenna [72, 73] investigated this in detail showing that as the decoherence in the system grows the spread of the walk gradually changes from the quantum walk shown above back to the classical binomial distribution. In the interim, we see a gradual change with an almost 'top-hat' distribution being found which is useful for random sampling. We note, in fig. 2.5, that the distribution of the walk is dependent on the initial state of the walker. In eq. (2.17) above, when the walker is started in one specific position and coin state, $|0,0\rangle$, that the distribution is 'skewed' to the left. This is due to the Hadamard coin treating the different coin states in different ways: the 'right-moving' coin state has a phase of -1 attached to it. If the walker was instead started in an initial state $|0,1\rangle$, we would see the

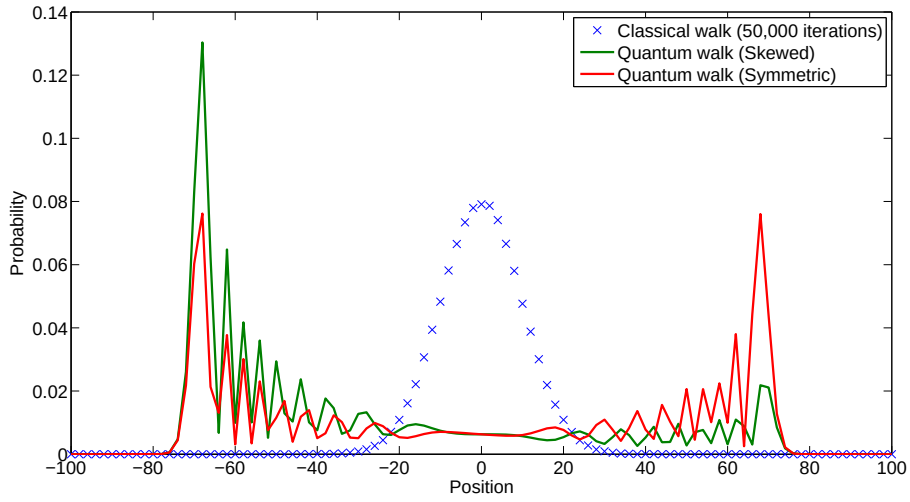

Figure 2.5: Classical (crosses) and quantum (solid lines) probability distributions for walks on a line after 100 timesteps. Only even positions are shown since odd positions are zero. The classical walk is averaged over 50,000 iterations of the random walk. A skewed quantum walk is shown with an initial state of $|0,0\rangle$ along with a symmetric quantum walk with an initial state of either $\sqrt{0.15}|0,0\rangle + \sqrt{0.85}|0,1\rangle$ or $1/\sqrt{2}(|0,0\rangle + i|0,1\rangle)$.

same skew in the distribution but this time to the right. This can be explained by the fact that whichever coin state the walker is started in incurs less destructive interference and thus skews in that directions. A symmetric distribution, also shown in fig. 2.5, can be obtained in various ways. Firstly, by using a general initial state of the form $|\psi\rangle = \cos\theta|0,0\rangle + \sin\theta|0,1\rangle$, Konno [74, 75] proved, analytically, values of θ which give the symmetric distribution. This leads to the amount of deconstructive and constructive interference becoming balanced, allowing the walk to spread in a symmetric fashion in both directions. The initial state required for this is $\sqrt{0.15}|0,0\rangle + \sqrt{0.85}|0,1\rangle$. Another way of accomplishing the symmetric distribution is to start the walk in a way that the two coin states will not interfere with each other. This can be done using the initial state $1/\sqrt{2}(|0,0\rangle + i|0,1\rangle)$. As the Hadamard coin we have introduced does not give any additional complex terms to the 0 coin state, each coin state will remain either real or complex thus not

interfering. Finally, the complex Hadamard coin,

$$H_c = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}, \tag{2.18}$$

can be used. By changing the coin to this more 'balanced' or 'symmetric' coin, then it will treat each coin state independently, thus producing a symmetric distribution.

The quantum walk on the line was analysed and extended further by placing it on a bounded line (on one or both sides). In [45, 76], this is investigated by placing an absorbing boundary on the line. This can be viewed as a partial measurement of the walker if it hits the boundary. The walk is measured after each step and ends in either the state representing the walker at the absorbing boundary or the state where the walker is at all other positions. A good example from [77], where the boundary is placed at $x = 0$, is as follows. If the walker is in the state

$$|\psi\rangle = \frac{1}{\sqrt{14}}(2|0,0\rangle - |1,0\rangle + 3|1,1\rangle), \tag{2.19}$$

then after the measurement step the walker will either end up in the state $|0,0\rangle$ with probability $2/7$ or the state $1/\sqrt{10}(-|\rangle + 3|1,1\rangle)$ with probability $5/7$. In the classical case, the probability to become absorbed, in the long time limit, is certain, $p = 1$. This is due to the fact that there is always an infinitely small possibility that the walker could be at any vertex. However, in the quantum case, if the walker is started in the correct initial state, the probability of being absorbed is $2/\pi$, in the long time limit of t. This probability changes depending on the initial state the walker is started in but is always finite. Although we do not discuss higher dimensional structures until the next section, this behaviour also holds on these structures [78].

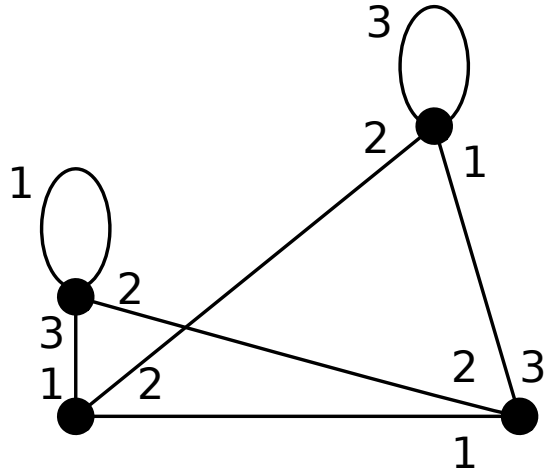

Figure 2.6: Example of a higher dimensional graph with varying degree at each vertex. Self loops are added at these vertices to make the graph d-regular. We also show the labelling scheme for the edges. We note that a self loop is taken to be the equivalent of one edge at a vertex so in the figure, all vertices are of degree three.

2.4.3 Discrete time quantum walk in higher dimensions

We can see from the previous section that the quantum walk exhibits interesting and very different behaviour to the classical walk even on the line. However, many interesting problems in computer science are defined in higher dimensions. We discuss here d-regular graphs so each vertex has the same number of edges. If the graph is not d-regular, one way of making it so is to add self loops at any vertex which has a lower degree than the maximum degree of the graph, fig. 2.6. In this case, the self loop is equivalent to just adding one edge per vertex. Other ways of accomplishing quantum walk dynamics on non-regular graphs are discussed in Chapters 5 and 8. In order to define the walk on these higher dimensional graphs, we require a new coin operator, of dimension d, in order to span the entire coin state space of the walker [79–82]. This can be any unitary operator of the required dimension. Clearly many different possibilities exist but we only mention the two most common (and natural) operators here. Firstly, the Grover coin,

$$G^{(d)} = \begin{pmatrix} \frac{2}{d} & \cdots & \frac{2}{d} \\ \vdots & \ddots & \vdots \\ \frac{2}{d} & \cdots & \frac{2}{d} \end{pmatrix} - I_d, \tag{2.20}$$

where d is the degree of the vertex and I_d is the identity operator of the same dimension. The Grover coin is symmetric but only balanced, i.e. it treats all directions in the same way - up to a phase factor, for the cases where $d = 2$ and $d = 4$. In the case of $d = 3$ and all higher dimensions, the coin treats one edge differently to the remaining $d - 1$.

The second coin operator we describe here is the DFT (discrete Fourier transform) coin,

$$DFT^{(d)} = \frac{1}{\sqrt{d}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w & w^2 & \dots & w^{d-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{d-1} & w^{2(d-1)} & \dots & w^{(d-1)(d-1)} \end{pmatrix}, \tag{2.21}$$

where $w = exp(2\pi i/d)$ which is a d-th root of unity. The DFT coin is clearly symmetric and balanced, it transforms each direction in a superposition of all directions so all are equally likely with a probability of $1/d$.

In addition to the coin operator, the conditional shift operator must also be modified. In the case of the line, it is easy to define as there are only two possible directions the walker can move in. In higher dimensions, the walker can move in any one of d directions. Kendon [81] treats this problem rigorously but the most important thing is to maintain a consistent labelling approach for each of the edges. We label the d edges from 1.....d as shown in fig. 2.6 thus creating a mapping between a vertex / edge pair and another vertex / edge pair,

$$S|x, c\rangle = \begin{cases} |v, c'\rangle & \text{if } (x, v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.22}$$

where $|x, c\rangle$ is the current position and coin state of the walker and $|v, c'\rangle$ is the related vertex / coin state (edge) pair.

The dynamics of the walk on higher dimensional structures has been studied briefly by Mackay et al. [80] and then in more detail by Tregenna et al. [83]. They

numerically studied the spreading of the quantum walk with varying coin operators and initial states. In [83], the walk was initially studied on a two dimensional lattice using two Hadamard coins, one for each pair of edges or directions, i.e. horizontal and vertical. In this case, the dynamics of the walk moving in perpendicular directions do not mix with each other and thus we get a distribution similar to that of the line but in both directions. In addition, the choice of initial coin state, provided it gives a symmetric distribution, makes no difference to the spread of the walk, which gives a standard deviation $\sqrt{2}$ greater than that of the line [80]. In order to give more interesting dynamics, the two coin operators we mentioned previously were then studied. In the case of the two dimensional lattice, $d = 4$, the Grover coin reduces to

$$G^{(4)} = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}, \tag{2.23}$$

and the DFT coin reduces to

$$DFT^{(4)} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}. \tag{2.24}$$

Similar dynamics are found for both these coins on a two dimensional lattice. Tregenna et al. found that the initial state of the walker on the lattice had a large impact on the spreading of the walker. Depending on the initial state, the walker can spread anywhere from a minimum possible spread to a maximum possible spreading (as defined in [83] by the second moment). However, there are subtle differences between the dynamics of the two coin operators. In the case of the Grover coin, most of the

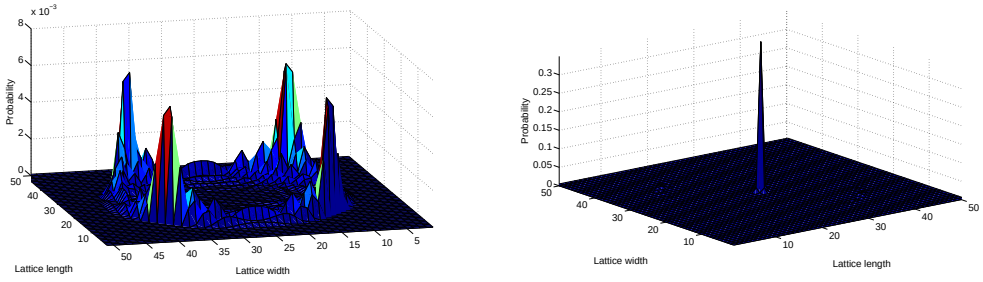

Figure 2.7: Dynamics of the quantum walk on a two dimensional Cartesian lattice using the Grover coin, eq. (2.23). LHS: Maximum spreading obtained using the initial state in eq. (2.26). RHS: Localisation obtained using the initial state in eq. (2.25). Note the different scales.

initial states, including the symmetric initial state,

$$|\psi\rangle_{sym} = \frac{1}{2}\left(|0,L\rangle + i|0,R\rangle + |0,D\rangle + i|0,U\rangle\right), \tag{2.25}$$

where L, R, D, U are the four directions the walker is able to move on the lattice, give a minimal spreading with the walker localising around the origin with high probability as seen in fig. 2.7. However, one specific state,

$$|\psi\rangle_{grover:max} = \frac{1}{2}\left(|0,L\rangle - |0,R\rangle + |0,D\rangle - |0,U\rangle\right), \tag{2.26}$$

gives a maximal spreading, again shown in fig. 2.7.

The DFT coin dynamics are the same in the sense that one specific state (different to the Grover coin),

$$|\psi\rangle_{dft:max} = \frac{1}{2}\left(|0,L\rangle + \frac{1-i}{\sqrt{2}}|0,R\rangle + |0,D\rangle - \frac{1-i}{\sqrt{2}}|0,U\rangle\right), \tag{2.27}$$

gives maximal spreading, but for all other (symmetric) initial states the distribution is not localised as much as in the case of the Grover coin, see fig. 2.8. In fact, it is spread across several of the vertices close to the origin with the choice of initial state determining how much of the probability is within the central peak.

The hitting time of a quantum walk can be defined as the time it takes for the walker to reach a specific vertex (with sufficiently high probability) from another. In

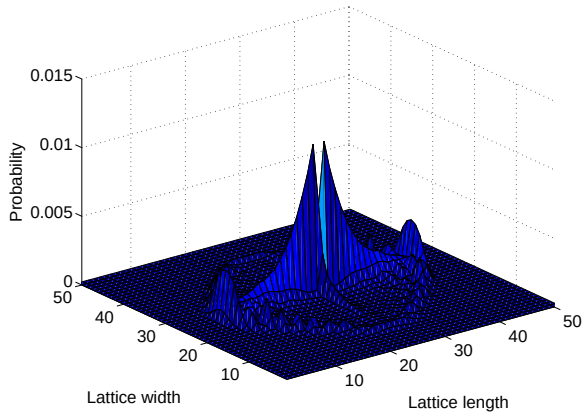

Figure 2.8: Dynamics of the quantum walk on a two dimensional Cartesian lattice using the DFT coin, eq. (2.24). Localisation dynamics obtained using the initial state in eq. (2.25).

higher dimensions, this notion has been studied in some detail. The hitting time on a structure is important algorithmically as it gives us an indication as to how quickly a solution to a problem can be reached. In [82], Kempe proves that a quantum walk can hit the opposite corner of a hypercube exponentially faster than in the classical case. The complete graph [84] and the two dimensional Cartesian lattice [85] also give improved results versus classical.

This unusual behaviour, for both the line and higher dimensions, compared to the classical random walk is one of the main reasons quantum walks have attracted so much attention in recent years. We discuss other unusual behaviour in sec. 2.6 and show how this behaviour can be used to our advantage in the form of quantum algorithms.

2.5 Connecting the continuous and discrete time quantum walks

Although the discrete and continuous time quantum walks are the direct quantum analogues of their individual classical counterparts, it is not obvious how each version of the quantum walk maps to the other. In the classical case, we can formulate a discrete time random walk as a Markov chain with N states. An initial probability

distribution, $p \in \mathbb{R}^N$, is transformed by a $N \times N$ transition matrix, M, to an updated distribution, $p' = Mp$, at each timestep. In order to transform this to the continuous time case, we firstly only allow the transformation M to occur with some small probability $\epsilon > 0$. By replacing the matrix M with $\epsilon M + (1-\epsilon)I$ we turn the walk into a *lazy random walk* as in [86]. Thus

$$p' - p = \epsilon(M - I)p. \tag{2.28}$$

Now, in the continuous time limit where $\epsilon \to 0$, each discrete time step of the walk is doing very little. Setting each time step to equal to an interval of ϵ then gives

$$\frac{\mathrm{d}}{\mathrm{d}t}p(t) = (M - I)p(t). \tag{2.29}$$

This Markov process is now a continuous time one in which the dynamics are generated by the matrix $M - I$.

However, in the quantum case the walks cannot simply be linked in the limiting case due to the fact the discrete time quantum walk takes place on a larger state space as it must include the additional degree of freedom provided by the coin. The first work to attempt to link the two types of quantum walk was given by Strauch [87] who gave a direct correspondence for the infinite line and the cubic lattice. Strauch proved that in the continuous limit the discrete time quantum walk actually becomes to two copies of the continuous time quantum walk which propagate in opposite directions.

Later, using an approach based on the quantisation of classical Markov chains introduced by Szegedy [88], Childs [86] proved a direct correspondence between the two types on quantum walk for general graphs. This work builds on previous studies in which it has been noted that, using the framework of Szegedy, the continuous and discrete time quantum walks can be connected in certain circumstances. Ambainis et al. [89] showed that the two can be connected when the continuous time quantum walk has a positive, symmetric matrix as its generator. This was generalised to an

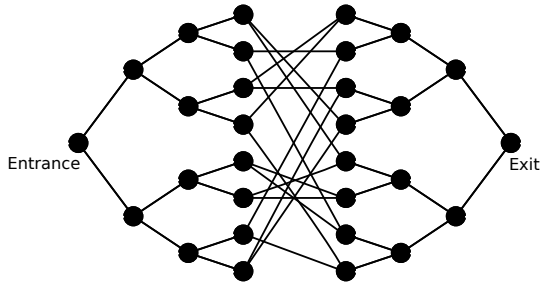

Figure 2.9: The 'glued-trees' graph of Childs et al. [65]. It consists of two binary trees linked randomly at the leaves.

arbitrary Hermitian matrix as long as the equivalent graph is bipartite by Reichardt and Spalek [90]. Childs [86] then further extended this to any general graph by constructing a 'lazy quantum walk' which is a discrete time process which is able to approximate continuous time dynamics in the small time limit.

2.6 Applications of the quantum walk

The quadratic enhancement of the quantum walk on a line (and also higher dimensions) caused much excitement in the quantum information community, especially from a computer scientist's perspective. The quantum walk has found many applications including its use in algorithm design and also as a transport model used in many physical systems.

The possibility of using the quantum walk to develop new quantum algorithms was quickly realised in the 'glued-trees' algorithm of Childs et al. [65]. Although a fairly artificial problem, Childs et al. show an exponential separation in the run time between the quantum and classical case. The walker is started at the root of a binary tree and the task is to find the exit: the root of another binary tree which is linked to the first randomly at the leaves, see fig. 2.9. In the classical case, the walker finds it easy to reach the centre of the graph but once there cannot identify which way is 'forwards' as there is a higher probability of the walker moving 'backwards'. As such, it takes an exponential time for the walker to reach the root of the second binary tree. In the quantum case, the problem can be mapped to a quantum walk

on the line and is only linear in the time it takes to reach the base of the second tree. This is an extension of work by Childs et al. [66] in which the binary trees are linked pair-wise at the leaves, which also gives an exponential speedup over the classical case. In the case of the 'glued-trees' algorithm, the additional set of edges is held by an oracle and thus the exponential speed up is proved with respect to this. These algorithms provide a 'proof of principle' that quantum walk algorithms can provide an exponential speedup compared to classical algorithms.

Following this, several other quantum walk algorithms were proposed. These provide either a quadratic or polynomial speed up over the classical case. One of the first of these was the quantum walk search algorithm introduced by Shenvi et al. [50] which is able to match the quadratic speed up of Grover's algorithm for searching an unsorted dataset. As the efficiency of this algorithm is the basis of several of the chapters of this thesis we discuss it in detail in Chapter 4.

Around the same time, Ambainis [67] gave a quantum walk algorithm for element distinctness. Given a function $f : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, N\}$ we have a black box which given any input i, returns the value of $f(i)$. We want to determine if two inputs i, j where $i \neq j$ returns the same value, i.e. $f(i) = f(j)$. The number of calls to the black box function is the measure of complexity of the algorithm. In the classical case, the complexity scales as $O(N)$. In the quantum case, the first quantum algorithm based on Grover's search was given by Buhrman et al. [91] which was able to solve the problem in $O(N^{3/4})$. The work of Ambainis [67] improves this using a different approach involving quantum walks and brings the complexity to $O(N^{2/3})$, which is optimal [92].

The seminal work of Ambainis [67] and also the quantum walk search algorithm [50] have both been used as subroutines in other new quantum walk algorithms. The most notable of these being the triangle finding problem [68], matrix product verification [93] and finding a marked subgraph [94]. These algorithms were generalised as 'subset finding' problems by Ambainis [69]. In addition to these quantum walk algorithms, there are several other algorithmic applications of the quantum

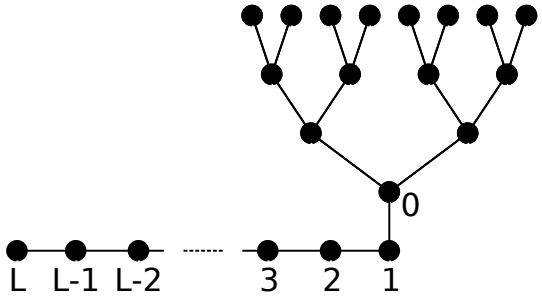

Figure 2.10: A formula evaluation tree as an undirected graph with a finite tail attached.

walk: uniform sampling [95] and other graph theoretic properties [96, 97]. For a good review of current advances in quantum algorithms, specifically quantum walk algorithms, see Ambainis [69, 98] and Santha [99] and also references therein.

More recently, a new quantum walk algorithm has been discovered for the evaluation of Boolean functions and thus so called 'span problems'. This work was pioneered by Farhi et al. [100] who showed a continuous time quantum walk algorithm which could evaluate a full binary AND-OR tree in $O(\sqrt{N})$ steps. This was translated, and subsequently improved, in the discrete time case by Ambainis et al. [89, 101, 102]. This allowed the evaluation of arbitrary Boolean formulae in $O(N^{1/2+o(1)})$ steps. The generalisation to span programs, which can be used for proving lower bounds on circuit size [103], was discovered by Reichardt and Spalek [90]. A general description of the basic idea is as follows. A formula tree is set up as an undirected graph with a tail attached, fig. 2.10. A quantum walk is started in a specific state across the vertices of the tail and is allowed to propagate. At the leaves of the formula tree, the operation which is performed depends on the value the leaf holds. If initialised in the right way, after $O(N^{1/2+o(1)})$ steps of the walk the the resultant state only changes if the formula evaluates to 1. If it evaluates to 0, the state remains almost the same so it is easy to distinguish the correct result just by measuring the state. The generalisation to span programs is the same but the graph is weighted, where the weights on the edges relate to the components of the span program. For a more complete description of the full algorithm and proofs, see the papers above and the recent review by Ambainis [104].

One of the most active areas of research using quantum walks is that of quantum simulation. The dynamics of a quantum system, the Hamiltonian, can be simulated using a quantum walk. One of the main ideas is that the Hamiltonian can be broken down into a sum of local Hamiltonians as suggested by Lloyd [38]. This was generalised for any sparse Hamiltonian [105] and improved by Berry et al. [106]. Childs improves on the these cases by using an approach based on the quantum walk. In [107, 108], a sparse Hamiltonian is broken down into several others whose graphs have all non-zero entries and are all star graphs. A quantum walk is then used to efficiently simulate each individual piece. This was extended in [86] to any non-sparse Hamiltonians, however, Childs and Kothari [109] show there is a limitation to which non-sparse Hamiltonians this is valid for. They show that graphs with small arboricity, a measure of how dense the graph is, can be efficiently simulated using the quantum walk approach.

Another application of quantum walks is as a model to understand transport and other phenomena in physical systems. Examples of this include, but are not restricted to, showing coherent quantum control of atoms in an optical lattice [110] and the simulation of classical annealing processes [111]. More recently, the continuous time quantum walk has also been used as a model to describe the high fidelity transport of excitons in photosynthetic systems [112–116]. For further details, a review of coherent transport using the continuous time quantum walk can be found in [117]. In addition to these models, other work has involved modifying the original walk to produce interesting results. The most notable of these include entanglement generation using quantum walks [118], quantum walks with anyons [119], community detection in social networks using quantum walks [120], quantum walks applied to the graph isomorphism problem [121], inhomogenous (position dependent) quantum walks [122–124] and multi-particle quantum walks [125–127]. Obviously these are just a highlight of current and recent research into quantum walks as the area is currently a hive of interest.

2.7 Experimental implementation

The theoretical analysis of quantum walks has progressed at a much faster rate than has been possible experimentally. Much of the control needed to manipulate single quantum particles, such as photons, is still in development. However, many schemes have been proposed since quantum walks were shown to be of algorithmic use in quantum computing. Some of these schemes have been implemented to show 'proof of principle' of the quantum walk on the line for small numbers of timesteps.

The first piece of work, although not directly related to quantum walks, was presented by Bouwmeester et al. [128]. They performed an experiment using classical optics to illustrate diffusion on a wave-mechanical version of the Galton board. The classical Galton board is a game in which balls are rolled along a board with pins at specific positions. The distribution of a number of balls having been rolled tends to a Gaussian distribution, just like a classical random walk. Without realising at the time, Bouwmeester et al. showed experimentally how classical wave-like interference can be used to show the distribution of the quantum walk. We note here that although the 1D quantum walk can be shown experimentally using just classical methods, this does not mean that entanglement plays no role in quantum walks in general. As soon as we move to higher dimensions of the quantum walk, the quantum entanglement would manifest itself in the level of classical resources then needed to implement it by classical means only. For a general review of entanglement in quantum walks and resources needed to implement quantum walks experimentally see Spreeuw [129].

The first experimental proposals with the quantum walk in mind were introduced shortly after the quantum walk was introduced with algorithmic applications in mind [44, 45]. The first proposal was shown by Travaglione and Milburn [130] who gave a scheme for implementing the discrete time quantum walk on either the line or the cycle using ion traps. They showed how one could encode the position and coin states of the walker in the motional and electronic states of the ion respectively. By applying various pulses to the ion trap, both the coin and shift operators can be

performed. In order to implement the walk on a cycle, the ion is 'walked' around a cycle in phase space. In this scheme, the decoherence of the internal states limits the number of steps the scheme is able to implement.

Another scheme to implement the discrete time quantum walk on the cycle was introduced by Sanders et al. [131] using cavity quantum electrodynamics. In this case the coin is implemented by a two level atom traversing a microwave cavity. The cavity starts in an initial coherent state and laser pulses implement the required operations. Depending on the state on the atom, the cavity field undergoes phase shifts, implementing the quantum walk in phase space. The decoherence time of the cavity limits the number of steps possible in this scheme.

Neutral atoms trapped in optical lattices provided the implementation of the next scheme proposed by Dür et al. [132]. They propose implementation of the discrete time quantum walk on the line or cycle in position space. The atoms are trapped in optical lattices and laser pulses are used to alter their internal state. The atoms then periodically shift, left or right, based on the internal state of the atom. At the time of the proposal they estimated that the technology at the time would have been able to implement several hundred steps of the quantum walk.

These proposals are generally seen as the most important initial schemes. Many others have been proposed since then including schemes using just linear optics [133], classical interference effects [134–137], cavity quantum electrodynamics [138], Bose-Einstein condensates trapped in optical lattices [139], ion traps [140] and quantum accelerator modes [141]. Although new, many of these schemes just extend or improve the original schemes proposed. However, more recently several new proposals have been made which may allow higher dimensional quantum walks to be realised. In the first of these schemes, Eckert et al. [142] uses the same idea as in [132] but uses trapping potentials of the optical lattice as the coin as opposed to the internal states of the atom. In order to extend this to a 2D lattice, four levels are formed by creating a tensor product of two trapping potentials and two hyperfine levels of the atom. In related work, Roldán and Soriano [143] also give a proposal for a higher

dimensional quantum walk. In this scheme, optical cavities are used where the frequency of the light field encodes the position state. The polarisation of the light field gives the two orthogonal directions of the position coordinates. This scheme uses physically different spatial paths through cavities to encode the coin state. In other work, Zou et al. [144] show a scheme for implementing a 1D walk with a higher dimensional coin. In this case linear optics are used to encode the position state as the orbital angular momentum of the photon and again spatially separated paths are used for the coin state.

The first experimental implementation of the quantum walk was shown by Du et al. [145]. They implemented the continuous time quantum walk on the cycle of 4 vertices using a two qubit NMR quantum computer. They show the periodicity of the quantum walk on this cycle as shown in [130] and numerically studied by Tregenna et al. [83]. Du et al. also experimentally show that quantum walk on this cycle on 4 vertices yields a uniform distribution at specific points.

Shortly after this initial implementation, Mandel et al. [146] experimentally showed coherent transport of neutral atoms in optical lattices. They showed coherences between atoms which were delocalised over 7 lattice sites. Although the work was not done in a quantum walk context, this in effect experimentally showing that the shift operator of the discrete time quantum walk can be implemented. This work is seen as a first step in the experimental implementation of the proposal of Dür et al. [132].

Implementation of the discrete time quantum walk was first experimentally shown by Ryan et al. [147] in a three qubit NMR quantum computer. This was shown on a circle of 4 vertices for 8 timesteps. In addition to showing this 'proof of principle' for the discrete time quantum walk, decoherence was added after each step to show the transition back to the classical walk.

Several other of the proposals above have also been experimentally implemented. The quantum quincunx of [131] was shown by Do et al. [148] but using linear optics over 5 timesteps. Zhang et al. [149] implemented the proposal of [144] for 3 steps

of the discrete time quantum walk using the orbital angular momentum of photons. The recent proposal of Xue et al. [140] using ion traps to show the discrete time walk was shown by Schmitz et al. [150]. The number of steps in the proposal is unlimited but experimental limitations mean the number of timesteps is restricted to 3. We note that the schemes detailed in [145] and [147] are experimental examples of quantum computations, whereas the proposals above are experimental examples of physical quantum walks.

In related work, Zähringer et al. [151] present a similar ion trap implementation of a quantum walker in phase space. However, they are able to realise up to 23 steps of the discrete time quantum walk. More recently, lattices of waveguides have been used by Perets et al. [152] to experimentally show the continuous time quantum walk with reflecting boundaries. Also using optical waveguides Peruzzo et al. [153] have recently shown quantum walks with two walkers using two entangled photons.

Finally, we mention three recent experimental results. The first of these was presented by Karski et al. [154], which give the first experimental realisation of the quantum walk in position space using Cs atoms trapped in an optical lattice, in a scheme similar to that proposed in [132]. Although quantum walks have previously been experimentally implemented, this is the first realisation of a quantum particle moving in position space coherently by controlled internal states, atomic spin. In the experiment, 10 steps of the discrete time quantum walk were shown and also reversed back to the initial state. These results show great control over the quantum state and coherences and should allow many other properties of the quantum walk to be explored. For example, this scheme allows the implementation of position dependent coin operators as in the inhomogeneous quantum walk of Linden and Sharam [122].

The second set of recent results are from Broome et al. [155] who use linear optics to realise a discrete time quantum walk. The coin state in encoded into the polarisation of the photon and the position of the walker is determined by spatially seperated paths. This is similar to the proposal of Zhao et al. [133] but additionally

allows tunable decoherence and control over operations at all points of the walk. They show 6 steps of the quantum walk both with varying levels of decoherence and also with absorbing boundaries. The fact this scheme allows tunable decoherence allows the study of the quantum to classical transition along with random sampling applications for algorithms. In addition, this scheme also allows position dependent coin operators as in the previous experiment.

The final experiment we mention is the first scaleable implementation of the discrete time quantum walk. Schreiber et al. [156] show a 5 step quantum walk using a loop of optical elements essentially allowing any number of steps of the quantum walk to be performed with a constant level of resources. They use the polarisation of photons for the coin state and the position of the walker is translated via the arrival time at the detectors. The coin state encoding could be changed to the orbital angular momentum to allow the implementation of higher dimensional coin operators. The limiting factor in this experiment is the quality of the optical elements used. Due to the large number of timesteps that are feasible in this experiment and the ability to apply varying coin operators to different position states, this experiment should provide a starting point for the implementation of a 'proof of principle' experiment of the quantum random walk search algorithm of Shenvi et al. [50].

Chapter 3

Universal quantum computation using the discrete time quantum walk

3.1 Introduction

A proof that continuous time quantum walks are universal for quantum computation, using unweighted graphs of low degree, has recently been presented by Childs [49]. In this chapter, we present a version based instead on the discrete time quantum walk. We show the discrete time quantum walk is able to implement the same universal gate set, and thus both discrete and continuous time quantum walks can be considered computational primitives. Additionally, we discuss perfect state transfer on graph structures and give a set of components on which the discrete time quantum walk provides perfect state transfer. Since completion and subsequent publication of this work, Underwood and Feder [157] have presented another construction combining the continuous time work of [49] and the discrete time case we describe here.

In [49], Childs extends the original results of Feynman [9], which can be interpreted, loosely, as showing that quantum walks are universal for quantum com-

putation. Feynman constructed a time independent Hamiltonian which can then be used to implement any quantum computation, the idea being to show that a physical quantum mechanical device for information processing was reasonable. As Childs points out, the dynamics of a time independent Hamiltonian can be implemented by a quantum walk. In this case, the graph the walk traverses is weighted and directed. In addition, in order to satisfy Hermiticity, the weights on opposite edges of the graph must occur as complex conjugate pairs. It is more physical, and clearer, to view Hamiltonian dynamics as a quantum walk on sparse, unweighted and undirected graphs. Childs [49] proves that a continuous time quantum walk, on an unweighted graph of bounded degree, is universal for quantum computation. This extends Feynman's work, as in his original construction, the degree of the representative graph grows with the size of the computation.

In his work, Childs was motivated to show universality of quantum walks for two reasons. Firstly, it shows the computational power of the quantum walk, proving that any quantum algorithm can be recast as a quantum walk algorithm. This fact alone shows motivation for the continued search for new quantum walk algorithms. Secondly, Childs points out there may be applications for his construction to quantum complexity theory. Feynman's original construction has previously been used to construct QMA- [158] and BQP-complete [159, 160] problems. It therefore seems reasonable to assume that this new construction may give new insights into quantum complexity.

Childs gives an explicit construction that converts a standard gate model computation into a graph, on which a continuous time quantum walk executes an algorithm by traversing the graph. The construction requires an exponentially large graph for the size of the input, needing 2^n wires for an n qubit input. The quantum walk takes place on this N-vertex graph, on which it is already known the walk can be simulated efficiently by a universal quantum computer using $poly(\log N)$ gates, provided there is a simple rule to compute the neighbours of any vertex [65]. Thus, by performing the quantum walk on a quantum computer, the binary encoding brings the resources

required back to the expected level. We discuss the resource requirement in more detail in sec. 3.4.

Our construction for the universal gate set in discrete time is similar to [49] but has maximum degree, d, of eight at any vertex as opposed to three in the continuous case. The continuous time walk can easily be propagated in one direction with no reflection at the vertices. The discrete time walk is not so straightforward, it can only be propagated in one direction by using a specific coin corresponding to the σ_x operation. Using this coin restricts the graph to vertices of degree two, providing no way to construct higher degree structures. Thus, we find we must use a double-edged wire to accomplish directional propagation. This solution has its roots in the connection between the continuous and discrete time walks. Strauch [87] has shown that, as we take the continuous limit of the discrete time walk on the line, we actually get two copies of the continuous time walk propagating in opposite directions. Childs [86] later showed a direct correspondence between the discrete and continuous time quantum walks on arbitrary graphs. In the same work, Childs shows how a discrete time walk can be used, at its limit of small eigenvalues, to approximate the continuous time walk. He uses this 'lazy' quantum walk approach to allow the discrete time walk to propagate in the same way as the continuous. This same approach could be used in this case to allow the computation to be performed on the same structures defined in [49]. However, this would require the discrete time walk to approach the limit at which it is doing very little at each timestep. This would then increase the overhead required to allow completely deterministic computation.

We start by describing perfect state transfer and work previously done in the context of spin chains. We also discuss perfect state transfer on graph structures using the quantum walk. However, most of this has been done using the continuous time quantum walk. We then move on to describe the universal gate set for quantum computation we are able to show using the discrete time quantum walk. This includes structures which exhibit perfect state transfer from one side of the graph

to another. Using these structures, we show how to create larger quantum circuits, before finally concluding by comparing our discrete time quantum walk structures to the continuous time construction of Childs [49].

3.2 Perfect State Transfer

Quantum walks have been used as quantum transport models in the context of spin chains for quantum communication [161–163], and more recently to investigate high fidelity exciton transfer in photosynthetic molecules [114–116]. The glued trees algorithm of Childs et al. [65], and the hitting time of a quantum walk on a hypercube [82], both showing exponential algorithmic speed ups, are also examples of quantum transport. All these examples of transfer phenomena allow an arbitrary state to be transferred from one vertex of an undirected graph to another. In the case of quantum communication or charge transfer, we require either perfect, or high fidelity, transfer to occur, whereas in algorithmic problems, we are more interested in the time to travel between the two points (as long as the probability of reaching the desired vertex is not exponentially small). As a preliminary to our quantum computation scheme, we discuss structures on which perfect state transfer can be achieved using the quantum walk, an important property for both quantum communication links and quantum computation.

Spin networks were initially investigated by Bose [161], who proposed using spin chains for short range quantum communication between distant qubits. A network of spins can be represented by an undirected graph, where the vertices represent the qubits (individual spins) and the edges represent the interactions or couplings between them. We want to be able to find a network which is able to provide perfect state transfer from one vertex to another. For quantum communication purposes, we ideally want some form of network which does not need dynamic control over the individual interactions. In the work of Bose [161], a 1D, unmodulated (all the couplings between neighbouring qubits are the same) chain of N qubits is initialised

with a specific state at one end,

$$|\Psi\rangle = |\psi\rangle|0\rangle^{\otimes N-1}, \tag{3.1}$$

where $|\psi\rangle$ is the state we wish to transfer. Allowing the system to evolve with the Hamiltonian in [161],

$$H_{\text{BOSE}} = J\sum_{n=1}^{N-1}(X_nX_{n+1} + Y_nY_{n+1} + Z_nZ_{n+1}), \tag{3.2}$$

where X, Y, Z are the standard Pauli matrices, Bose wanted to transfer the state perfectly to the opposite end of the spin chain to give the output state

$$|\Psi\rangle = |0\rangle^{\otimes N-1}|\psi\rangle. \tag{3.3}$$

For $N = 2$ and $N = 3$, the transfer of the state from the first qubit to the end qubit can be achieved with perfect fidelity. Christandl et al. [163] showed later this same perfect state transfer is not possible for $N > 3$ in this setting. However, Christandl et al. [162] gave an alternative arrangement by placing the spins in a hypercubic pattern as opposed to a 1D chain. This allows perfect state transfer for any N spin hypercube. In addition, by modulating the strength of the couplings between neighbouring spins, perfect state transfer is also possible for a 1D spin chain of any number of spins [163]. For a good review on perfect state transfer using spin chains, see Bose [164] or Kay [165].

The propagation of a state through spin systems follows the same dynamics as a continuous time quantum walk. In addition, the properties of instantaneous mixing and periodic cycles are closely related to perfect state transfer and have been studied in detail for the continuous time quantum walk [166–169]. Tamon et al. proved that this instantaneous mixing can be achieved on cycles of 2, 3 and 4 vertices only. The continuous time quantum walk can also be shown to have perfect state transfer properties on many other graphs including the hypercube [79]. For a recent review

on the perfect state transfer and mixing properties of the continuous time quantum walk see Kendon and Tamon [170].

Due to the additional degree of freedom provided by the coin in the discrete time quantum walk, the perfect state transfer properties have been studied in much less detail. This is partly due to the analytical analysis being much more complex, and also as the continuous time quantum walk is generally seen as a more suitable model for both biological and physical systems. One method to investigate the dynamics of the discrete time quantum walk was introduced by Feldman and Hillery [171–173]. This involves attaching semi-infinite tails to the start and end points of the graph to be studied, thus the time to propagate from one tail to the other can then be found. However, although this method is well suited to the task, it has not, as yet, been used to investigate perfect state transfer on graphs. The opposite of the same problem, i.e. no transfer occurs at all, has been studied by Krovi and Brun [174–176] where the hitting time of the discrete time quantum walk was studied. They characterised the cases where the hitting time of the quantum walk becomes infinite, and therefore no mixing or transfer can occur at all. These results highlight the importance of symmetry in the structure being walked upon for perfect state transfer.

For the discrete time quantum walk, slightly larger cycles show exact periodic behaviour than in the continuous time case. Travaglione and Milburn [130] showed that a cycle of 4 vertices has a periodicity of 8 timesteps, after which the entire state returns to the starting position. Tregenna et al. [83] showed more periodic cycles exist, cycles of 2, 3, 4, 5, 6, 8 and 10 were shown numerically to be periodic by varying both the bias and phase in the coin. Perfect state transfer occurs at half the periodic cycle for even cycles, where we obtain the entire state at the opposite point of the cycle as shown in fig. 3.1.

For our case of using the walk for computation, we require the walk to travel perfectly in a single direction. On the structures mentioned, the quantum walk travels around the cycle in both directions and interferes to produce perfect state

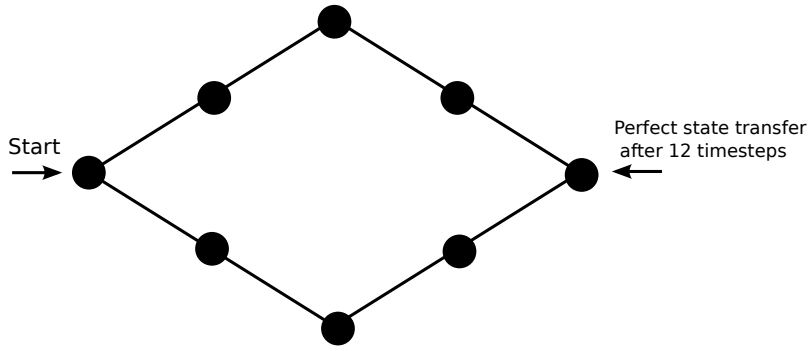

Figure 3.1: Cycle of 8 vertices which gives perfect state transfer from the initial vertex to the opposite vertex after half of the period, 12 timesteps. The entire state returns to the initial vertex in a full period, 24 timesteps.

transfer. Using a completely biased coin (which is just a σ_x operation),

$$H_{bias} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{3.4}$$

we can make the state transfer perfectly around the cycle in a single direction. However, if we then try to attach another structure to the cycle, this periodicity is broken in both cases. The Grover coin, eq. (2.20), can be used to overcome part of this problem at vertices with an equal number of input edges as output edges. For any vertex of even degree, it will transfer the entire state from all the input edges to all the output edges provided the inputs are all equal in both amplitude and phase, as shown in eq.(3.5), where d is the degree of the vertex in question. These results led us to the designs that work for universal computation.

$$G^{(d)} \begin{pmatrix} \alpha \\ \vdots \\ \alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 2/d-1 & 2/d & \dots & 2/d \\ 2/d & 2/d-1 & \dots & 2/d \\ \vdots & \vdots & \ddots & \vdots \\ 2/d & 2/d & \dots & 2/d-1 \end{pmatrix} \begin{pmatrix} \alpha \\ \vdots \\ \alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \alpha \\ \vdots \\ \alpha \end{pmatrix}. \tag{3.5}$$

3.3 Universal Gate Set

We now show how we construct a universal gate set with the discrete time quantum walk. Although the gate set we implement is the same as in [49], the structures used to propagate the discrete time walk are different. The gate set used is the standard universal set comprising the controlled-not (C-NOT) gate,

$$C\text{-NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{3.6}$$

the single qubit Hadamard,

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{3.7}$$

and the phase shift gate (we implement the specific phase shift known as the $\frac{\pi}{8}$ gate),

$$P(\frac{\pi}{8}) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}. \tag{3.8}$$

These gates create a universal set that can implement any quantum computation [177].

In order to represent quantum states, Childs defines his computational basis states as quantum wires. The other gates required for universality are then attached to wires and used to connect them together. The computation is represented as a quantum walk on these wires and structures, where the computation flows from input to output (left to right in our diagrams). Note that this encoding is not meant to be implemented directly. The wires represent computational basis states rather than qubits, thus the model does not represent a physical architecture. Instead, the underlying graph structure created would be used to help 'program' a quantum computer. We first show how to construct a simple wire, along which the quantum

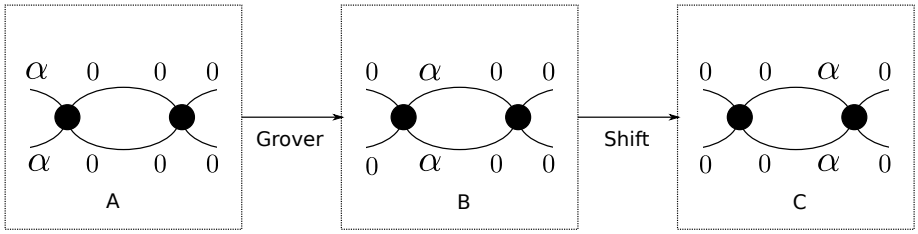

Figure 3.2: Grover coin and shift operation acting on a vertex of degree $d = 4$. Section A shows the initial state, B shows the state after the Grover coin is applied, and finally C is after the shift operation.

walk will propagate naturally in one direction. We use two edges per wire to ensure no reflection occurs at a vertex. We distribute the walker across the two edges which then recombine at the next vertex. As the split is equal, the Grover coin in effect moves both halves to the output edges of the vertex. Figure 3.2 shows this operation and the 'shift' to the next vertex in explicit steps. The Grover coin, eq. (2.20), is used at each vertex of degree $d = 4$. The initial and final vertices are in effect degree four, if we include other edges attached to either end. Figure 3.3 shows the basic wire we use. The computation would start with the amplitude at the initial vertex spread equally across the pair of edges in a wire. For example, the state $\alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$, would be split thus,

$$|\psi\rangle_{initial} = \frac{1}{\sqrt{2}} \left[\alpha|0\rangle_a + \alpha|0\rangle_b + \beta|1\rangle_a + \beta|1\rangle_b\right], \tag{3.9}$$

where the subscript a refers to the top line of the wire and subscript b is the bottom line. The walk propagates left to right on the wire deterministically, in this case reaching the incoming edges of the final vertex in four timesteps. These wires form the basic connections in the computation.

The simplest gate to construct is the C-NOT. It is trivial to implement by just exchanging the wires of the second qubit. The C-NOT gate is shown in fig. 3.4 and shows how the second qubit is flipped but the first qubit is untouched.

The phase gate, eq. (3.8), requires the addition of a relative phase to one wire or computational basis state in relation to the other. To accomplish this, but still have only one coin operator for each vertex of the same degree, we modify the basic

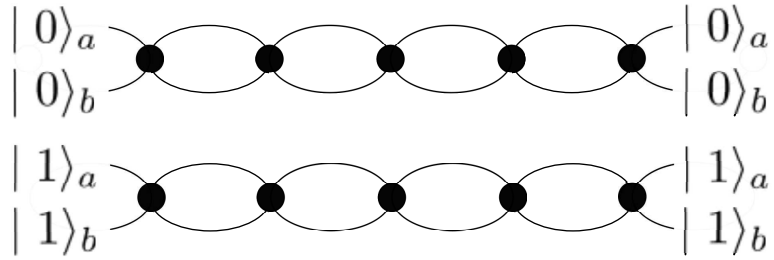

Figure 3.3: Basic wire used to propagate the quantum walk from left to right only. At a vertex of degree $d = 4$, the Grover diffusion coin is used. The initial state is split across the pair of edges in the wire, eq. (3.9).

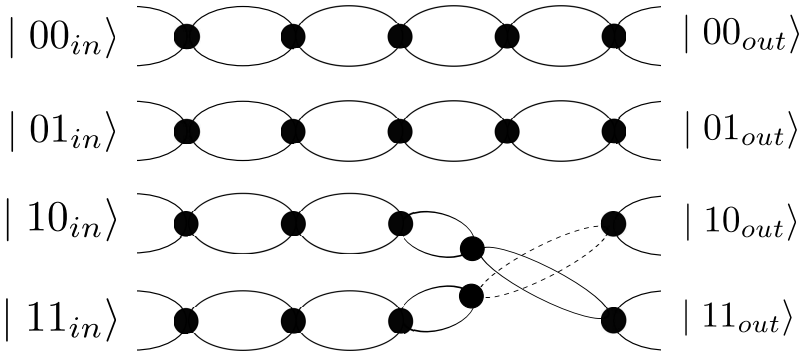

Figure 3.4: Structure used to implement a C-NOT gate. In this case, the first qubit is the control and the second is the target. The target qubit's wires are interchanged and the control qubit is left unaltered. The dotted lines represent wires passing underneath the solid lines - there is no interaction between these wires.

wire and add a phase factor, $e^{i\phi}$, to it,

$$G_{\phi}^{(4)} = e^{i\phi} G^{(4)}. \tag{3.10}$$

Thus, as the walk propagates along a basic wire, it now picks up a phase of $e^{i\phi}$ each time it passes through a vertex of degree $d = 4$. For the wires shown in fig. 3.3, the walker would pick up a phase of $e^{5i\phi}$ as it passes through the final vertex. The phase added here is arbitrary and can be set to any value so long as it is set to the same value for all vertices of degree $d = 4$. As we are looking to implement a $\pi/8$ gate, we set it as follows: $\phi = -\pi/4$. In order to add a relative phase of $\pi/4$ between the $|0\rangle$ and $|1\rangle$ wires, we insert the structure in fig. 3.5 into the graph at the required

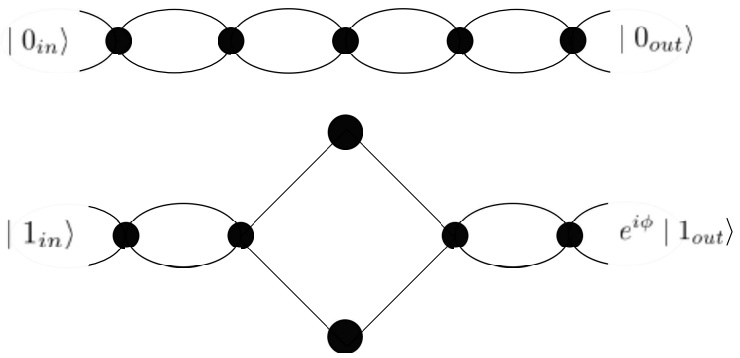

Figure 3.5: Phase gate structure. The $d = 2$ Grover coin, eq. (3.11), is used at the vertices of degree $d = 2$. The $|1\rangle$ wire will pick up a phase of $e^{i\phi}$ relative to the $|0\rangle$ wire. In our construction, we actually obtain the operation corresponding to a phase of $e^{i\frac{\pi}{4}}$ as we set $\phi = -\pi/4$.

point. In this structure there are also vertices of degree $d = 2$, at which we use the Grover coin at its limit of degree $d = 2$, again the same as the σ_x operation,

$$G^{(2)} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{3.11}$$

We add no phase to eq. (3.11) so as the walker passes through these vertices no phase is picked up. In fig. 3.5, the walker propagates along the $|1\rangle$ wire and picks up a phase of $e^{-i\pi}$ in four timesteps as it only passes through four vertices of degree four. However, the $|0\rangle$ wire picks up a phase of $e^{-5i\frac{\pi}{4}}$ in the same number of timesteps as all its vertices are of degree four. Relative to the $|0\rangle$ wire, the $|1\rangle$ wire will pick up a phase of $e^{i\frac{\pi}{4}}$. Therefore, using the structure described here we obtain the operation in eq. (3.8).

The last gate in the universal set is the Hadamard gate. This requires an interaction between the two computational basis states. The structure we use to perform this operation is shown in fig. 3.6. This looks complex in relation to the other gates we have shown so we break it up to explain it more clearly. Sections A and C of the structure are each two phase gates giving a relative phase of i to the $|1\rangle$ wire before and after the main section of the gate (B). Section B of the structure combines the two inputs from the $|0\rangle$ and $|1\rangle$ wires and then splits this across the outputs equally.

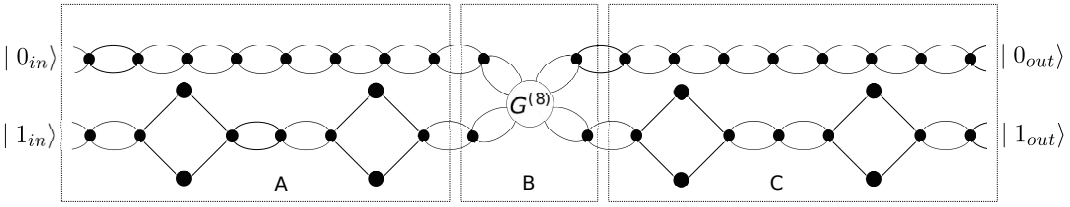

Figure 3.6: Hadamard gate structure. Sections A and C each add a relative phase of i to the $|1\rangle$ wire. Section B performs the unitary transformation of eq. (3.12) to the incoming state. The structure adds a global phase of $3\pi/4$ to the wires.

The structure here is similar to the basis changing gate in [49]. In order to obtain the desired operation on this structure, we have designed a coin for vertices of degree $d = 8$:

$$G^{(8)} = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & i & i & -1 \\ 0 & 0 & 0 & 0 & i & 1 & -1 & i \\ 0 & 0 & 0 & 0 & i & -1 & 1 & i \\ 0 & 0 & 0 & 0 & -1 & i & i & 1 \\ i & -1 & 1 & i & 0 & 0 & 0 & 0 \\ -1 & i & i & 1 & 0 & 0 & 0 & 0 \\ 1 & i & i & -1 & 0 & 0 & 0 & 0 \\ i & 1 & -1 & i & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3.12}$$

This operator combines the complex Hadamard operator,

$$H_c = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}, \tag{3.13}$$

and the σ_x,

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{3.14}$$

in a tensor product form,

$$G^{(8)} = (H_i \otimes H_i) \otimes \sigma_x, \tag{3.15}$$

with the top two and bottom two rows of the $H_i \otimes H_i$ matrix rearranged. This rearrangement ensures the outputs come out in the same order as the input states.

This gate adds a global phase of $3\pi/4$. The phase gates at the start and end of the central section give us the relative phase of -1 required for the Hadamard operation. We note here that the choice of where to place the phases in this construction is arbitrary. The same result can be achieved by using the degree four Grover coin, eq. (2.23), with no phase at vertices of degree four, and the degree two Grover coin, eq. (3.11), with a phase of $\pi/4$ at vertices of degree two. However, by placing the phases on the Grover coin in the fashion we described previously, eqs. (3.10, 3.11), means that the global phase added by the Hadamard gate, eq. (3.12), corresponds to the phase added by a wire of the same length.

3.4 Constructing Quantum Circuits

Thus far, each gate we have described only acts on one or two qubits. However, non-trivial quantum computers involve many qubits. We now describe how to link these wires and structures together to form larger circuits. Figure 3.8 shows the underlying graph structure of the circuit in fig. 3.7. The graph structure is obtained by connecting together wires and structures so that the walk flows from left to right. For this reason, we designed our wires and structures to both input and output from vertices of degree four, thus making it simple to link them together. The initial state of the computation is set on all or a subset of the vertices on the left hand side of the graph, with the amplitude at each vertex split across the incoming edges. This initial state can be thought of as the first column of vertices in the graph structure in superposition, with each subsequent column of vertices representing a further timestep. For example, in fig. 3.8 this initial column of vertices is the set prior to the Hadamard structures. The walker is propagated across the graph structure, from left to right deterministically, for the required number of timesteps. We therefore do not require the addition of momentum filters or separators as in the continuous time case. Our structures all propagate the walker at the same speed, meaning output from the wires will be synchronised throughout the computation. Finally, the walker picks up a global phase of $-\pi/4$ per vertex that is not part of a gate that

changes the phase, so all the wires also stay synchronized in phase. Thus, we know with certainty that, after the required number of timesteps, the walker will have a distribution over just the output vertices on the right hand side of the graph. Once the computation has been completed, we measure the output vertices. We will find the walker at just one of these vertices, representing the output of the computation.

The graph structure in fig. 3.8 is clearly larger in size than its equivalent representation in the circuit model, fig. 3.7. In fact, for a general n-qubit computation the equivalent graph will have 2^n wires, one for each combination of computational basis states. Similarly, we require more gate structures than in the circuit model. Single qubit structures are repeated 2^{n-1} times and for the C-NOT gate we need 2^{n-2} structures. As an example, we can see the phase gate acting on the third qubit in fig. 3.7 is repeated four times in the underlying graph structure of fig. 3.8, one for each combination of wires involving this qubit. Although this seems as though we would lose any form of quantum speed up due to the exponential number of gates required in the underlying graph, this is not the case. Consider simulating a classical random walk on a classical computer, the N-vertex graph is represented in $\log_2 N$ bits of memory with each vertex having a unique binary number as a label. In a similar fashion, if we simulate a quantum walk on a quantum computer, the N-vertex graph can be represented by $\log_2 N$ qubits. Therefore, if we encode our graph using qubits, we can describe the 2^n wires in just n qubits. By manipulation of a single qubit we can affect all combinations of wires associated with that qubit. As the state moves across the graph, the adjacent vertices must be established. In complex graphs the description of the graph and its connections is often exponential in size and an oracle must be used to store it [65]. The graphs produced here are of bounded degree and have a regularity stemming from the repetition of gate structures on combinations of wires involving a specific qubit. Due to the labelling of the wires, we know where to place each structure based on one bit in the label, thus we can efficiently describe the graph. For example, consider the second C-NOT gate in fig. 3.7, which operates on the third qubit with the second qubit as control. We

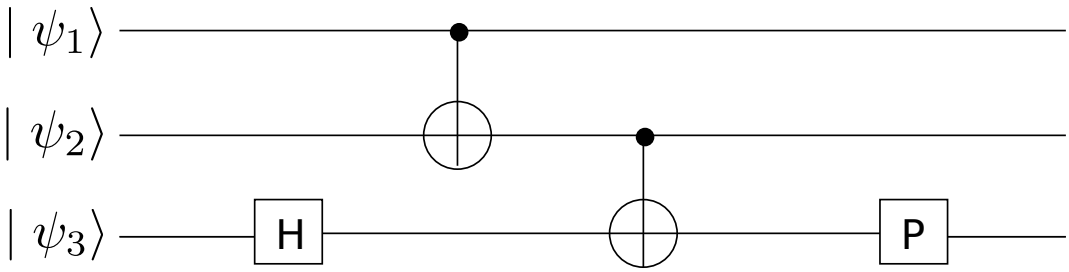

Figure 3.7: Quantum circuit on three qubits. A Hadamard operation is performed on qubit three followed by two C-NOT gates. Finally, a phase gate is applied on qubit three. The underlying graph of this structure is shown in fig. 3.8.

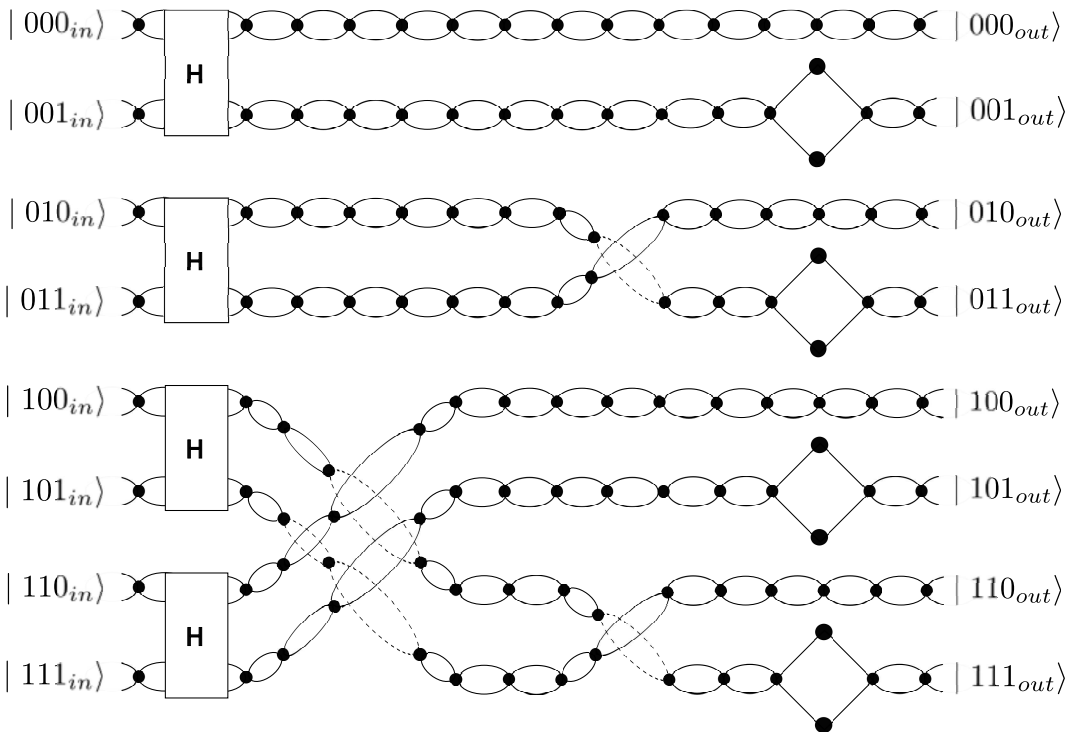

Figure 3.8: Graph to represent the quantum circuit in fig. 3.7. The Hadamard structure, H, is the same as in fig. 3.6. The dotted lines represent wires passing underneath the solid lines - there is no interaction between these wires.

can see from fig. 3.8 that it is easy to identify where the C-NOT structures should be placed. The labelling of the wires shows that the middle bit determines which combinations of wires relate to the second qubit having a value of 1. Similarly, we can also identify which wire it should link to by the last bit in the labelling scheme, it needs to be flipped relative to the original wire, i.e. $|010\rangle$ links to $|011\rangle$.

3.5 Discussion

In this chapter, we have described an alternative to the continuous time construction in [49], using the discrete time quantum walk. This shows the discrete time quantum walk is universal, therefore any quantum algorithm can be reformulated as a discrete time quantum walk algorithm. It also confirms that the discrete and continuous time walks are both computational primitives and thus computationally equivalent. This equivalence is dependent on the number of steps in both cases to be of the same order. Our gate constructions require twice the number of edges compared to the continuous time case, but the same number of wires. Our phase gate requires an additional timestep in relation to the continuous time phase gate construct. The number of timesteps required for a computation is also the same as the continuous time case, but with a small overhead depending on the number of phase gates required.

Another difference in the two constructions is the degree of the graphs produced. In the continuous time case, the maximum degree of any vertex in the graph is three. In the discrete time case, we use vertices of higher degree to ensure directional propagation. In most of the structures this involves a doubling of the degree at a vertex, as shown in the case of the basic wire and the C-NOT structure. The Hadamard structure we propose here however, does not follow this doubling. It would seem reasonable, from the equivalent degree three structure in the work by Childs, that it may be possible to decompose our Hadamard structure into one with degree six vertices. The doubling of degree at vertices would then correspond directly to the continuous case. In the discontinuous time case of [157], the construction combines the benefits of both schemes by allowing the walker to take discrete steps of continuous evolution. This scheme, like the discrete time case we describe here, is deterministic for all input states, but has the added benefit that the additional coin degree of freedom is not required, thus reducing the resources needed.

Although this model is universal for quantum computation, neither this or the continuous model of Childs [49] is designed to be implemented physically. Instead it is meant to show the power of the quantum walk as an algorithmic tool. As any

quantum algorithm can be defined as a quantum walk algorithm, it provides new motivation that many new quantum walk algorithms can be found. In addition, the construction provided here may give new insights into quantum complexity theory.

Chapter 4

Searching

4.1 Introduction

Searching is undoubtedly one of the most basic problems in computer science. In this context, searching is not just restricted to a physical database but could also be searching through a state space for an entry which fulfills a specific clause such as the constraint satisfiability problem (k-SAT). In k-SAT, we are given a set of clauses, $C_1, C_2.....C_m$, each of which contains k boolean variables which can take values of 0 or 1. The problem is to determine if there is any assignment of these variables such that all m logical clauses are satisfied simultaneously. Clearly as k (and m) increases, the search space increases exponentially. Although many tricks can be played to search data efficiently, such as sorting or using specialised data structures, searching through such a large, unsorted dataset can be a lengthy process. In fact, the complexity of such a task scales linearly with the size of the dataset, N, to be searched. Intuitively, it is easy to see this must be the case as every item must be checked in turn until the specific item is found. On average, half the items will have to be checked before the correct one is located. This leads to the best classical scaling which can be achieved, $O(N)$.

One of the most important quantum algorithms discovered thus far is the searching algorithm of Grover [13]. He showed that an item could be found from a set

of N in a time quadratically faster than the classical case, $O(\sqrt{N})$. Grover uses a technique known as amplitude amplification to increase the probability of finding the desired item from an initially uniform distribution. Although in this thesis we only consider the case of finding a single marked item from a dataset, the algorithm introduced by Grover also applies to a search for multiple items as shown in detail by Boyer et al. [35]. In the case where we have a set of M items, the search algorithm is able to amplify all M to a constant value in $O(\sqrt{N/M})$ steps. Obviously when we measure the resulting state we only obtain one of the M desired items at random. We also note that Grover's algorithm has been shown to be both optimal and also one of the few quantum algorithms which is provably faster than any possible classical algorithm [34].

One application of the quantum searching algorithm could be to break certain classical cryptographic systems, for example the commonly used Data Encryption Standard (DES). In DES, a message is encryted using a 56-bit secret key which two (or more) parties must agree upon prior to starting communication. If a third (hostile) party eavesdrops and manages to intercept both matching parts of a single message, the clear and the ciphertext, it would be possible to find the matching key which maps one to the other. The problem classically is that the optimal method, to isolate the correct key, would be an exhaustive search of all possible keys until the correct one is found. This means searching through 2^{55} keys which would take approximately one year (if one billion keys could be checked every second). If the same search could be done using the quantum search algorithm, the problem could be solved with just 185 million steps [178].

Due to the quantum search algorithm being of such practical importance, much interest has been shown in Grover's algorithm in order to try and make it more robust and easier to implement. Several years after Grover introduced his algorithm, Shenvi et al. [50] gave a quantum walk based search algorithm which was able to match the quadratic speed up of Grover's algorithm. This quantum walk search algorithm has been studied in detail and many improvements have been made since

its introduction. In fact, due to the many uses of searching in algorithms, the quantum walk search algorithm has become a standard tool in developing new quantum algorithms. For a good review see Santha [99].

After discussing the most efficient classical searching method, we move to the quantum case and describe the main parts of Grover's algorithm. Later in this chapter we show how the quantum walk can be easily modified to turn it into the search algorithm of Shenvi et al. [50]. Their work was originally based on the hypercube, but was later extended to d-dimensional lattices [179]. We describe the search algorithm, along with its scaling behaviour, on a two dimensional Cartesian lattice, before concluding this chapter with a review of known results for the quantum walk search algorithm on various structures.

4.2 Classical search algorithm

Consider a phone book where the entries are ordered alphabetically. If given a name, it is easy to find the corresponding phone number as we have knowledge of the ordering and structure of the database. If we take the reverse of this, and are given a number and asked to find the corresponding name, the situation becomes much more difficult and it is less clear how to solve. The obvious way is to just start from the beginning and work through every entry until the correct one is found. It is clear that, on average, we must check half the entries before the correct one is located. In fact, this is actually the optimal classical strategy for searching a large unordered set giving rise to a classical runtime of $O(N)$. Checking each entry sequentially is the equivalent of a magnetic tape storage system. We must work from one end of the tape to another in order to eventually find the required item. Although technological advances have lowered access times, e.g. hard drives, the basic scaling of the runtime is still the same.

In this thesis, we only consider the case where the dataset is unordered and is to be searched only once. In classical searching, if the data to be searched is to be used on many occasions, i.e. a physical database, the data could be sorted into some

logical order. In the case of the phone book, alphabetical ordering is the standard, and most logical, form of ordering. By sorting the dataset in this fashion, other searching methods or specific data structures, e.g. a binary search tree, can then be used to search the data. These methods or structures can reduce the complexity of the search considerably. For example, the run time for a search on a binary search tree is $O(\log N)$ in the average case. Obviously, the initial sorting of the data will take at least $O(N)$ steps so this does not reduce the time to search an unordered list. However, the dataset only needs to be sorted once so this is just a fixed overhead. If the data is to be searched on many occasions, this overhead is often worth the cost. For a good review of all searching and sorting methods currently used in classical computer science, see any good textbook on data structures and algorithms, for example Knuth [180].

4.3 Grover's algorithm

In his seminal work, Grover [13] provided a quantum algorithm which is able to find a specific item from an unordered dataset in a time quadratically faster, $O(\sqrt{N})$, than the classical case, $O(N)$. We firstly describe the algorithm in general terms, before moving on to show a geometric visualisation. Grover's algorithm uses a technique, often referred to as amplitude amplification, to increase the amplitude of the desired state in a system. An oracle is used which can recognise (but does not know) the item we are searching for. As the algorithm works in superposition, many items can be checked simultaneously. People are often confused by the role of the oracle, asking if the oracle already knows the item we are looking for then what is the point of the algorithm. The answer to this is that the oracle does not know which item we are looking for. However, it can recognise the item easily once it is found. This property is often used in problems in computer science. In the example of the phone book, it is easy to check whether the number found is the one we are searching for, but that does not mean we already knew its location.

The main steps of Grover's algorithm are shown in fig. 4.1 and can be described

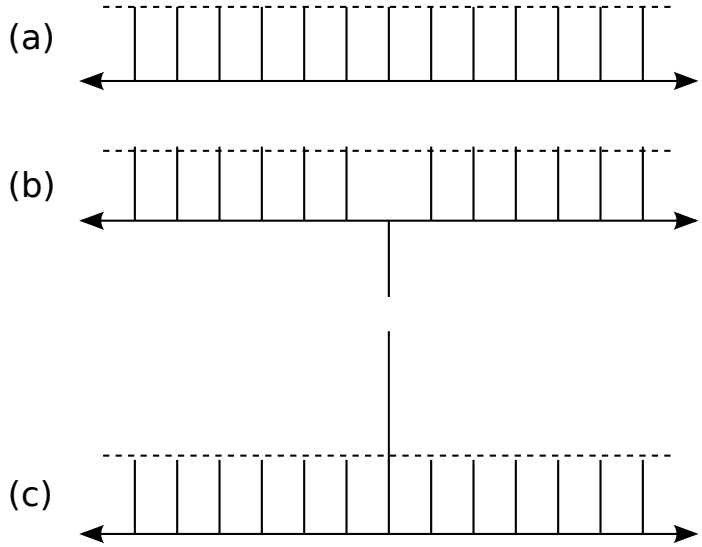

Figure 4.1: Graphical representation of a step of Grovers algorithm. The dotted line represents the average of the entire state. (a) The state is prepared in a uniform superposition over all N items. (b) The oracle is applied which negates the component of the state we are searching for. (c) The diffusion operated is applied which has the effect of inverting the state about the average.

as follows:

(1) Prepare the initial state in an equal superposition of all possible states as in fig. 4.1 (a). In the case of a search over an unsorted database of N items this is

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x\rangle, \tag{4.1}$$

(2) Apply an oracle, C, where $C(x) = 1$ if the state $|x\rangle$ being checked matches the required item or $C(x) = 0$ otherwise. This acts as

$$C|x\rangle \rightarrow \begin{cases} -|x\rangle & C(x) = 1 \\ |x\rangle & C(x) = 0 \end{cases}$$

This operation is the equivalent of a rotation of π radians to the state which represents the item we are looking for as shown in fig. 4.1 (b).

(3) Transform the state using the diffusion operator, G, which is defined as

$$G_{i,j} = \begin{cases} 2/N & \text{if} \quad i \neq j \\ -1 + 2/N & \text{if} \quad i = j \end{cases}$$

The diffusion operation used here is often referred to as the *inversion about average* operator. This is due to the fact that after the operator has been applied, the amplitude in each part of the state is as much below the average of the whole state as it was above the average previously (and vice versa). We can see this more clearly by rewriting the diffusion operator in the following form,

$$G \equiv -I + 2P, \tag{4.2}$$

where I is the identity matrix of the required dimension and P is

$$P_{i,j} = \frac{1}{N}, \tag{4.3}$$

for i, j from 1 to N. It is easy to see that by applying P to any vector $\overline{v}$ will return a vector where each entry is equal to the average of all the components. Applying the diffusion operator to a vector $\overline{v}$ therefore gives

$$G\overline{v} = -\overline{v} + 2P\overline{v}. \tag{4.4}$$

The component i of the state can therefore be written as

$$Gv_i = -v_i + 2A, \tag{4.5}$$

where A is the average of the components of the original vector $\overline{v}$. Rewriting this as

$$Gv_i = A + (A - v_i), \tag{4.6}$$

gives us exactly the inversion about the average of the vector. Figure 4.1 (c) shows

how this 'inversion about the average' operation increases the state we are searching for, while the remaining states only differ slightly.

(4) Repeat steps (2) and (3) $\sqrt{N}$ times. Each time these steps are repeated, the required state is amplified by $O(1/\sqrt{N})$. As these steps are repeated $\sqrt{N}$ times, the amplitude, and thus the probability of the state we are looking for reaches $O(1)$. The resulting state is then measured and we will find the required state with a probability of at least $1/2$. Improvements have been made to the algorithm since its introduction which allow the required item to be found with almost unit probability, [36, 181, 182].

4.3.1 Geometric visualisation

Although we have just described the main steps of the algorithm, it is often useful to show it in a geometric fashion to aid clarity. In this sense, the main part of the algorithm, parts (2) and (3), can be described as a rotation in the two dimensional plane which is spanned by the states representing the desired item and the remaining states we are not searching for. We can define general states for both the item we are searching for, $|\alpha\rangle$, and for all the components which are not the item we are searching for, $|\beta\rangle$, as

$$|\alpha\rangle \equiv |x_m\rangle, \tag{4.7}$$

and

$$|\beta\rangle \equiv \frac{1}{\sqrt{N-1}} \sum |\, x\rangle, \tag{4.8}$$

where $|x_m\rangle$ and $|x\rangle$ are the components of the state for the items we are and are not searching for respectively. The initial state of the algorithm can then be written as

$$|\psi\rangle = \sqrt{\frac{1}{N}}|\alpha\rangle + \sqrt{\frac{N-1}{N}}|\beta\rangle. \tag{4.9}$$

We can see from this expression that the state lies in the two dimensional space which is spanned by $|\alpha\rangle$ and $|\beta\rangle$. The two main parts of the algorithm, the oracle and the diffusion operators, are just reflections in this space, thus creating a rotation

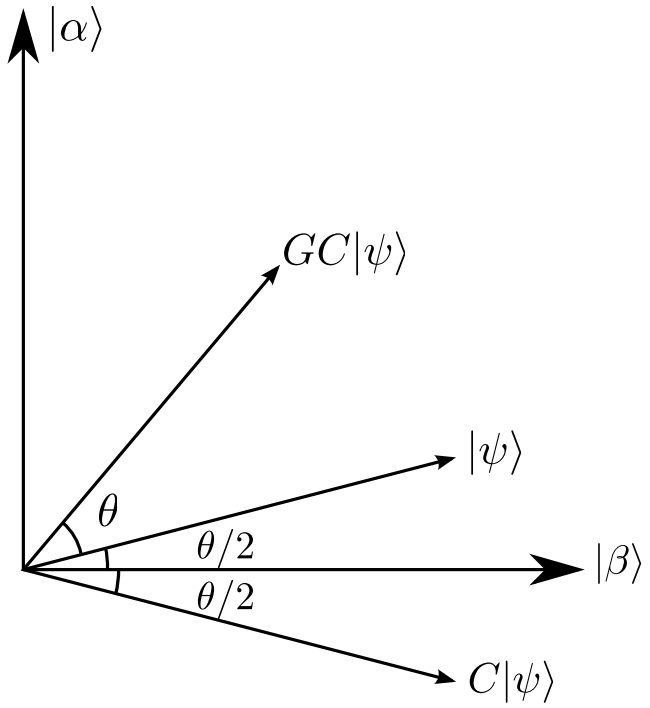

Figure 4.2: Geometrical representation of a step of Grovers algorithm. The state is reflected about $|\beta\rangle$ after application of the oracle. The diffusion operator then performs another reflection, this time about the average of the state, which is very close to the original $|\psi\rangle$. As a result, we see a rotation by θ towards $|\alpha\rangle$.

overall. We show this rotation graphically in fig. 4.2. After each iteration of the algorithm, the initial state is rotated closer to the state we are searching for. Setting $\cos\theta/2 = \sqrt{(N-1)/N}$, the initial state becomes

$$|\psi\rangle = \sin\frac{\theta}{2}|\alpha\rangle + \cos\frac{\theta}{2}|\beta\rangle. \tag{4.10}$$

We first apply the oracle giving

$$|\psi\rangle = \cos\frac{\theta}{2}|\beta\rangle - \sin\frac{\theta}{2}|\alpha\rangle, \tag{4.11}$$

followed by the diffusion operator, G, to obtain

$$|\psi\rangle = \sin\frac{3\theta}{2}|\alpha\rangle + \cos\frac{3\theta}{2}|\beta\rangle. \tag{4.12}$$

On each iteration, the state is rotated by a rotation angle θ equal to $\sqrt{1/N}$ radians. As such, after $\sqrt{N}$ iterations of the main part of the algorithm, the state becomes very close the required state, in this case $|\alpha\rangle$.

4.4 Quantum walk search algorithm

A few years after Grover [13] introduced his quantum searching algorithm, Shenvi et al. [50] introduced a search algorithm based on the discrete time quantum walk which was able to match the quadratic speed up of Grover. We describe here how Shenvi et al. [50] were able to modify the quantum walk into a search algorithm. In their work, they analysed the search algorithm on a hypercube. Here, we show how the quantum walk search algorithm is applied to a 2D Cartesian lattice. The data points we wish to search are layed out as the vertices of an undirected graph. The edges then represent the specific connections between data points. At the edges of the lattice we impose periodic boundary conditions, in effect turning the graph into a torus. Our aim is to find one data item, a specific vertex, out of the set of data to be searched. We start the walker in an equal superposition of all the possible sites in the lattice, and the coin in an equal superposition of all directions,

$$|\psi\rangle = \frac{1}{\sqrt{dN}} \sum_{x=1}^{N} \sum_{c=1}^{d} |x, c\rangle, \tag{4.13}$$

where d is the degree of the vertices in the graph and N is the total number of vertices. If we let the walker evolve in a natural fashion, using the Grover coin eq. (2.20), we would find a flat distribution identical to the starting state at any point in time. This is because the same operation, eq. (2.23) in the case of the 2D lattice, is being performed at every vertex and the walker cannot distinguish a specific vertex from any other. We note here that not all initial states would be stationary or unchanged. The initial state described here is stationary as it is an eigenstate. We need to use a different coin operator for the marked state in order to introduce a bias into the walk. We show later it is optimal to invert the phase of

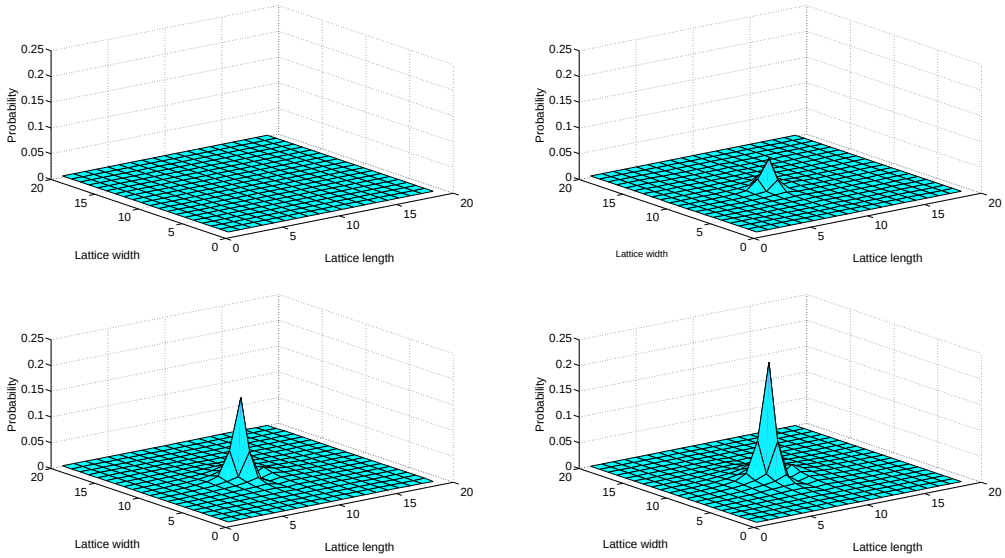

Figure 4.3: Probability distribution of a discrete time quantum walk search on 400 vertices arranged in a 20×20 square with periodic boundary conditions, evolved for 0, 10, 20 and 32 timesteps. The marked vertex is at position 190.

the $G^{(4)}$ coin operator from eq. (2.23) giving

$$G_m^{(4)} = \frac{1}{2}\begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{pmatrix}. \tag{4.14}$$

Figure 4.3 shows how the distribution of the walker evolves with time for a 20×20 lattice, i.e. $N = 400$. We see that using a different coin creates a defect in the walk and the probability coalesces on the marked state over time. As the walk progresses, the probability at the marked state cannot keep increasing without limit. In fact, we see in fig. 4.4 that the probability at the marked state has periodic behaviour with the first peak occuring at roughly $t = (\pi/2)\sqrt{N} \simeq 32$, with maximum probability for $N = 400$ of around 0.23. This can be increased as close to 1 as desired by standard amplification techniques (repeating the search a few times). We see that subsequent peaks occur at other integer multiples of this initial time, $t = n(\pi/2)\sqrt{N}$ where $n = 2, 3, 4 \ldots$.

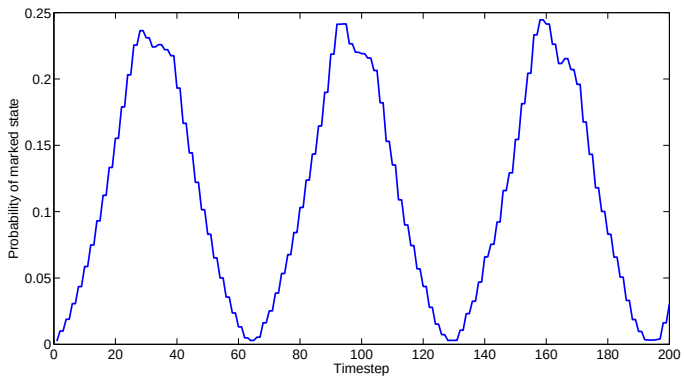

Figure 4.4: Probability of the marked state over 200 timesteps on a 20×20 grid with periodic boundary conditions. The marked vertex is at position 190.

The coin operator used at the marked state does not have to be the negated Grover coin as we have used above. It can be any unitary operator which creates a bias relative to the dynamics of the walk with the standard Grover coin, eq. (2.20). Although it is assumed that the negated coin is the optimum, as it is the furthest coin away from the identity operator, this has never been, as far as we know, investigated. In order to explore how the marked state coin affects the search result, we can introduce a phase into the marked state coin operator, eq. (4.14),

$$G^{(4)}_{\phi,m} = e^{i\phi} G^{(4)}_{m}, \tag{4.15}$$

where $0 \leq \phi \leq \pi$. The standard $G^{(4)}$ coin operator, eq. (2.23), corresponds to $\phi = 0$, and the marked coin operator used before, $G^{(4)}_m$ eq. (4.14), corresponds to $\phi = \pi$. Figure 4.5 shows the effect of varying the phase, ϕ, on the maximum probability of the marked state. It can easily be seen that the largest probability of finding the marked state is when $\phi = \pi$. A bias can also be introduced by a matrix of the form

$$G^{(4)}_{\delta} = \begin{pmatrix} \delta & a+ib & a+ib & a+ib \\ a+ib & \delta & a+ib & a+ib \\ a+ib & a+ib & \delta & a+ib \\ a+ib & a+ib & a+ib & \delta \end{pmatrix}, \tag{4.16}$$

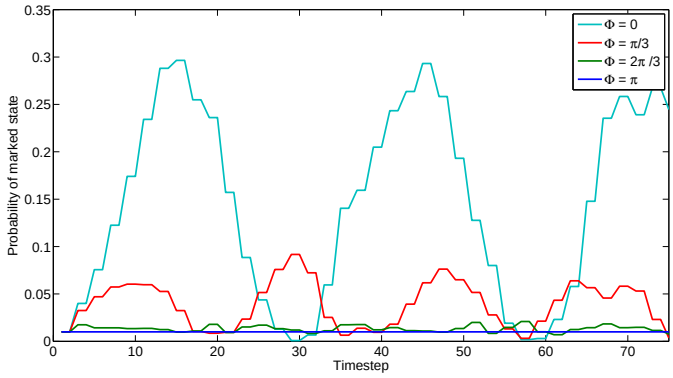

Figure 4.5: Probability of marked state over 75 timesteps for $N = 100$ i.e., a 10×10 grid with marked state at 45, using the marked state coin in eq. (4.15) with $\phi = 0$, $\pi/3$, $2\pi/3$ and π.

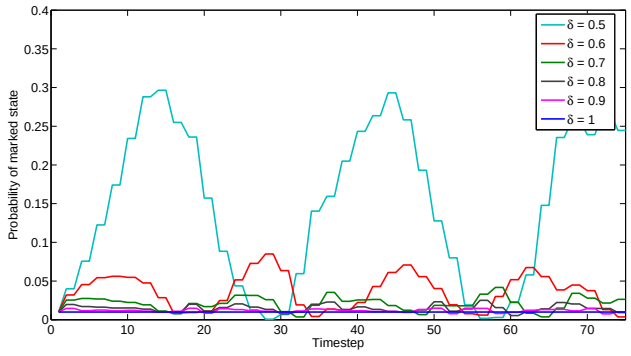

Figure 4.6: Probability of marked state over 75 timesteps for $N = 100$ i.e., a 10×10 grid with marked state at 45, using the marked state coin in eq. (4.16) with $\delta = 0.5$, 0.6, 0.7, 0.8, 0.9 and 1.

where $0.5 \leq \delta \leq 1$. We solve this to give expressions for a and b in terms of the bias parameter δ in order to interpolate between the identity and the marked state operator, $G_m^{(4)}$ eq. (4.14). Taking $\delta = 1$ makes the marked state coin into the identity operator, while $\delta = 0.5$ corresponds to the $G_m^{(4)}$ operator. We show in fig. 4.6 how varying the bias of this coin operator affects the maximum probability of the marked state, showing the maximum probability is obtained at $\delta = 0.5$. These variations were chosen to preserve the symmetry of the coin operator, and we see they justify our choice of the optimal marked state coin operator.

As we have now shown how the quantum walk can be turned into a search algorithm, we are interested in how quickly the quantum walker finds the marked

state. That is, we want to know when the probability of the walker being present at the marked state is at a maximum. As this probability is periodic and we want the algorithm to be efficient, the subsequent peaks are not of interest: we want to know when the first peak occurs. Although it would be ideal to measure the walker at the precise timing of the maximum in the first peak, this is not strictly necessary. In fact, as can be seen in fig. 4.4, the peaks are quite broad, so even if an error occurs in when to measure, it only means a somewhat lower probability of finding the marked state, this is only a constant extra overhead on the amplification. For example, if the state of the walker was measured at half the optimal number of timesteps ($t = (\pi/4)\sqrt{N} \simeq 16$), the probability of the walker being measured in the marked state is roughly half that of the maximum possible ($p \approx 0.1$).

In later chapters, we discuss the algorithmic efficiency of the search algorithm on various graph structures. It is important we define here what factors of efficiency we are interested in. As we are looking to finding a specific item from a set of many, we must consider how likely it is the walker coalesces at the marked state. The maximum probability of the walker at the marked state, i.e. the maximum value of the first peak, varies with the size of the dataset (for the 2D Cartesian lattice). In this case, the theoretical value of $O(1/\log_2 N)$ from Ambainis [69] is numerically confirmed in our results in fig. 4.7 with a small prefactor of just over 2. The second factor we are interested in is the number of timesteps it takes to reach this maximum probability. The scaling of the time to find the marked state with the size of the dataset, N, for the 2D Cartesian lattice is shown in fig. 4.8. We see a scaling of $O(\sqrt{N})$ here, also with a prefactor of 2. In order to compare our results in later chapters to previous work, we must consider the total algorithmic complexity of the quantum walk search algorithm. In the case of the 2D Cartesian lattice, the maximum probability scales as $O(1/\log_2 N)$. As such, we must use amplitude amplification techniques to increase this to a constant value. This has previously been shown to take $O(\sqrt{\log N})$ time steps [36]. This makes the total algorithmic complexity $O(\sqrt{N \log N})$ for the 2D lattice, in agreement with the recent results of

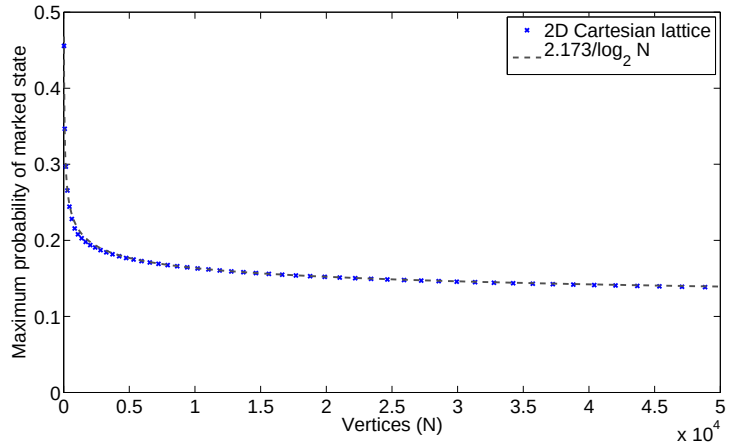

Figure 4.7: Maximum of the first peak in the probability of being at the marked state for different sized data sets, using the optimal marked state coin in eq. (4.14) on a 2D lattice of size $\sqrt{N} \times \sqrt{N}$, plotted against N (crosses). Also shown is the closest fit to our data, $2.173/\log_2 N$ (dashes).

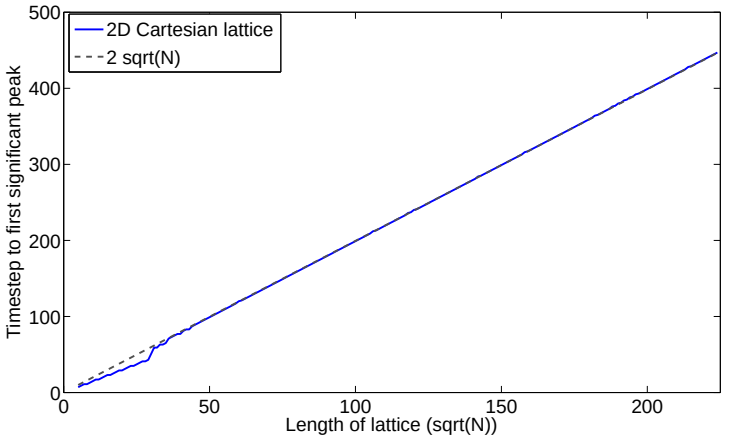

Figure 4.8: Time step at which the first peak in the probability of being at the marked state occurs for different sized data sets, using the optimal marked state coin in eq. (4.14), plotted against $\sqrt{N}$. Also shown is the closest fit to our data, $2\sqrt{N}$.

Tulsi [183] and Magniez et al. [85]. These scalings are not the same for all graph structures. In particular, on a cubic lattice, the maximum probability scales as a constant value $O(1)$. As such, only a constant number of amplification steps are needed to bring the probability to ≈ 1, thus the total algorithmic complexity is just $O(\sqrt{N})$.

4.5 Review of recent results for the searching algorithm

Since the introduction of the quantum walk by Aharonov et al. [44] and Ambainis et al. [45], there has been a surge of interest into quantum walk algorithms in general as already discussed, sec. 2.6. However, the quantum walk search algorithm of Shenvi et al. [50] in particular has received significant interest as it is fundamental to a wide range of problems in computer science.

The quantum walk search algorithm [50] is able to find a marked item in a time quadratically faster than the classical case, which is known to be an optimal and provable speedup [34]. In [50], the items of the dataset are layed out as the vertices of an undirected graph, specifically a hypercube of dimension $\lceil \log_2 N \rceil$, on which the quantum walk can be solved analytically [79]. The constant prefactors to the $O(\sqrt{N})$ scaling on the hypercube have been determined analytically in work by Hein and Tanner [184]. Other recent work by Potoček et al. [185] has improved the original algorithm by adding an additional coin dimension, allowing the probability of the marked state to approach unity after just one run of the algorithm. This brings the running time of the quantum walk search algorithm very close to the optimal for searching an unsorted dataset, $\pi/4\sqrt{N}$. Zalka [186] has previously shown that, for a probability of finding the marked state to be one, this is the best that can be achieved.

However, the hypercube studied in [50] is a highly connected but non-physical structure. In order to make the algorithm more physical, the study of the search algorithm on lower dimensional structures was started by Benioff [187]. He considered the additional cost of the time it would take a robot searcher to move between different spatially separated data points on d-dimensional lattices, stating that in two spatial dimensions, D, no speedup was apparent. Subsequently, Aaronson and Ambainis [188] introduced an algorithm based on a divide and conquer approach, contradicting this claim with a run time of $O(\sqrt{N})$ in dimensions $D \geq 3$ and $O(\sqrt{N}\log^{3/2} N)$ when $D = 2$ as shown in table 4.1.

Around the same time as this work, Childs and Goldstone [189] gave another

Work	$D = 2$	$D = 3$	$D = 4$	$D \geq 5$
Aaronson and Ambainis [188] (2003)	$O(\sqrt{N}\log^{3/2} N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$
Childs and Goldstone [189] (CTQW 2003)	$O(N)$	$O(N^{5/6})$	$O(\sqrt{N}\log N)$	$O(\sqrt{N})$
Childs and Goldstone [190] (CTQW 2004)	$O(\sqrt{N}\log N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$
Ambainis et al. [179] (DTQW 2004)	$O(\sqrt{N}\log N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$
Tulsi [183] (2008)	$O(\sqrt{N\log N})$	-	-	-
Magniez et al. [85] (2008)	$O(\sqrt{N\log N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$
Patel and Rahaman [191, 192] (2010)	$O(\sqrt{N\log N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$

Table 4.1: Summary of runtimes of quantum search algorithms in various dimensions.

algorithm, this time based on the continuous time quantum walk. They showed a runtime of $O(N)$ for $D = 2$, $O(N^{5/6})$ for $D = 3$, $O(\sqrt{N}\log N)$ for $D = 4$ and $O(\sqrt{N})$ for $D \geq 5$. This algorithm is not as efficient as the one introduced in [188], but does represent the first quantum walk search algorithm defined in continuous time. Shortly after this work, Ambainis et al. [179] gave a discrete time quantum walk search algorithm, improving on the original work of Shenvi et al. [50]. Childs and Goldstone [190] later improved their continuous time algorithm by using the Dirac Hamiltonian and hence an additional degree of freedom which can be thought of as adding a coin to the continuous time quantum walk. This approach was able to match that of the discrete time quantum walk search algorithm of Ambainis et al. [179]. These results are summarised in table 4.1. Up to this point, it remained an important open question as to whether the full quadratic speedup could be achieved in two spatial dimensions.

It took several years for any further improvements to be found in two spatial dimensions. Tulsi [183] then managed to improve the run time for $D = 2$ by a $\sqrt{\log N}$ factor to $O(\sqrt{N\log N})$ using a modified version of the algorithm with ancilla qubits. In the previous cases, the probability of the marked state scaled logarithmically with the size of the data set, $O(1/\log_2 N)$. In his work, Tulsi is able to control this probability using the ancilla qubits to give a constant scaling of the probabil-

ity at the marked state, $O(1)$, thus removing the need for the $\sqrt{\log N}$ amplitude amplification steps. During the years prior to the work by Tulsi, several advances were made in establishing a theory of quantum walk search algorithms. This was pioneered by Szegedy [88] who was able to introduce a method to quantise classical Markov chains (classical random walks on graphs) based on the previous work of Ambainis [67]. This framework is similar to other work by Ambainis et al. [179] and both have been used to develop algorithms which give complexity gains compared to the basic Grover search [68, 93, 193]. Building on all these approaches, Magniez et al. [194] developed a quantum walk search algorithm for any quantum walk based on a reversible, ergodic (a stationary distribution can be found) classical Markov chain. This extends previous work as the algorithm is applicable to a much larger class of Markov chains. It also combines previous ideas into one coherent theory of quantum walk search algorithms. Following this work, Magniez et al. [85] gave a similar theory for the hitting times of quantum walks. They prove that, given a reversible, ergodic classical random walk, the hitting time of the equivalent quantum walk is quadratically faster than the classical case. In addition, they actually prove this speedup is tight for a large class of these quantum walks where the unitary operation is a reflection. It is well known that the hitting time of a classical random walk on a 2D lattice is $O(N \log N)$. Therefore, the equivalent quantum walk hitting time would be $O(\sqrt{N \log N})$ which then matches the run time of Tulsi [183]. Magniez et al. [85] also show they can find the probability of the marked state in a constant fashion, thus extending the result of Tulsi [183] to the larger class of quantum walks which are based on reversible, ergodic Markov chains. In fact, this result has recently been tightened further by Krovi et al. [195], showing that the classical Markov chain, which forms the basis of the quantum walk, need only be reversible. These results tend to indicate that it is unlikely the additional $\sqrt{\log N}$ factor in the run time of the algorithm in two spatial dimensions can be removed. Other recent work shows similar results, including Marquezino et al. [196] who show the mixing time of a quantum walk on a two dimensional toroidal lattice is also $O(\sqrt{N \log N})$.

Additionally, a different approach to the searching problem, using the staggered lattice fermion formalism, has been put forward by Patel and Rahaman [191, 192] to give the same run time. In related work, Hein and Tanner [197] give a detailed analysis of the search algorithm on d-dimensional lattices in terms of the level dynamics near an avoided crossing. They find the same additional $\sqrt{\log N}$ factor in the run time of the algorithm. They also give analytical expressions for the prefactors to the basic scaling of both the time to find the marked state and also the maximum probability the marked state reaches. All of these results lend further weight that the two dimensional case is the critical dimension and it is unlikely that the full quadratic speedup is possible.

The quantum walk search algorithm has provided the basis for various other applications. It has previously been used as a subroutine in the algorithms of Ambainis [67] and Magniez et al. [68]. Hillery et al. [94] have also shown an algorithm based on the quantum walk search algorithm which is able to find a marked subgraph of the complete graph. More recently, it has been identified by Hein and Tanner [198] that the search algorithm can be used to allow wave communication across d-dimensional lattices. They prove that by starting the walker at a marked state as opposed to a uniform superposition, a signal can be transmitted across the lattice to a receiving marked state, the location of which does not have to be known. In fact, they also show that if the receiver moves position, the signal follows and communication is still possibly. The speed of propagation of the signal is limited to the same scaling as the quantum walk search algorithm. This provides another application of the quantum walk search algorithm.

In the next few chapters, we investigate the factors which affect the quantum walk search algorithm. In the next chapter, we show how the algorithm fails on the infinite line and the choices of boundary conditions possible. We then extend this to show other structures with non periodic boundary conditions including the two and three dimensional lattices and certain fractal structures. In Chapter 6, we investigate the dependence of the algorithm on the spatial dimension by interpolating between

structures of differing dimension. Chapters 7 and 8 highlight secondary dependencies on the connectivity and the regularity of the structure being search respectively.

Chapter 5

Quantum walk search algorithm on non-periodic structures

5.1 Introduction

In this chapter, we show structures on which the quantum walk search algorithm fails or is less efficient than on d-dimensional lattices or the hypercube. We show that just because a structure is higher dimensionally, it doesn't necessarily mean that the quantum walk search algorithm can perform efficiently on it. We start with the basic 1D line and show that the algorithm fails, which can be explained via simple arguments. We show here how changing the coin and the boundary conditions affects the search. We then study other lattices with non-periodic boundary conditions: the Bethe lattice (or Cayley tree), fractal structures, specifically the Sierpinski triangle and carpet, and finally the basic two and three dimensional lattices with fixed boundary conditions. We note that during the completion of this work, related results have been presented by Agliari et al. [199, 200] and also Berry and Wang [201]. The results presented by these authors confirm, and extend, the results we show in our work on searching on fractal structures.

5.2 Quantum walk search on the line

We now examine how the quantum walk search algorithm behaves for data arranged on a line. Szegedy [88] previously proved that a quantum walk search approach can only find a marked state in time $O(N)$ with probability $1/N$. We confirm this numerically considering both a line segment, and a loop (cycle) where periodic boundary conditions are applied at the ends. The loop is less physical (for most tape storage, the ends are far apart from each other), but allows us to investigate the behaviour of the algorithm without the edge effects the ends introduce. For the line segment, we use a reflecting boundary condition. This means we have to use a different coin at the edges, which has the effect of introducing two spurious marked states.

Since symmetry is important in the quantum walk search algorithm in higher dimensions, we also investigate a more symmetric version of the Hadamard operator,

$$H_\delta^{(\mathrm{sym})} = \begin{pmatrix} \sqrt{\delta} & i\sqrt{1-\delta} \\ i\sqrt{1-\delta} & \sqrt{\delta} \end{pmatrix}. \tag{5.1}$$

For $\delta = 0.5$, this reduces to

$$H_c = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}. \tag{5.2}$$

We vary δ from zero to one, to see how this affects the performance. The initial state for the quantum walk algorithm on the line needs to match the symmetry of the chosen coin operator. For the Hadamard, we use

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^{N} \frac{1}{\sqrt{2}} (|x,0\rangle + i|x,1\rangle), \tag{5.3}$$

and for the symmetric coin operator we use

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^{N} \frac{1}{\sqrt{2}} (|x,0\rangle + |x,1\rangle), \tag{5.4}$$

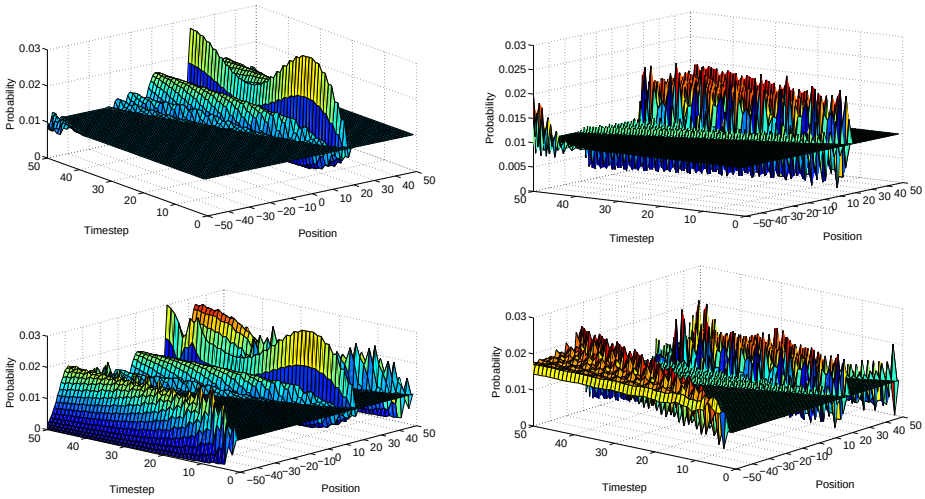

Figure 5.1: Probability distribution of the quantum walk search of $N = 101$ items arranged on a line, after 50 time steps with marked state at position 20, for a symmetric coin and periodic boundary conditions (top left), Hadamard and periodic (top right), symmetric and reflecting (bottom left), and Hadamard and reflecting (bottom right).

where N is the total number of vertices.

First we contrast the symmetric coin operator, eq. (5.2) with the standard Hadamard operator, eq. (2.15). In fig. 5.1 we see the contrast between the two possible boundary conditions (periodic and reflecting). We show each condition with both the coin operators described previously, eqs. (2.15) and (5.2), for all dynamics (negating the phase for the marked state operator). We find that the symmetric coin operator gives a more smoothly varying probability distribution about the marked state and we can see oscillations in the probability of finding the marked state. Superficially, these results look similar to the square lattice. However, the square lattice has a peak probability at the marked state of around 0.3 for a 10×10 square lattice with $N = 100$. On a line with $N = 101$, the peak in the probability is only around 0.028, This is not significantly larger than the uniform distribution, which has a probability of 0.01 for any site. The Hadamard coin operator varies over a period of around seven time steps, regardless of the other parameters, and shows slightly higher probability spreading out from the marked state in two soliton-like waves. This supports the case for symmetry being important for quantum walk

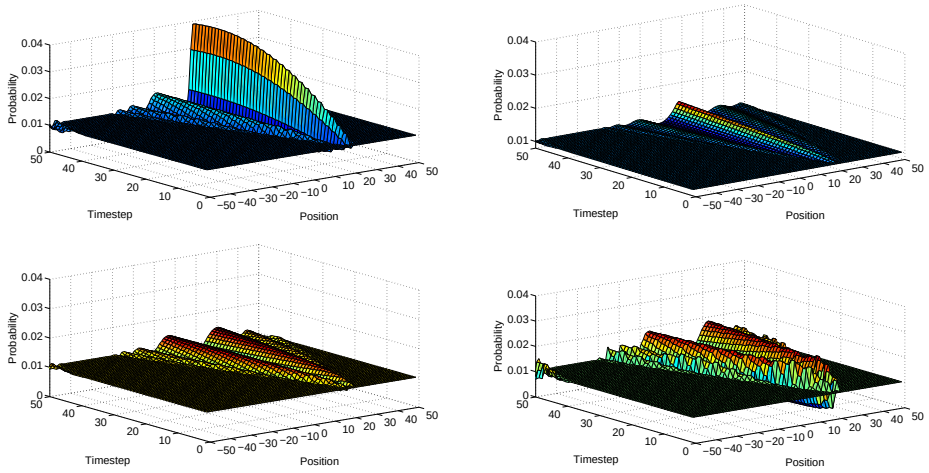

Figure 5.2: Probability distribution of the quantum walk search of $N = 101$ items arranged on a line, after 50 time steps with marked state at position 20, for a symmetric coin and periodic boundary conditions with the marked coin operator given by eq. (5.1) with $\delta = 0.15$ (top left), $\delta = 0.45$ (top right), $\delta = 0.65$, (bottom left) and $\delta = 1$ (bottom right).

searching. Reflecting boundary conditions also produce spreading soliton-like waves from the boundaries for both symmetric and Hadamard coins.

Concentrating on the case of a symmetric coin operator and periodic boundary conditions, we now consider what happens when δ is varied. For δ increasing from zero to a half, the period of the oscillations increases towards infinity as $\delta \to 0.5$, the value for which the marked coin operator becomes the same as the unmarked coin operator and the distribution remains uniform. Increasing δ above 0.5, we find that instead of finding the marked state, in effect it "un-finds" it with the probability of being in the marked state decreasing below the uniform distribution. This is due to the fact the coin is now biased in the wrong direction and so is giving away more and more probability instead of retaining it. Figure 5.2 shows the variation in probability distribution with δ, compare with $\delta = 0$ from fig. 5.1 (top left). For the symmetric quantum walk with periodic boundary conditions, evolving for longer times with $\delta \simeq 0.45$ eventually results in a peak in the probability of the marked state approaching $2\pi/N$, but only after around $5N$ or more time steps, see fig. 5.3. This is obviously not useful, either in the size of the peak probability, which we would like to scale with at least $\log N$ as it does for the search in two dimensions,

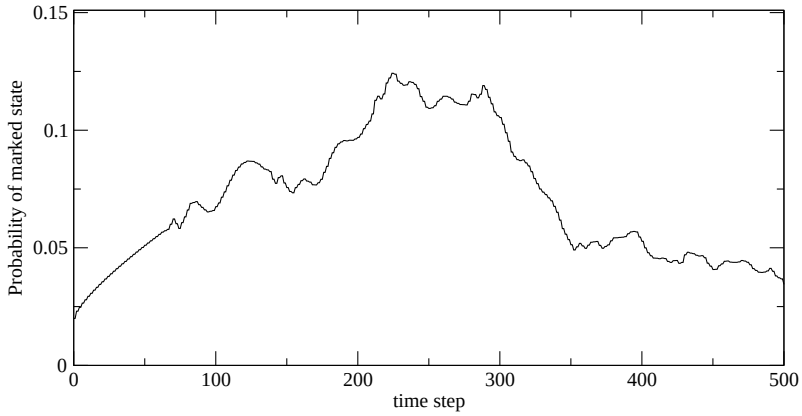

Figure 5.3: Probability of the marked state for each time step for a line of $N = 50$ sites with periodic boundary conditions and $\delta = 0.45$, run for 500 time steps.

nor with the number of time steps, which far exceeds the classical worst case of N.

The quantum walk search algorithm used for data on a line is thus completely ineffective for the parameters we have considered: it does not find the marked state with significant probability even when run for as long as the worst case classical time of N steps for N items. Of course, we can easily specify a quantum version of the classical algorithm that does find the marked state in N steps. For example, start in the state $|0, 1\rangle$ and use the identity as the coin operator everywhere except the marked state. This causes the walk to hop deterministically along the line. At the marked state, use σ_x for the coin operator. This will flip the coin from $|1\rangle$ to $|0\rangle$ and thus reverse the direction of the walker. If the position of the walker is measured after N steps, the current location allows you to work out where it turned round, and thus locate the marked state. This method uses only a single measurement. If you allow measurements at every step, then of course you can immediately find out if the walker has arrived at the marked state by testing the state of the coin.

A classical random walk searching algorithm on a line, with equal probability of moving left and right, would take $O(N^2)$ to find a marked item. Using the techniques by Magniez et al. [85] and Krovi et al. [195], a quantum analogue of this classical walk can be defined which would give a quadratic speed up to $O(N)$. However, as shown by Szegedy [88], the value of this maximum probability at the marked state would be $1/N$ thus rendering the algorithm ineffective.

5.3 Bethe lattice

The Bethe lattice (or Cayley tree) is a general structure which can have any fixed degree at all of its vertices. Its connectivity is somewhat different to the previous examples in that there are no loops in it, giving a tree-like structure with 'branches' stemming from a vertex indefinitely. We work with a finite sized segment based on around a central vertex. A piece with vertices of degree three is shown in fig. 5.4. It can be seen from this example that the vertices on the branches form 'shells' around the central vertex. The number of vertices in each shell is,

$$N_s = d(d-1)^{s-1} \quad \text{where } s > 0, \tag{5.5}$$

where N_s is the number of vertices in shell s and d is the degree of the vertices. The coin used in the degree three case of the Bethe lattice is the Grover coin of dimension three,

$$G^{(3)} = \frac{1}{3}\begin{pmatrix} -1 & 2 & 2 \\ 2 & -1 & 2 \\ 2 & 2 & -1 \end{pmatrix}, \tag{5.6}$$

and marked state operator is just this same coin inverted as with the other structures,

$$G_m^{(3)} = \frac{1}{3}\begin{pmatrix} 1 & -2 & -2 \\ -2 & 1 & -2 \\ -2 & -2 & 1 \end{pmatrix}, \tag{5.7}$$

We can't impose periodic boundary conditions on the Bethe lattice without creating loops in the structure. Instead, at the 'ends' of the branches, we reflect the amplitude back upon itself. This is accomplished using the Grover coin at its limit, $d = 2$, which is the σ_x operation as shown in eq. (3.11).

Figure 5.5 shows how the maximum probability of the marked state varies with both the size of the dataset and the position of the marked state, that is which shell the marked state is present within. We find that if the marked state was present

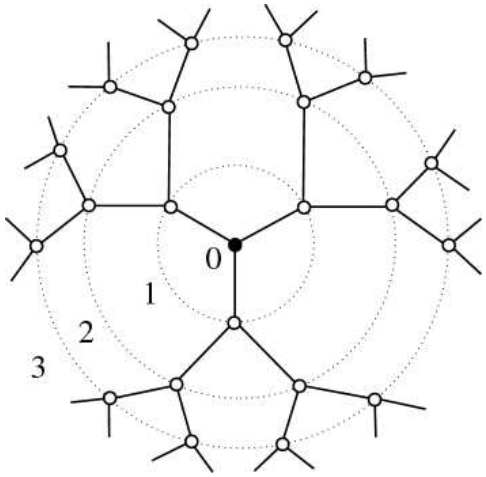

Figure 5.4: A segment of a Bethe lattice with fixed degree $d = 3$. Three shells are shown here emanating from the central vertex.

either at the central vertex or in the first shell then the algorithm is actually almost optimal in scaling, close to $O(\sqrt{N})$. The probability at this point is also high enough to allow the marked state to be distinguished from the remaining superposition. In fact, as the probability scales as $O(1)$, this is the total complexity and so would be optimal. This is an unrealistic scenario though and so the Bethe lattice would never, in general, be efficient for the search algorithm. Although we have only shown the degree three Bethe lattice here, we have also studied the Bethe lattice of degree four, and it performs in a similar fashion. In contrast to the 2D Cartesian lattice, the position of the marked state in the Bethe lattice (which shell it occurs in) strongly affects the efficiency of the search algorithm. As the marked state moves away from the central vertex, the probability of the marked state is significantly lower than if the marked state is the central vertex. In fact, by the time the marked state is in the fourth shell, the probability of the marked state does not get much higher than the value of the initial coefficient. At this level, there is no way to distinguish the marked state from any others. This is similar to the search on a line here where we only see a small increase in probability at the marked state. This behaviour is caused by the connectivity of the structure itself. As there are no cycles in the Bethe lattice, the probability is split between the 'branches' of the structure and so only a portion of the probability can converge on the marked state. This localisation of

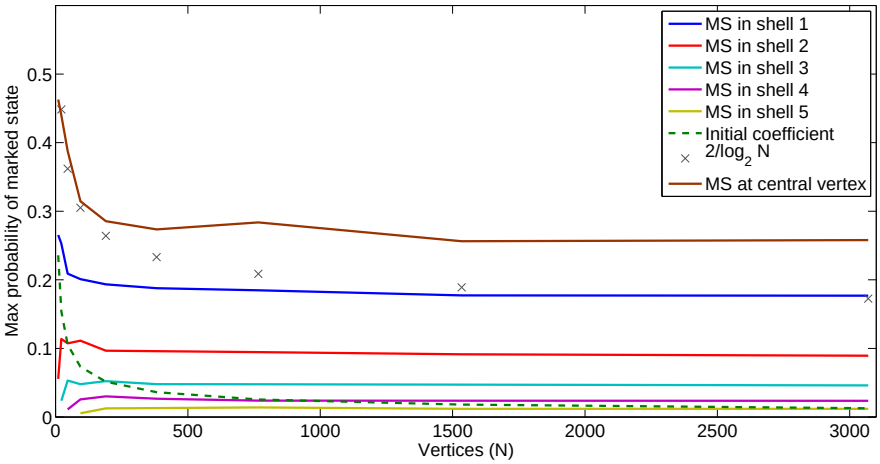

Figure 5.5: Maximum of the first peak in the probability of being at the marked state for different sized data sets on a Bethe lattice of degree three, plotted against N (solid), for varying positions of the marked state (MS). Also shown is the initial coefficient (dashes) and $2/\log_2 N$ (crosses) for comparison.

probability in portions of the structure away from the marked state means the walker will be unable to coalesce at the marked state with any significant probability. As this maximum probability scales in a constant fashion with the number of vertices, $O(1)$, it makes no difference how long the search algorithm is run for.

The time to find the marked state also exhibits unusual characteristics. We see in fig. 5.6 that as we move further from the central vertex, the timestep to the first significant peak actually reduces. However, as already mentioned, the probability at this point is so low that the marked state could never be distinguished. As the marked state only accumulates a small portion of the probability, the time to get to this amount would get faster, hence the decrease in time to 'find' the marked state. We note that the 'kinks' in this graph are most likely due to finite size effects.

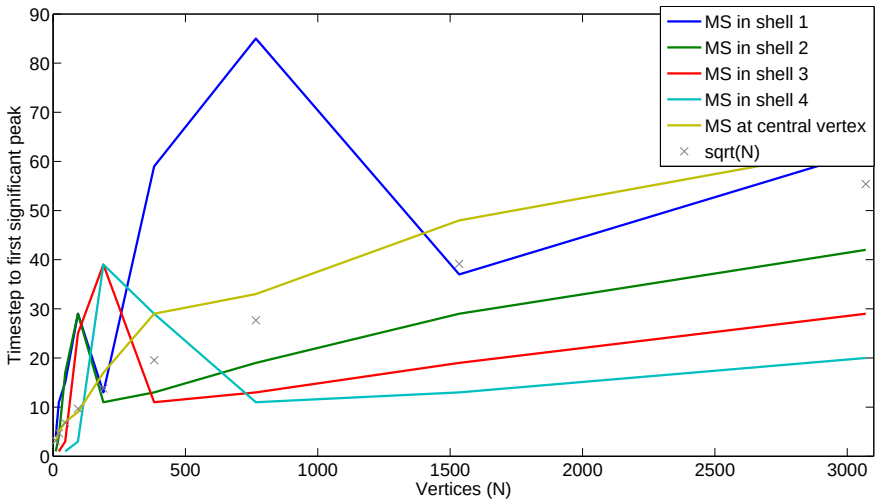

Figure 5.6: Time step at which the first significant peak in the probability of being at the marked state occurs for different sized data sets plotted against N, for a Bethe lattice of degree three, for varying positions of the marked state (MS). Also shown is $\sqrt{N}$ (crosses) for comparison.

5.4 Fractal structures

We now study the search algorithm on two fractal structures, the Sierpinski triangle and carpet. These fractals, due to their structure, have a non-integer value of spatial dimension. This is due to the fact they have a self similarity at different scales. Our aim here was to use fractal structures as a way to interpolate between spatial dimensions. However, in these cases, as with the Bethe lattice, there is no way to apply periodic boundary conditions to the edges of the structures. We investigate if this is an appropriate way to interpolate between spatial dimensions and also if the search algorithm is adversely affected by the reflection effects introduced by the edges and 'holes' in the structures.

5.4.1 Sierpinski triangle

The Sierpinski triangle, shown in fig. 5.7, is a self similar structure created by reducing the initial generating triangle by 1/2 and repeating the smaller structure three times within the previous triangle. We call the initial triangle generation $g = 1$ and each subsequent generation is a higher integer value. It has a spatial dimension of $D = 1.585$.

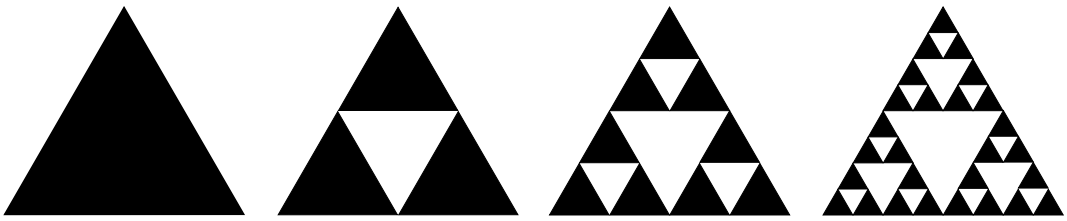

Figure 5.7: Representation of the Sierpinski triangle. We show the first four generations of construction.

We spread the initial state of the walker equally across all the vertices and edges of the structure. We only study the triangle up to generation 7, as generating the graph to represent the fractal became computationally intractable after this. Obviously, in this type of graph there will be vertices of varying degree. At these, we use the Grover coin, eq. (2.20), of the correct dimension at each vertex. In the standard quantum walk search algorithm, the position of the marked state has no effect on the algorithm. In the case of fixed boundary conditions, this is not the case, as already seen in the case of the Bethe lattice, as the edges introduce additional reflection effects which hamper the ability of the walker to coalesce on the marked state.

We firstly show how the position of the marked state effects the search algorithm by considering a single generation of the fractal structure, generation 5 (123 vertices). In figs. 5.8 and 5.9, we show a representation of the Sierpinski triangle as an undirected graph. In fig. 5.8, we show the maximum probability of the marked state, for each different position the marked state can take, as the label on each vertex. Figure 5.9 is the same representation, but this time showing the time to find the marked state for each vertex. We see that the algorithm is more efficient when

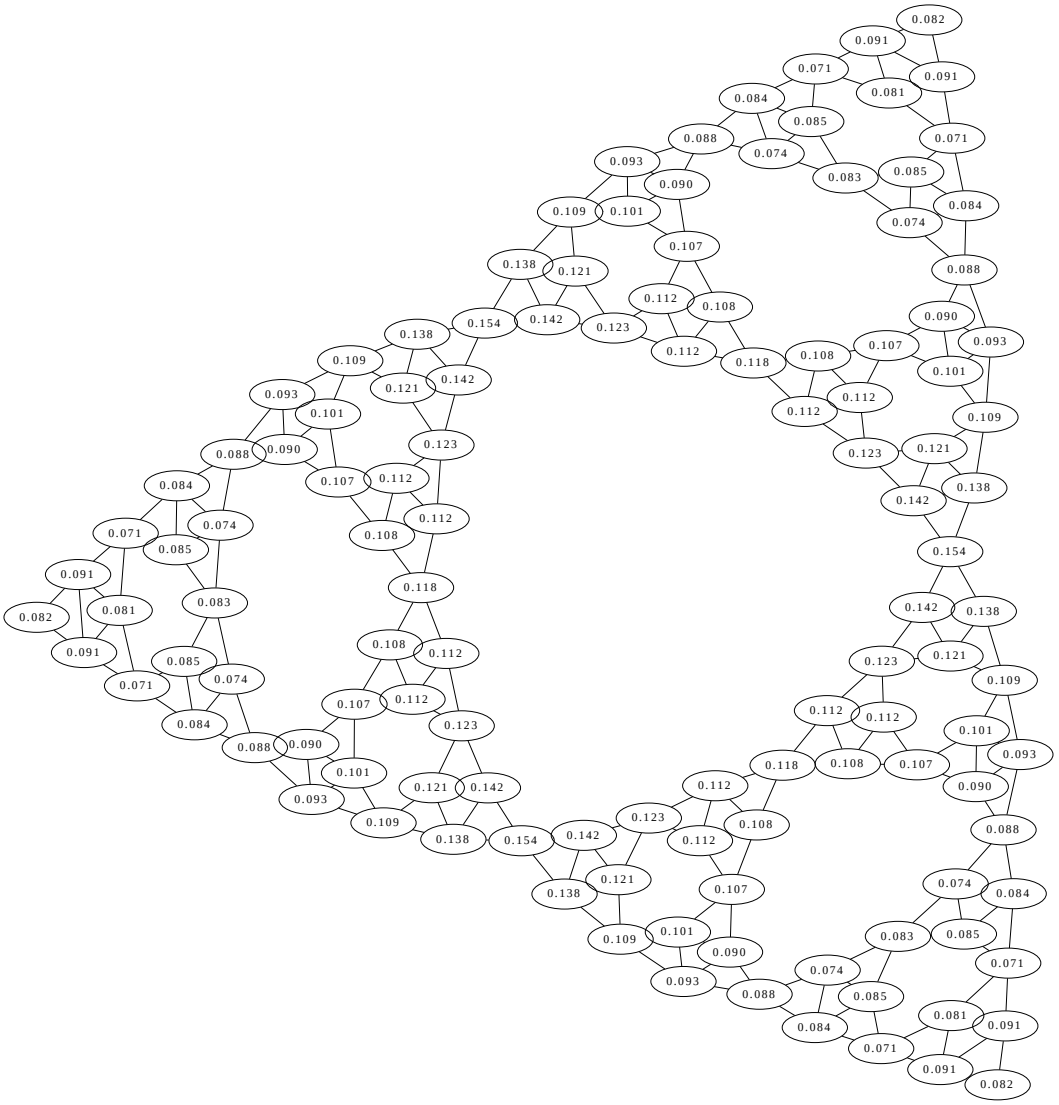

Figure 5.8: Representation of the Sierpinski triangle as an undirected graph for generation 5. We show the maximum probability of the marked state for every vertex.

the marked state is in certain parts of the fractal structure than in others. This is similar in behaviour to the Bethe lattice, but the difference between sites is not as pronounced. We show the same information as in the graphical representations in fig. 5.10. These plots also show the basic scaling for the maximum probability of the marked state and the time to find it for a basic lattice in two spatial dimensions. We can see that when the marked state is present at almost any vertex, the maximum probability the marked state is less than that of the basic scaling of the

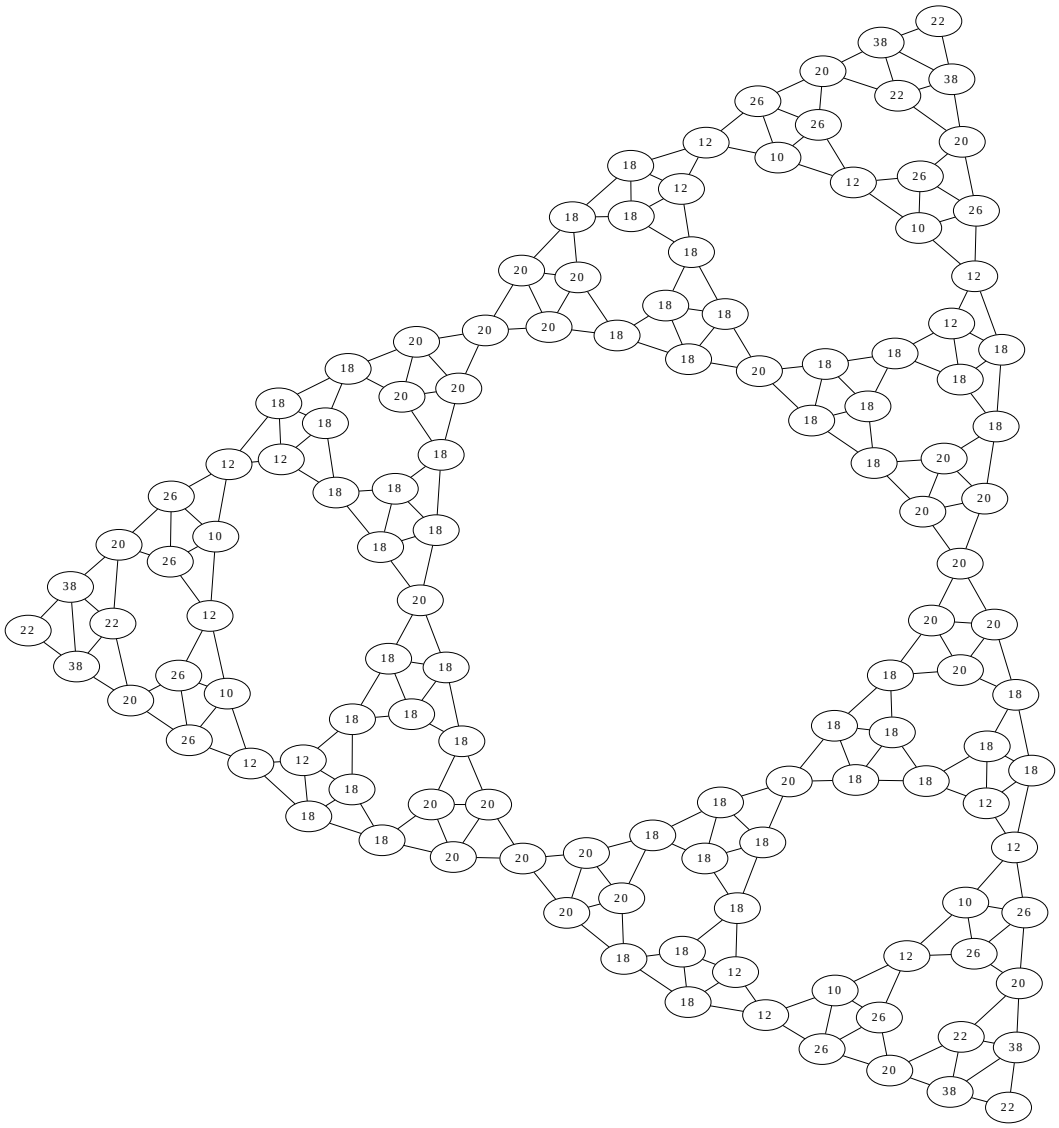

Figure 5.9: Representation of the Sierpinski triangle as an undirected graph for generation 5. We show the time to find the maximum probability of the marked state for every vertex.

2D lattice. This clearly shows that the algorithm is less efficient than in the basic two dimensional case, with some vertices not able to retain much probability at all. In these cases, there would be no way to distinguish the marked state from the surrounding probability distribution, hence the algorithm would fail. Similarly, the time to find the marked state is higher, in general, than the basic 2D scaling of the algorithm, again making the algorithm less efficient. Finally, in fig. 5.11, we show how the maximum probability of the marked state and the time to find it vary, for

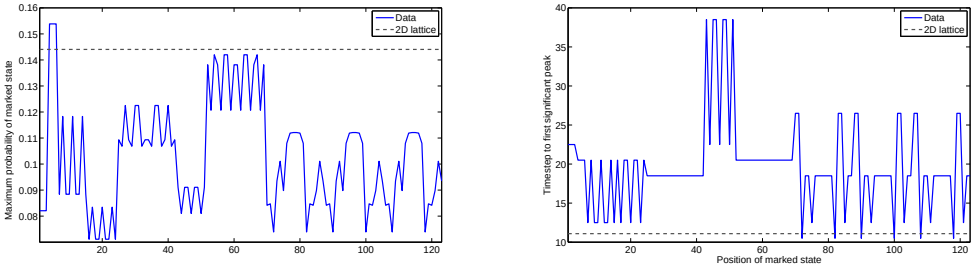

Figure 5.10: LHS: Maximum probability of the marked state for the Sierpinski triangle of generation 5. Also shown is the scaling of the maximum probability of the marked state for the basic 2D lattice. RHS: Time to find the maximum probability of the marked state for the Sierpinski triangle of generation 5. Also shown is the scaling of the time to find the marked state for the basic 2D lattice.

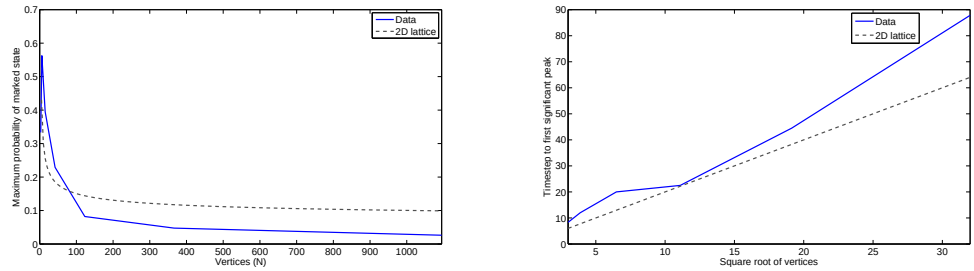

Figure 5.11: LHS: Plot to show how the maximum probability of the marked state varies with the size of the dataset for the Sierpinski triangle. RHS: Plot to show how the time to find the maximum probability of the marked state varies with the size of the dataset for the Sierpinski triangle.

a specific vertex (corner), with the size of the dataset (generation of the fractal). We also show the scaling for the basic 2D lattice for both the maximum probability and the time to find it. In the case of the maximum probability, the scaling for the Sierpinski triangle is lower than the basic 2D lattice and is also dropping at a rate faster than the logarithmic 2D scaling. Although hard to tell what will happen for larger values of N, this indicates that the fractal would be unsuitable for use as the substrate for the quantum walk search algorithm. In a similar fashion, the time to find the marked state seems to be increasing at a rate faster than the basic 2D lattice, again showing the algorithm fails on the fractal structure.

5.4.2 Sierpinski carpet

The Sierpinski carpet, shown in fig. 5.12, is another fractal structure, this time with spatial dimension $D = 1.898$. It is generated by splitting an initial square into nine segments and removing the central piece. This process is then iterated per generation of construction.

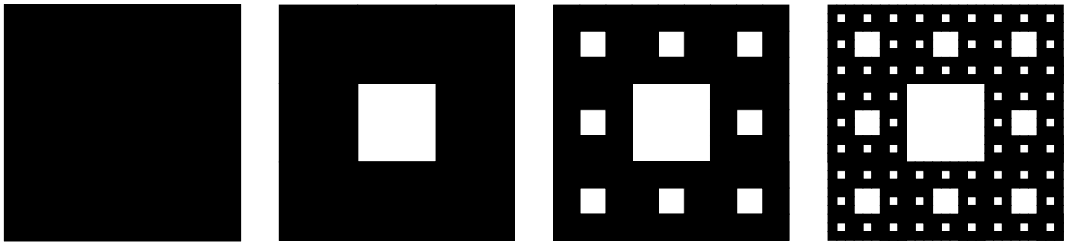

Figure 5.12: Representation of the Sierpinski carpet. We show the first four generations of construction.

The walker is started in the same state as in the case of the Sierpinski triangle, spread across all possible vertices and edges. We show our results on the Sierpinski carpet in the same way as the triangle. Firstly, the maximum probability and the time to find this probability are shown in figs. 5.13 and 5.14 for a single generation of the fractal, generation 3 (96 vertices). We see that, in general, as the marked state is moved closer to the 'centre' of the structure, the maximum probability of the marked state increases. In the same sense, the time to find this maximum probability reduces. Both of these indicate the algorithm becomes more efficient when the marked state is most central. This is confirmed in fig. 5.15 where we see that for many of the vertices, the maximum probability of the marked state is higher or very similar to the 2D lattice scaling. However, the time to find the marked state is significantly higher than in the basic 2D case.

We finally show, in fig. 5.16, the scaling of the algorithm with the size of the dataset. The scaling of the carpet, in contrast to the triangle, is higher than the basic 2D lattice. However, it seems to be dropping at a rate faster than the basic 2D lattice. It is hard to tell if this is truly the case from the limited generations we are able to simulate, but it does seem reasonable as when the lattice becomes

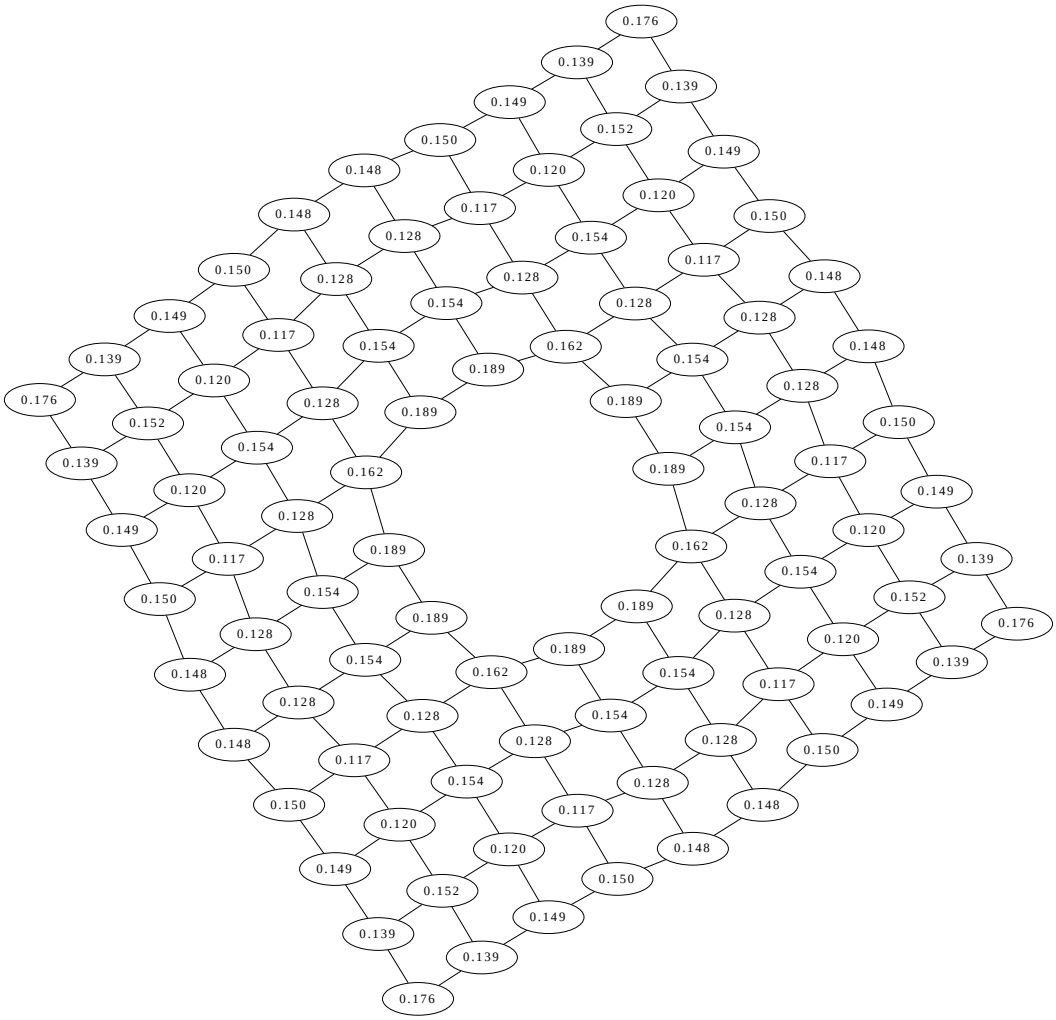

Figure 5.13: Representation of the Sierpinski carpet as an undirected graph for generation 3. We show the maximum probability of the marked state for every vertex.

bigger, the central 'hole' becomes bigger, thus limiting the possible centrality of the marked state. The time to find the marked state seems to scale in a similar way to the 2D lattice but with just a higher prefactor. These factors indicate that, although less efficient as the basic 2D lattice, the Sierpnski carpet may possibly be used for the quantum walk search algorithm for relatively small N. However, as N increases and the 'holes' in the structure have more effect on the walker, the algorithm will start to fail as the probability of the marked state will become too low. However, the Sierpinksi carpet is certainly a more efficient data structure than the Sierpinski triangle, due to the fact it is, in general, more symmetric.

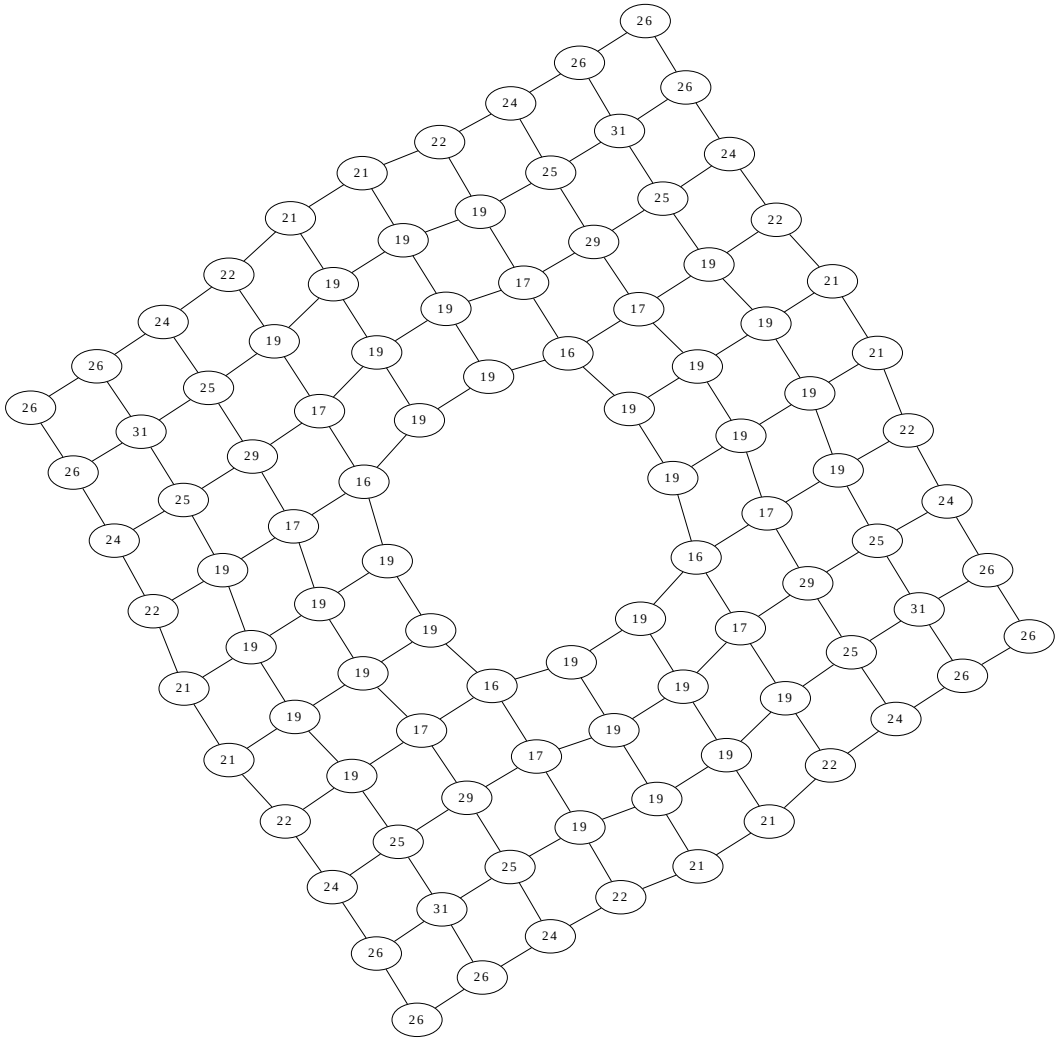

Figure 5.14: Representation of the Sierpinski carpet as an undirected graph for generation 3. We show the time to find the maximum probability of the marked state for every vertex.

5.5 Non-periodic regular lattices

We now consider the case of regular lattices with fixed boundary conditions. We are interested in the effect the edges of the structure have on the efficiency of the search algorithm. The walk is distributed across the structure as a uniform superposition over all the possible vertices and edges, just as in the periodic case. At the corners, or edges, we use the Grover coin, eq. (2.20), of the correct dimension, i.e. two at the corners and three at any edge. As with the fractal structures, the position of the marked state also affects the algorithm in this case. We study both two and

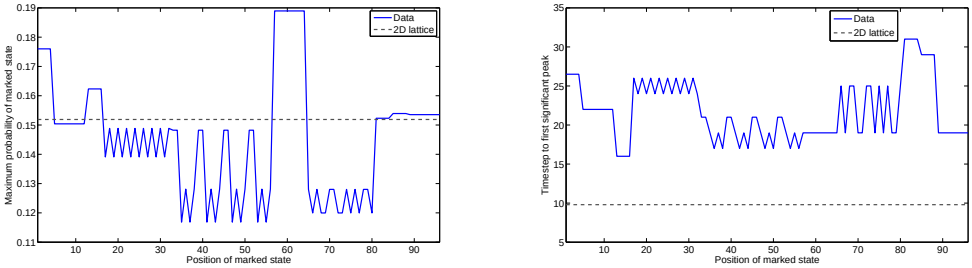

Figure 5.15: LHS: Maximum probability of the marked state for the Sierpinski carpet of generation 3. Also shown is the scaling of the maximum probability of the marked state for the basic 2D lattice. RHS: Time to find the maximum probability of the marked state for the Sierpinski carpet of generation 3. Also shown is the scaling of the time to find the marked state for the basic 2D lattice.

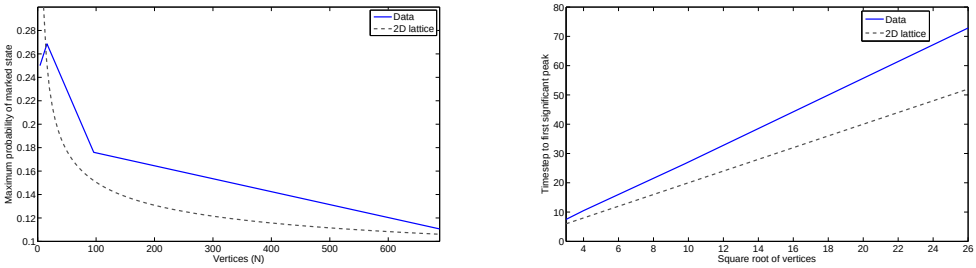

Figure 5.16: LHS: Plot to show how the maximum probability of the marked state varies with the size of the dataset for the Sierpinski carpet. RHS: Plot to show how the time to find the maximum probability of the marked state varies with the size of the dataset for the Sierpinski carpet.

three dimensional lattices with non-periodic boundary conditions to see how both the boundary conditions and the position of the marked state affect the algorithmic efficiency.

5.5.1 Two dimensional lattices

We see in fig. 5.17 how the maximum probability of the marked state varies depending on the location of the marked state for a fixed lattice size (10,000 vertices). We see that, in general, the closer the marked state is to the centre of the structure, the higher the maximum probability of the marked state. This is due to the fact that the effects due to reflection from the edges of the structure is minimised. A similar plot to show how the time to find the marked state is shown in fig. 5.18. This shows that the closer the marked state is to the centre of the lattice, the quicker the walker

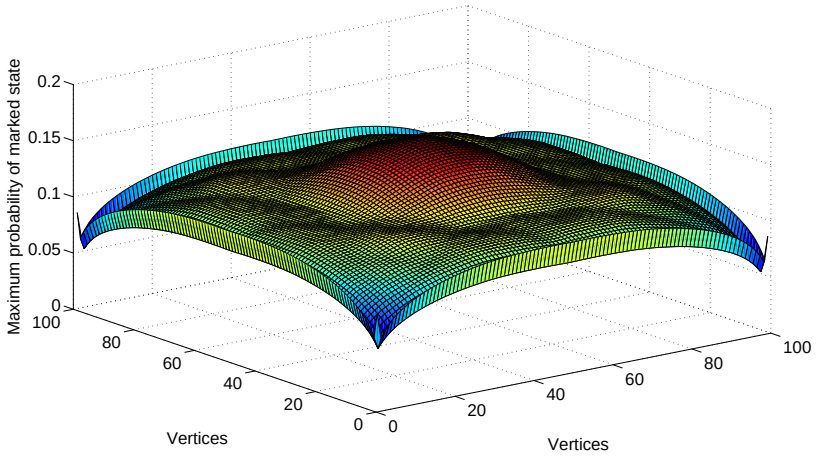

Figure 5.17: Probability distribution of the quantum walk search of $N = 10,000$ items arranged on as a two dimensional Cartesian lattice. We show how the maximum probability of the marked state varies based on the position of the marked state.

is able to coalesce on the marked state. We note from both figures that even though the Grover coin does not treat all directions in the same fashion, the distributions that result are symmetric, i.e. each corner has the same maximum probability and time step to this probability.

We now show the effect of the fixed boundary conditions as the lattice grows in size. Figure 5.19 shows how the maximum probability of the marked state varies with lattice size, for a corner vertex, the most central vertex and an intermediary vertex. We see that the scaling of the algorithm when the marked state is at the most central vertex matches that of the standard search algorithm with periodic boundaries. However, we see when the marked state is moved towards the corners of the lattice, the maximum probability of the marked state drops. The basic scaling remains the same, indicating the algorithm is still viable, but has a lower prefactor as the marked state becomes less central. We see a similar behaviour in the scaling of the time to find the marked state. Again, when the marked state is the most central vertex, the scaling of the algorithm matches that of lattices with periodic boundary conditions. The prefactor to the scaling of the time to find the marked state increases as the marked state is moved towards the corners of the structure.

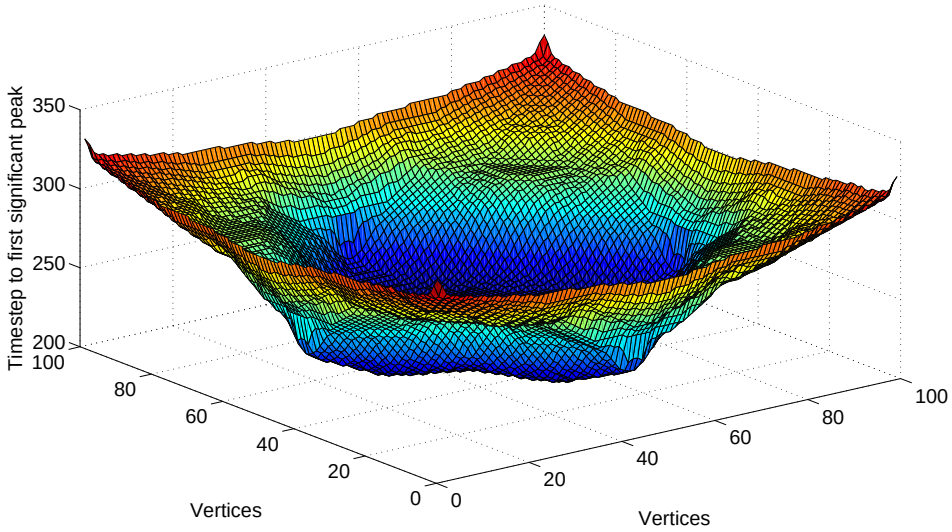

Figure 5.18: Distribution of the time to find the marked state of the quantum walk search of $N = 10,000$ items arranged as a two dimensional Cartesian lattice. We show how the time to find the maximum probability of the marked state varies based on the position of the marked state.

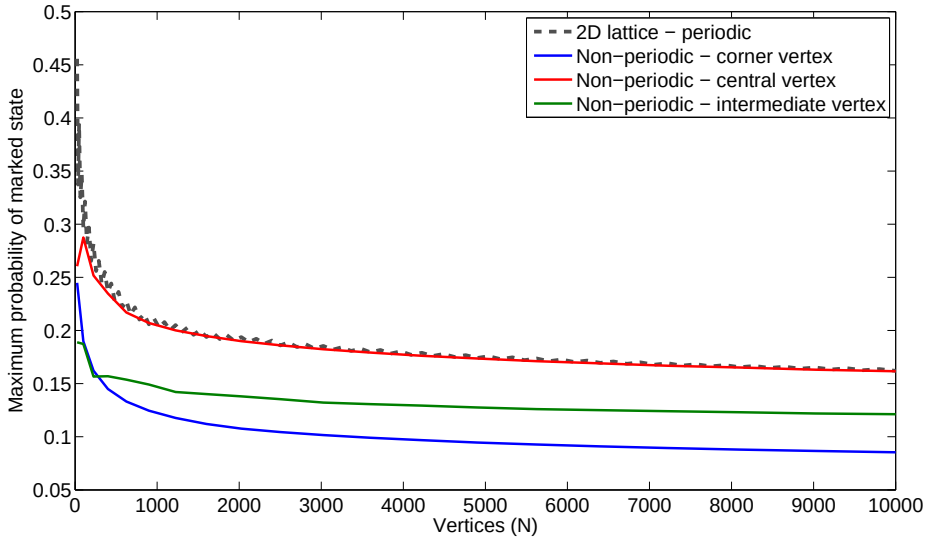

Figure 5.19: Maximum probability of the marked state for varying positions of the marked state on a non-periodic 2D lattice. Also shown is the scaling for a 2D Cartesian lattice with periodic boundary conditions for comparison.

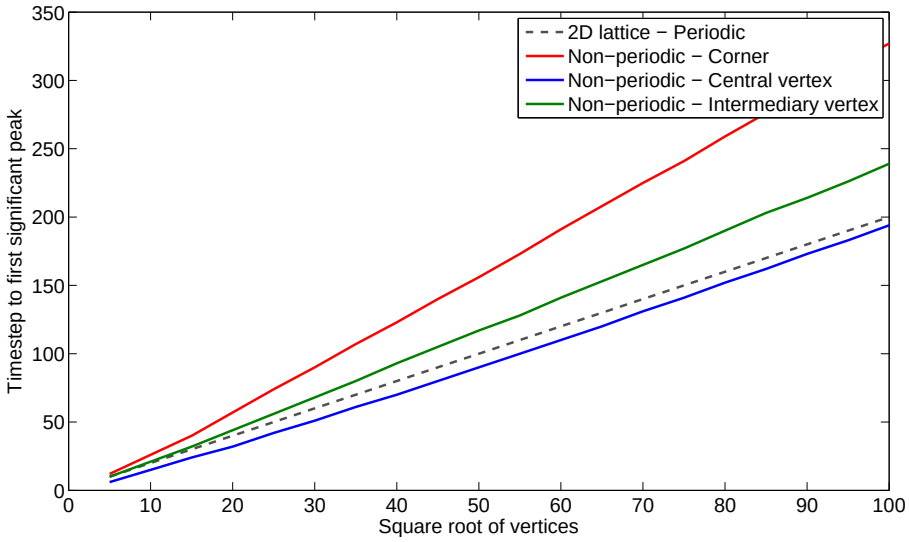

Figure 5.20: Time step at which the first significant peak in the probability of being at the marked state occurs for varying positions of the marked state on a non-periodic 2D lattice. Also shown is the scaling for a 2D Cartesian lattice with periodic boundary conditions for comparison.

We see from all of the above that the lack of periodic boundary conditions does not cause the algorithm to fail. However, as the marked state becomes less central, thus increasing the effects of reflection from the edges of the structure, the efficiency of the algorithm decreases.

5.5.2 Three dimensional lattices

In the case of three dimensions, we see the same behaviour as in the two dimensional case. Figures 5.21 and 5.22 show how the maximum probability of the marked state and the time to find this probability vary with the position of the marked state. Obviously it is harder to show this graphically and so we show 'slices' of the three dimensional lattice. In the case of the maximum probability of the marked state, we show these slices on top of one another, and see that as the slice becomes closer to the centre of the lattice, the maximum probability increases.

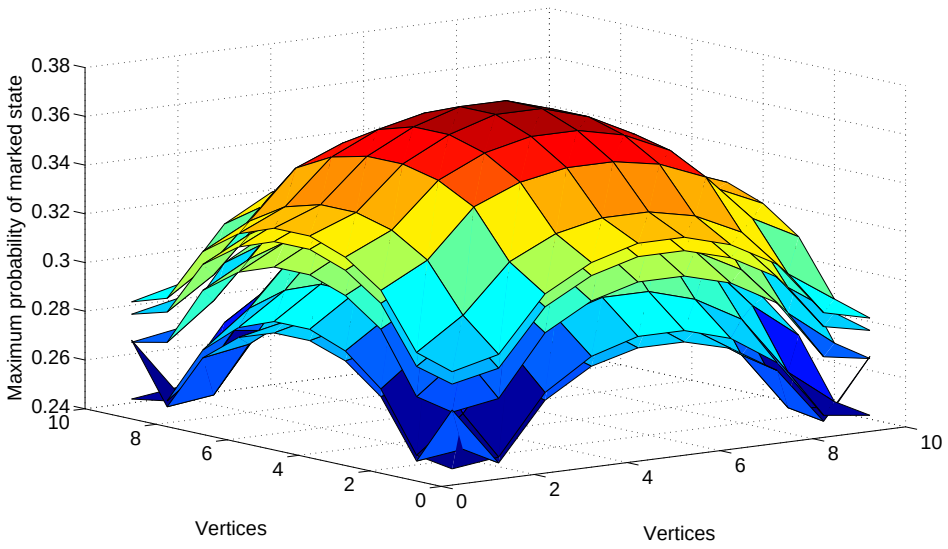

Figure 5.21: Probability distribution of the quantum walk search of $N = 1,000$ items arranged on as a non-periodic three dimensional cubic lattice. We show various 'slices' of the lattice to indicate how the maximum probability of the marked state varies based on the position of the marked state. As the slices become more central, the probability increases and thus the slice is shown on top of the others.

The behaviour of the time to find the marked state is similar. As the slice approaches the centre of the structure, the time to find the marked state reduces and so the slices are shown under one another. However, the differences between the slices is much smaller so we only show the central slice, the edge slice and one in between.

We see the scaling of the algorithm with the size of the lattice in figs. 5.23 and 5.24. It is clear that, just like the two dimensional case, the basic algorithmic scaling of both the maximum probability and the time to find it is the same as in the periodic case. When the marked state is at the most central vertex, the prefactor to the scaling of both almost matches the standard algorithm. As the marked state is moved towards the corners of the lattice, the prefactors for the scaling of the maximum probability and the time to find it, increase and decrease respectively, thus reducing the efficiency of the algorithm as a whole.

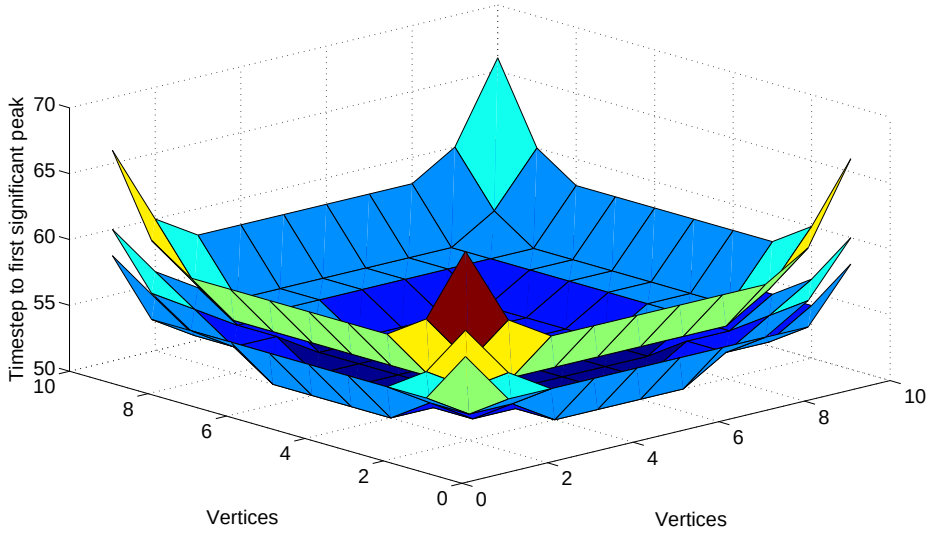

Figure 5.22: Distribution of the time to find the marked state of the quantum walk search of $N = 1,000$ items arranged on as a non-periodic three dimensional cubic lattice. We show various 'slices' of the lattice to indicate how the time to find the maximum probability of the marked state varies based on the position of the marked state. As the slices become more central, the time to find the marked state decreases and thus the slice is shown below the others.

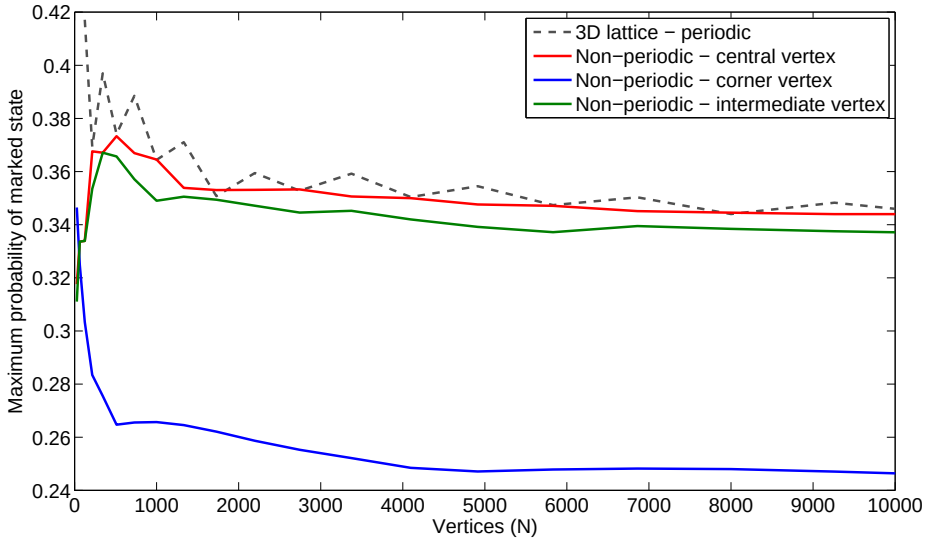

Figure 5.23: Maximum probability of the marked state for varying positions of the marked state on a non-periodic three dimensional cubic lattice. Also shown is the scaling for a three dimensional cubic lattice with periodic boundary conditions for comparison.

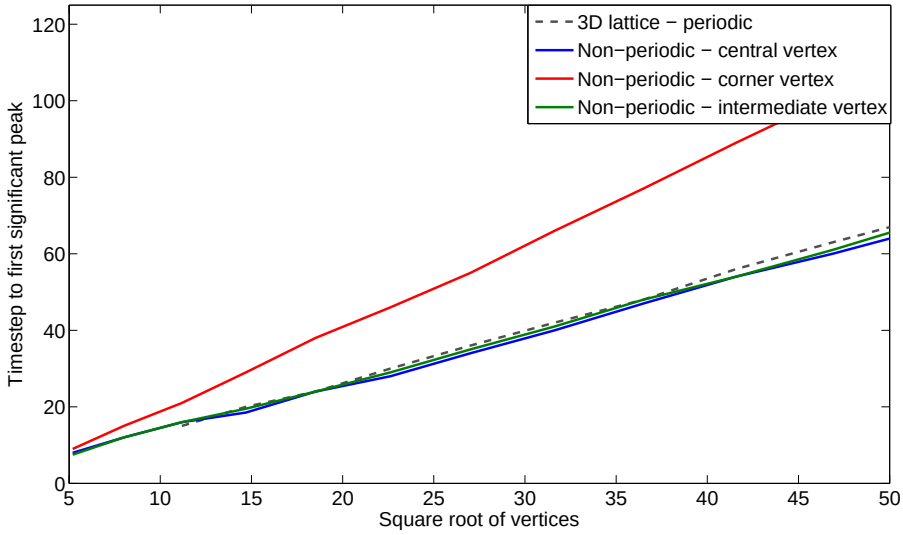

Figure 5.24: Time step at which the first significant peak in the probability of being at the marked state occurs for varying positions of the marked state on a non-periodic three dimensional cubic lattice. Also shown is the scaling for a three dimensional cubic lattice with periodic boundary conditions for comparison.

5.6 Discussion

In this chapter, we have discussed the quantum walk search algorithm on the line and cycle, the Bethe lattice, the Sierpinski carpet and triangle, and finally the basic 2D and 3D lattices but with fixed boundary conditions.

In the case of the line and cycle, we show that although the walker does coalesce on the marked state, the maximum probability of the marked state does not rise much above the initial superposition, $1/N$, thus making it almost impossible to distinguish the marked state. Varying the boundary conditions and the symmetry of the coin used gives a smoother probability distribution indicating that there is also a dependence on symmetry in the algorithm.

The Bethe lattice and the two fractal structures we study, the Sierpinski triangle and carpet, are also inefficient as data structures for the quantum walk search algorithm to search upon. In these cases, due to the boundary conditions, the position of the marked state has an effect on the efficiency of the algorithm. This highlights that, although the quantum walk search algorithm has a strong dependence on the

spatial dimension of the structure being walked upon, this does not necessarily mean that a higher dimensional structure makes the algorithm efficient. In fact, in the case of the Bethe lattice the algorithm fails completely. For non-periodic structures in general, it seems the reflection effects we get from the edges of the structure hamper the ability of the algorithm to coalesce on the marked state. We again note that the symmetry of a structure has an effect on the search algorithm. This is confirmed by the fact that the Sierpinski carpet performed more efficiently than the Sierpinski triangle, mainly due to the fact that it is a more symmetric structure. However, the efficiency does seem to drop off in higher generations of the Sierpinski carpet as the central 'hole' becomes larger and the other 'holes' in the structure have more effect on the walker. From these results, although we highlight some interesting behaviour, it seems like using fractal structures is not a good way to interpolate between structures of differing spatial dimension. Instead, we show two other ways in which we can do this in chapter 6, using a tunnelling operator and also lattices of varying height and depth.

Although these same reflection effects are present in the basic 2D and 3D lattices with fixed boundary conditions, they have less of an effect on the efficiency of the algorithm. This again highlights the need for symmetry in the search algorithm, both in the coin used and also the structure being walked upon. In the case of the basic lattices, the scaling is pretty similar to the scaling for the lattices with periodic boundary conditions. The position of the marked state does still have an impact on the maximum probability of the marked state and the time to find it, but seems to still follow the periodic scaling. Therefore, although the reflection effects do have an adverse effect on the algorithm, they are not as predominant as symmetry effects in the structure being searched.

Chapter 6

Effect of dimensionality on the quantum walk search algorithm

6.1 Introduction

In this chapter, we investigate how the spatial dimension of the database arrangement affects the searching algorithm. We already know, from chapter 4, that the basic scaling of the algorithm is heavily dependent upon the spatial dimension of the structure in question. Both the scaling of the maximum probability of the marked state and the time to find this probability alter for structures of differing spatial dimension. We are interested here in how this basic scaling changes when the spatial dimension is altered. We do this in two ways, firstly by introducing a simple form of tunnelling, which allows us to interpolate between structures of varying spatial dimension, and secondly by using lattices of varying height (1D-2D) and depth (2D-3D).

We introduce a simple form of tunnelling using a tunnelling coin operator in sec. 6.2. We then numerically study how the algorithm performs, in sec. 6.3, as we gradually change the underlying structure from a 1D line to a 3D cubic lattice. In sec. 6.4, we then discuss how we vary the height (2D) and the depth (3D) of the lattices, before giving our results as to how the algorithm performs on these lattices.

We end the chapter by discussing our results for both variations.

The algorithmic efficiency of the quantum walk search algorithm using standard two and three dimensional lattices has been well studied in previous work as summarised in chapter 4. In three spatial dimensions, the algorithm scales as $O(\sqrt{N})$. This comes from two parts: the scaling of the maximum probability of the marked state and also the time to find this maximum probability. In the three dimensional case, the maximum probability of the marked state scales in a constant fashion, $O(1)$, so has no effect on the total algorithmic complexity. The time to find this maximum probability, and hence find the marked state, scales with the size of the cubic lattice as $O(\sqrt{N})$. This makes the total algorithmic complexity of the quantum walk search algorithm in three spatial dimensions $O(\sqrt{N})$. The two dimensional case is more complex with a total complexity of $O(\sqrt{N \log N})$. This is made up from a scaling of $O(1/\log_2 N)$ for the maximum probability of the marked state, and $O(\sqrt{N})$ for the time to find this. As the maximum probability of the marked state scales logarithmically, this must be amplified to a constant value. This has previously been shown to take $\sqrt{\log N}$ steps using amplitude amplification [36]. By combining these additional amplification steps, we obtain the total complexity of $O(\sqrt{N \log N})$. We note here that in previous (analytical) results for the quantum walk search algorithm [179], the time to find the marked state is given as $O(\sqrt{N \log N})$, thus giving a total algorithmic complexity of $O(\sqrt{N} \log N)$. However, our numerical results [2] seem to suggest that the scaling of the time to find the marked state is actually $O(\sqrt{N})$, giving a total complexity of $O(\sqrt{N \log N})$, therefore matching the scaling of Tulsi [183] and Magniez et al. [85].

6.2 Tunnelling operator

We now describe a modified coin operator we will use in the search algorithm to gradually vary the connectivity of the structures studied. We introduce a simple form of tunnelling to allow us to gradually vary the substrate to be walked upon from one form to another. A simple example is changing a 2D Cartesian lattice

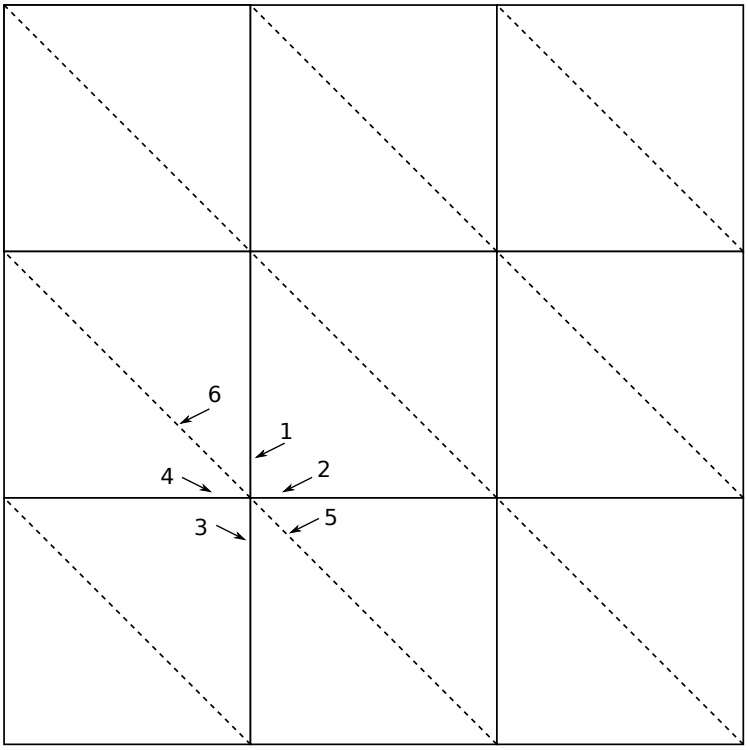

Figure 6.1: An example of how we can change one lattice into another by adding or removing edges. Here we change a square lattice, degree $d = 4$, into a triangular lattice, degree $d = 6$. In this example, $d = 6$ and $t = 2$ in the tunnelling matrix we introduce.

into a triangular lattice by introducing one diagonal link across each square of the lattice, see fig. 6.1.

In order to achieve the quantum walk dynamics we require, we must use a different coin operator. The only condition on this operator is that it must be unitary. As such, we 'design' a new coin operator which incorporates a single tunnelling parameter, c, which will allow us to vary the strength of specific tunnelling edges. We define d to be the degree of the vertex in question as used previously in the Grover coin, eq. (2.20), and t to be the number of tunnelling edges. For a d-dimensional vertex, the first $(d-t)$ edges in our labelling scheme are normal and the last t edges are tunnelling. In fig. 6.1, edges 1-4 are normal, fixed edges creating a 2D lattice and edges 5-6 are tunnelling edges which convert the lattice to a triangular one. The

general matrix for the desired coin operator would be as follows:

$$T_{d,t} = \begin{pmatrix} a & b & b & \dots & b & c & c & c & \dots & c \\ b & a & b & \dots & b & c & c & c & \dots & c \\ . & . & . & \dots & . & . & . & . & \dots & . \\ . & . & . & \dots & . & . & . & . & \dots & . \\ . & . & . & \dots & . & . & . & . & \dots & . \\ b & b & b & \dots & a & c & c & c & \dots & c \\ c & c & c & \dots & c & e & f & f & \dots & f \\ c & c & c & \dots & c & f & e & f & \dots & f \\ . & . & . & \dots & . & . & . & . & \dots & . \\ . & . & . & \dots & . & . & . & . & \dots & . \\ . & . & . & \dots & . & . & . & . & \dots & . \\ c & c & c & \dots & c & f & f & f & \dots & e \end{pmatrix} \tag{6.1}$$

where the number of columns of b's is $d-t-1$ and the number of columns of f's is $t-1$. We want to be able to rewrite this in terms of just one tunnelling parameter, c, which represents the coupling between the normal and tunnelling edges. As the dynamics of the walk must be reversible, we must ensure that the coin produced is unitary. As such,

$$(T_{d,t})\,(T_{d,t})^{\dagger} = \mathbb{I}_d, \tag{6.2}$$

must hold, where $(T_{d,t})^{\dagger}$ is the hermitian conjugate of the general matrix. Using this unitarity condition we can write five equations which must be satisfied for unitarity to hold,

$$a^2 + (d-t-1)b^2 + tc^2 = 1, \tag{6.3}$$

$$2ab + (d-t-2)b^2 + tc^2 = 0, \tag{6.4}$$

$$(d-t)c^2 + e^2 + (t-1)f^2 = 1, \tag{6.5}$$

$$(d-t)c^2 + 2ef + (t-2)f^2 = 0, \tag{6.6}$$

and

$$[a + (d - t - 1)b + e + (t - 1)f]c = 0. \tag{6.7}$$

We want to solve these equations in terms of d, t and c. Taking eq. (6.4) from eq. (6.3) we obtain an expression for a in terms of b only,

$$\begin{aligned} a^2 - 2ab + b^2 &= 1, \\ (a - b)^2 &= 1, \\ a &= b - 1. \end{aligned} \tag{6.8}$$

We obtain a similar expression for e in terms of f by taking eq. (6.6) from eq. (6.5),

$$\begin{aligned} e^2 - 2ef + f^2 &= 1, \\ (e - f)^2 &= 1, \\ e &= f - 1. \end{aligned} \tag{6.9}$$

By substituting these new expressions back into equations (6.3) and (6.5), we obtain two quadratic equations,

$$(b - 1)^2 + (d - t - 1)b^2 + tc^2 = 1 \tag{6.10}$$

and

$$(d - t)c^2 + (f - 1)^2 + (t - 1)f^2 = 1. \tag{6.11}$$

Taking eq. (6.10), we simplify and solve giving an expression for b in terms of just d, t and c,

$$\begin{aligned} (b - 1)^2 + (d - t - 1)b^2 + tc^2 &= 1, \\ b^2 - 2b + 1 + (d - t - 1)b^2 + tc^2 &= 1, \\ (d - t)b^2 - 2b + tc^2 &= 0. \end{aligned} \tag{6.12}$$

Solving, by choosing the positive root in this case gives

$$b = \frac{1 + \sqrt{1 - (d-t)tc^2}}{d-t}. \tag{6.13}$$

Similarly, an expression for f can be obtained from eq. (6.11),

$$\begin{aligned} (d-t)c^2 + (f-1)^2 + (t-1)f^2 &= 1, \\ (d-t)c^2 + f^2 - 2f + 1 + (t-1)f^2 &= 1, \\ tf^2 - 2f + (d-t)c^2 &= 0. \end{aligned} \tag{6.14}$$

Solving, by choosing the negative root here gives

$$f = \frac{1 - \sqrt{1 - (d-t)tc^2}}{t}. \tag{6.15}$$

We now have expressions for a, b, e and f in terms of just d, t and c from equations (6.8), (6.13), (6.9) and (6.15). Finally, we must check that these expressions maintain unitarity in eq. (6.7). For c not equal to zero this becomes,

$$\begin{aligned} (b-1) + (d-t-1)b + (f-1) + (t-1)f &= 0, \\ b - 1 + bd - bt - b + f - 1 + tf - f &= 0, \\ (d-t)b + tf &= 2. \end{aligned} \tag{6.16}$$

It is easy to see that by the choice of roots chosen in solving for b and f that this condition holds when we substitute in eqs. (6.13) and (6.15).

This operator allows us to vary the 'strength' of certain edges in the structure we wish to walk on. It holds for any degree of vertex but the number of tunnelling edges must be at most half the degree, i.e. $t \leq d/2$. By using vertices of degree six with two tunnelling edges, we have a basic 2D Cartesian lattice gradually becoming

a triangular lattice as in fig. 6.1. In this case, setting $c = 0$ we obtain

$$T_{6,2} = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 1 & 0 & 0 \\ 1 & 1 & -1 & 1 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix}, \tag{6.17}$$

and setting $c = 2/d = 1/3$ gives

$$T_{6,2} = \frac{1}{3}\begin{pmatrix} -2 & 1 & 1 & 1 & 1 & 1 \\ 1 & -2 & 1 & 1 & 1 & 1 \\ 1 & 1 & -2 & 1 & 1 & 1 \\ 1 & 1 & 1 & -2 & 1 & 1 \\ 1 & 1 & 1 & 1 & -2 & 1 \\ 1 & 1 & 1 & 1 & 1 & -2 \end{pmatrix}. \tag{6.18}$$

These choices of c represent the extremes of the operator when the structure would be either a basic 2D Cartesian lattice, eq. (6.17), or a triangular lattice, eq. (6.18). Any other values of c where $0 \leq c \leq 2/d$ would give varying strengths of tunnelling across the tunnelling edges.

6.3 Results using the tunnelling operator

Using the tunnelling operator introduced above, we numerically study how the algorithm is effected by the change in spatial dimension. We use the operator to interpolate between the line (1D) and a Cartesian lattice (2D) and then between a Cartesian lattice (2D) and a cubic lattice (3D). This is done by, in the 1D-2D case, using vertices of degree four with two tunnelling edges, with the correct connectivity, to interpolate between a collection of 1D lines to a fully connected 2D Cartesian lat-

tice. In the same fashion, we interpolate between a set of 2D Cartesian lattices and a fully connected cubic lattices by using vertices of degree six with two tunnelling edges, again with the correct connectivity.

The search algorithm we use is the same as in the original work by Shenvi et al. [50], with a small change to the initial state. Although we must still start the walker in a uniform superposition over all vertices, the distribution over the edges must be altered slightly to account for the strength of the tunnelling edges. In other words, we must distribute the state over the edges with a weighting to match the tunnelling strength as follows

$$(d-t)\alpha + tp\alpha = \frac{1}{\sqrt{N}}, \tag{6.19}$$

where p is the tunnelling probability, α is the state on each edge and N is the number of vertices. The tunnelling probability is just the tunnelling parameter, c, rescaled to lie between 0 and $2/d$. In this way, the tunnelling probability matches the proportion of the initial state which is placed on the tunnelling edges. This initial state gives a probability distribution, where there is no marked state, which is periodic over two timesteps as can easily be checked. Although this is not stationary as in the case of the basic non-tunnelling lattices, the fact that it returns to the same state after only two timesteps means it will give rise to the same dynamics.

6.3.1 1D line - 2D lattice

We study here the quantum walk search algorithm, interpolating between a 1D lattice or line and a 2D Cartesian lattices. If there is no possibility of tunnelling, the structure will just be a collection of unlinked 1D lattices. When the edges are present, the structure becomes connected, turning it into a 2D Cartesian lattice. Here we run the algorithm on a 2D Cartesian lattice where the edges which link the 1D lattices together are tunnelling. We show the a basic unit in fig. 6.2. The solid lines are the fixed, normal edges and the dashed lines are tunnelling edges. This structure allows us to gradually change the strength of the edges which make the structure two dimensional, hence interpolating between the 1D lattice and the 2D

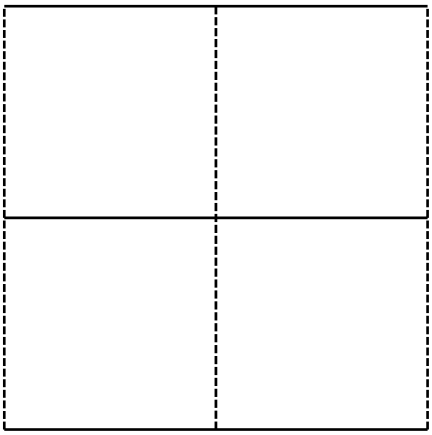

Figure 6.2: Basic unit of the structure we use to interpolate between the one dimensional and the two dimensional Cartesian lattice. The solid lines represent the fixed, normal edges whereas the dashed lines represent the edges we set to be tunnelling.

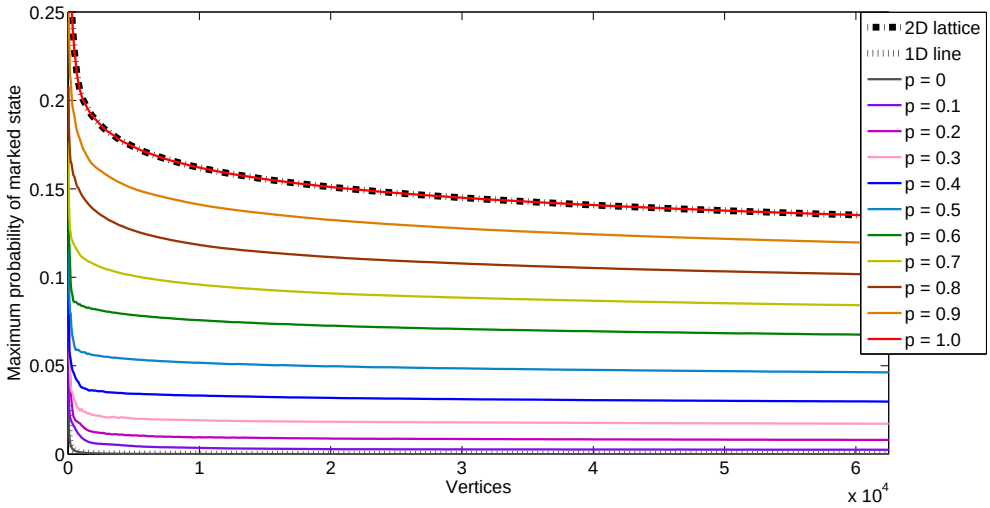

Figure 6.3: Plot to show how the maximum probability of the marked state varies with both the size of the lattice and the tunnelling strength as a one dimensional lattice is gradually changed into a two dimensional Cartesian lattice.

Cartesian lattice.

We see in fig. 6.3 how the maximum probability of the marked state varies as the tunnelling strength is increased and the structure changes from 1D to 2D. We can see that as soon as the additional edges exist, effectively turning the structure from one spatial dimension to another, the scaling of the probability changes from the 1D scaling of $1/N$ to the 2D lattice scaling of $1/\log_2 N$. As the tunnelling probability increases, we see a change in the prefactor to the maximum probability

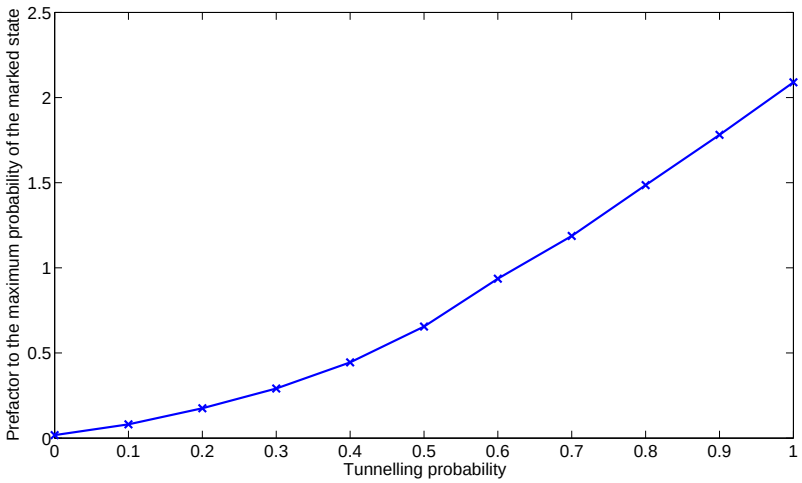

Figure 6.4: Plot to show how the prefactor to the scaling of the maximum probability of the marked state, obtained from the data in fig. 6.3, varies with the tunnelling strength as a one dimensional line is gradually changed into a two dimensional Cartesian lattice.

of the marked state as shown in fig. 6.4. We can see that for tunnelling probabilities of $p \approx 0.5$ and above, there seems to be a linear scaling in the prefactor. However, below this probability we see a change to this scaling of the prefactor. This is due to the scaling changing to the 1D $1/N$ scaling of the line, and the search is starting to fail completely.

In the 1D case, we already know that the search fails as the maximum probability never reaches a high enough level to allow it to become distinguishable from the remaining superposition. As such, the time to find the marked state does not scale in the a sensible fashion. In fact, it can often become artificially small due to the fact it doesn't take long at all to reach a probability only very slightly larger than then initial state. As such, we only show the scaling of the time to find the marked state for probabilities greater than $p = 0.5$. We firstly show, in fig. 6.5, how the time to find the marked state varies with both the size of the dataset and the tunnelling probability. As we would expect, as the tunnelling strength is increased, the time to find the marked state decreases. We then show how this prefactor to the scaling of the time to find the marked state varies with the tunnelling probability. Again, we only show this for the higher tunnelling probabilities.

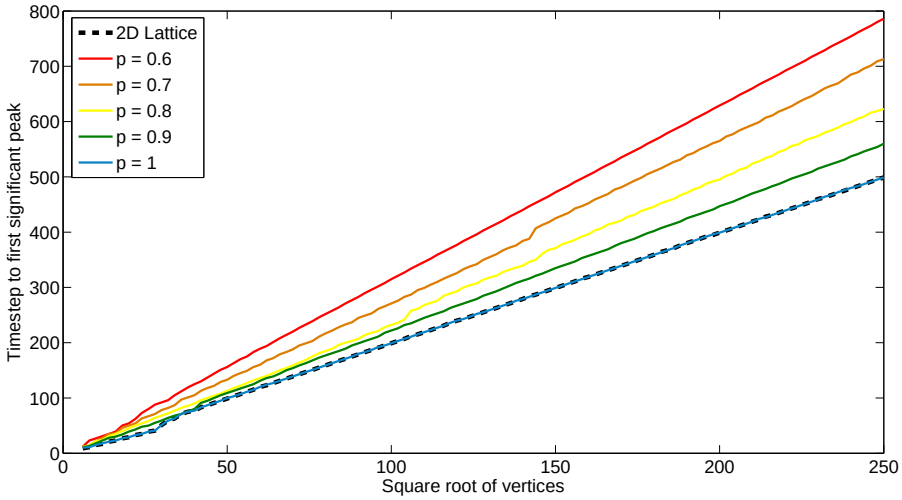

Figure 6.5: Plot to show how the time to find the marked state varies with both the size of the lattice and the tunnelling strength as a one dimensional lattice is gradually changed into a two dimensional Cartesian lattice.

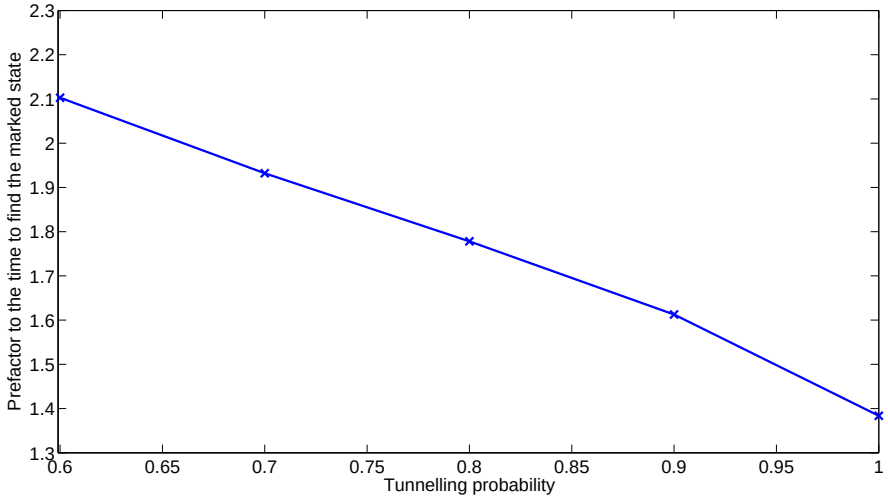

Figure 6.6: Plot to show how the prefactor to the scaling of the time to find the marked state, obtained from the data in fig. 6.5, varies with the tunnelling strength as a one dimensional line is gradually changed into a two dimensional Cartesian lattice.

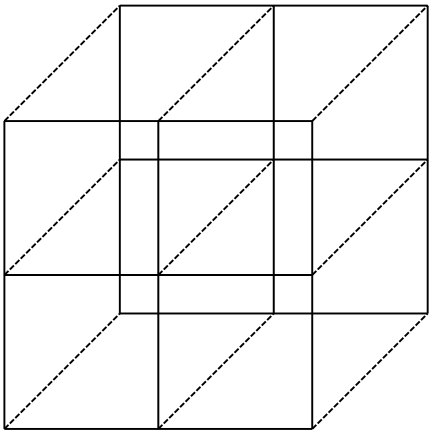

Figure 6.7: Basic unit of the structure we use to interpolate between the two dimensional Cartesian and the three dimensional cubic lattice. The solid lines represent the fixed, normal edges whereas the dashed lines represent the edges we set to be tunnelling.

6.3.2 2D lattice - 3D lattice

As in the case of changing from the 1D line to the 2D lattice, we see a similar behaviour in the 2D to 3D case. Here we run the algorithm on a 3D cubic lattice where the edges which link the 'slices' of 2D lattices together are tunnelling. We show the a basic unit in fig. 6.7. The solid lines are the fixed, normal edges and the dashed lines are tunnelling edges. This structure allows us to gradually change the strength of the edges which make the structure three dimensional, hence interpolating between the 2D Cartesian lattice and the 3D cubic lattice.

The maximum probability of the marked state, shown in fig. 6.8, changes from the $1/\log_2 N$ scaling in the 2D case to the constant $O(1)$ scaling as soon as the additional edges even have a small weighting attached to them. At low tunnelling strengths, we see the probability dropping initially before gradually recovering towards a constant value for higher lattice sizes. At these higher sizes, it is easy to see that the scaling is constant for any tunnelling strength with just varying prefactors. Figure 6.9 shows how this prefactor to the scaling of the maximum probability of the marked state changes as we increase the tunnelling probability. The sharp drop at the low tunnelling probabilities is most probably due to the fact the scaling hasn't reached the constant value as we can only simulate up to a fixed lattice size.

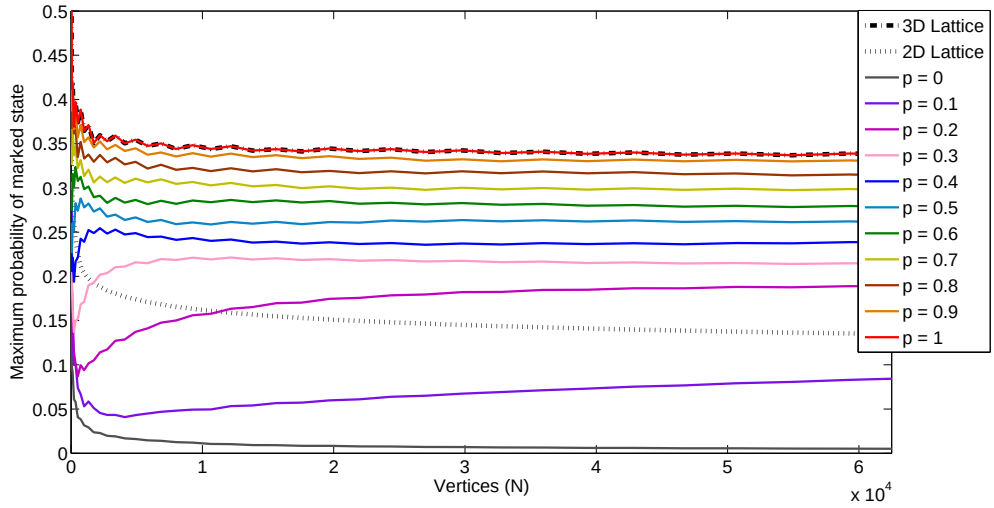

Figure 6.8: Plot to show how the maximum probability of the marked state varies with both the size of the lattice and the tunnelling strength as a two dimensional lattice is gradually changed into a three dimensional Cartesian lattice.

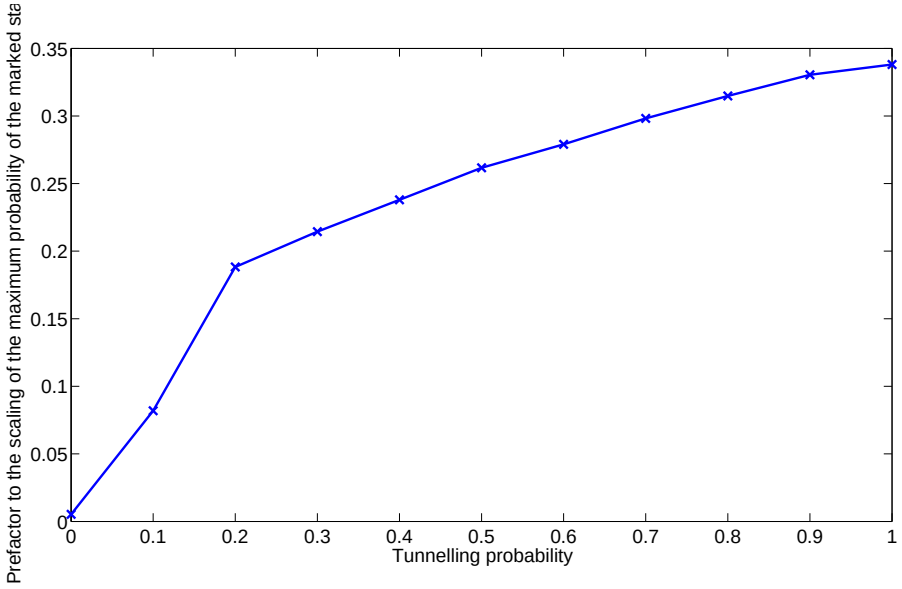

Figure 6.9: Plot to show how the prefactor to the scaling of the maximum probability of the marked state, obtained from the data in fig. 6.8, varies with the tunnelling strength as a two dimensional lattice is gradually changed into a three dimensional Cartesian lattice.

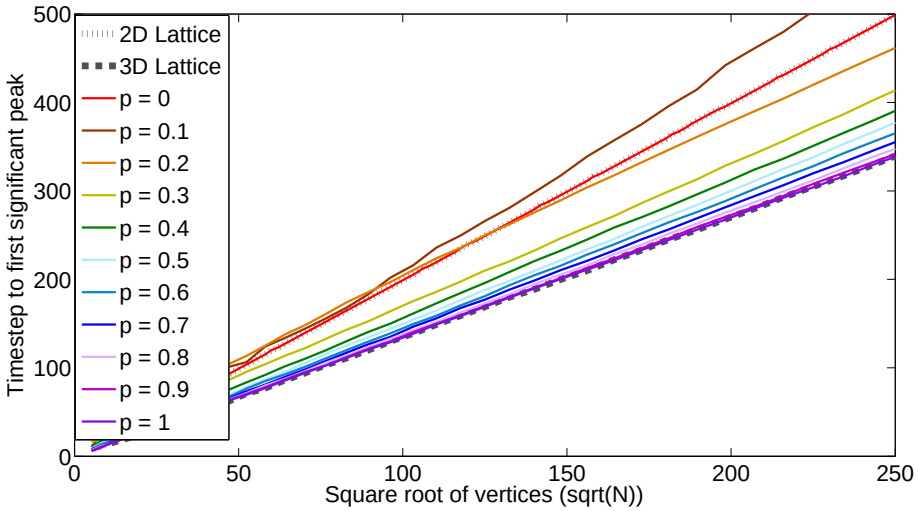

Figure 6.10: Plot to show how the time to find the marked state varies with both the size of the lattice and the tunnelling strength as a two dimensional lattice is gradually changed into a three dimensional Cartesian lattice.

In addition, we note here that when the probability of tunnelling is zero, the scaling does not match that of the basic 2D lattice. At $p = 0$, the structure is in effect a collection of 2D lattices which are unlinked. The initial state is still spread across all these individual lattices and due to the connectivity of the structure, only the amplitude in one of the lattices (that with the marked state present) is able to coalesce on the marked state. As such, the scaling is just reduced by a constant factor as can be seen in fig. 6.8.

The time to find the marked state follows a similar behaviour. The basic scaling of the time to find the marked state is the same in both two and three dimensions, $O(\sqrt{N})$. We see in fig. 6.11 that, in general, the time to find the marked state (the prefactor to the basic scaling) decreases as the tunnelling strength increases, thus making the algorithm more efficient. We show a plot of how this prefactor varies with the tunnelling strength in fig. 6.11. We do note that at the very low tunnelling probabilities, $p <\approx 0.3$, the scaling of the time to find the marked state does not follow the same behaviour. This is due to the fact that the marked state has not yet reached a constant value, as mentioned previously. If we were able to run the algorithm on much larger sized lattices, we should find the probability of the marked

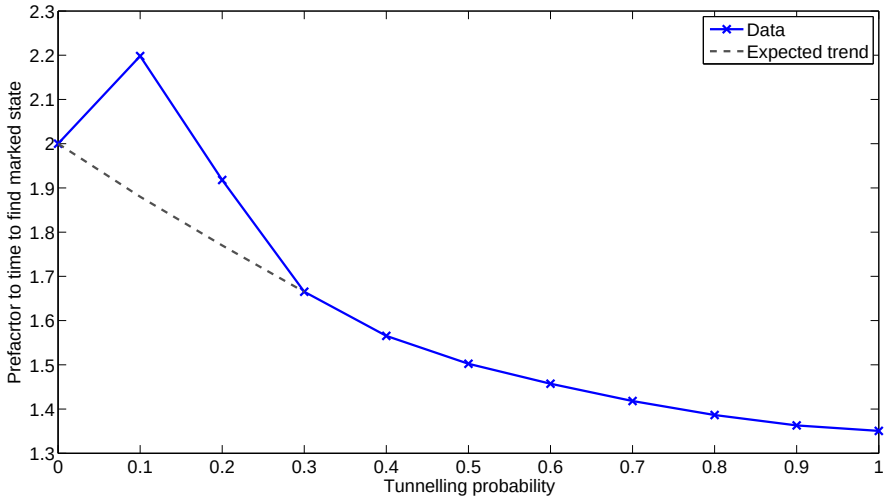

Figure 6.11: Plot to show how the prefactor to the scaling of the time to find the marked state, obtained from the data in fig. 6.10, varies with the tunnelling strength as a two dimensional lattice is gradually changed into a three dimensional Cartesian lattice.

state stabilising and thus the time to find the marked state matching the quadratic speedup in scaling. We show the expected trend to the scaling of the prefactors to the time to find the marked state in fig. 6.11.

6.4 Lattices of varying height and depth

In this section, we discuss lattices of varying height and depth to give a different avenue of investigation of the dependence on spatial dimension. We vary the height of the lattice in the case of the 2D lattice and the depth of the lattice in the case of the 3D lattice. For the 2D lattice, at low heights of the lattice, it can, in effect, be considered almost like a 1D line, where we know the search cannot find the marked state efficiently. We are interested here in how the scaling of the probability of the marked state and the time to find it changes as the lattice height (depth) is gradually increased, eventually becoming a fully symmetric 2D (3D) lattice.

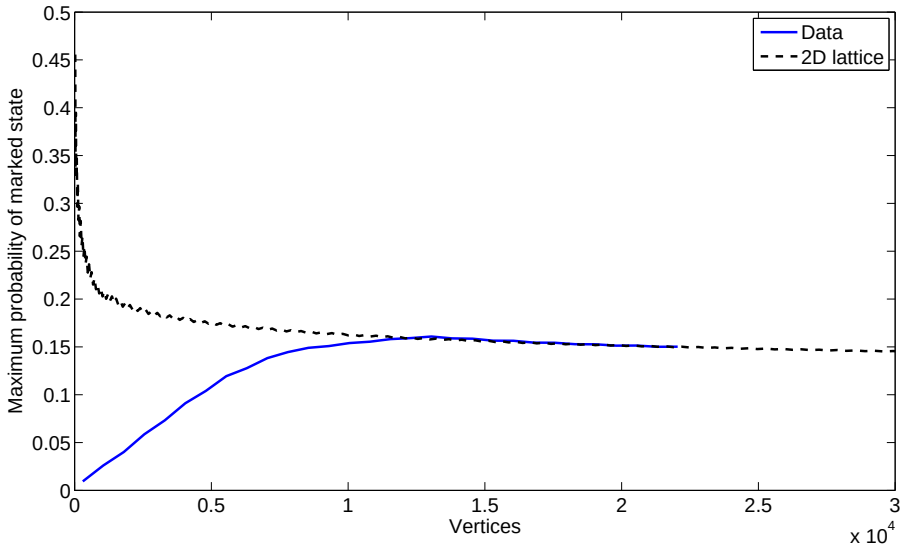

Figure 6.12: Plot to show how the maximum probability of the marked state varies as a two dimensional lattice is gradually increased in height, while keeping the length fixed, to become a two dimensional square lattice. We plot this against the equivalent number of vertices for that lattice.

6.4.1 1D - 2D lattice

In order to interpolate between the 1D line and the 2D Cartesian lattice, we maintain the width of the lattice and vary the height gradually. This changes the lattice from a 1D line to eventually end at a 2D Cartesian lattice where both dimensions are the same.

We firstly show how the probability of the marked state is affected as we maintain the length of the lattice but vary the height. This is shown in fig. 6.12 along with the same probability for the fully square 2D Cartesian lattice. In this case, we maintained the length of the 1D lattice at 150 vertices and increased the height gradually from $h = 2$ through to $h = w$, where w is the width of the lattice and h is the height. We see that as the lattice grows in height, i.e. becomes a square lattice, the maximum probability of the marked state gradually increases to eventually match that of the 2D lattice at roughly $h = 70$, half that of the lattice width.

We then ran the search algorithm on lattices of fixed height, but varying length (with a fixed number of total vertices) to show how the maximum probability of the

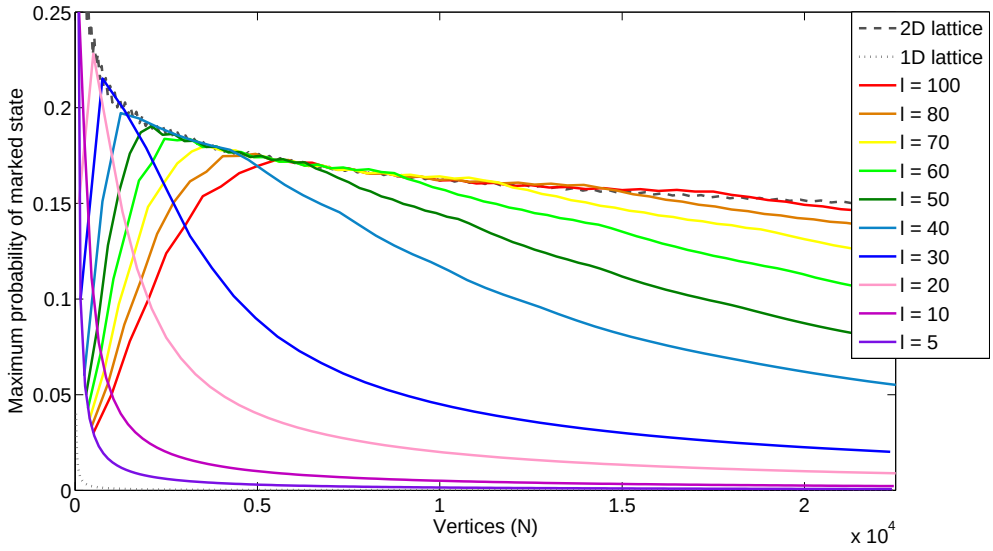

Figure 6.13: Plot to show how the maximum probability of the marked state varies as a two dimensional lattice is gradually increased in height to become a two dimensional Cartesian lattice.

marked state and the time to find this probability varied. As we see in fig. 6.13, the scaling of the maximum probability does not change immediately as we saw when using the tunnelling operator. Instead, we see a gradual change from the $1/N$ scaling of the 1D line to the $1/\log_2 N$ scaling of the square 2D Cartesian lattice. When the height is small, i.e. $h = 5 - 30$, we see the scaling is basically the same as that of the 1D line. This is due to the fact that as the walker is moving around the lattice periodically, it wraps around (vertically) very quickly and so the boundary effects cause it to see the lattice more like a line. As the height of the lattice increases, the walker takes longer to pass around the lattice and thus these boundary effects become less dominant. Due to numerical size restrictions, it becomes hard to tell when the scaling becomes logarithmic, but we see at heights around 70 the behaviour is very similar to that of the 2D lattice. We note here that the initial peaks of the scaling for low numbers of vertices is just due to the lattice being, in effect, taller in height than width.

Due to the search algorithm failing when the scaling is $1/N$, we only show lattices of larger height when studying the time to find the marked state. Figure 6.14 shows

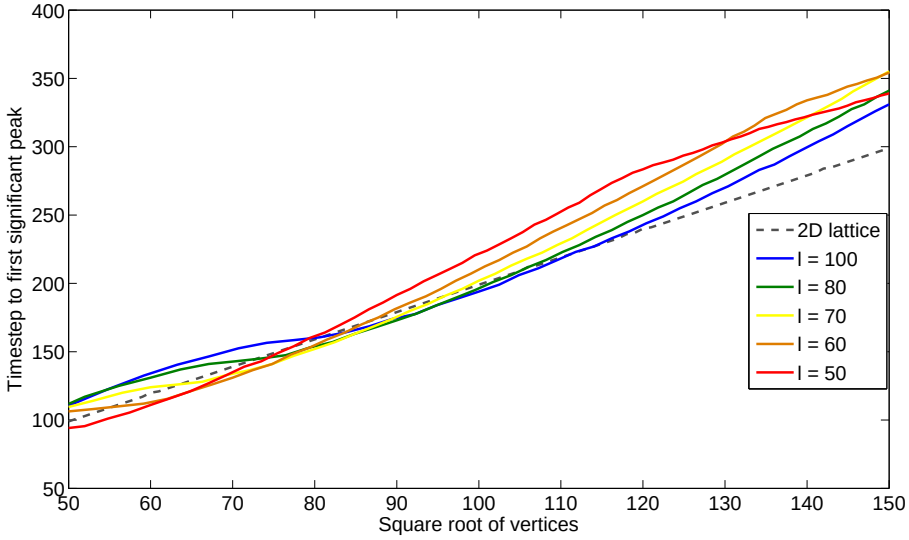

Figure 6.14: Plot to show how the time to find the marked state varies as a two dimensional lattice is gradually increased in height to become a two dimensional Cartesian lattice.

that the time to find the marked state seems to follow a different scaling than the full square 2D Cartesian lattice. Even when $h = 100$ the scaling does not quite match the basic 2D lattice. This is probably just a numerical artifact of how we find the time to the first significant peak, due to the fact the scaling of the probability has not quite reached the full logarithmic scaling yet. However, it could be following a $\sqrt{N \log N}$ scaling, but is hard to tell with numerical restrictions on the size of the lattice we are able to study.

6.4.2 2D - 3D lattice

In order to interpolate between the 2D Cartesian lattice and the 3D cubic lattice, we firstly maintain both the width and height of the lattice and vary the depth gradually. This changes the lattice from a 2D Cartesian lattice to eventually end at a 3D cubic lattice where all dimensions are the same.

We perform the same analysis as in the 1D-2D case, firstly showing how the maximum probability of the marked state varies for a fixed width and height of lattice (30x30), while varying the depth. Figure 6.15 shows how this probability

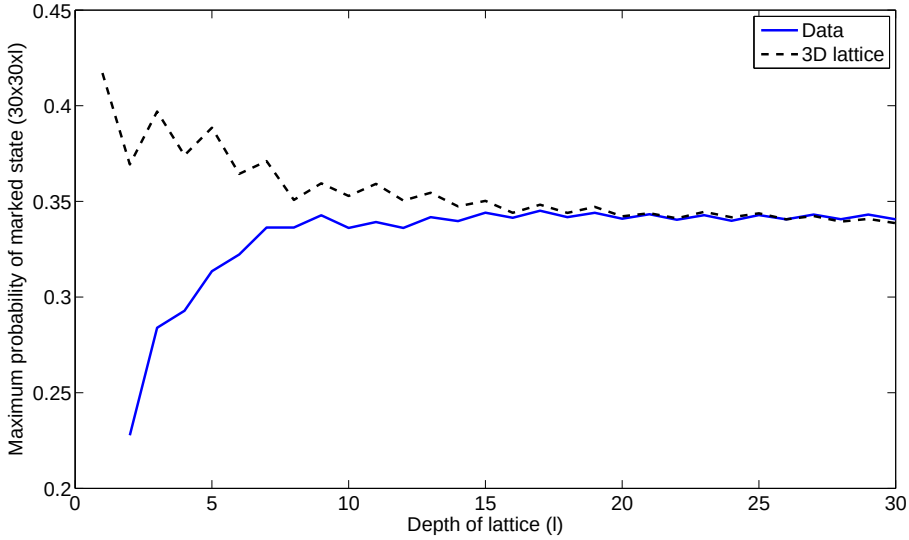

Figure 6.15: Plot to show how the maximum probability of the marked state varies as a three dimensional lattice is gradually increased in depth, while keeping the width and height fixed, to become a three dimensional cubic lattice.

varies, along with the equivalent scaling for the fully cubic 3D lattice, showing the scaling matches at roughly $l = 15$, where l is the depth of the lattice.

We then fixed the depth of the lattice and altered the width and height of the lattice (with a fixed number of vertices) for each run of the search algorithm. Figure 6.16 shows how the maximum probability of the marked state varies for differing depths of the lattice. We again see a gradual change in scaling from the basic 2D logarithmic scaling to the constant scaling of the cubic lattice. This is in contrast to the almost instantaneous change in scaling we see when interpolating using the tunnelling operator.

The basic scaling of the time to find the marked state is unaffected by the change in the depth of the lattice. The prefactor to this scaling though does decrease as we increase the depth, changing from that of the basic 2D lattice to almost match that of the cubic lattice for even lattices of modest depth. It is much easier to see how the time to find the marked state changes than in the 1D-2D case as the algorithm succeeds in all cases.

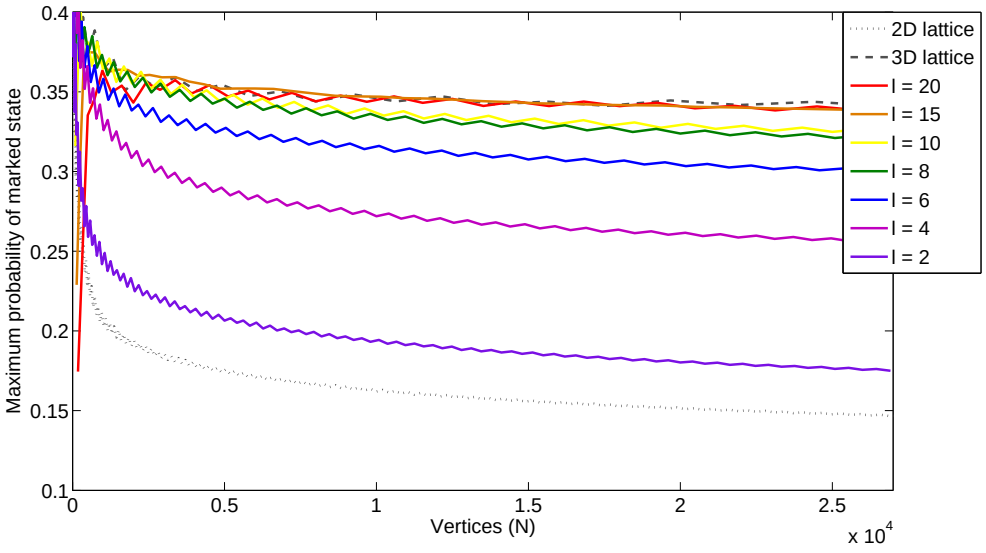

Figure 6.16: Plot to show how the maximum probability of the marked state varies as a three dimensional lattice is gradually increased in depth to become a three dimensional cubic lattice.

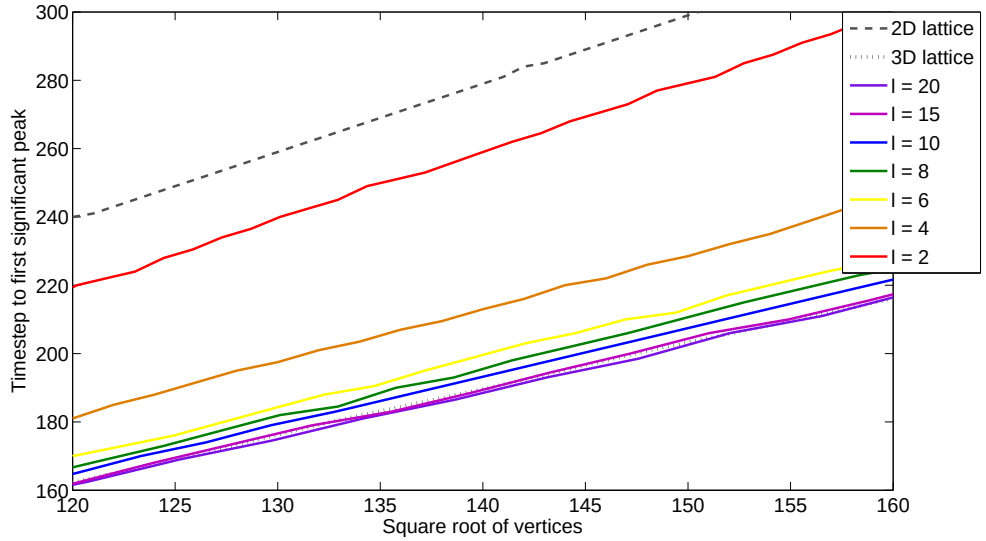

Figure 6.17: Plot to show how the time to find the marked state varies as a three dimensional lattice is gradually increased in depth to become a three dimensional cubic lattice.

6.5 Discussion

In this chapter, we have shown two different ways in which we can interpolate between structures of differing spatial dimension. Firstly, we used a tunnelling operator to vary specific edges of a lattice enabling us to gradually change the spatial dimension of the lattice. In this case, we found a sudden change in the scaling of the maximum probability of the marked state as soon as there was even a very small probability of the edges existing. This seems to indicate that the 'strength' of the edges in the lattice is of little importance, with the dependence on the specific spatial dimension taking precedence. However, we did find that the prefactor to the scaling of this probability did vary with the strength of the tunnelling edges, increasing as the tunnelling strength increased. These effects were more pronounced in the change from a 2D Cartesian lattice to a 3D cubic lattice as in these cases the algorithm always succeeds. In the case of a 1D line, the algorithm can fail as it can only find the marked state with a probability of $O(1/N)$, which is clearly of no use. The basic scaling of the time to find the marked state is not affected by the change in dimensionality, we note though that the prefactor to the scaling decreases as the tunnelling strength is increased, hence the algorithm becomes more efficient.

The other case we considered is the case of lattices with varying height or depth, for example, a 3D lattice with fixed width and height but of varying depth. Although this structure is still strictly three dimensional, when the depth is very low and the width (height) is large, the quantum walker will see the structure as almost a basic 2D Cartesian lattice. Suprisingly, in this case we see a gradual change in scaling in the maximum probability of the marked state. At low heights or depths of the lattice, the scaling is almost the same as the lower spatial dimensional structure gradually changing to the higher dimensional structure scaling as the height (depth) increases to become equal to that of the other dimensions. This highlights the importance of full symmetry in the quantum walk search algorithm.

Chapter 7

Effects of connectivity on the quantum walk search algorithm

7.1 Introduction

In this chapter, we investigate how important the connectivity of the database arrangement is for the searching algorithm. As discussed in chapters 3 and 6, the basic scaling of the algorithm is heavily dependent upon the spatial dimension of the structure in question. Here, we numerically study how the connectivity, in a specific spatial dimension, affects the prefactors to this scaling. Although it is unlikely that the runtime of the search algorithm on a 2D Cartesian lattice can be reduced to the optimal $O(\sqrt{N})$, it may be possible to reduce any constant overhead associated with the run time. While the hypercube and 2D Cartesian lattices have been studied in detail, little has been covered on other structures due to their connectivity making them hard to analyse analytically. However, in related work, Abal et al. [202] have shown analytically that the complexity of the search algorithm on the hexagonal lattice is $O(\sqrt{N}\log N)$, matching the search on the Cartesian lattice in [69] but with a differing prefactor to the scaling of the algorithm. In addition, highly symmetric graphs such as the complete graph were studied by Reitzner et al. [203] showing the additional connectivity does not allow the search to beat the opti-

mal lower bound of $O(\sqrt{N})$. The hitting time on the complete graph has also been studied recently by Santos and Portugal [84], proving this is also $O(\sqrt{N})$. We show how the connectivity of regular structures affects the efficiency of the search algorithm in terms of the prefactors to the scaling of both the maximum probability of the marked state and also the time to find this maximum probability. This chapter forms the basis of two pieces of published work [2, 3].

We use the tunnelling coin operator introduced in Chapter 5 which allows us to model the search algorithm on structures where there is a probability of additional connections existing. For example, our tunnelling operator allows us to interpolate between running the search algorithm on a hexagonal lattice, with degree $d = 3$, and the 2D Cartesian lattice, $d = 4$. This allows us to analyse how the search algorithm is affected by a gradual change in the degree of the underlying substrate by changing the tunnelling strength of the additional edges. This extends the initial studies of [2] by considering an interpolation between lattices with fixed degree. After introducing the structures we wish to perform the search algorithm upon, we show our results for both two and three dimensional structures with varying connectivity before then concluding this chapter by discussing our results.

7.2 Results

Using the tunnelling operator, described in sec. 6.2, we now present our results for simulating the quantum walk search algorithm on both two and three dimensional structures, gradually varying the connectivity. We show that the prefactors to the scaling of the algorithm for both the maximum probability of the marked state and the time to find the marked state are dependent on the connectivity of the underlying structure.

7.2.1 Two dimensional structures

Using the tunnelling matrix we have introduced, we ran the search algorithm on 2D lattices ranging from $d = 3$, a hexagonal lattice, through to $d = 8$, a Cartesian

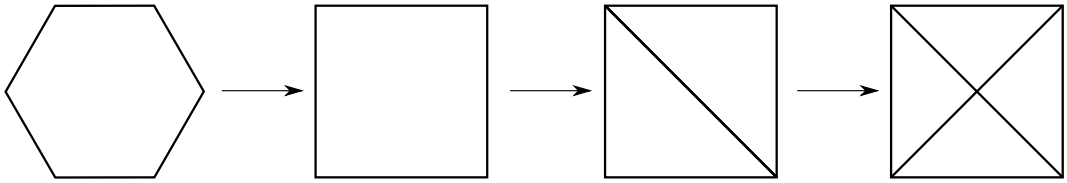

Figure 7.1: The 2D lattices we interpolate between using the tunnelling matrix. We change gradually from a hexagonal lattice, $d = 3$, to a 2D Cartesian lattice with diagonals included, $d = 8$. We show here just the building block of each lattice. We note that in the case of the 2D Cartesian lattice with diagonals, there is no vertex at the central point where the edges cross.

lattice with diagonals added as shown in fig. 7.1, for varying lattices sizes from 6^2 (36) vertices up to 250^2 (62500) vertices. As in fig. 7.1, we gradually changed the degree of the structure we performed the search algorithm on. This was split into intermediate steps, firstly from the 2D hexagonal lattice ($d = 3$) to the square lattice ($d = 4$), the square lattice to the triangular lattice ($d = 6$), eventually ending at the more highly connected Cartesian lattice with diagonals ($d = 8$). We spread the walker in the same fashion as eq. (6.19) to ensure we distribute the state evenly based on the tunnelling strength of the edges.

We show in fig. 7.2 how the time to find the marked state varies with both the size of the lattice and the connectivity. We see that as the connectivity increases, the time to find the marked state decreases, hence the efficiency of the algorithm increases. As the time to find the marked state scales as $O(\sqrt{N})$, we fit to each of the data sets in fig. 7.2 to obtain the prefactor to the scaling of the time to find the marked state. Figure 7.3 shows how this prefactor to the scaling changes with the degree of the underlying structure being searched.

In fig. 7.4, we show how the maximum probability of the marked state varies with both the size of the dataset and the connectivity of the structure. We see that, in general, as the connectivity of the structure being searched increases, the maximum probability of the marked state also increases. A larger prefactor to this scaling means fewer repeats of the algorithm are required to bring the success probability close to unity. Figure 7.5 shows how this prefactor to the scaling of $O(1/\log_2 N)$ varies with the degree of the structure being searched.

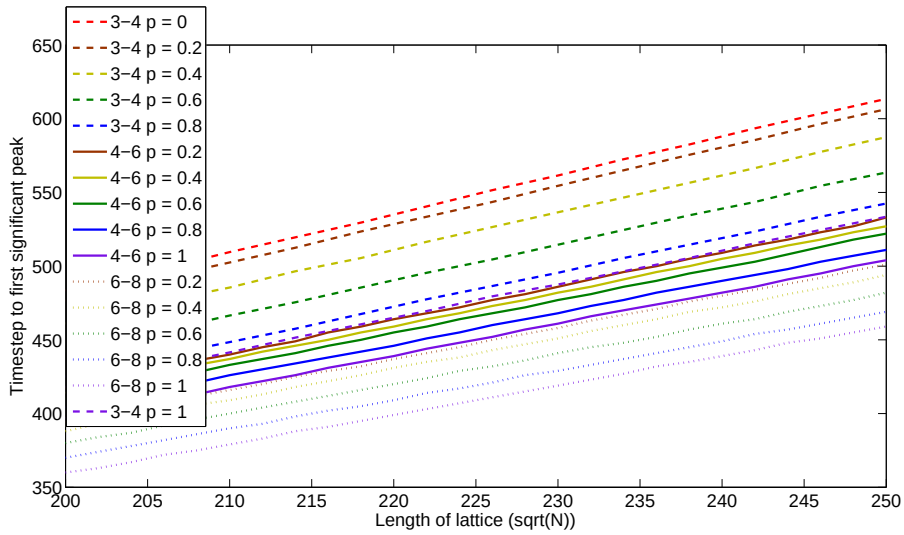

Figure 7.2: Scaling of the time to find the marked state with the size of the lattice with varying connectivity in two dimensions. It is clear that as the connectivity of the structure increases, the time to find the marked state decreases. Note that this is a zoomed in plot showing only larger lattices sizes, data for $\sqrt{N} < 200$ has been omitted to improve clarity.

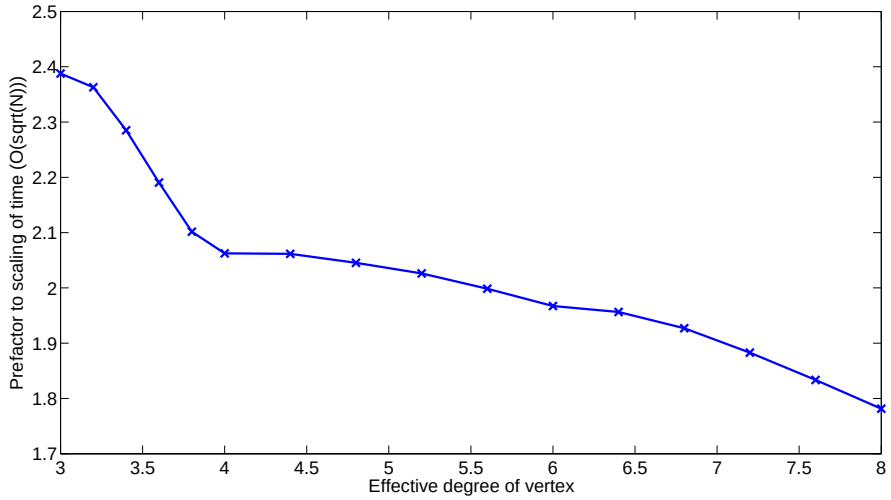

Figure 7.3: Plot to show how the prefactor to the scaling, obtained from the data shown in fig. 7.2, of the time to find the marked state of $O(\sqrt{N})$ changes with the degree of the two dimensional structure being searched.

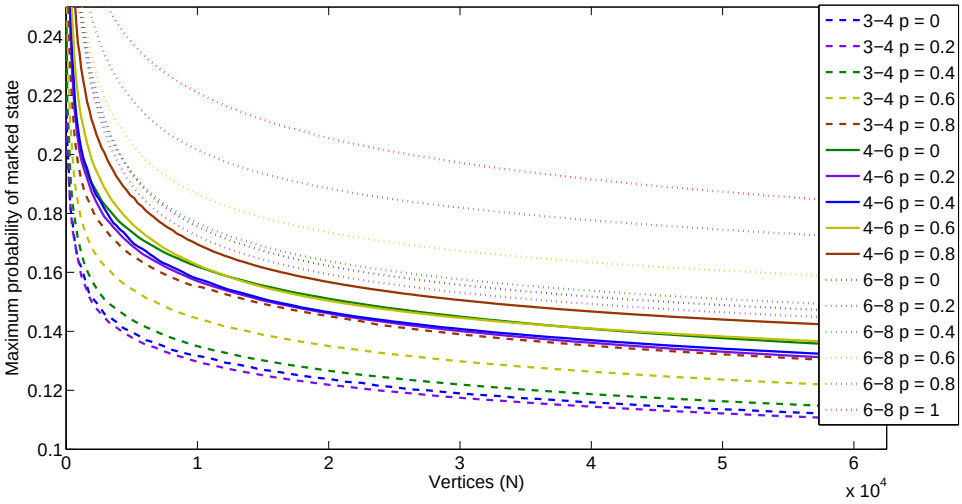

Figure 7.4: Scaling of the maximum probability of the marked state with the size of the lattice with varying connectivity in two dimensions. In general, as the connectivity of the structure increases, the maximum probability of the marked state also increases.

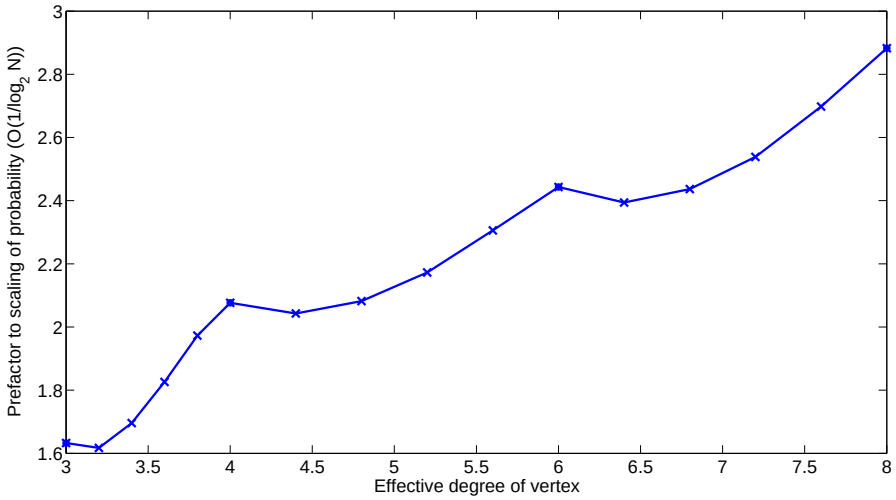

Figure 7.5: Plot to show how the prefactor to the scaling, obtained from the data in fig. 7.4, of the maximum probability of the marked state of $O(1/\log_2 N)$ changes with the degree of the two dimensional structure being searched.

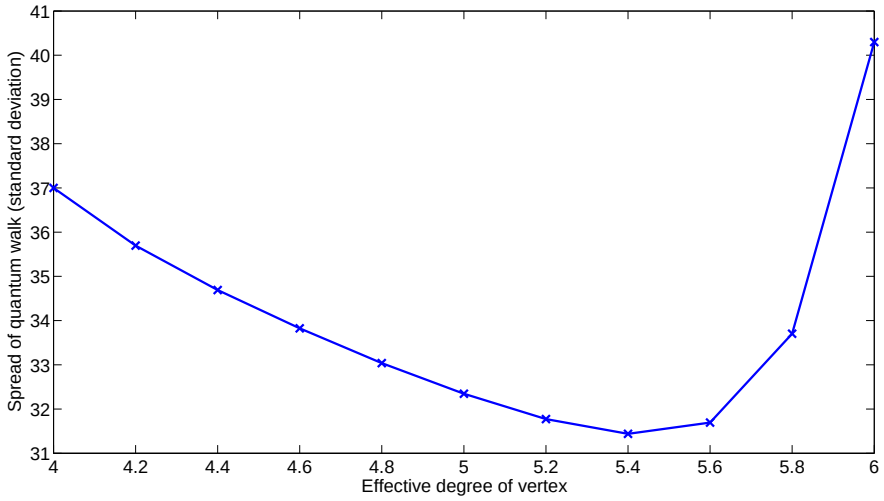

Figure 7.6: Plot to show how the spreading of the quantum walk, characterised by eq. (7.1), changes with the degree of the two dimensional structure being searched.

The 'dips' and revivals in the scaling seem counter intuitive, but appear to arise from the dynamics of the walk on these structures where the symmetry is partially broken (low tunnelling strength). In order to confirm this, we briefly examined the basic dynamics of the quantum walk while varying the tunnelling strength. We started the walker at a specific vertex in the graph, as opposed to an equal superposition, and allowed it to propagate outwards in order to determine its dynamics. We define the spread of the walker as

$$\langle r \rangle = \sum_{i=1}^{N} p_i s_i, \tag{7.1}$$

where p_i is the probability of the walker being at vertex i and s_i is the shortest path distance from the position of the initial state to vertex i. Using this metric for the rate of spreading, we explored how this was affected by the tunnelling strength. Figure 7.6 shows this spreading on a 2D Cartesian lattice gradually being turned into a triangular lattice as in fig. 6.1. We found at low tunnelling strengths, where the symmetry breaking is most obvious, the spread, $\langle r \rangle$, dropped. As the tunnelling strength was raised, the quantum walk was able to recover and $\langle r \rangle$ increased back to

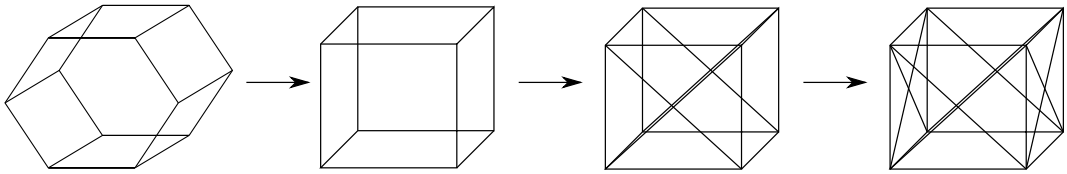

Figure 7.7: The 3D lattices we interpolate between using the tunnelling matrix. We change gradually from a 3D hexagonal lattice, $d = 5$, through to a cubic lattice with diagonals added on the faces, $d = 14$. We show here just the building block of each lattice. We note here that vertices are only present at the eight corners of the cubic structures, there are no vertices present where the edges cross.

the value of the original lattice, before increasing further as the tunnelling strength reached its maximum value, i.e. the new lattice. Although this is not an exhaustive study of the quantum walk dynamics when we include tunnelling edges, this behaviour does match the results we find for the search algorithm. While the variation of $\langle r \rangle$ does not match the scaling of the probability of the marked state directly, the basic quantum walk dynamics do not have any reflection effects from the edges of the structure. Due to the periodic boundary conditions imposed in the searching algorithm, we find slightly different behaviour which relate to the extra interference effects. We found similar results for all the lattices studied in both two and three dimensions.

7.2.2 Three dimensional structures

We now consider three dimensional lattices, using the tunnelling matrix to study structures ranging from $d = 5$, a 3D hexagonal lattice, through to $d = 14$, a cubic lattice with additional diagonals added as shown in fig. 7.7. We ran the search algorithm for varying lattices sizes from 3^3 (27) vertices up to 40^3 (64000) vertices. As in the two dimensional case, we did not just change the lattice from $d = 5$ to $d = 14$ in one go. We split this into intermediary steps, fig. 7.7, changing firstly from the 3D hexagonal lattice ($d = 5$) to the cubic lattice ($d = 6$), the cubic lattice to one with diagonals added to one face ($d = 10$), eventually ending with a cubic lattice with diagonals added on two faces ($d = 14$). We split the initial state across the vertices and edges in the same way as in the two dimensional case.

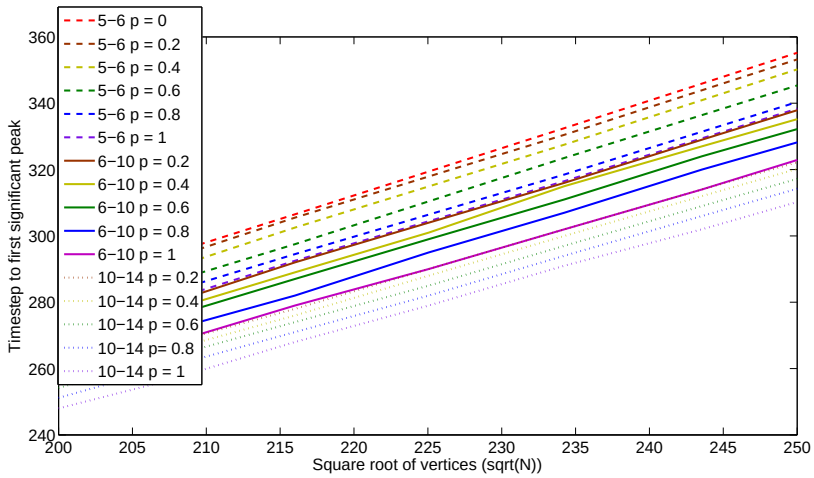

Figure 7.8: Scaling of the time to find the marked state with the size of the lattice with varying connectivity in three dimensions. It is clear that as the connectivity of the structure increases, the time to find the marked state decreases. Note that this is a zoomed in plot showing large lattices sizes, data for $\sqrt{N} < 200$ has been omitted to improve clarity.

We show in fig. 7.8 how the time to find the marked state varies with both the size of the lattice and the connectivity. It is clear that as the connectivity increases, the time to find the marked state decreases, hence the efficiency of the algorithm increases. As the time to find the marked state scales as $O(\sqrt{N})$, we fit to each of these to obtain the prefactor to the scaling of the time to find the marked state. Figure 7.9 shows how this prefactor to the scaling changes with the degree of the underlying structure being searched.

In the three dimensional case, the maximum probability of the marked state scales in a constant fashion, $O(1)$. As such, this scaling does not affect the complexity of the algorithm as in the two dimensional case. However, we do note that this constant value of probability does affect how many times we must run the algorithm to ensure we have the correct result. We show in fig. 7.10 that, in general, as the connectivity of the structure being searched increases, the maximum probability of the marked state also increases.

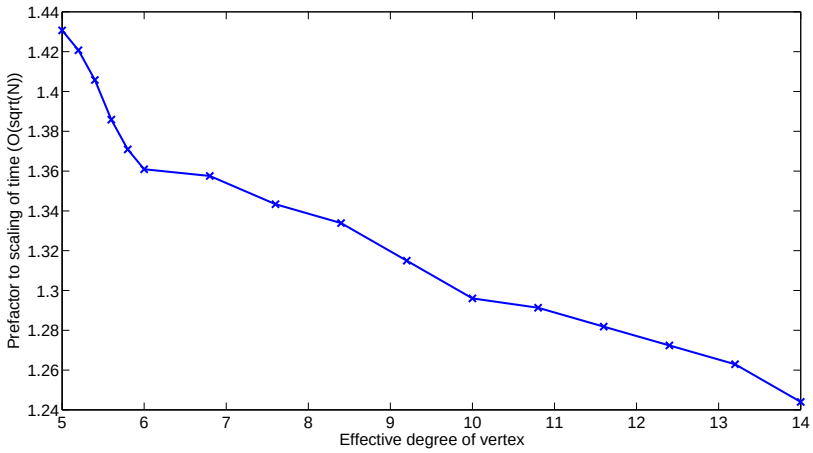

Figure 7.9: Plot to show how the prefactor to the scaling, obtained from the data shown in fig. 7.8, of the time to find the marked state of $O(\sqrt{N})$ changes with the degree of the three dimensional structure being searched.

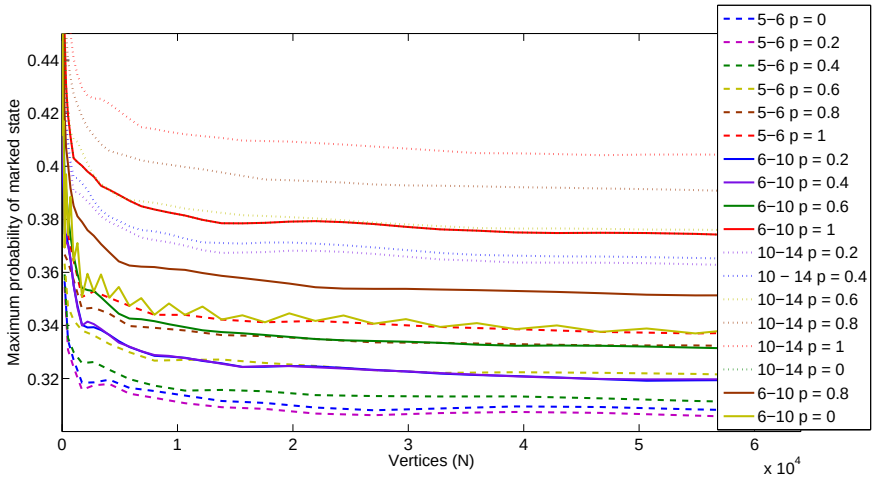

Figure 7.10: Scaling of the maximum probability of the marked state with the size of the lattice with varying connectivity in three dimensions. In general, as the connectivity of the structure increases, the maximum probability of the marked state also increases.

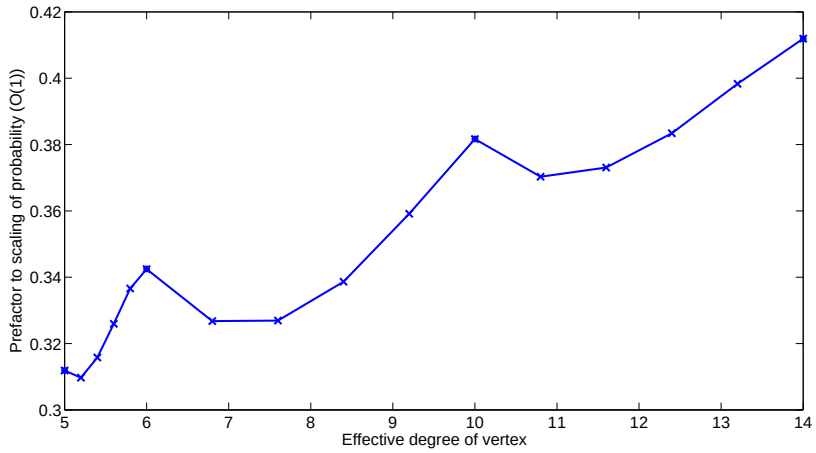

Figure 7.11: Plot to show how the prefactor obtained from the data in fig. 7.10, of the maximum probability of the marked state changes with the degree of the three dimensional structure being searched.

The closer we can get this prefactor to unity, the lower the number of times we must run the algorithm. Figure 7.11 shows how this prefactor to the probability of finding the marked state varies with the degree of the structure being searched. We find the same 'dips' and recurrences in the scaling as in the two dimensional case which can be explained in the same way as previously.

7.3 Discussion

We have investigated numerically how the quantum walk search algorithm introduced by Shenvi et al. [50] is affected by varying connectivity in regular lattices. We use our simple model of tunnelling to allow us to interpolate between structures such as the square lattice ($d = 4$) and the triangular lattice ($d = 6$). With this model, we are able to identify how the prefactors to the scaling of both the maximum probability of the marked state and the time to find the marked state vary with the connectivity of the structure.

The basic scaling of the time to find the marked state, $O(\sqrt{N})$, is not affected by the increase in connectivity but we find the prefactor to this scaling reduces as the connectivity of the structure being searched increases. This is due to the additional

paths the walker can take to coalesce on the marked state, thus increasing the efficiency of the algorithm in both two and three dimensions.

The maximum probability of the marked state is also affected by the connectivity of the underlying structure. We find that additional connectivity does not affect the basic scaling of $O(1/\log_2 N)$ in the two dimensional case. Only moving to three spatial dimensions allows the walker to find the marked state with a constant probability, $O(1)$. However, we do note that in both two and three dimensions the prefactors to this scaling, in general, increase as the connectivity of the structure increases. Again, this increases the efficiency of the algorithm as it may not have to be repeated so many times. We also find that the probability of the marked state does not increases uniformly with the additional connectivity. We see the prefactor in the scaling drop and then recover itself before increasing as the tunnelling strength increases. This is due to the dynamics of the quantum walk on a structure with some broken symmetry, i.e. low tunnelling strength between vertices. We briefly investigated the dynamics of the walk by starting the walker in a single location and monitoring how quickly it spread outwards with varying tunnelling strengths. This confirmed our results for the search algorithm as we found that the spread of the quantum walk also dropped for lower tunnelling strengths before recovering and eventually increasing at higher tunnelling probabilities. However, this work on the spreading of the walk compared to tunnelling strength is by no means exhaustive and it would be interesting to look more deeply into this in the future.

Chapter 8

Quantum walk searching on percolation lattices

8.1 Introduction

In the previous chapters, we have only considered perfect, regular lattices with no defects. In this chapter, we consider a simple form of noise (disorder) and are interested in how this effects the efficiency of the search algorithm. Previous work by Keating et al. [204] has highlighted the effect of Anderson localisation in continuous time quantum walks and also Krovi and Brun [174–176] have shown how defects and a lack of symmetry in continuous time quantum walks can have an impact on the spreading of the walk. Both these factors suggest that the search algorithm will fail as soon as any level of disorder is introduced into the lattice. However, in contrast to these results, the study of the transport properties of discrete time quantum walks on 1D and 2D percolation lattices has recently been presented by Leung et al. [205]. They show that the spreading of a discrete time quantum walk, on a 2D percolation lattice, follows a fractional scaling, i.e. $\langle r \rangle \propto T^{\alpha}$ where $\langle r \rangle$ is the spread of the quantum walk and T is the number of timesteps. This seems to be in contradiction to the previous work in the continuous time context. In addition, Abal et al. [206] have investigated how the quantum walk search algorithm

performs in the presence of decoherence, specifically phase errors in the coin operator. As previously mentioned, we assume that we have a quantum computer with error correction available, and as such are not interested in these errors. Instead, we are interested in any disorder that could be present in an imperfect data structure. We aim to establish how much, if any, disorder the search algorithm can tolerate or if it fails completely. In order to do this, we use percolation lattices to allow us to vary the level of disorder in the lattice.

8.2 Percolation lattices

A percolation lattice is a lattice, for example a 2D Cartesian lattice, which has vertices (site percolation) or edges (bond percolation) randomly missing. The probability, p, of a vertex or edge existing determines the amount of disorder present in the lattice. As the probability increases there reaches a point, p_c, where the structure changes from a set of smaller, unconnected pieces into one larger piece which is almost all connected. At probabilities $p \geq p_c$, there will, in general, be a path from one side of the lattice to the other. We note here that this is only the case for structures with dimension two or more. It clear that any one dimensional lattice must be fully connected in order for a path to exist from one side of the lattice to the other, i.e. $p_c = 1$. Figure 8.1 shows an example of a 2D bond percolation lattice with varying probability of an edge existing. A path from one side of the lattice can clearly be seen for probabilities greater than or equal to the critical percolation probability, $p_c = 0.5$. This percolation threshold is only for bond percolation on a 2D square lattice. Although site and bond percolation lattices exhibit similar behaviour, the critical percolation probability differs, for site percolation $p_c = 0.5928.....$ Other lattices have varying critical probabilities depending on their structure, with many efficient numerical methods developed to calculate them [207, 208]. In this chapter, we are only interested in two and three dimensional lattices, and we summarise the critical percolation probabilities of these in table 8.1. It is fairly obvious that at this critical percolation threshold, the properties of the lattice change significantly. For

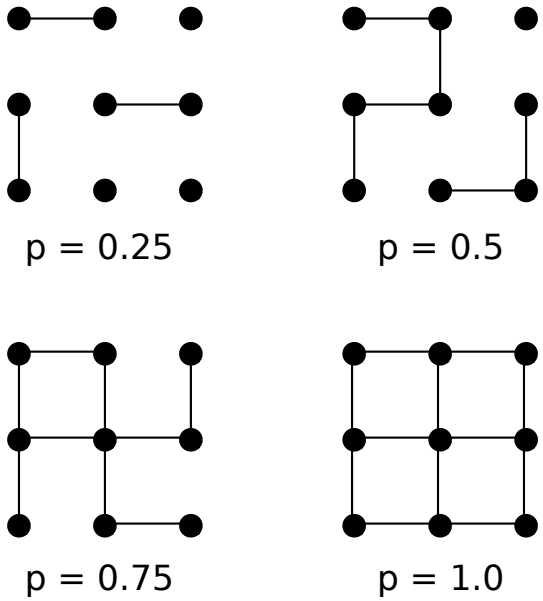

Figure 8.1: An example of a 2D bond percolation lattice with varying levels of disorder determined by the probability of an edge existing. The critical percolation probability, $p_c = 0.5$, for bond percolation clearly shows a path from one side of the lattice to the other.

Table 8.1: Summary of critical percolation probabilities for two and three dimensional lattices

Lattice	Bond	Site
2D	0.5	0.5928.....
3D	0.2488....	0.3116.....

lattices with a percolation probability below the percolation threshold, it is clear that many of the sites in the lattice will be unreachable, whereas above the threshold the opposite is true (though perhaps through a less direct route than in a fully connected lattice). Due to their transport properties, percolation lattices are widely used to model various phenomena including forest fires, disease spread and the size and movement of oil deposits. For a good introduction to both the theory and use of percolation lattices, see for example Stauffer and Aharony [209].

8.3 Search algorithm on percolation lattices

We are using the percolation lattices as a description for the database arrangement that we wish to run the quantum walk search algorithm upon. As the disorder

introduced by using percolation lattices is random, we ran the search algorithm on many different percolation lattices (5000), and averaged over the results. It is obvious that at low probabilities of vertices (or edges) existing, that there may be sections of the graph that the quantum walk is unable to reach. In fact, at very low probabilities, it is likely that the marked state will be in a small, unconnected region of the lattice where it will never be 'found'. In these cases, this means the marked state will only ever be able to attain a small portion of the total probability. We set the condition on the algorithm that the probability of the marked state must reach at least twice the value of the initial superposition in order for it to succeed. Similarly, the time to find this maximum probability is artificially smaller than it should be if the entire lattice was connected. This is due to the walker only having to coalesce on the marked state over a small piece of the lattice. In order to combat this, we set the time to find the marked state as zero if the algorithm failed. If it succeeded, we took the reciprocal of the time to find the marked state. After averaging over many different percolation lattices, we again took the reciprocal of this averaged time in order to give a clearer view on how the algorithm scaled with time. We also set the probability of the marked state to be zero if the algorithm failed.

In order to run the quantum walk search algorithm on percolation lattices, we have to deal with the fact that the lattice is not d-regular. In this setting, we cannot just add self loops to make the lattice regular as in [77] as we want to know exactly how the disorder affects the algorithm. Instead, we take the Grover coin for the degree of the vertex in question and 'pad' it out with the identity operator for the edges that are missing. For example if we have a vertex with just edge 3 missing, the operator would be

$$G_{1,2,4}^{perc} = \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} & 0 & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 1 & 0 \\ \frac{2}{3} & \frac{2}{3} & 0 & -\frac{1}{3} \end{pmatrix}, \tag{8.1}$$

where $G^{perc}_{1,2,4}$ represents the Grover coin with edges 1, 2 and 4 present. In the case of a two dimensional percolation lattice, there are 16 combinations of edges that can be present / missing. For a three dimensional percolation lattice, this increases to 64 combinations. In order to deal with this, we maintain the labelling of the edges as previously and assign a binary number to each edge, depending on whether an edge is present or not. The example above, eq. (8.1), would therefore be 1101. This creates the 2^d combinations we require. There is then a fixed mapping between each binary number and the correct coin for each vertex.

In addition to the coin operator changing, we must also modify the initial state to account for the missing vertices or edges. This could be done in several ways. We try to stick as closely to the initial state of the basic quantum walk search algorithm by just splitting the state into an equal superposition over all the possible edges present.

8.4 Results

We present our results for the quantum walk search algorithm on both two and three dimensional percolation lattices. Due to the computational time required for averaging over many lattices, we only consider site percolated lattices in this work, though we expect a qualitatively similar behaviour in lattices with edge percolation.

8.4.1 Two dimensional percolation lattices

We now show our initial results for the quantum walk search algorithm on two dimensional site percolation lattices. We firstly show, fig. 8.2, how the maximum probability of the marked state varies with both the size of the dataset and the percolation probability. We see, as we would think intuitively, that as the percolation probability drops and the structure becomes less connected, the maximum probability of the marked state decreases. We note that the scaling of the maximum probability initially maintains the logarithmic scaling of the basic 2D lattice before eventually reverting to the scaling of the line, $1/N$, at lower percolation probabili-

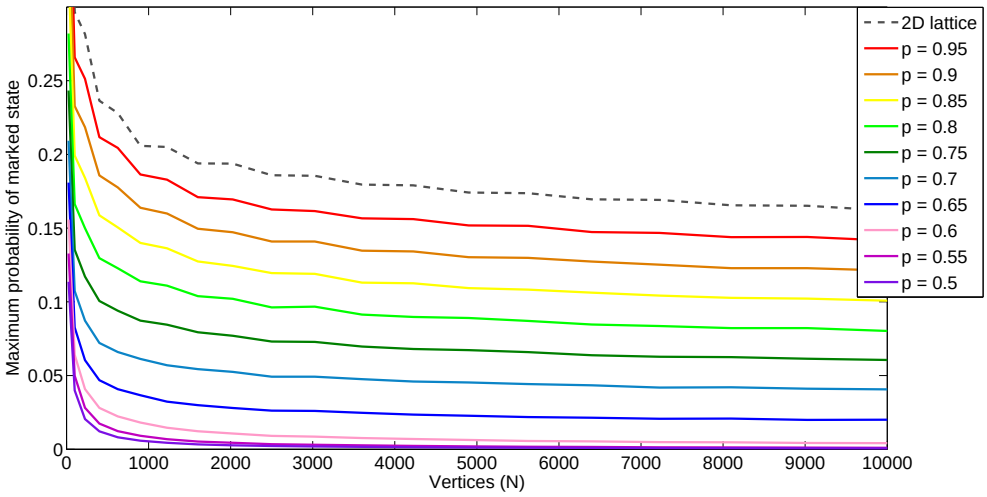

Figure 8.2: Plot to show how the maximum probability of the marked state varies with the size of the dataset and percolation probability in two dimensions.

ties. In the case of site percolation, this change in scaling seems to occur at roughly probabilities below $p \approx 0.65$, not significantly higher than the critical percolation threshold. This is expected as at the critical threshold, the structure has in general a single path from one side to the other, effectively a 1D lattice. Our numerical results match this behaviour, with the scaling of the probability of the marked state matching that of the line at this point. At percolation probabilities higher than the critical threshold, we see a change in the prefactor to the scaling of the maximum probability of the marked state. We show this prefactor to the logarithmic scaling in fig. 8.3. It is easy to see that as soon as the percolation probability passes the critical threshold, $p_c = 0.5928....$, the scaling increases in a linear fashion. We also note here, after investigation on a finer scale, that there is a gradual change in this prefactor scaling around the critical percolation threshold.

The time to find the marked state follows a similar behaviour, gradually changing from the quadratic scaling of the 2D lattice to a classical linear scaling. We show the time to find the marked state for site percolation in fig. 8.4. We see that when $p = 0.6$, the scaling of the time to find the marked state is very similar to the classical run time, $O(N)$.

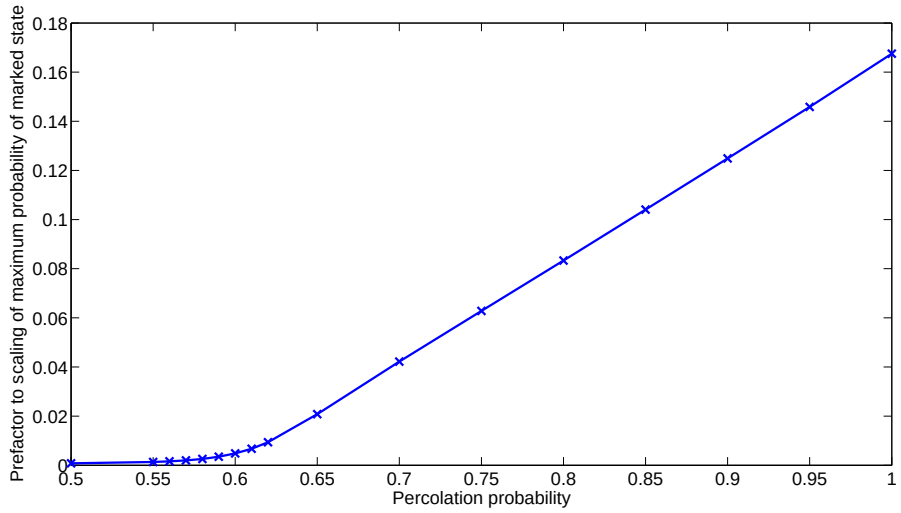

Figure 8.3: Plot to show how the prefactor to the scaling of the maximum probability of the marked state varies with the size of the dataset and the percolation probability for site percolation in two dimensions.

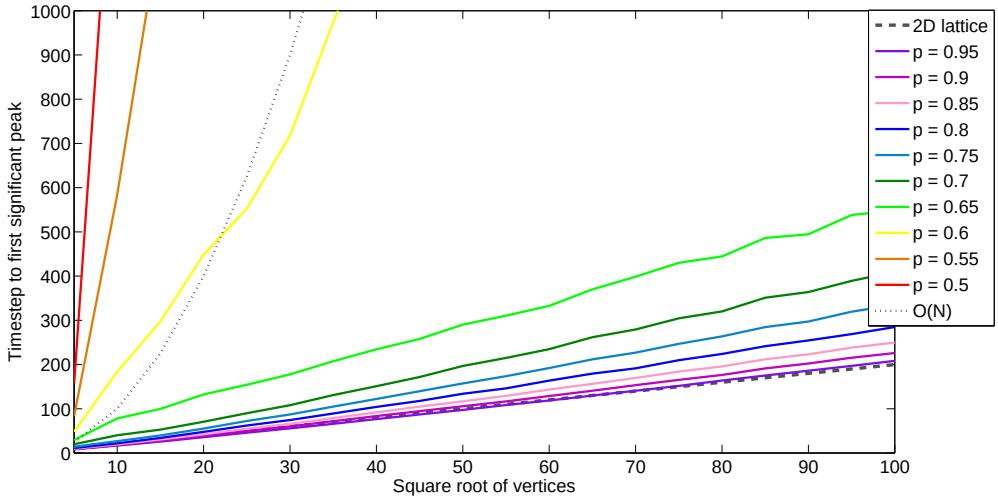

Figure 8.4: Plot to show how the time to find the marked state varies with the size of the dataset and the percolation probability for site percolated lattices in two dimensions.

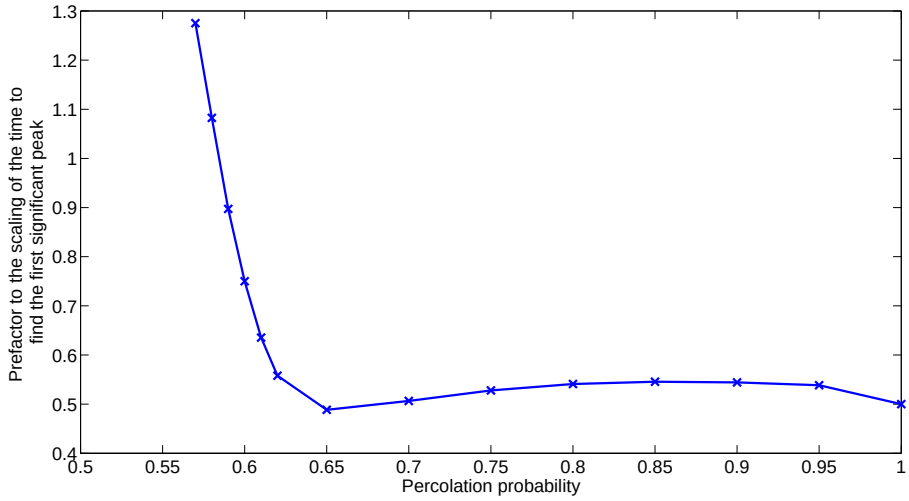

Figure 8.5: Plot to show how the prefactor to the scaling of the time to find the maximum probability of the marked state varies with the size of the dataset and the percolation probability for site percolation in two dimensions.

The kinks in this scaling (and the other percolation probabilities) are just from averaging over many percolation lattices. Given more time, a higher number could be run and thus a smoother scaling obtained. It can be seen that the time to find the marked state seems to retain the quadratic quantum speed up, even in the presence of a non-trivial level of disorder. As in the work of Leung et al. [205], it seems as though the scaling of the time to find the marked state may follow a fractional scaling from quadratic back to linear as,

$$T \propto N^{\alpha}, \tag{8.2}$$

where T is the time to find the marked state and N is the size of the dataset. We follow the analysis in [205] to establish how the scaling of the time to find the marked state varies with the percolation probability. We show, in fig. 8.5, how the value of the coefficient α varies as the level of disorder is increased. We can see the quadratic speedup is maintained, $\alpha \approx 0.5$, for percolation probabilities of roughly $p > 0.65$. Below this probability, the quantum speed up disappears gradually to end at the classical run time when $p = p_c$. This is for the same reason as in the scaling of the

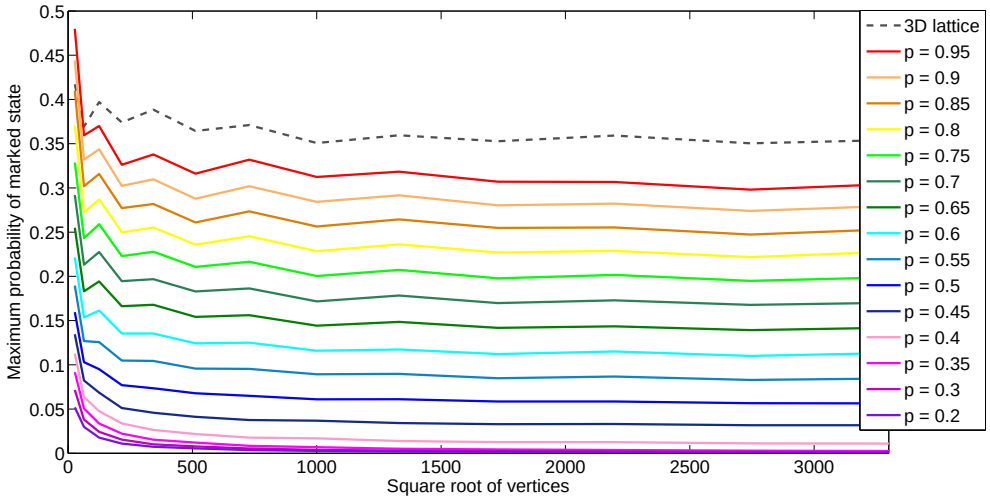

Figure 8.6: Plot to show how the maximum probability of the marked state varies with the percolation probability in three dimensions.

maximum probability of the marked state, at the critical threshold the structure is effectively a line. Below the critical threshold, the algorithm fails (the marked state is probably in a disconnected region). We note here that the coefficient rises and falls slightly for percolation probabilities higher than p_c. This is most probably due to the fact that percolation lattices are random in nature, and we only average over a specific number. If we averaged over more, then we would see a more constant scaling of the coefficient at $\alpha = 0.5$, i.e. a full quadratic speed up.

8.4.2 Three dimensional percolation lattices

We now turn our attention to three dimensional site percolation lattices. We follow the same analysis as in the two dimensional case. We firstly show, fig. 8.6, how the maximum probability of the marked state varies as the percolation probability is decreased. We see, as in the two dimensional case, that the basic scaling of the maximum probability matches that of the three dimensional lattice until the percolation probability drops to roughly the critical percolation threshold, $p_c = 0.3116....$ We show in fig. 8.7, how the prefactor to this scaling of the maximum probability varies with the percolation probability. In the same way as the two

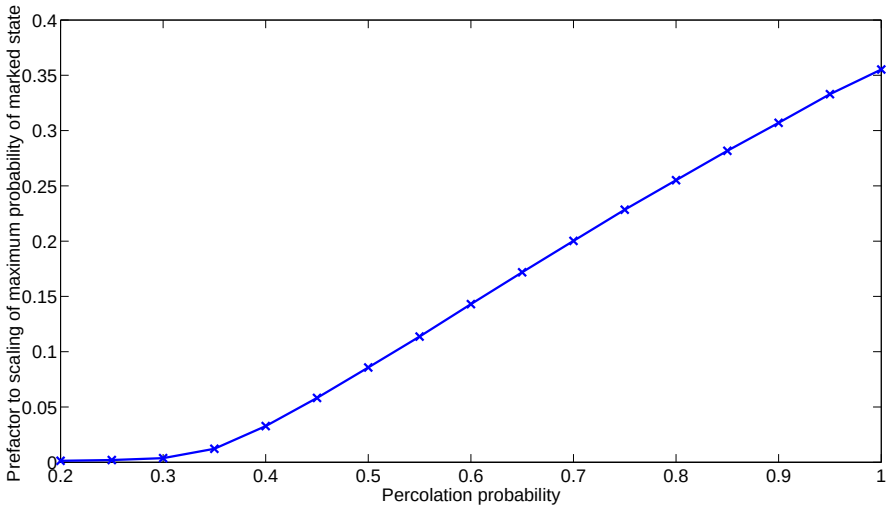

Figure 8.7: Plot to show how the prefactor to the scaling of the maximum probability of the marked state varies with the size of the dataset and the percolation probability for site percolation in three dimensions.

dimensional case, we see an almost linear scaling the prefactor once the percolation probability has passed the critical threshold. The scaling here doesn't seem to be as close as in the two dimensional case. This is probably because in the case of three dimensional percolation lattices, there are many more combinations of lattice which can be created. Averaging over more of these lattices would most probably give a smoother fit.

The time to find the marked state, in the three dimensional case, follows the same behaviour as in the two dimensional percolation lattices. We show in fig. 8.8, how the time to find the marked state varies with the percolation probability. We see, fig. 8.9, as in the two dimensional case, that the scaling coefficient, α, gradually changes from the quadratic speed up to the classical run time. Again, we note that the quadratic speed up is maintained for a non-trivial amount of disorder before gradually changing to the classical run time at the point $p = p_c$. We do note, as in the two dimensional case, that the coefficient falls slightly for percolation probabilities higher than p_c. This can be explained in the same way as the two dimensional percolation lattices, and averaging over more lattices should give a constant value of the coefficient α.

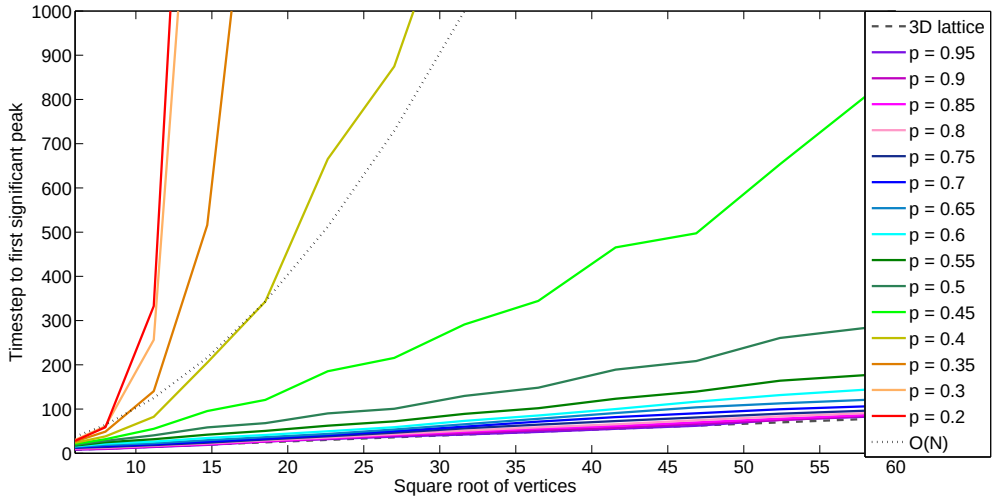

Figure 8.8: Plot to show how the time to find the marked state varies with the size of the dataset and the percolation probability for site percolation lattices in three dimensions.

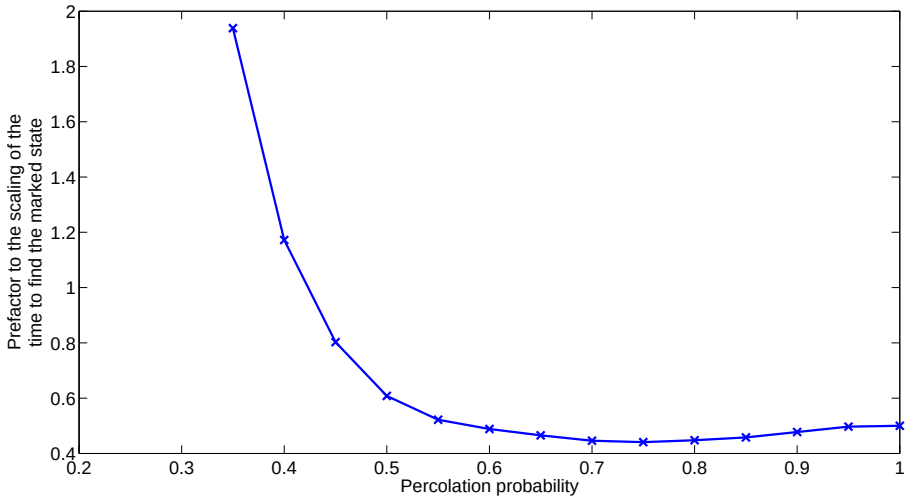

Figure 8.9: Plot to show how the prefactor to the scaling of the time to find the maximum probability of the marked state varies with the size of the dataset and the percolation probability for site percolation in three dimensions.

8.5 Discussion

In this chapter we have studied both two and three dimensional percolation lattices as a way to model disorder in the quantum walk search algorithm. We are interested in how the algorithm performs with increasing disorder. We use percolation lattices as a random substrate for the database arrangement we wish to search.

We find, in both the two and three dimensional cases, that as the level of disorder increases, the maximum probability of the marked state decreases. Whilst the percolation probability is higher than the critical percolation threshold the basic scaling of the maximum probability of the marked state matches that of the basic lattice (in that spatial dimension). Once the percolation probability drop to the critical threshold, this scaling changes to that of the line, $1/N$. This is expected as at this point the structure is effectively a line. We also note the prefactor to the scaling of the maximum probability of the marked state increases linearly once the percolation probability is greater than the critical threshold.

The time to find the marked state follows a similar behaviour. We find that as the disorder increases, the time to find the marked state also increases. Surprisingly though, we note that the quadratic speed up is maintained for a non-trivial level of disorder, before gradually reverting to the classical run time, $O(N)$, as the disorder reaches the critical percolation threshold. This seems to match the results of [205], which show a fractional scaling for the spreading of the quantum walk from a maximal quantum spreading to a classical spreading at and below the critical threshold. However, this is in contrast to the work of Keating et al. [204] and Krovi and Brun [174–176] who highlight the effect of localisation on the quantum walk when defects are introduced into the substrate.

Both these factors indicate that the quantum walk search algorithm seems to be more robust to the effects of disorder and symmetry than the basic spreading of the quantum walk. This could be due to the fact that the initial state of the walker is spread across the whole lattice. We have seen that the algorithm becomes less efficient as the disorder increases, but at percolation probabilities greater than the

critical threshold, the algorithm still seems to be viable, although more amplification of the result may be required.

Chapter 9

Generation of topologically useful entangled states

9.1 Introduction

One of the hardest challenges in experimental quantum information processing is to create and maintain entanglement at varying points throughout the computation. In this chapter, we describe an abstract experimental proposal for the generation of cluster states, a universal resource for cluster state quantum computation. We introduce our basic scheme and the states we are able to prepare in Sec. 9.2. We extend the scheme in Sec. 9.3 before ending the chapter in sec. 9.4 by discussing its benefits and applications.

Cluster state quantum computation, [14], is a different paradigm in quantum computation than the circuit model. It is also known as measurement based quantum computation (MBQC). In MBQC, a general graph state is produced and then each site is entangled with its neighbour(s) by a controlled phase operation (C-Phase),

eq. (9.1), entangling the two together as

$$C-PHASE = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{9.1}$$

In the case of two qubits they are firstly prepared in the $|+\rangle$ state,

$$\begin{aligned} |\psi\rangle &= \frac{1}{\sqrt{2}}(|0_1\rangle + |1_1\rangle) \otimes \frac{1}{\sqrt{2}}(|0_2\rangle + |1_2\rangle), \\ |\psi\rangle &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle). \end{aligned} \tag{9.2}$$

The C-Phase entangling operation is then applied between the two qubits to leave the resultant entangled state,

$$C-PHASE|\psi\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle). \tag{9.3}$$

Single qubit rotations and measurements are used to progress the computation. These measurements are based on the previous results which are fed forward. As all the entanglement is generated when the graph state is produced, no entanglement needs to be generated during the computation, which experimentally is challenging. A cluster state is just a specific graph state, a square lattice. When first introduced it was hoped that this lack of ad-hoc entanglement would mean the experimental implementation would be much easier. However, this has not been the case, although work by Rudolf et al. has shown it to be feasible, [210]. Since its conception, many other schemes for cluster state preparation have also been introduced, [211–214].

Recently, there has been a stimulus in work on cluster state generation using photons, [215–218], and also topological error correction in cluster states, [28–30, 219, 220]. The photonic module, [215], is essentially a 'plug and play' cluster state generator. It can deterministically create cluster states and also uses the mobility

of the photons to enable any output to be connected to any input. In addition to generating cluster states, another focus of the work is on topological error correction. Raussendorf and Goyal [30] introduced the concept of using multiple qubits to encode one logical qubit in an attempt to make it fault tolerant. This is in contrast to the traditional cluster state in which the qubits are not protected from loss channels or errors in the system.

Due to the recent work on topological error correction and the idea of MBQC, the motivation for this work was to develop a scheme to produce a universal resource for MBQC. We also wanted a way to prepare these states which could be scaled up easily. We numerically modelled states we could theoretically prepare. We found we could create many interesting topologies with various possible applications. The recent work by Elham Keshafi et al. [221] on ancilla driven quantum computing has unusual 'twisted graph states' as a resource which our scheme could produce. The unit cell for topological error correction, [29], can also be created (with some Z measurements to remove qubits).

9.2 Basic Scheme

We now present our scheme for the generation of graph states which would be useful in quantum information processing. The scheme we describe is abstract and we do not initially define an architecture in which this could be physically realised. We show some of the useful states we can produce (by numerical simulation) and then discuss the drawbacks of this scheme.

9.2.1 Scheme

Consider a small grid of sites as in Figure 9.1. We imagine vertices of a graph moving across this grid horizontally and vertically by one site in the grid for each arbitrary timestep. If two of these vertices meet at a site we say a link or edge is formed between them. This edge represents the C-Phase entanglement generated between the two vertices which represent qubits. The vertices enter the grid in

the direction of the arrows as a stream of atoms, one entering for each arbitrary timestep. We extend this further and say that each of these sites could be active or inactive, forming an edge only when the site is active. An active site is considered to perform the C-Phase operation between the two vertices (qubits). Therefore instead of creating edges between static vertices, we build our graph states by moving the vertices through a set of 'edge joining' interaction regions, which can be switched on or off. This could be a grid of collisional cavities through which atomic beams are passed and entangled together as proposed by Blythe and Varcoe, [222].

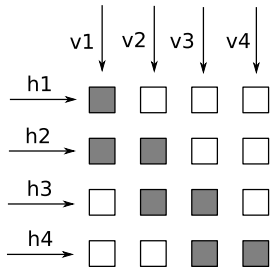

Figure 9.1: Structure we use to create our graph states. We imagine vertices moving in the direction of the arrows advancing by one site in the grid for each timestep. A dark site indicates it is active and so an edge is formed when two vertices pass in the same timestep.

As we have introduced the concept of the vertices moving with each timestep, we must also address the notion of when (in time) a vertex enters the grid. We do this by assigning a generation to each timestep. Using the grid in fig. 9.1 the vertices entering the first row of the grid will be labelled as $h_1 g_4 - h_1 g_3 - h_1 g_2 - h_1 g_1$. This implies that generation 1 passes into the grid first and therefore after four timesteps would be at the site in the top right hand corner of the grid. Using this labelling, it is easy to keep track of which generations of vertices are interacting together. We show how the vertices interact with the active sites and different generations in fig. 9.2. The mix of generations forming edges is obviously highly dependent on the size of the grid and also which sites are active. This dependency and our ability to change these factors allows various different structures to be created. Some of the structures produced by numerical simulation are shown next along with the grid pattern of active sites required to produce them.

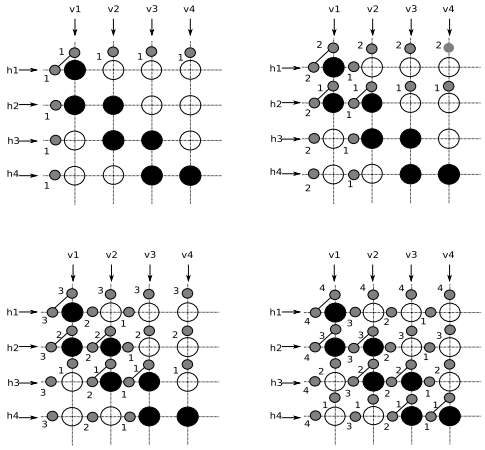

Figure 9.2: First four timesteps of a 4x4 grid. The vertices enter in the direction of the arrows. The black circles indicate active collision sites wheras white indicates the site is switched off.

Due to the vertices moving across the grid as a stream, we find that we get a number of the same structures created. The actual number produced depends on which sites are active and the number of timesteps taken. This scales as $t-(\sqrt{N}-1)$ where t is the number of timesteps and N is the number of sites in the grid. However, some of these structures are incomplete. These come about from the vertices that have only partially traversed the grid when the number of timesteps are completed. These multiple and incomplete structures are shown in fig. 9.3. In the work by Blythe and Varcoe, [222], the sequence of atoms is pulsed at specific times to allow the creation of one structure as opposed to a continuous stream here.

9.2.2 States produced

We show several different cluster states we have produced by numerical simulation in order to show the variety of topologies our scheme can produce. All show the structure produced and the grid of active sites required to produce it. The number of timesteps was 10 for all of the examples. We can see the creation of a cluster state for universal quantum computing in figs. 9.4, 9.5 and 9.6. In fig. 9.7 we see additional links from a central structure. These could be used as ancilla qubits for algorithmic or error correction purposes. Finally in figs. 9.8 and 9.9 we see the initial unit cell of both cubic and and hexagonal lattices. If all sites are active we obtain the

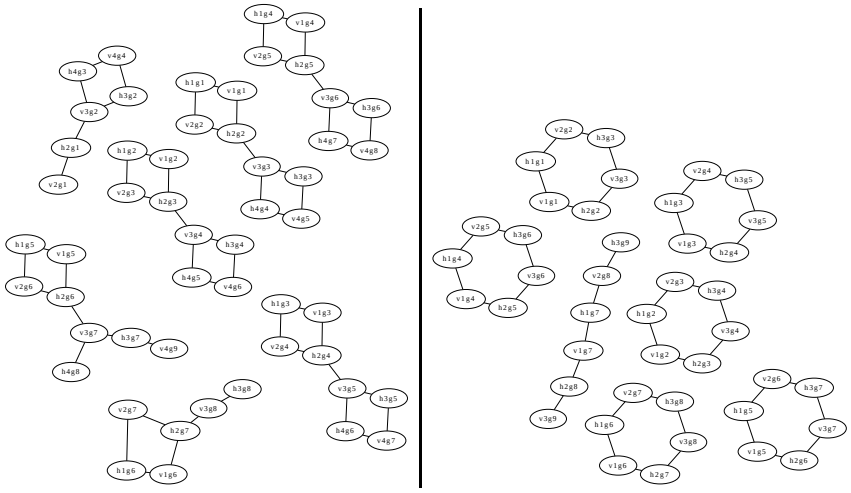

Figure 9.3: Multiple structures produced after 10 timesteps. The incomplete structures can easily be seen. Left - Full set of structures produced from the grid in fig. 9.5. Right - Full set of structures produced from the grid in fig. 9.9.

most highly connected state possible in this scheme. This is a cube with its corners connected diagonally as shown in fig. 9.10. This is in essence a superposition of all other structures that could be produced. Other more exotic structures can be created using a different combination of active sites.

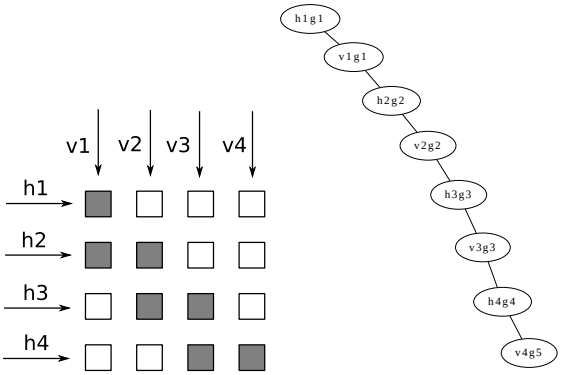

Figure 9.4: Grid of active sites and structure produced. The structure produced is a 1D lattice.

9.2.3 Drawbacks

In the examples we have shown it is clear that each structure created can only have a maximum of eight vertices due to the size of the grid. Obviously the grid is not static in size, it can be be a square of any dimension N, $\sqrt{N} \times \sqrt{N}$. We find that

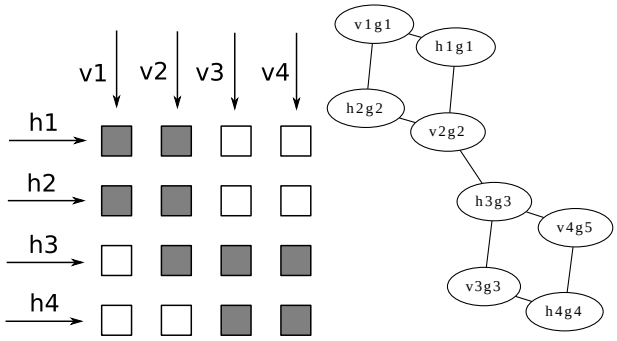

Figure 9.5: Grid of active sites and structure produced. The structure produced is the formation of a 2D lattice but not all the bonds have been formed.

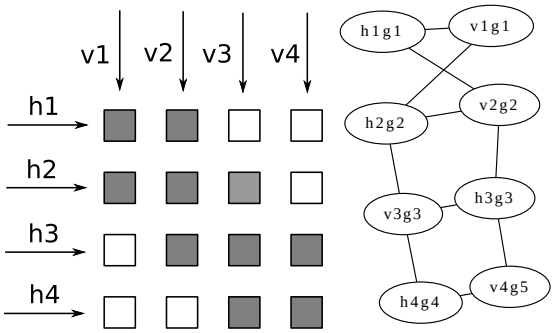

Figure 9.6: Grid of active sites and structure produced. The structure produced is a 2D lattice of depth two.

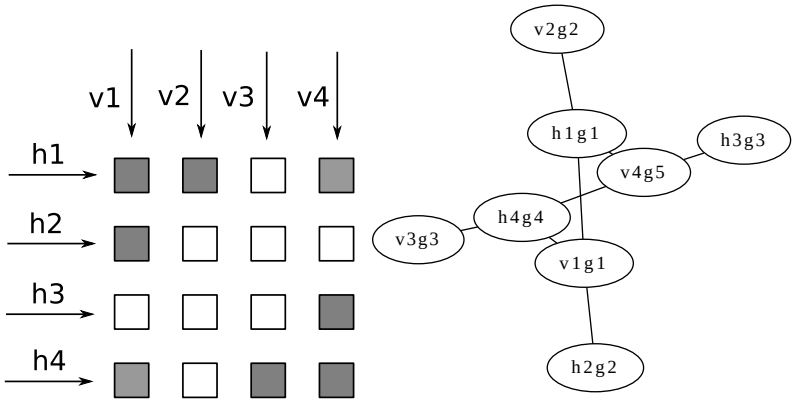

Figure 9.7: Grid of active sites and structure produced. The structure produced is a square lattice with an additional link on each vertex.

as the grid increases in size, the number of vertices in each structure also increases. The number of vertices present in any structure created can only be a maximum of $2\sqrt{N}$. Similarly, the maximum number of edges a vertex can form is $\sqrt{N}$, depending on how many active sites a vertex passes through. This restricts both the size and

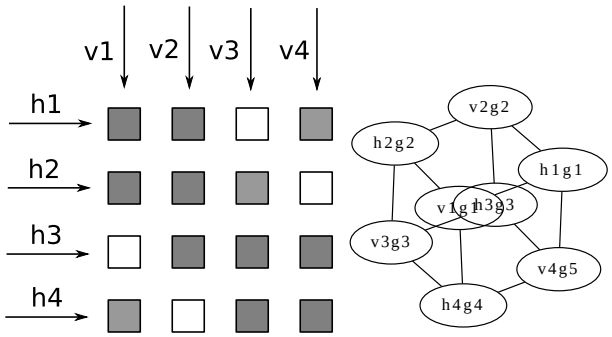

Figure 9.8: Grid of active sites and structure produced. The structure produced is a cube.

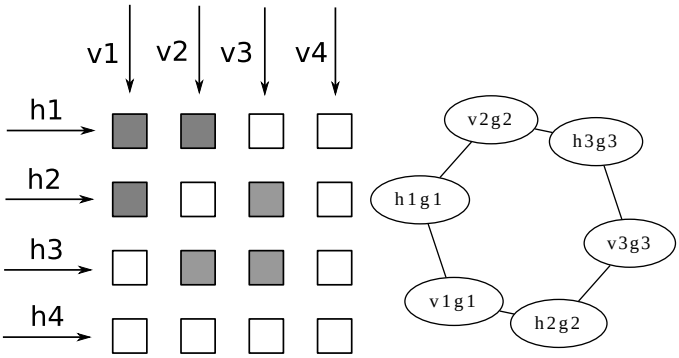

Figure 9.9: Grid of active sites and structure produced. The structure produced is a single cell of a hexagonal lattice.

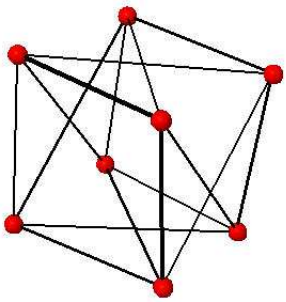

Figure 9.10: Structure produced when all sites are active. This is the most highly connected structure that can be produced from this model.

the topology of the structures we are able to create. However, as the grid increases in size we do not have to stick to a specific pattern across the entire grid. Another option is to repeat an existing one on the diagonal. For example, if we have an 8 x 8 grid we could have any one of the patterns from figs. 9.4 to 9.9 repeated twice on the diagonal. This will create additional structures of the same form which could then be linked to form larger structures. This can be achieved by activating the adjacent skew diagonal elements between the repeated pattern. This is shown in figs. 9.11 and 9.12 and it is clear that any of the previous structures described could be linked in this way. The only thing that would limit the number of structures we could link together would be the size of the grid that could be constructed. However, as the grid is expanded in this way there are many inactive sites meaning it will probably not scale particulary well in a physically realisable device.

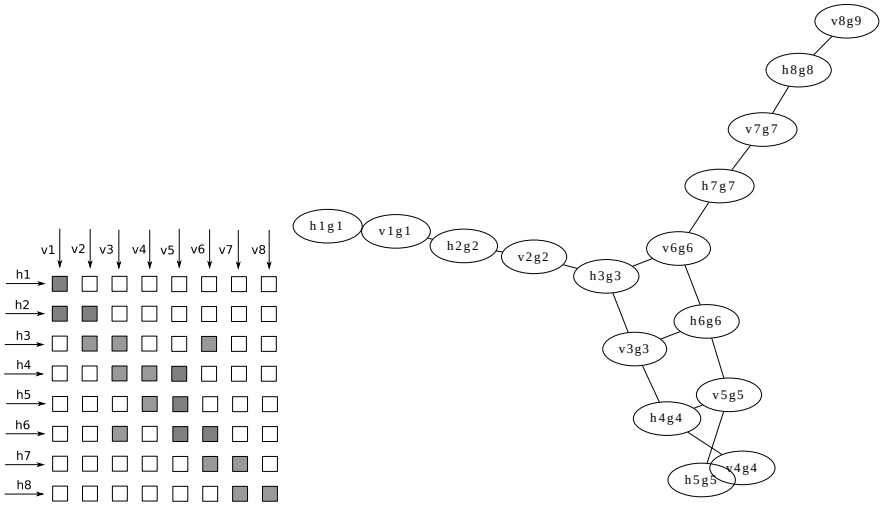

Figure 9.11: Grid of active sites and structure formed for the pattern in fig. 9.5 repeated and 'linked' together. If the number of timesteps was increased then more of the structure would link together at each timestep. The grid has two patterns which would create a 1D lattice, the skew diagonals link the two to form the two depth 2D lattice.

9.3 Extended Scheme

We now modify our scheme to allow the creation of larger structures using the same initial grid. We show some of the useful states we can produce (by numerical

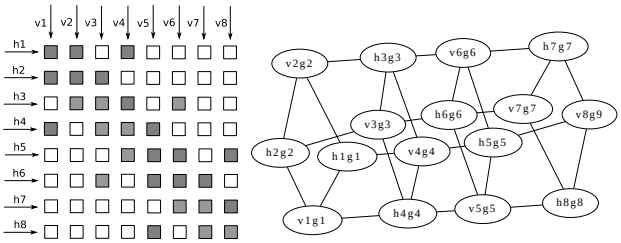

Figure 9.12: Grid of active sites and structure formed for the pattern in fig. 9.7 repeated and 'linked' together. The grid has two patterns which would create cubes, the skew diagonals link the two to form a 'chain' of cubes.

simulation).

9.3.1 Scheme

In order to solve some of the drawbacks in the initial scheme, we amend it slightly allowing the construction of much larger structures. In the extended scheme, we allow the vertices (qubits) to enter from either side of the grid in any pattern. This is shown in fig. 9.13 which shows the grid 'filling' up for the first few timesteps. Due to the change in where and when the vertices meet, we find the structures created are in effect 'infinite' in length, the only limiting factor is the number of timesteps the system is run for (assuming we can keep the system coherent for this time). This would allow any number of computational steps to be performed on the qubits. If we stick to creating just a 2D lattice (cluster state) then the size of the grid increases linearly with the depth of the cluster produced. This is clearly shown next where we show some of the examples of states produced, again by numerical simulation. The size of the grid compared to the structure produced is now much smaller than the original scheme.

Allowing the vertices to enter the grid in this fashion means that only four copies of the same structure are obtained. As is shown by the examples, the structures are much more similar in comparison to the original scheme. As only four structures are produced for any grid pattern it means that the connections are formed much faster. This ensures much of the structure is always complete and it is just the start and end which has less connections. This is shown in fig. 9.14 and is due to two factors.

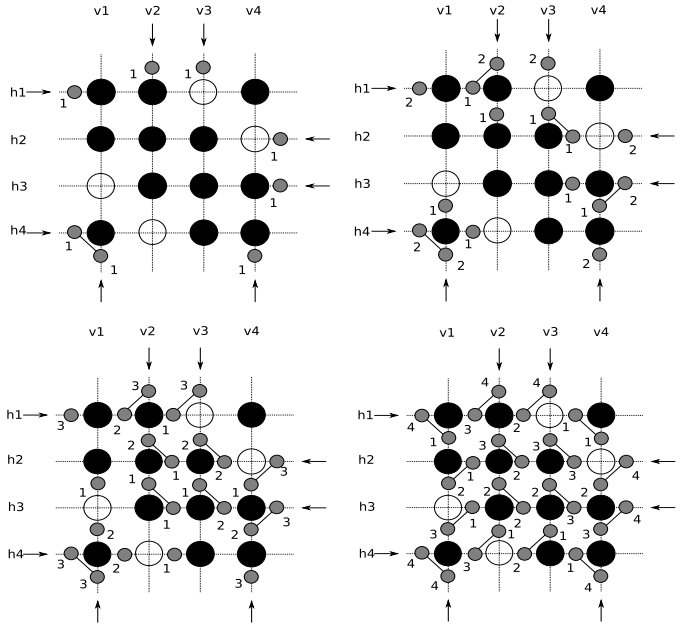

Figure 9.13: First four timesteps of a 4x4 grid as it 'fills up'. The vertices enter in the direction of the arrows. The black circles indicate active collision sites wheras white indicates the site is switched off.

The first is due to the grid filling up at the start of the generation of the cluster which can be solved by starting the computation $\sqrt{N}$ steps later. The second is at the end of the structure which is incomplete as some of the vertices have not passed through the entire grid when we reach the number of required timesteps. This can be solved by ensuring the number of timesteps is an additional $\sqrt{N}$ than required for the computation. Therefore, overall we have a constant overhead of $2\sqrt{N}$ timesteps to add to any computation.

Our initial motivation was to generate a scaleable scheme for cluster state generation. We also found that we could easily create much more interesting structures as in the original scheme. If certain sites are activated we find the structure can loop and join itself again creating 3D structures and 'rings'. Other options are to repeat the basic pattern of the grid down the diagonal and then link them together. This could be used to form a 'ring of rings' for example. One interesting thing about copying the pattern in this sense is that we can create a 'ring' of any number of other structures. These individual structures can also have any number of vertices linked together.

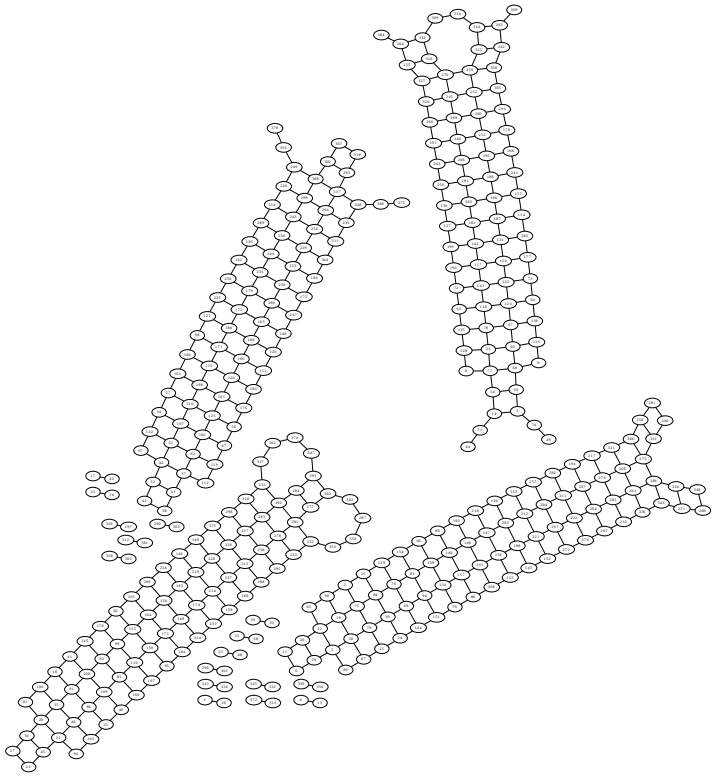

Figure 9.14: An example of the four structures created. The structures show the incomplete parts at the start and end.

In order to show that we produce four individual structures, we describe the first four timesteps using a simple 2x2 grid. This will produce four 1D chains of entangled atoms as shown in fig. 9.16. We note here that as the atoms only pass through a specific number of collisional zones, the timing is known and they pass each other in orthogonal directions there is no problem in distinguishing the atoms in the chain. Figure 9.15 shows in detail the first four timesteps of generating the 1D cluster states.

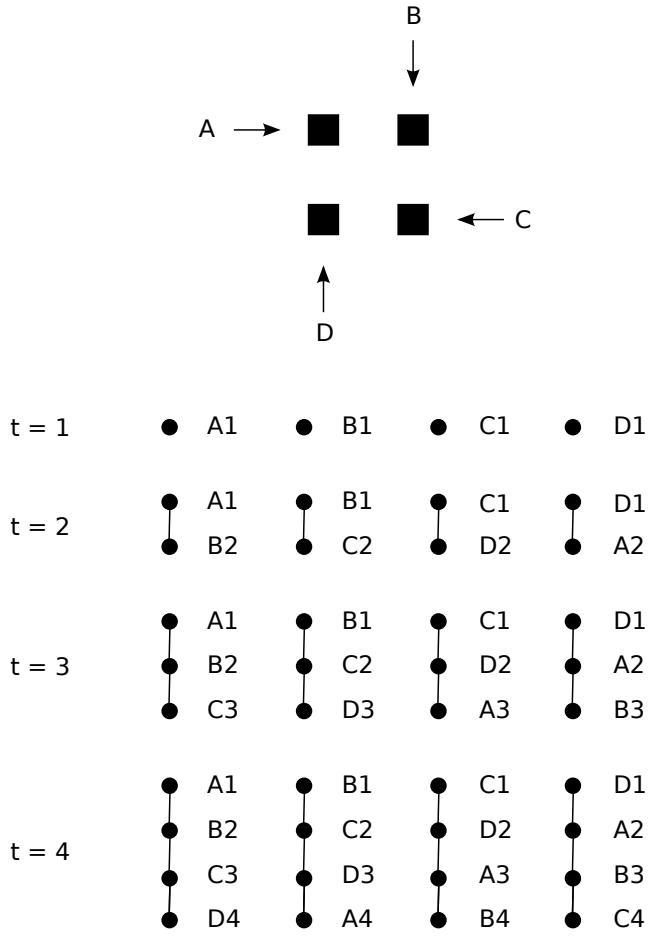

Figure 9.15: Diagrammatic representation of the first four timesteps of the generation of 1D cluster states.

9.3.2 States produced

We show several different structures we have produced by numerical simulation in order to show the variety of topologies our extended scheme can produce. All show the structure produced and the grid of active sites required to produce it.

We can see the creation of a cluster state for universal quantum computing in figs. 9.16, 9.17 and 9.18 and can clearly see the linear scaling of the grid with cluster depth. Figures 9.19 and 9.20 show how activating specific sites allows depth four or eight clusters to loop around to form a cube or octagon respectively. The 3D structures shown in figs. 9.19 and 9.20 can be repeated in a larger grid and then linked together in a similar way. This then achieves a 'ring of rings' topology where the number of vertices in either ring is just dependent on the size of the initial grid.

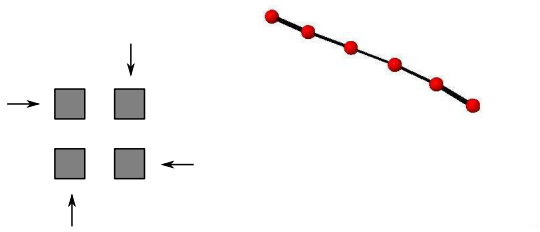

Figure 9.16: Grid of active sites and structure produced. 1D cluster with no restricted length.

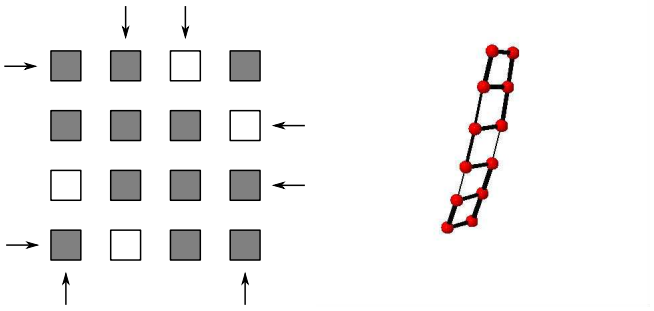

Figure 9.17: Grid of active sites and structure produced. 2D cluster of depth two with no restricted length.

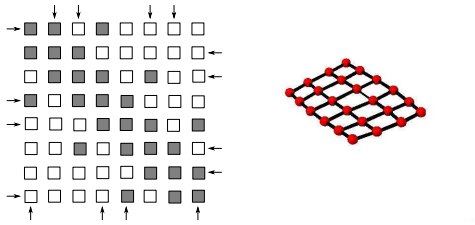

Figure 9.18: Grid of active sites and structure produced. 2D cluster of depth four with no restricted length.

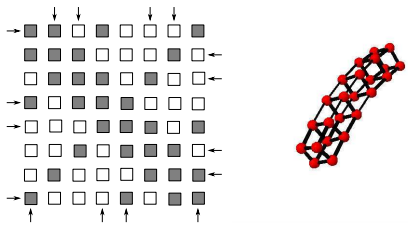

Figure 9.19: Grid of active sites and structure produced. 2D cluster of depth four looped round to form a cube of no restricted length.

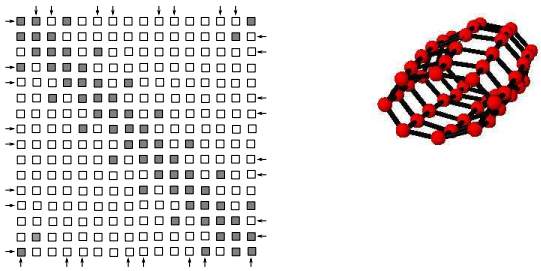

Figure 9.20: Grid of active sites and structure produced. 2D cluster of depth 8 looped round to form an octagon of no restricted length.

9.4 Discussion

We have presented a scheme to allow the creation of graph states. Our scheme could be applied to various architectures, however we do note its relevance to the cavity QED scheme previously mentioned, [222]. In this way we envisage the vertices as atoms and the edges of the graph as entanglement between these atoms. This architecture would seem to lend itself to quantum information processing applications. The basic cluster state produced by the extended scheme is a universal resource for measurement based quantum computing. The scaling here is better than many other schemes that have been proposed. We only need to double the size of the grid to get a structure of double the depth (double the qubits in the cluster state). It is clear that doing this will create many inactive collision sites and as such is wasteful. However, if experimentally implemented these inactive sites need not be implemented, instead just creating the active zones and controlling the timing of the atoms is sufficient. Obviously, maintaining the timing and coherence of the qubits during the computation would still represent a significant challenge. As the computation can be run for an arbitrary time, the length of the structure is in effect 'infinite' as long as the system remains coherent. The unit cell for topological error correction can also be created (with some Z measurements to remove qubits).

In the scheme, we have assumed that when two vertices pass each other at an active site a full link or edge is always formed. This is an ideal case and we intend

to amend our numerical simulations to include a probability of an edge forming. We would imagine there to be some critical probability similar to percolation theory where the structures are formed / not formed. This could mean the structures could be used in percolation problems when studying stuctures with broken or missing links. We notice that as there are multiple copies of the same structure it is unlikely that the same bonds will form in each one if there is chance of an error. As such we may be able to use some form of 'majority rules' or entanglement distillation scheme, [223], to ensure we have a complete structure. This would however remove many of the additional structures that are in effect created for 'free'.

The additional structures could easily be used to our advantage. This could be one of two options - multiple copies running either the same computation or different parts of a program. If we ran the same program on all the copies this would give a higher probability of success if there was a possibility of error as discussed above. It would also mean a probabilistic algorithm would have to be run less times as we would be in effect running it four times in one run. The other option would be to use each structure as a 'thread' in a multithreaded quantum computer. Activating specific sites in the grid at certain times would then allow the connection of the structures temporarily to allow communication between threads. This would be much more complex than the previous option but could also give benefits such as speed for example. The main problem with the multithreading idea would be communication between threads. This would be in the timing of the links between structures to pass information from one structure to another. It would also require longer coherence times as the states may need to be 'stored' temporarily if communication is blocked with another structure to avoid a read-write error between threads in the same way as classical multithreading.

We intend to extend this work to provide a more physical setting in which we can discuss errors and implementation more thoroughly. The scheme we present here can not produce every structure as the vertices (qubits) cannot be directed from output to input as in [215]. We will address this in a specific architecture. Another

possibility of the scheme is useful applications of the more exotic structures shown in the extended scheme. This would most likely be in the area of topological error correction mentioned briefly above. The unit cell can be created which is the basis of the error correcting schemes introduced by Raussendorf and Harrington in [29]. An extension of error correction in this scheme would be to use multiple qubits in order to encode one logical qubit. This redundancy allows for fault tolerance but how many qubits we could use would obviously depend on how many qubits could physically be realisable.

Chapter 10

Conclusions

In this chapter, we conclude the thesis with a summary of our results. We discuss the main results of each section of the thesis: Universal quantum computation, Quantum walk search algorithm and also the cluster state generation scheme. We also give details of further work which could be carried out.

10.1 Universal computation using the discrete time quantum walk

In Chapter 3, we gave a scheme showing the discrete time quantum walk is universal for quantum computation. We give an explicit graph construction which maps a standard circuit model computation, to a graph on which a discrete time quantum walk traverses (from left to right), thus performing the computation. We also discussed the difference in resources, in the size of the graph and gates required, compared to the circuit model. The scheme we present is an extension of work done by Childs [49], who showed the same results for the continuous time quantum walk. The structures we show for our universal gate set are similar in construction to the work of Childs, but require a higher degree at each vertex. This is due to the additional degree of freedom provided by the coin in the discrete time case. We note here that since this work was completed and published, Underwood and Feder [157] showed

universal computation using a 'discontinuous' time quantum walk, which combines both the results we show here, and the ones of Childs [49]. This construction takes discrete steps of continuous evolution, utilising the advantages of both schemes. In this discontinuous time construction, the walker takes advantage of perfect state transfer to perform a deterministic computation for all states, as opposed to the continuous time case, but is able to do this without the addition of the coin degree of freedom, required in our discrete time construction. All these constructions show the power of the quantum walk, and that any quantum algorithm can be recast as a quantum walk algorithm.

Further work in this area would involve trying to decompose our gate structures to reduce the degree at each vertex. This would then reduce the overhead required in the computation. In addition, this work has opened up new avenues of exploration for perfect state transfer. As pointed out in [170], the Grover coin transfer introduced in this work can be used to create arbitrary graphs which allow perfect state transfer. This could have implications for the possible development of routing algorithms for quantum networks or new quantum algorithms where the problem can be cast as a large graph structure, i.e. k-SAT.

10.2 Efficiency of the quantum walk search algorithm

Chapters 5-8 deal with the main part of this thesis, investigating factors which affect the efficiency of the quantum walk search algorithm. All the work done here shows that there are many factors which affect the efficiency of the quantum walk search algorithm. Past research has mainly focused on the spatial dimension of the structure being searched. This was also our initial motivation and although there is clearly a strong dependence on this, there are also secondary dependencies on both the connectivity of the structure, the boundary conditions and also any disorder (symmetry breaking) present.

We started in Chapter 5 by discussing the algorithm on the 1D line where we know, by simple argument, there can be no quantum speed up. We discussed the

factors which can affect the algorithm on the line, before extending to other lattices which have fixed boundary conditions, including fractal structures. The work here highlights the importance of symmetry in the quantum walk search algorithm, in both the coin operator and the structure to be used to represent the dataset.

This work could be extended in two ways. Firstly, the work on fractal structures could be extended to include those fractals which have a spatial dimension between 2 and 3. Although the fractal structures studied are not viable as a database arrangement for the search algorithm, other higher dimesional ones may be. It would also be of interest to complete a more thorough study of the dynamics of the discrete time walk on fractal structures in general. Secondly, in the case of the search on non-periodic structures, specifically the two and three dimensional lattices, we take both boundaries to be non-periodic. It would be interesting to see if the efficiency of the algorithm varied if just one of the boundaries was fixed and the other was allowed to be periodic.

Chapter 6 investigates the effect of the spatial dimension on the algorithm. We investigated this dependence in two ways. We firstly introduced a simple form of tunnelling to allow us to interpolate between structures of differing spatial dimension. For example, using this tunnelling operator we are able to gradually change a 2D Cartesian lattice to a 3D cubic lattice. Secondly, we studied lattices of differing dimensions. In the 3D case, we maintain the width and height of the lattice and vary the depth of the lattice. This allows us to change between a 2D square Cartesian lattice to a 3D cubic lattice gradually, in a different way to the tunnelling operator. We find, using the tunnelling operator, that the search algorithm is highly dependent on the spatial dimension, changing in scaling as soon as additional edges are present, no matter the strength of these edges. In contrast to this, as we vary the dimensions of the lattice, we see a gradual change from the scaling of one spatial dimension to another. This shows that although the spatial dimension of the structure in question is of high importance, the symmetry also plays an important factor.

In Chapter 7, we use the same tunnelling operator introduced in Chapter 6 to

study the effect of connectivity on the algorithm. We maintain the spatial dimension of the structure being walked upon, but gradually change the connectivity. We find that as the connectivity of the structure increases, the maximum probability of the marked state also increases. This is due to the additional paths the walker is able to take to coalesce on the marked state. In the same way, the time to find the marked state decreases, thus increasing the efficiency of the algorithm. This increased efficiency would then lead to less amplification of the result, i.e. less repeated runs of the algorithm. This may be useful when experimentally running the algorithm, as there may be physical restraints on the topology of the database arrangement, or a cost associated with each connection.

Further work here would involve extending the study of the basic quantum walk on structures with varying connectivity / spatial dimension using the tunnelling matrices. We only touched on the basic dynamics here, continuing this to other structures could allow the investigation of mixing and hitting times on lattices with varying connectivity. In addition, there are numerous other ways we could increase (or decrease) the connectivity to investigate how the quantum walk search algorithm behaves. For example, in the case of the three dimensional lattices, the degree of each vertex of the underlying structure could be increased to 22 by allowing diagonal connections across the initial cubic structure.

Finally in Chapter 8, we study how disorder (symmetry) affects the algorithm by studying it on percolation lattices. By varying the probability of a vertex (or edge) existing, we can easily vary the underlying database arrangement, changing the level of disorder present. We find that the search algorithm is able to tolerate small amounts of disorder, maintaining the quadratic speed up, before gradually losing any speed up to match the classical run time. In fact, it is at the critical percolation threshold that the algorithm returns to the classical run time before failing completely for percolation probabilities below this level. At the critical percolation threshold, there is in general, a single path from one side of the structure to the other, thus the structure is effectively a 1D lattice where we already know that no

quantum speed up is possible.

Further work here would likely involve running the simulations of the search algorithm on larger percolation lattices to establish a large N scaling. Also, averaging over many more percolation lattices would also give a better idea of the scaling behaviour in a smoother fashion. In addition, we would also like to compare the efficiency of the algorithm on both site and edge percolated lattices. This should be qualitatively similar as it has already been shown that the spreading of the quantum walk of the two types of percolated lattice is similar, as shown in [205].

From the work in this thesis, it is clear that the factors which affect the quantum walk search algorithm are complex and hard to isolate. There are several way, in addition to those just mentioned, to extend the work we have shown here. Firstly, the study of the search algorithm on other families of graphs could shed light on the interdependency of the factors which affect the efficiency of the algorithm. These could include planar or general graphs of which we have completed some initial studies. General graphs can often be considered as small world networks, which then opens up a variety of possibilities in the realm of complex or social networks as preliminary studied in [120].

Additional work could involve trying to establish if there is a modified approach of the search algorithm which could match the full quadratic speed up in two spatial dimensions. This seems unlikely as previously mentioned, but no proof has been shown that such an algorithm cannot exist. Some recent attempts have enabled the run time in two dimensions to be lowered to the current fastest of $\sqrt{N \log N}$, but currently it seems a radical new approach may be needed to improve upon it (if it can be at all).

Another avenue of research could be the addition of multiple marked states to the algorithm. In both Grover's algorithm and the initial quantum walk search algorithm, the addition of multiple marked states, M, just changes the scaling thus, $O(\sqrt{N}/M)$ for the case of $D > 2$. Therefore, one of the answers we are searching for can be found in a time quicker than just one state by a factor of M. It would be

interesting to consider whether this same scaling holds in all the structures we have studied. It could be that in some of the non-periodic structures this is not the case.

Finally, recent work into viable quantum memories casts problems in a quantum walk setting where the spreading needs to be minimised [224]. This is in effect the opposite problem to that which we have considered. In the search algorithm, we are trying to ensure the walker is able to spread and search through a structure as efficiently as possible. Using knowledge of these same factors, it may be possible to establish an underlying graph (and dual) which is able to minimise these same dynamics, thus allowing a topological quantum memory to exist for longer timescales.

10.3 Generation of topologically entangled states

In Chapter 9, we introduced an experimental proposal to create cluster states, a universal resource for quantum computation. This scheme is abstract and could be applied to various architectures, though we note the relevance to the cavity QED scheme of Varcoe and Blythe [222]. Streams of atoms would pass through cavities (or electric field zones) which mediate a C-Phase interaction. This entangles any atoms which pass through the cavity at the same point in time. The scaling of the cluster state fairs better in this scheme than other proposals. In our scheme, we see a linear increase in the depth of the cluster state: When we double the size of the grid, we get a doubling of the depth of the cluster state produced. The scheme can be used to create other useful entangled states including the unit cell for topological error correction. It can also be used to create 'structures within structures' which could provide uses in fault tolerant quantum computation.

There is much further work which could be accomplished in this area. Firstly, we assume ideal conditions in all parts of the scheme. The timing of the streams of atoms must be exact in order for the atoms to become maximally entangled in the cavities. Investigation of the effect these errors have on the scheme and the cluster state it produces would be of use if the scheme was to be experimentally realised. If

atoms in the stream were missing or the timing of the streams became unsynchronised, this may destroy the entire cluster state that is produced. However, it may be possible to use some kind of distillation scheme to produce a full cluster from several incomplete structures. There may be some critical level of atoms present / 'on-time' to allow a viable cluster to be produced, similar to the percolation threshold in percolation lattices. In addition to errors, investigation of the possibility of using the additional structures produced to create some kind of multithreaded quantum processor would also be interesting. The main drawback here would be the propagation of errors through the scheme and communication / storage of information between the 'threads'.

Appendix A

Example of source code

Basic source code for a simulation of the quantum walk search algorithm on a 2D Cartesian lattice.

```
clear all
d = 4; %%%degree of graph
la = 1;
latticewidth = 10;
p=1;
marked = zeros(ceil(sqrt(latticewidth*latticewidth) * pi),latticewidth
    -4);
maximumprob = zeros(latticewidth-4,1);
for l = 5:5:latticewidth
    save('data/loop.txt','l','-ascii','-append');
    check = 0;
    checka = 0;
    timesteps = 30;
    trans = zeros(d,d);
    for a = 1:1:d
        for b = 1:1:d
            if a==b
                trans(a,b) = -1 + 2/d;
            else
               trans(a,b) = 2/d;
            end
```

```
    end
end
vertices = l^2;
rootv = sqrt(vertices);
numberrows = vertices/rootv;
shift = zeros(vertices,d);
operator = zeros(vertices,d);
timesteps = ceil(sqrt(vertices) * pi);
markedstate = 20;
markedtrans = -trans;
unmarkedtrans=trans;
coeff = 1/sqrt(vertices*d);
coefflist(la) = coeff;
for a = 1:1:vertices
    for b = 1:1:d
        shift(a,b) = coeff;
    end
end
prob = zeros(vertices,ceil(timesteps));
probmarked = zeros(ceil(timesteps),1);
for a = 1:1:vertices
    for b = 1:1:d
        prob(a) = prob(a) + abs(shift(a,b))^2;
    end
end
for t = 2:1:timesteps
    for a = 1:1:vertices
        for b = 1:1:d
            if shift(a,b) ~=0
                if a == markedstate
                    trans = markedtrans;
                else
                    trans = unmarkedtrans;
                end
                c = zeros(d,1);
                c(b,1) = shift(a,b);
```

```
            c = trans*c;
            for e = 1:1:d
                operator(a,e) = operator(a,e) + c(e);
            end
        end
    end
end
check = 0;
for a = 1:1:vertices
    for b = 1:1:d
        check = check + abs(operator(a,b))^2;
    end
end
shift = zeros(vertices,d);
for a = 1:1:vertices
    for b = 1:1:d
        if operator(a,b)~=0
            if a<=rootv
                %bottom row
                if a==1
                    %bottom left
                    if b == 1
                        shift(a+rootv,b+2) = shift(a+rootv,b
                            +2) + operator(a,b);
                    elseif b == 2
                        shift(a+1,b+2) = shift(a+1,b+2) +
                            operator(a,b);
                    elseif b == 3
                        shift(a+vertices-rootv,b-2) = shift(a+
                            vertices-rootv,b-2) + operator(a,b
                            );
                    elseif b == 4
                        shift(a+rootv-1,b-2) = shift(a+rootv
                            -1,b-2) + operator(a,b);
                    end
                elseif a==rootv
```

```
                %bottom right
                if b == 1
                    shift(a+rootv,b+2) = shift(a+rootv,b
                        +2) + operator(a,b);
                elseif b == 2
                    shift(a-rootv+1,b+2) = shift(a-rootv
                        +1,b+2) + operator(a,b);
                elseif b == 3
                    shift(a+vertices-rootv,b-2) = shift(a+
                        vertices-rootv,b-2) + operator(a,b
                        );
                elseif b == 4
                    shift(a-1,b-2) = shift(a-1,b-2) +
                        operator(a,b);
                end
            else
                if b == 1
                    shift(a+rootv,b+2) = shift(a+rootv,b
                        +2) + operator(a,b);
                elseif b == 2
                    shift(a+1,b+2) = shift(a+1,b+2) +
                        operator(a,b);
                elseif b == 3
                    shift(a+vertices-rootv,b-2) = shift(a+
                        vertices-rootv,b-2) + operator(a,b
                        );
                elseif b == 4
                    shift(a-1,b-2) = shift(a-1,b-2) +
                        operator(a,b);
                end
            end
        elseif a>vertices-rootv
            %top row
            if a==(vertices-rootv)+1
                %top left
                if b == 1
```

```
            shift(a-vertices+rootv,b+2) = shift(a-
                vertices+rootv,b+2) + operator(a,b
                );
        elseif b == 2
            shift(a+1,b+2) = shift(a+1,b+2) +
                operator(a,b);
        elseif b == 3
            shift(a-rootv,b-2) = shift(a-rootv,b
                -2) + operator(a,b);
        elseif b == 4
            shift(a+rootv-1,b-2) = shift(a+rootv
                -1,b-2) + operator(a,b);
        end
    elseif a==vertices
        %top right
        if b == 1
            shift(a-vertices+rootv,b+2) = shift(a-
                vertices+rootv,b+2) + operator(a,b
                );
        elseif b == 2
            shift(a-rootv+1,b+2) = shift(a-rootv
                +1,b+2) + operator(a,b);
        elseif b == 3
            shift(a-rootv,b-2) = shift(a-rootv,b
                -2) + operator(a,b);
        elseif b == 4
            shift(a-1,b-2) = shift(a-1,b-2) +
                operator(a,b);
        end
    else
        if b == 1
            shift(a-vertices+rootv,b+2) = shift(a-
                vertices+rootv,b+2) + operator(a,b
                );
        elseif b == 2
```

```
                shift(a+1,b+2) = shift(a+1,b+2) +
                    operator(a,b);
            elseif b == 3
                shift(a-rootv,b-2) = shift(a-rootv,b
                    -2) + operator(a,b);
            elseif b == 4
                shift(a-1,b-2) = shift(a-1,b-2) +
                    operator(a,b);
            end
        end
    elseif mod(a,rootv)==0 && a~=rootv && a~= vertices
        %rhs
        if b == 1
            shift(a+rootv,b+2) = shift(a+rootv,b+2) +
                operator(a,b);
        elseif b == 2
            shift(a-rootv+1,b+2) = shift(a-rootv+1,b
                +2) + operator(a,b);
        elseif b == 3
            shift(a-rootv,b-2) = shift(a-rootv,b-2) +
                operator(a,b);
        elseif b == 4
            shift(a-1,b-2) = shift(a-1,b-2) + operator
                (a,b);
        end
    elseif mod(a-1,rootv)==0 && a~=1 && a~=(vertices-
        rootv+1)
        %lhs
        if b == 1
            shift(a+rootv,b+2) = shift(a+rootv,b+2) +
                operator(a,b);
        elseif b == 2
            shift(a+1,b+2) = shift(a+1,b+2) + operator
                (a,b);
        elseif b == 3
```

```
                    shift(a-rootv,b-2) = shift(a-rootv,b-2) +
                        operator(a,b);
                elseif b == 4
                    shift(a+rootv-1,b-2) = shift(a+rootv-1,b
                        -2) + operator(a,b);
                end
            else
                if b == 1
                    shift(a+rootv,b+2) = shift(a+rootv,b+2) +
                        operator(a,b);
                elseif b == 2
                    shift(a+1,b+2) = shift(a+1,b+2) + operator
                        (a,b);
                elseif b == 3
                    shift(a-rootv,b-2) = shift(a-rootv,b-2) +
                        operator(a,b);
                elseif b == 4
                    shift(a-1,b-2) = shift(a-1,b-2) + operator
                        (a,b);
                end
            end
        end
    end
end
checka = 0;
for a = 1:1:vertices
    for b = 1:1:d
        checka = checka + abs(shift(a,b))^2;
    end
end
operator = zeros(vertices,d);
for a = 1:1:vertices
    for b = 1:1:d
        prob(a,t) = prob(a,t) + abs(shift(a,b))^2;
    end
end
```

```
            probmarked(:) = prob(markedstate,:);
            for ms = 1:1:length(probmarked)
                marked(ms,la) = probmarked(ms);
            end
            if prob(markedstate,t)>maximumprob(la,1)
                maximumprob(la,1) = prob(markedstate,t);
                maximumtime(la,1) = t;
            end
        end
        maxprob(la) = max(prob(markedstate,:));
        maxproba(la) = max(probmarked);
        pmax = max(marked(:,la));
        pa = find(marked(:,la)>=(pmax*0.5));
        centre = ceil((pa(1) + pa(length(pa)))/2);
        maxpr(la) = marked(pa(1),la);
        maxti(la) = pa(1);
        save(['data/mprob' int2str(l) '.txt'],'probmarked','-ascii','-
            append');
        [j,k] = max(prob(markedstate,:));
        maxtime(la) = k;
        [m,n] = max(probmarked);
        maxtimea(la) = n;
        probabove = find(probmarked>=(max(prob(markedstate,:))*0.75));
        for ab = 1:1:length(probabove)
           probabovearray(la,ab) = probabove(ab);
           probsabovearray(la,ab) = probmarked(probabove(ab));
        end
        for k = 1:1:length(probmarked)
           probm(k,l) = probmarked(k);
        end
        axes(la) = vertices;
        la = la+1;
    end
    for a = 1:1:length(5:5:latticewidth)
        maxt(a) = probabovearray(a,length(probabovearray(a)));
        mint(a) = probabovearray(a,1);
```

```
    w = maxt(a) - mint(a);
    if w==0
        widtht(a) = 1;
    else
        widtht(a) = w;
    end
    centret(a) = mint(a) + widtht(a)/2;
    minp(a) = probsabovearray(a,1);
    maxp(a) = max(probsabovearray(a,:));
    centrep(a) = probsabovearray(ceil(widtht(a)/2));
end
save(['/Users/neillovett/Desktop/QRW/twodmaxprob' int2str(latticewidth
    ) '.txt'],'maxprob','-ascii','-append')
save(['/Users/neillovett/Desktop/QRW/twodcentret' int2str(latticewidth
    ) '.txt'],'centret','-ascii','-append')
save(['/Users/neillovett/Desktop/QRW/twodaxes' int2str(latticewidth)
    '.txt'],'axes','-ascii','-append')
```

Bibliography

[1] N. B. Lovett, S. Cooper, M. Everitt, M. Trevers, and V. Kendon. Universal quantum computation using the discrete-time quantum walk. *Physical Review A*, 81:42330, 2010.

[2] N. B. Lovett, M. Everitt, M. Trevers, D. Mosby, D. Stockton, and V. Kendon. Spatial search using the discrete time quantum walk. *Accepted to Natural Computation, Proceedings of Physics and Computation 2009, Ponta Delgada, Azores, 2009. Arxiv preprint arXiv:1010.4705*, 2009.

[3] N. B. Lovett, M. Everitt, and V. Kendon. Effect of connectivity on the coined quantum walk search algorithm. *Submitted to Theory of Computing Systems as part of CiE post proceedings*, 2010.

[4] N. B. Lovett and B. T. H. Varcoe. Generation of topologically useful entangled states. *Accepted to International Journal of Unconventional Computation, special issue - New Worlds of Computation. Arxiv preprint arXiv:1101.1220*, 2011.

[5] P. A. M. Dirac. *The principles of quantum mechanics*. Oxford University Press, USA, 1981. ISBN 0198520115.

[6] R. Landauer. Information is Physical. *Physics Today*, page 23, 1991.

[7] Y. Manin. Computable and uncomputable. *Sovetskoye Radio (Moscow)*, 1980.

[8] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[9] R. P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16: 507–531, 1986.

[10] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 400:97–117, 1985.

[11] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th annual symposium on foundations of computer science (FOCS), 1994*, pages 124–134. IEEE, 1995.

[12] P. W. Shor. Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal of Computing*, 26:1484, 1997.

[13] L. K. Grover. A fast quantum mechanical algorithm for database search. *Proceedings of the 28th annual ACM symposium on theory of computing (STOC), 1996*, page 212, 1996.

[14] R. Raussendorf and H. J. Briegel. A One-Way Quantum Computer. *Physical Review Letters*, 86:5188–5191, 2001.

[15] M. A. Nielsen and I. L. Chuang. Quantum computation and information. *Cambridge University Press, Cambridge, UK*, 2000.

[16] J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Physical Review Letters*, 74:4091–4094, 1995.

[17] D. Kielpinski, C. Monroe, and D. J. Wineland. Architecture for a large-scale ion-trap quantum computer. *Nature*, 417:709, 2002.

[18] D. Loss and D. P. DiVincenzo. Quantum computation with quantum dots. *Physical Review A*, 57:120–126, 1998.

[19] J. M. Taylor, H. A. Engel, W. Dür, A. Yacoby, C. M Marcus, P. Zoller, and

M. D. Lukin. Fault-tolerant architecture for quantum computation using electrically controlled semiconductor spins. *Nature Physics*, 1:177–183, 2005.

[20] E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409:46, 201.

[21] N. A. Gershenfeld and I. L. Chuang. Bulk spin-resonance quantum computation. *Science*, 275:350, 1997.

[22] D. G. Cory, A. F. Fahmy, and T. F. Havel. Ensemble quantum computing by NMR spectroscopy. *Proceedings of the National Academy of Sciences of the United States of America*, 94:1634, 1997.

[23] J. E. Mooij, T. P. Orlando, L. Levitov, L. Tian, C. H. Van der Wal, and S. Lloyd. Josephson persistent-current qubit. *Science*, 285:1036, 1999.

[24] A. M. Steane. Error correcting codes in quantum theory. *Physical Review Letters*, 77:793–797, 1996.

[25] E. Knill and R. Laflamme. Theory of quantum error-correcting codes. *Physical Review A*, 55:900–911, 1997.

[26] P. W. Shor. Fault-tolerant quantum computation. In *Proceedings of the 37th annual symposium on the foundation of computer science (FOCS), 1996*, page 56. Published by the IEEE Computer Society, 1996.

[27] A. Y. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303:2–30, 2003.

[28] R. Raussendorf, J. Harrington, and K. Goyal. On measurement-based quantum computation with the toric code states. *Annals of Physics*, 321:2242, 2006.

[29] R. Raussendorf and J. Harrington. Fault-Tolerant Quantum Computation with High Threshold in Two Dimension. *Physical Review Letters*, 98:190504, 2007.

[30] R. Raussendorf, J. Harrington, and K. Goyal. Topological fault-tolerance in cluster state quantum computation. *New Journal of Physics*, 9:199, 2007.

[31] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proceedings: Mathematical and Physical Sciences*, 439:553–558, 1992.

[32] E. Bernstein and U. Vazirani. Quantum complexity theory. In *Proceedings of the 25th annual ACM symposium on theory of computing (TOCS), 1993*, page 20. ACM, 1993.

[33] D. R. Simon. On the power of quantum computation. In *Proceedings of the 35th annual symposium on the foundations of computer science (FOCS), 1994*, pages 116–123. IEEE Comput. Soc. Press, 1994.

[34] E. Bernstein, C. Bennett, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal of Computing*, 26:1510–1523, 1997.

[35] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46:493–505, 1998.

[36] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Information*, volume 305, pages 53–74. American Mathematical Society, Providence, RI., 2002.

[37] G. Brassard and P. Høyer. An exact quantum polynomial-time algorithm for Simon's problem. In *Proceedings of the 5th Israeli Symposium on theory of computing and systems, 1997*, pages 12–23. IEEE, 1997.

[38] S. Lloyd. Universal quantum simulators. *Science*, 273:1073–1073, 1996.

[39] C. Zalka. Simulating quantum systems on a quantum computer. *Proceedings: Mathematical, Physical and Engineering Sciences*, 454:313–322, 1998.

[40] K. L. Brown, W. J. Munro, and V. M. Kendon. Using Quantum Computers for Quantum Simulation. *Entropy*, 12:2268–2307, 2010.

[41] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Quantum computation by adiabatic evolution. *Arxiv preprint arXiv:000.1106*, 2000.

[42] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292:472, 2001.

[43] E. Farhi, J. Goldstone, and S. Gutmann. Quantum adiabatic evolution algorithms with different paths. *Arxiv preprint arXiv:020.8135*, 2002.

[44] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani. Quantum walks on graphs. In *Proceedings of the 33rd annual ACM symposium on theory of computing (STOC)*, pages 50–59. ACM, 2001.

[45] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous. One-dimensional quantum walks. In *Proceedings of the 33rd annual ACM symposium on theory of computing (STOC)*, pages 37–49. ACM, 2001.

[46] M. Mosca. Quantum algorithms. *Encyclopedia of Complexity and Systems Science. Arxiv preprint arXiv:0808.0369*, 2008.

[47] J. Smith and M. Mosca. Algorithms for Quantum Computers. *Handbook of Natural Computing, Arxiv preprint arXiv:1001.0767*, 2010.

[48] A. M. Childs and W. van Dam. Quantum algorithms for algebraic problems. *Reviews of Modern Physics*, 82:1–52, 2010.

[49] A. M. Childs. Universal computation by quantum walk. *Physical Review Letters*, 102:180501, 2009.

[50] N. Shenvi, J. Kempe, and K. B. Whaley. A quantum random-walk search algorithm. *Physical Review A*, 67:52307, 2003.

[51] T. Schoning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of 40th annual symposium on foundations of computer science (FOCS), 1999*, pages 410–414. IEEE, 2002.

[52] C. H. Papadimitriou. On selecting a satisfying truth assignment. In *Proceedings of 32nd annual symposium on foundations of computer science (FOCS), 1991*, pages 163–169. IEEE Comput. Soc. Press, 1991.

[53] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51:671–697, 2004.

[54] M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM (JACM)*, 38:1–17, 1991.

[55] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge Univ Press, 2005.

[56] A. Sinclair. *Algorithms for random generation and counting: a Markov chain approach*. Springer, 1993.

[57] R. Motwani and P. Raghavan. Randomized algorithms. *ACM Computing Surveys (CSUR)*, 28:33–37, 1996.

[58] C. H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

[59] Y. Aharonov, L. Davidovich, and N. Zagury. Quantum random walks. *Physical Review A*, 48:1687–1690, 1993.

[60] R. P. Feynman and A. R. Hibbs. *Quantum mechanics and path integrals*, volume 13. McGraw-Hill New York, 1965.

[61] D. A. Meyer. From quantum cellular automata to quantum lattice gases. *Journal of Statistical Physics*, 85:551–574, 1996.

[62] D. A. Meyer. On the absence of homogeneous scalar unitary cellular automata. *Physics Letters A*, 223:337–340, 1996.

[63] J. Watrous. Quantum simulations of classical random walks and undirected graph connectivity. In *Proceedings of 14th annual IEEE conference on computational complexity, 1999*, pages 180–187. IEEE, 2002.

[64] E. Farhi and S. Gutmann. Quantum computation and decision trees. *Physical Review A*, 58:915–928, 1998.

[65] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the 35th annual ACM symposium on theory of computing (STOC), 2003*, pages 59–68. ACM, 2003.

[66] A. M. Childs, E. Farhi, and S. Gutmann. An example of the difference between quantum and classical random walks. *Quantum Information Processing*, 1:35–43, 2002.

[67] A. Ambainis. Quantum walk algorithm for element distinctness. In *Proceedings of the 45th annual IEEE symposium on foundations of computer science (FOCS), 2004*, pages 22–31. IEEE, 2004.

[68] F. Magniez, M. Santha, and M. Szegedy. Quantum algorithms for the triangle problem. In *Proceedings of the 16th annual ACM-SIAM symposium on discrete algorithms (SODA)*, pages 1109–1117. Society for Industrial and Applied Mathematics, 2005.

[69] A. Ambainis. Quantum walks and their algorithmic applications. *International Journal of Quantum Information*, 1:507–518, 2003.

[70] D. Aldous and J. Fill. *Reversible Markov chains and random walks on graphs.* University of California Press, Berkeley, 2002.

[71] A. Nayak and A. Vishwanath. Quantum walk on the line. *DIMACS Technical Report 2000-43, Arxiv preprint arXiv:0010117*, 2000.

[72] V. Kendon and B. Tregenna. Decoherence can be useful in quantum walks. *Physical Review A*, 67:42315, 2003.

[73] V. Kendon. Decoherence in quantum walks - a review. *Mathematical Structures in Computer Science*, 17:1169–1220, 2007.

[74] N. Konno. Quantum random walks in one dimension. *Quantum Information Processing*, 1:345–354, 2002. ISSN 1570-0755.

[75] N. Konno, T. Namiki, and T. Soshi. Symmetry of distribution for the one-dimensional Hadamard walk. *Interdisciplinary Information Sciences*, 10:11–22, 2004. ISSN 1340-9050.

[76] E. Bach, S. Coppersmith, M. P. Goldschen, R. Joynt, and J. Watrous. One-dimensional quantum walks with absorbing boundaries. *Journal of Computer and System Sciences*, 69:562–592, 2004.

[77] J. Kempe. Quantum random walks: an introductory overview. *Contemporary Physics*, 44:307–327, 2003.

[78] T. Yamasaki, H. Kobayashi, and H. Imai. Analysis of absorbing times of quantum walks. *Physical Review A*, 68:12302, 2003.

[79] C. Moore and A. Russell. Quantum walks on the hypercube. *Proceedings of 6th international workshop on randomization and approximation techniques in computer science (RANDOM)*, pages 164–178, 2002.

[80] T. D. Mackay, S. D. Bartlett, L. T. Stephenson, and B. C. Sanders. Quantum walks in higher dimensions. *Journal of Physics A: Mathematical and General*, 35:2745, 2002.

[81] V. Kendon. Quantum walks on general graphs. *International Journal of Quantum Information*, 4:791, 2003.

[82] J. Kempe. Quantum random walks hit exponentially faster. In *Lecture Notes*

in Computer Science - Proceedings of RANDOM'03, volume 2764, pages 354–369. Springer, 2003.

[83] B. Tregenna, W. Flanagan, R. Maile, and V. Kendon. Controlling discrete quantum walks: coins and initial states. *New Journal of Physics*, 5:83, 2003.

[84] R. A. M. Santos and R. Portugal. Quantum Hitting Time on the Complete Graph. *To appear in International Journal of Quantum Information. Arxiv preprint arXiv:0912.1217*, 2009.

[85] F. Magniez, A. Nayak, P. C. Richter, and M. Santha. On the hitting times of quantum versus random walks. In *Proceedings of the 20th annual ACM-SIAM symposium on discrete algorithms (SODA)*, pages 86–95. Society for Industrial and Applied Mathematics, 2009.

[86] A. M. Childs. On the relationship between continuous-and discrete-time quantum walk. *Communications in Mathematical Physics*, 294:581–603, 2010.

[87] F. W. Strauch. Connecting the discrete and continuous-time quantum walks. *Physical Review A*, 74:30301, 2006.

[88] M. Szegedy. Quantum Speed-Up of Markov Chain Based Algorithms. In *Proceedings of 45th annual IEEE symposium on foundations of computer science (FOCS)*, pages 32–41. IEEE, 2004.

[89] A. Ambainis, A. M. Childs, B. W. Reichardt, R. Spalek, and S. Zhang. Any AND-OR Formula of Size N can be Evaluated in time $N^{1/2+o(1)}$ on a Quantum Computer. In *Proceedings of the 48th annual IEEE symposium on foundations of computer science, (FOCS), 2007*. IEEE Computer Society, 2007.

[90] B. W. Reichardt and R. Spalek. Span-program-based quantum algorithm for evaluating formulas. In *Proceedings of the 40th annual ACM symposium on theory of computing (STOC)*, pages 103–112. ACM, 2008.

[91] H. Buhrman, C. Durr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. De Wolf. Quantum algorithms for element distinctness. In *16th annual IEEE conference on computational complexity*, pages 131–137. IEEE, 2002.

[92] Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *Proceedings of 43rd annual conference on the foundations of computer science (FOCS)*, pages 513–519, 2002.

[93] H. Buhrman and R. Spalek. Quantum verification of matrix products. *Proceedings of the 17th annual ACM-SIAM symposium on discrete algorithms (SODA), 2004*, 2004.

[94] M. Hillery, D. Reitzner, and V. Buzek. Searching via walking: How to find a marked subgraph of a graph using quantum walks. *Physical Review A*, 81: 062324, 2009.

[95] P. C. Richter. Almost uniform sampling via quantum walks. *New Journal of Physics*, 9:72, 2007.

[96] A. M. Childs and R. Kothari. Quantum query complexity of minor-closed graph properties. *To appear as part of proceedings of the 28th international symposium on theoretical aspects of computer science (STACS), 2011*, 2010.

[97] E. Feldman, M. Hillery, H. W. Lee, D. Reitzner, H. Zheng, and V. Buzek. Finding structural anomalies in graphs by means of quantum walks. *Physical Review A*, 82:040301, 2010.

[98] A. Ambainis. New Developments in Quantum Algorithms. *Mathematical Foundations of Computer Science*, pages 1–11, 2010.

[99] M. Santha. Quantum walk based search algorithms. *Theory and Applications of Models of Computation*, pages 31–46, 2008.

[100] E. Farhi, J. Goldstone, and S. Gutmann. A quantum algorithm for the Hamiltonian NAND tree. *Theory of Computing*, 4:169–190, 2008.

[101] A. Ambainis. A nearly optimal discrete query quantum algorithm for evaluating NAND formulas. *Arxiv preprint arXiv:0704.3628*, 2007.

[102] A. M. Childs, B. W. Reichardt, R. Spalek, and S. Zhang. Every NAND formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. *Proceedings of the 48th IEEE symposium on foundations of computer science (FOCS)*, 2007.

[103] M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the 8th annual conference on the structure in complexity theory*, pages 102–111. IEEE, 2002.

[104] A. Ambainis. Quantum algorithms for formula evaluation. *To appear in Proceedings of the NATO advanced research Workshop 'Quantum Cryptography and Computing: Theory and Implementations', Arxiv preprint arXiv:1006.3651*, 2010.

[105] D. Aharonov and A. Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the 35th annual ACM symposium on theory of computing (STOC)*, pages 20–29. ACM, 2003.

[106] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270:359–371, 2007.

[107] A. M. Childs and R. Kothari. Simulating sparse Hamiltonians with star decompositions. *Lecture Notes in Computer Science - Proceedings of theory of quantum computation, communication and cryptography (TQC), 2010*, 6519: 94–103, 2010.

[108] D. W. Berry and A. M. Childs. Black-box Hamiltonian simulation and unitary implementation. *arXiv:0910.4157*, 2009.

[109] A. M. Childs and R. Kothari. Limitations on the simulation of non-sparse Hamiltonians. *Quantum Information and Computation*, 10:669–684, 2010.

[110] C. M. Chandrashekar and R. Laflamme. Quantum phase transition using quantum walks in an optical lattice. *Physical Review A*, 78:22314, 2008.

[111] R. D. Somma, S. Boixo, H. Barnum, and E. Knill. Quantum simulations of classical annealing processes. *Physical Review Letters*, 101:130504, 2008.

[112] O. Mülken, V. Bierbaum, and A. Blumen. Coherent exciton transport in dendrimers and continuous-time quantum walks. *Journal of Chemical Physics*, 124:124905, 2006.

[113] S. E. Gregory, R. C. Tessa, L. R. Elizabeth, T. M. Tae-Kyu Ahn, et al. Evidence for wavelike energy transfer through quantum coherence in photosynthetic systems. *Nature*, 446:782–786, 2007.

[114] M. Mohseni, P. Rebentrost, S. Lloyd, and A. Aspuru-Guzik. Environment-assisted quantum walks in photosynthetic energy transfer. *Journal of Chemical Physics*, 129:174106, 2008.

[115] M. B. Plenio and S. F. Huelga. Dephasing-assisted transport: quantum networks and biomolecules. *New Journal of Physics*, 10:113019, 2008.

[116] P. Rebentrost, M. Mohseni, I. Kassal, S. Lloyd, and A. Aspuru-Guzik. Environment-assisted quantum transport. *New Journal of Physics*, 11:033003, 2009.

[117] O. Mülken and A. Blumen. Continuous-time quantum walks: Models for coherent transport on complex networks. *Physics Reports*, 2011.

[118] S. E. Venegas-Andraca and S. Bose. Quantum Walk-based Generation of Entanglement Between Two Walkers. *Arxiv preprint arXiv:0901.3946*, 2009.

[119] G. K. Brennen, D. Ellinas, V. Kendon, J. K. Pachos, I. Tsohantjis, and Z. Wang. Anyonic quantum walks. *Annals of Physics*, 325:664–681, 2010.

[120] D. I. Tsomokos. Community Detection in Complex Networks with Quantum Random Walks. *Arxiv preprint arXiv:1012.2405*, 2010.

[121] J. K. Gamble, M. Friesen, D. Zhou, R. Joynt, and S. N. Coppersmith. Two-particle quantum walks applied to the graph isomorphism problem. *Physical Review A*, 81:52313, 2010.

[122] N. Linden and J. Sharam. Inhomogeneous quantum walks. *Physical Review A*, 80:52327, 2009.

[123] N. Konno. Localization of an inhomogeneous discrete-time quantum walk on the line. *Quantum Information Processing*, 9(3):405–418, 2010. ISSN 1570-0755.

[124] Y. Shikano and H. Katsura. Localization and fractality in inhomogeneous quantum walks with self-duality. *Physical Review E*, 82(3):031122, 2010.

[125] M. Stefanak, S. M. Barnett, B. Kollar, T. Kiss, and I. Jex. Directional correlations in quantum walks with two particles. *To appear in New Journal of Physics. ArXiV preprint arXiv:1102.4445*, 2011.

[126] Y. Omar, N. Paunković, L. Sheridan, and S. Bose. Quantum walk on a line with two entangled particles. *Physical Review A*, 74(4):042304, 2006.

[127] PK Pathak and GS Agarwal. Quantum random walk of two photons in separable and entangled states. *Physical Review A*, 75(3):032351, 2007.

[128] D. Bouwmeester, I. Marzoli, G. P. Karman, W. Schleich, and J. P. Woerdman. Optical galton board. *Physical Review A*, 61:13410, 1999.

[129] R. J. C. Spreeuw. Classical wave-optics analogy of quantum-information processing. *Physical Review A*, 63:62302, 2001.

[130] B. C. Travaglione and G. J. Milburn. Implementing the quantum random walk. *Physical Review A*, 65:32310, 2002.

[131] B. C. Sanders, S. D. Bartlett, B. Tregenna, and P. L. Knight. Quantum quincunx in cavity quantum electrodynamics. *Physical Review A*, 67:42305, 2003.

[132] W. Dür, R. Raussendorf, V. M. Kendon, and H. J. Briegel. Quantum walks in optical lattices. *Physical Review A*, 66:52319, 2002.

[133] Z. Zhao, J. Du, H. Li, T. Yang, Z. B. Chen, and J. W. Pan. Implement quantum random walks with linear optics elements. *Arxiv preprint arXiv:0212149*, 2002.

[134] P. L. Knight, E. Roldán, and J. E. Sipe. Quantum walk on the line as an interference phenomenon. *Physical Review A*, 68:20301, 2003.

[135] P. L. Knight, E. Roldán, and J. E. Sipe. Optical cavity implementations of the quantum walk. *Optics Communications*, 227:147–157, 2003.

[136] M. Hillery, J. Bergou, and E. Feldman. Quantum walks based on an interferometric analogy. *Physical Review A*, 68:32314, 2003.

[137] H. Jeong, M. Paternostro, and M. S. Kim. Simulation of quantum random walks using the interference of a classical field. *Physical Review A*, 69:12310, 2004.

[138] T. Di, M. Hillery, and M. S. Zubairy. Cavity QED-based quantum walk. *Physical Review A*, 70:32304, 2004.

[139] C. M. Chandrashekar. Implementing the one-dimensional quantum (Hadamard) walk using a Bose-Einstein condensate. *Physical Review A*, 74: 32307, 2006.

[140] P. Xue, B. C. Sanders, and D. Leibfried. Quantum walk on a line for a trapped ion. *Physical Review Letters*, 103:183602, 2009.

[141] Z. Y. Ma, K. Burnett, M. B. d'Arcy, and S. A. Gardiner. Quantum random walks using quantum accelerator modes. *Physical Review A*, 73:13401, 2006.

[142] K. Eckert, J. Mompart, G. Birkl, and M. Lewenstein. One and two-dimensional quantum walks in arrays of optical traps. *Physical Review A*, 72:12327, 2005.

[143] E. Roldán and J. C. Soriano. Optical implementability of the two-dimensional quantum walk. *Journal of Modern Optics*, 52:2649–2657, 2005.

[144] X. Zou, Y. Dong, and G. Guo. Optical implementation of one-dimensional quantum random walks using orbital angular momentum of a single photon. *New Journal of Physics*, 8:81, 2006.

[145] J. Du, H. Li, X. Xu, M. Shi, J. Wu, X. Zhou, and R. Han. Experimental implementation of the quantum random-walk algorithm. *Physical Review A*, 67:42316, 2003.

[146] O. Mandel, M. Greiner, A. Widera, T. Rom, T. W. Hänsch, and I. Bloch. Coherent transport of neutral atoms in spin-dependent optical lattice potentials. *Physical Review Letters*, 91:10407, 2003.

[147] C. A. Ryan, M. Laforest, J. C. Boileau, and R. Laflamme. Experimental implementation of a discrete-time quantum random walk on an NMR quantum-information processor. *Physical Review A*, 72:62317, 2005.

[148] B. Do, M. L. Stohler, S. Balasubramanian, D. S. Elliott, C. Eash, E. Fischbach, M. A. Fischbach, A. Mills, and B. Zwickl. Experimental realization of a quantum quincunx by use of linear optical elements. *Journal of the Optical Society of America B*, 22:499–504, 2005.

[149] P. Zhang, X. F. Ren, X. B. Zou, B. H. Liu, Y. F. Huang, and G. C. Guo. Demonstration of one-dimensional quantum random walks using orbital angular momentum of photons. *Physical Review A*, 75:52310, 2007.

[150] H. Schmitz, R. Matjeschk, C. Schneider, J. Glueckert, M. Enderlein, T. Huber, and T. Schätz. Quantum walk of a trapped ion in phase space. *Physical Review Letters*, 103:90504, 2009.

[151] F. Zähringer, G. Kirchmair, R. Gerritsma, E. Solano, R. Blatt, and C. F. Roos. Realization of a quantum walk with one and two trapped ions. *Physical Review Letters*, 104:100503, 2010.

[152] H. B. Perets, Y. Lahini, F. Pozzi, M. Sorel, R. Morandotti, and Y. Silber-

berg. Realization of quantum walks with negligible decoherence in waveguide lattices. *Physical Review Letters*, 100:170506, 2008.

[153] A. Peruzzo, M. Lobino, J. C. F. Matthews, N. Matsuda, A. Politi, K. Poulios, X. Q. Zhou, Y. Lahini, N. Ismail, et al. Quantum Walks of Correlated Photons. *Science*, 329:1500, 2010.

[154] M. Karski, L. Forster, J. M. Choi, A. Steffen, W. Alt, D. Meschede, and A. Widera. Quantum walk in position space with single optically trapped atoms. *Science*, 325:174, 2009.

[155] M. A. Broome, A. Fedrizzi, B. P. Lanyon, I. Kassal, A. Aspuru-Guzik, and A. G. White. Discrete single-photon quantum walks with tunable decoherence. *Physical Review Letters*, 104:153602, 2010.

[156] A. Schreiber, K. N. Cassemiro, V. Potoček, A. Gábris, P. J. Mosley, E. Andersson, I. Jex, and C. Silberhorn. Photons walking the line: A quantum walk with adjustable coin operations. *Physical Review Letters*, 104:50502, 2010.

[157] M. S. Underwood and D. L. Feder. Universal quantum computation by discontinuous quantum walk. *Physical Review A*, 82:042304, 2010.

[158] A. Y. Kitaev, A. H. Shen, A. Shen, and M. N. Vyalyi. *Classical and quantum computation*. American Mathematical Society, 2002.

[159] P. Wocjan and S. Zhang. Several natural BQP-complete problems. *Arxiv preprint arXiv:0606179*, 2006.

[160] D. Janzing and P. Wocjan. A simple Promise BQP-complete matrix problem. *Theory of Computing*, 3:61–79, 2007.

[161] S. Bose. Quantum communication through an unmodulated spin chain. *Physical Review Letters*, 91:207901, 2003.

[162] M. Christandl, N. Datta, A. Ekert, and A. J. Landahl. Perfect state transfer in quantum spin networks. *Physical Review Letters*, 92:187902, 2004.

[163] M. Christandl, N. Datta, T. C. Dorlas, A. Ekert, A. Kay, and A. J. Landahl. Perfect transfer of arbitrary states in quantum spin networks. *Physical Review A*, 71:32312, 2005.

[164] S. Bose. Quantum communication through spin chain dynamics: an introductory overview. *Contemporary Physics*, 48:13–30, 2007.

[165] A. Kay. A Review of Perfect State Transfer and its Application as a Constructive Tool. *International Journal of Quantum Information*, 8:641, 2010.

[166] A. Ahmadi, R. Belk, C. Tamon, and C. Wendler. On mixing in continuous-time quantum walks on some circulant graphs. *Quantum Information and Computation*, 3:611–618, 2003.

[167] L. Fedichkin, D. Solenov, and C. Tamon. Mixing and decoherence in continuous-time quantum walks on cycles. *Quantum Information and Computation*, 6:263, 2005.

[168] W. Carlson, A. Ford, E. Harris, J. Rosen, C. Tamon, and K. Wrobel. Universal mixing of quantum walk on graphs. *Quantum Information and Computation*, 7:738, 2006.

[169] A. Best, M. Kliegl, S. Mead-Gluchacki, and C. Tamon. Mixing of Quantum Walks on Generalized Hypercubes. *International Journal of Quantum Information*, 6:1135–1148, 2008.

[170] V. Kendon and C. Tamon. Perfect state transfer in quantum walks on graphs. *Journal of Computational and Theoretical Nanoscience*, 8:422–433, 2010.

[171] E. Feldman and M. Hillery. Quantum walks on graphs and quantum scattering theory. In *Proceedings of the international conference on coding theory and quantum computing*, volume 381, page 71. American Mathematical Society, 2005.

[172] E. Feldman and M. Hillery. Scattering theory and discrete-time quantum walks. *Physics Letters A*, 324:277–281, 2004.

[173] E. Feldman and M. Hillery. Modifying quantum walks: a scattering theory approach. *Journal of Physics A: Mathematical and Theoretical*, 40:11343, 2007.

[174] H. Krovi and T. A. Brun. Quantum walks on quotient graphs. *Physical Review A*, 75:62332, 2007.

[175] H. Krovi and T. A. Brun. Hitting time for quantum walks on the hypercube. *Physical Review A*, 73:32341, 2006.

[176] H. Krovi and T. A. Brun. Quantum walks with infinite hitting times. *Physical Review A*, 74:42334, 2006.

[177] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52:3457–3467, 1995.

[178] G. Brassard. Searching a quantum phone book. *Science*, 275:627–628, 1997. ISSN 0036-8075.

[179] A. Ambainis, J. Kempe, and A. Rivosh. Coins make quantum walks faster. In *Proceedings of the 16th annual ACM-SIAM symposium on discrete algorithms (SODA)*, pages 1099–1108. Society for Industrial and Applied Mathematics, 2005.

[180] D. E. Knuth. *The Art of Computer Programming Volumes 1-3 Boxed Set*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1998.

[181] P. Høyer. Arbitrary phases in quantum amplitude amplification. *Physical Review A*, 62:52304, 2000.

[182] G. L. Long. Grover algorithm with zero theoretical failure rate. *Physical Review A*, 64:22307, 2001.

[183] A. Tulsi. Faster quantum-walk algorithm for the two-dimensional spatial search. *Physical Review A*, 78:12310, 2008.

[184] B. Hein and G. Tanner. Quantum search algorithms on the hypercube. *Journal of Physics A: Mathematical and Theoretical*, 42:085303, 2009.

[185] V. Potoček, A. Gábris, T. Kiss, and I. Jex. Optimized quantum random-walk search algorithms on the hypercube. *Physical Review A*, 79:12325, 2009.

[186] C. Zalka. Grover′s quantum searching algorithm is optimal. *Physical Review A*, 60:2746–2751, 1999.

[187] P. Benioff. Space searches with a quantum robot. *Quantum Computation and Information*, 305:1, 2002.

[188] S. Aaronson and A. Ambainis. Quantum search of spatial regions. In *Proceedings of the 44th annual IEEE symposium on foundations of computer science (FOCS), 2003*, pages 200–209. IEEE, 2003.

[189] A. M. Childs and J. Goldstone. Spatial search by quantum walk. *Physical Review A*, 70:22314, 2004.

[190] A. M. Childs and J. Goldstone. Spatial search and the Dirac equation. *Physical Review A*, 70:42312, 2004.

[191] A. Patel and Md. A. Rahaman. Search on a Hypercubic Lattice through a Quantum Random Walk: I. d>2. *Physical Review A*, 82:032330, 2010.

[192] A. Patel, K. S. Raghunathan, and Md. A. Rahaman. Search on a Hypercubic Lattice through a Quantum Random Walk: II. d=2. *Physical Review A*, 82: 032331, 2010.

[193] F. Magniez and A. Nayak. Quantum complexity of testing group commutativity. *Automata, Languages and Programming*, pages 1312–1324, 2005.

[194] F. Magniez, A. Nayak, J. Roland, and M. Santha. Search via quantum walk. In *Proceedings of the 39th annual ACM symposium on theory of computing (STOC)*, pages 575–584. ACM, 2007.

[195] H. Krovi, F. Magniez, M. Ozols, and J. Roland. Finding is as easy as detecting for quantum walks. *Automata, Languages and Programming*, pages 540–551, 2010.

[196] F. L. Marquezino, R. Portugal, and G. Abal. Mixing Times in Quantum Walks on Two-Dimensional Grids. *Physical Review A*, 82:042341, 2010.

[197] B. Hein and G. Tanner. Quantum search algorithms on a regular lattice. *Physical Review A*, 82:12326, 2010.

[198] B. Hein and G. Tanner. Wave communication across regular lattices. *Physical Review Letters*, 103:260501, 2009.

[199] E. Agliari, A. Blumen, and O. Mülken. Quantum walk approach to search on fractal structures. *Physical Review A*, 82:012305, 2010.

[200] E. Agliari, A. Blumen, and O. Mülken. Dynamics of continuous-time quantum walks in restricted geometries. *Journal of Physics A: Mathematical and Theoretical*, 41:445301, 2008.

[201] S. D. Berry and J. B. Wang. Quantum walk-based search and centrality. *Physical Review A*, 82:042333, 2010.

[202] G. Abal, R. Donangelo, F. L. Marquezino, and R. Portugal. Spatial search in a honeycomb network. *To appear in Mathematical Structures in Computer Science. Arxiv preprint arXiv:1001.1139*, 2010.

[203] D. Reitzner, M. Hillery, E. Feldman, and V. Bužek. Quantum searches on highly symmetric graphs. *Physical Review A*, 79:12323, 2009.

[204] J. P. Keating, N. Linden, J. C. F Matthews, and A. Winter. Localization and

its consequences for quantum walk algorithms and quantum communication. *Physical Review A*, 76:12315, 2007.

[205] G. Leung, P. Knott, J. Bailey, and V. Kendon. Coined quantum walks on percolation graphs. *New Journal of Physics*, 12:123018, 2010.

[206] G. Abal, R. Donangelo, F. L. Marquezino, A. C. Oliveira, and R. Portugal. Decoherence in Search Algorithms. *Proceedings of the 29th Brazilian computer society congress (SEMISH)*, pages 293–306, 2009.

[207] Z. V. Djordjevic, H. E. Stanley, and A. Margolina. Site percolation threshold for honeycomb and square lattices. *Journal of Physics A: Mathematical and General*, 15:L405, 1982.

[208] T. Gebele. Site percolation threshold for square lattice. *Journal of Physics A: Mathematical and General*, 17:L51, 1984.

[209] D. Stauffer and A. Aharony. *Introduction to percolation theory.* Taylor & Francis, 1992. ISBN 0748402535.

[210] P. Walther, K. J. Resch, T. Rudolph, E. Schenck, H. Weinfurter, V. Vedral, M. Aspelmeyer, and A. Zeilinger. Experimental one-way quantum computing. *Nature*, 434:169–176, 2005.

[211] J. Cho and H. W. Lee. Generation of Atomic Cluster States through the Cavity Input-Output Process. *Physical Review Letters*, 95:160501, 2005.

[212] P. Dong, Z. Y. Xue, M. Yang, and Z. L. Cao. Generation of cluster states. *Physical Review A*, 73:033818, 2006.

[213] D. Gonta, S. Fritzsche, and T. Radtke. Generation of two-dimensional cluster states by using high-finesse bimodal cavities. *Physical Review A*, 79:062319, 2009.

[214] H. Wunderlich, C. Wunderlich, K. Singer, and F. Schmidt-Kaler. Two-dimensional cluster-state preparation with linear ion traps. *Physical Review A*, 79:052314, 2009.

[215] S. J. Devitt, A. D. Greentree, R. Ionicioiu, J. L. O'Brien, W. J. Munro, and L. C. L. Hollenberg. Photonic module: An on-demand resource for photonic entanglement. *Physical Review A*, 76:52312, 2007.

[216] A. M. Stephens, Z. W. E. Evans, S. J. Devitt, A. D. Greentree, A. G. Fowler, W. J. Munro, J. L. OBrien, K. Nemoto, and L. C. L. Hollenberg. Deterministic optical quantum computer using photonic modules. *Physical Review A*, 78: 32318, 2008.

[217] S. J. Devitt, A. G. Fowler, A. M. Stephens, A. D. Greentree, L. C. L. Hollenberg, W. J. Munro, and K. Nemoto. Architectural design for a topological cluster state quantum computer. *New Journal of Physics*, 11:083032, 2009.

[218] R. Ionicioiu and W. J. Munro. Constructing 2D and 3D cluster states with photonic modules. *International Journal of Quantum Information*, 8:149, 20010.

[219] S. Bravyi and R. Raussendorf. A fault-tolerant one-way quantum computer. *Physical Review Letters*, 76:022304, 2007.

[220] A. G. Fowler and K. Goyal. Topological cluster state quantum computing. *Quantum Information and Computation*, 9:721–738, 2009.

[221] E. Kashefi, D. K. L. Oi, D. Browne, J. Anders, and E. Andersson. Twisted Graph States for Ancilla-driven Universal Quantum Computation. *Electronic Notes in Theoretical Computer Science*, 249:307–331, 2009.

[222] B. T. H. Varcoe and P. J. Blythe. A cavity-QED scheme for cluster-state quantum computing using crossed atomic beams. *New Journal of Physics*, 8: 231, 2006.

[223] A. Acin, J. I. Cirac, and M. Lewenstein. Entanglement percolation in quantum networks. *Nature Physics*, 3:256–259, 2007.

[224] J. R. Wootton and J. K. Pachos. Bringing order through disorder: Localisation of errors in topological quantum memories. *Arxiv preprint arXiv:1101.5900*, 2011.