Math. Struct. in Comp. Science (2019), *vol.* 29, *pp.* 389–429. © Cambridge University Press 2018
doi:10.1017/S0960129518000051 First published online 16 May 2018

The quantum walk search algorithm: factors affecting efficiency[†]

NEIL B. LOVETT[*], MATTHEW EVERITT[‡],
ROBERT M. HEATH[§] and VIV KENDON[¶]

[*]*School of Physics and Astronomy, University of Leeds, Woodhouse Lane,Leeds, LS2 9JT, United Kingdom and Institute for Quantum Information Science, University of Calgary, 2500 University Drive NW, Calgary, Alberta, T2N 1N4, Canada*
[‡]*School of Physics and Astronomy, University of Leeds, Woodhouse Lane, Leeds, LS2 9JT, United Kingdom*
[§]*School of Physics and Astronomy, University of Leeds, Woodhouse Lane, Leeds, LS2 9JT, United Kingdom*
Email: Robert.Heath@glasgow.ac.uk
[¶]*Department of Physics, Durham University, South Road, Durham, DH1 3LE, United Kingdom*
Email: viv.kendon@durham.ac.uk

Received 19 October 2011; revised 1 June 2012

We carry out a numerical study of the quantum walk search algorithm of Shenvi, Kempe and Whaley Shenvi *et al.* (2003) and the factors that affect its efficiency in finding an individual state from an unsorted set. Previous work has focused purely on the effects of the dimensionality of the dataset to be searched. In the current paper we consider the effects of interpolating between dimensions, the connectivity of the dataset and the possibility of disorder in the underlying substrate: all these factors affect the efficiency of the search algorithm. We show that in addition to the strong dependence on the spatial dimension of the structure to be searched, there are also secondary dependencies on the connectivity and symmetry of the lattice, with greater connectivity providing a more efficient algorithm. We also show that the algorithm can tolerate a non-trivial level of disorder in the underlying substrate.

1. Introduction

Searching is undoubtedly one of the most basic problems in computer science and computational physics. In this context, searching is not just restricted to a physical

[†] Unfortunately there has been an unintended delay in the publication of this article. Cambridge University Press apologises for the delay.
[*] Neil Lovett was funded by the UK Engineering and Physical Sciences Research Council.
[¶] Viv Kendon was funded by a UK Royal Society University Research Fellowship.
[§] Robert Heath's current address is School of Engineering, University of Glasgow, Glasgow, G12 8QQ, United Kingdom.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

database but could also be searching through a state space for an entry that fulfills a specific clause such as the constraint satisfiability problem (k-SAT). The classical complexity of such a task scales linearly with the size N of the dataset 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 that can be achieved, *viz.* $O(N)$.

One of the most important quantum algorithms discovered to date is Grover's searching algorithm (Grover 1996). Grover showed that an item could be found from a set of N in a time quadratically faster than the classical case, *viz.* $O(\sqrt{N})$. Grover's algorithm has been shown to be both optimal and also one of the few quantum algorithms that is provably faster than any possible classical algorithm (Bernstein *et al.* 1997).

Several years after the introduction of this algorithm, Shenvi *et al.* (2003) gave a quantum search algorithm based instead on the discrete-time quantum walk, which was first introduced with algorithmic applications in mind in Aharonov *et al.* (2001) and Ambainis *et al.* (2001). This quantum walk approach to the search problem is able to match the quadratic speed up of Grover's algorithm. The 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 (Santha 2008). The quantum walk has also recently been shown to be universal for quantum computation, and hence a computational primitive (Childs 2009; Lovett *et al.* 2010; Underwood and Feder 2010), again showing it to be a powerful tool.

Shenvi *et al.* (2003) lays out the items of the dataset 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 (Moore and Russell 2002). Other recent work by Potoček *et al.* (2009) has improved the original algorithm by adding an additional coin dimension, which allows 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, *viz.* $\pi/4\sqrt{N}$. Zalka (1999) had 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 Shenvi *et al.* (2003) 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 begun in Benioff (2002), which 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, $D = 2$, no speedup was apparent. However, Aaronson and Ambainis (2003) subsequently introduced an algorithm based on a divide and conquer approach that contradicted this claim with a run time of $O(\sqrt{N})$ in dimensions $D \geqslant 3$ and $O(\sqrt{N}\log^{3/2} N)$ when $D = 2$.

Around the same time, Childs and Goldstone (2004a) gave another algorithm, this time based on the continuous-time quantum walk, which was first introduced in Farhi and Gutmann (1998). Childs and Goldstone showed a run time of: $O(N)$ for $D = 2$; $O(N^{5/6})$

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

Table 1. *Summary of run times of quantum search algorithms in various dimensions.*

Work	$D = 2$	$D = 3$	$D = 4$	$D \geqslant 5$
AA03 (2003)	$O(\sqrt{N}\log^{3/2} N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$
CG03 (2003)	$O(N)$	$O(N^{5/6})$	$O(\sqrt{N}\log N)$	$O(\sqrt{N})$
CG04 (2004)	$O(\sqrt{N}\log N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$
AKR04 (2004)	$O(\sqrt{N}\log N)$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$
Tulsi (2008)	$O(\sqrt{N\log N})$	-	-	-
Magniez et al. (2008)	$O(\sqrt{N\log N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$
Patel et al. (2010)	$O(\sqrt{N\log N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$

for $D = 3$; $O(\sqrt{N}\log N)$ for $D = 4$; and $O(\sqrt{N})$ for $D \geqslant 5$. This algorithm is not as efficient as the one introduced in Aaronson and Ambainis (2003), but does represent the first quantum walk search algorithm defined in continuous time. Shortly after, Ambainis *et al.* (2005) gave a discrete-time quantum walk search algorithm, improving on the original work in Shenvi *et al.* (2003) by using an additional $\log 2d$ qubits of extra memory. Childs and Goldstone (2004b) 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 algorithm of Ambainis *et al.* (2005). These results are summarised in Table 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 (2008) 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 to give $O(1/\log_2 N)$. In his work, Tulsi was able to control this probability using the ancilla qubits to give a constant scaling of the probability at the marked state, *viz.* $O(1)$, thus removing the need for the $\sqrt{\log N}$ amplitude amplification steps. During the years prior to the work by Tulsi, several advances had been made in establishing a theory of quantum walk search algorithms. This was pioneered by Szegedy (2004), who was able to introduce a method for quantising classical Markov chains (classical random walks on graphs) based on the previous work of Ambainis (2004). This framework is similar to other work by Ambainis *et al.* (2005), and both have been used to develop algorithms that give complexity gains compared with the basic Grover search (Magniez *et al.* 2005; Magniez and Nayak 2005; Buhrman and Spalek 2004). Building on this, Magniez *et al.* (2007) 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.* (2009) gave a similar theory for the hitting times of quantum walks. They proved that, given a reversible, ergodic classical random walk, the hitting time of the equivalent quantum walk is quadratically faster than the

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

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 two-dimensional 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 (2008). Magniez *et al.* (2009) also showed that they could find the probability of the marked state in a constant fashion, thus extending the result of Tulsi (2008) to the larger class of quantum walks based on reversible, ergodic Markov chains. In fact, this result has recently been tightened further by Krovi *et al.* (2010), which showed that the classical Markov chain, which forms the basis of the quantum walk, does not need to be state transitive. In much of the previous work, this state transitivity was assumed. These results tend to indicate that it is unlikely that 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: for example Marquezino *et al.* (2010) showed that 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, which uses the staggered lattice fermion formalism, was put forward in Patel and Rahaman (2010) and Patel *et al.* (2010) as giving the same run time. In related work, Hein and Tanner (2010) gave a detailed analysis of the search algorithm on d-dimensional lattices in terms of the level dynamics near an avoided crossing. They found the same additional $\sqrt{\log N}$ factor in the run time of the algorithm in two spatial dimensions. They also gave analytical expressions for the prefactors to the basic scaling of both the time to find the marked state and the maximum probability the marked state reaches. Finally, recent work by Ambainis *et al.* (2013) proves that the $\sqrt{\log N}$ factor can also be removed by adding a classical post-processing step checking the $O(\sqrt{N})$ neighbourhood of the measurement outcome. This classical post-processing is possible as they prove that the probability of being within this neighbourhood is constant, that is, $O(1)$. All of these results lend further weight to the belief that the two-dimensional case is the critical dimension, and it is unlikely that the full quadratic speed up is possible.

Almost all previous studies of the quantum walk search algorithm have focused on the dependence the algorithm has on the spatial dimension of the structure being searched. Little has been done to explore other factors that may affect the run time, and hence the efficiency of the algorithm. This is because the connectivity or lack of symmetry within interesting structures makes them hard to analyse analytically. However, Abal *et al.* (2010) showed 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 Ambainis (2003), but with a differing prefactor to the scaling of the algorithm. In addition, highly symmetric graphs such as the complete graph were studied in Reitzner *et al.* (2009), which showed that the additional connectivity does not allow the search to beat the optimal lower bound of $O(\sqrt{N})$. The hitting time on the complete graph has also been studied recently in Santos and Portugal (2010), which proved that this is also $O(\sqrt{N})$. Finally, the constant prefactors to the $O(\sqrt{N})$ scaling on the hypercube and d-dimensional lattices have been determined analytically in Hein and Tanner (2009) and Hein and Tanner (2010).

In order to obtain a broad overview of a range of factors, we chose to carry out a numerical investigation for the current paper. Quantum walks with single walkers are easy to simulate classically, allowing us to map out the territory in detail, identifying

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

whether and where there are questions worth tackling analytically, especially for cases where the lack of symmetry or presence of disorder in the structures makes analytical results extremely challenging. Numerical study also allows us to apply a new technique for analysing the dimension-dependent properties, which we call 'tunnelling', where extra edges are gradually 'turned on' to change the lattice from a two-dimensional to a three-dimensional structure. The specific factors we study here, dimensionality, connectivity and disorder, affect the efficiency of the search algorithm by altering the prefactors to the scaling of both the maximum probability of the marked state and the time to find this maximum probability. Analytical access to these prefactors might be possible in simple cases through the correspondence between the quantum walk of Ambainis *et al.* (2005) and the Szgedy walk Szegedy (2004) established by Magniez *et al.* (2009). This allows the quantum prefactors to be obtained from the corresponding classical Markov chain. For the more complex search structures we consider in the current paper, numerical estimates of the classical prefactors would be required. Unfortunately, it is harder to calculate the classical cases numerically since, because of the quadratically slower spreading compared with the quantum case, much longer simulations on larger lattices have to be performed – see Kendon *et al.* (2011) for a relevant example on percolation lattices. This puts the corresponding classical results out of reach: for example, our quantum walks on 5×10^4 sites would need to be compared with classical random walks on 2.5×10^9 sites.

1.1. *Structure of the paper*

The paper is organised as follows. In Section 2, we describe the discrete-time quantum walk and how it can easily be modified to become a search algorithm. We then investigate how the quantum walk search algorithm is affected by the dimensionality of the underlying substrate in Section 3. We do this by introducing a simple form of tunnelling that allows us to interpolate between structures of differing spatial dimension.

In Section 4, we study how varying the connectivity of regular structures affects the prefactors to the scaling of the algorithm. This also uses the same form of tunnelling, and extends previous work in Lovett *et al.* (2011). In Section 5, we investigate disorder in the underlying substrate. We model this using percolation lattices in both two and three dimensions to establish how much, if any, disorder the search algorithm can tolerate whilst still maintaining a quantum speed up.

We conclude with a discussion of our findings in Section 6.

2. The quantum walk search algorithm

2.1. *Discrete-time quantum walk on the infinite 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

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

position and $c \in \{0, 1\}$ is the state of the coin. The walker is started at the origin with an internal coin state of 0. At each time step, 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

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

and the shift operation S acts on the basis states thus

$$\begin{aligned} S|x, 0\rangle &= |x - 1, 0\rangle \\ S|x, 1\rangle &= |x + 1, 1\rangle. \end{aligned} \tag{2}$$

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

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

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 Equation (3), which causes some probabilities to be amplified or decreased at each time step. This leads to the different behaviour compared to its classical counterpart: spreading at a rate proportional to t, which is 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 in both Ambainis *et al.* (2001) and Nayak and Vishwanath (2000).

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 that the coin operator is now deterministic. However, if we were to

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

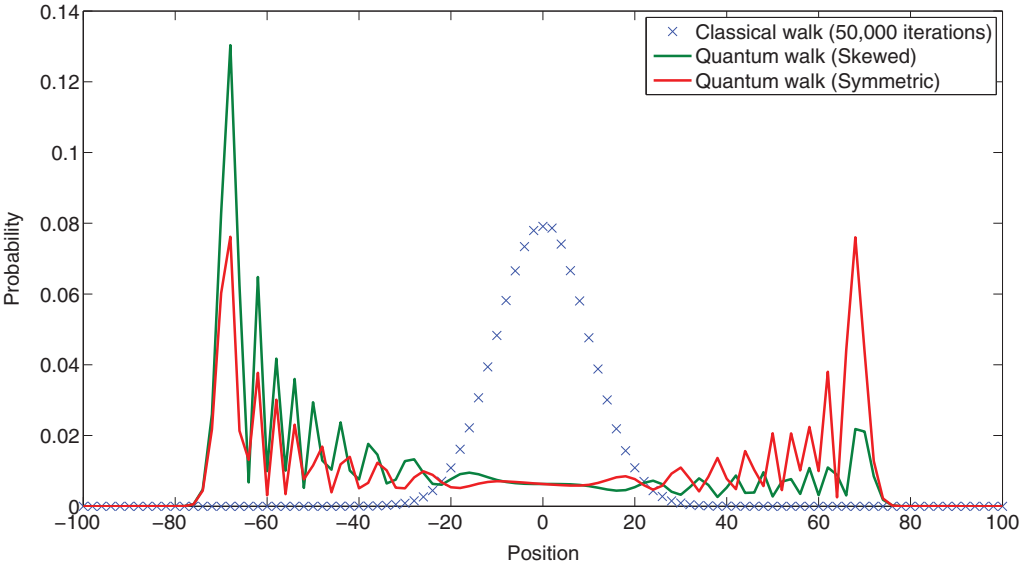


Fig. 1. Classical (crosses) and quantum (solid lines) probability distributions for walks on a line after 100 time steps. Only even positions are shown, since odd positions are zero. The classical walk is averaged over 50,000 iterations of the random walk. A skewed quantum walk is shown with an initial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

measure the coin after the required number of time steps, we would get a random output as in the classical case. We show both the classical and quantum probability distributions after 100 steps in Figure 1. If the walk is imperfect and some decoherence is allowed, we can see the gradual change from the quantum case back to the classical. Kendon and Tregenna (2003) investigated this in detail, and showed 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. For a review of the effects of decoherence in quantum walks, see Kendon (2007).

2.2. *Discrete-time quantum walk in higher dimensions*

We can see 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. In order to define the walk for these higher dimensions, we require a new coin operator of dimension d in order to span the entire coin state space of the walker (Moore and Russell 2002; Mackay *et al.* 2002; Kendon 2003; Kempe 2003a). This can be any unitary operator of the required dimension. Clearly, many different possibilities exist, but we will only mention the most common operator

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

here, namely, the Grover coin:

$$G^{(d)} = \begin{pmatrix} \frac{2}{d} & \cdots & \frac{2}{d} \\ \vdots & \ddots & \vdots \\ \frac{2}{d} & \cdots & \frac{2}{d} \end{pmatrix} - I_d, \tag{4}$$

where d is the degree of the vertex and I_d is the identity operator of the same dimension. The Grover coin is not symmetric, but it is balanced, that is, it treats all directions in the same way (up to a phase factor) for the cases where $d = 2$ and $d = 4$. In the $d = 3$ case, and all higher dimensions, the coin treats one edge differently to the other $d - 1$. 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 (2003) treats this problem rigorously, but the most important thing is to maintain a consistent labelling approach for each of the edges.

The dynamics of the walk on higher dimensional structures was studied in Mackay *et al.* (2002) and then in more detail in Tregenna *et al.* (2003). They carried out a numerical study of the spreading of the quantum walk with varying coin operators and initial states. 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 Tregenna *et al.* (2003) by the second moment). Using the Grover coin on a two-dimensional Cartesian lattice, that is, with $d = 4$ and

$$G^{(4)} = \frac{1}{2} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}, \tag{5}$$

most of the initial states, including the symmetric initial state,

$$|\psi\rangle_{sym} = \frac{1}{2}\left(|0, L\rangle + i|0, R\rangle + |0, D\rangle + i|0, U\rangle\right), \tag{6}$$

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 shown in Figure 2. However, one specific state,

$$|\psi\rangle_{max} = \frac{1}{2}\left(|0, L\rangle - |0, R\rangle + |0, D\rangle - |0, U\rangle\right), \tag{7}$$

gives a maximal spreading, which is also shown in Figure 2. Inui *et al.* (2004) proved this localisation analytically for two-dimensional lattices. In addition to these results, some analytical results have been shown for d-dimensional lattices. Grimmett, Janson and Scudo (2004) proved that in the limit $n \to \infty$, X_n/n converges weakly, where X_n is the position at time n in the case of the infinite line. Gottlieb (2005) later extended this result to show convergence on d-dimensional lattices.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

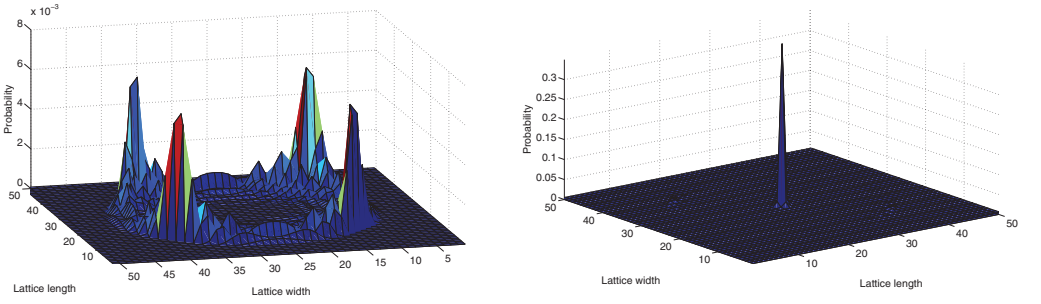


Fig. 2. Dynamics of the quantum walk on a two-dimensional Cartesian lattice using the Grover coin, Equation (5). The plot on the left-hand side shows the maximum spreading obtained using the initial state in Equation (7). The plot on the right-hand side shows the localisation obtained using the initial state in Equation (6). Note the different scales.

2.3. *The discrete-time quantum walk search algorithm*

We now describe how Shenvi *et al.* (2003) was 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 two-dimensional Cartesian lattice. The data points we wish to search are laid out as the vertices of an undirected graph. The edges then represent the specific connections between data points. We impose periodic boundary conditions at the edges of the lattice, in effect turning the graph into a torus. Our aim is to find a single data item, a specific vertex, within 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, that is,

$$|\psi\rangle = \frac{1}{\sqrt{dN}} \sum_{x=1}^{N} \sum_{c=1}^{d} |x, c\rangle, \tag{8}$$

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 of Equation (4), we would find a flat distribution identical to the starting state at any point in time. This uniform distribution is an eigenstate of the Grover coin operator. We need to use a different coin operator for the marked state in order to introduce a bias into the walk. It is optimal to invert the phase of the $G^{(4)}$ coin operator of Equation (5), as first suggested in Ambainis *et al.* (2005) and confirmed numerically in Lovett *et al.* (2011), giving

$$G_m^{(4)} = \frac{1}{2} \begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{pmatrix}. \tag{9}$$

Figure 3 shows how the distribution of the walker evolves with time for a 20×20 lattice, that is, $N = 400$. 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, Figure 4 shows that the probability at the marked state has a periodic behaviour, with the first

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

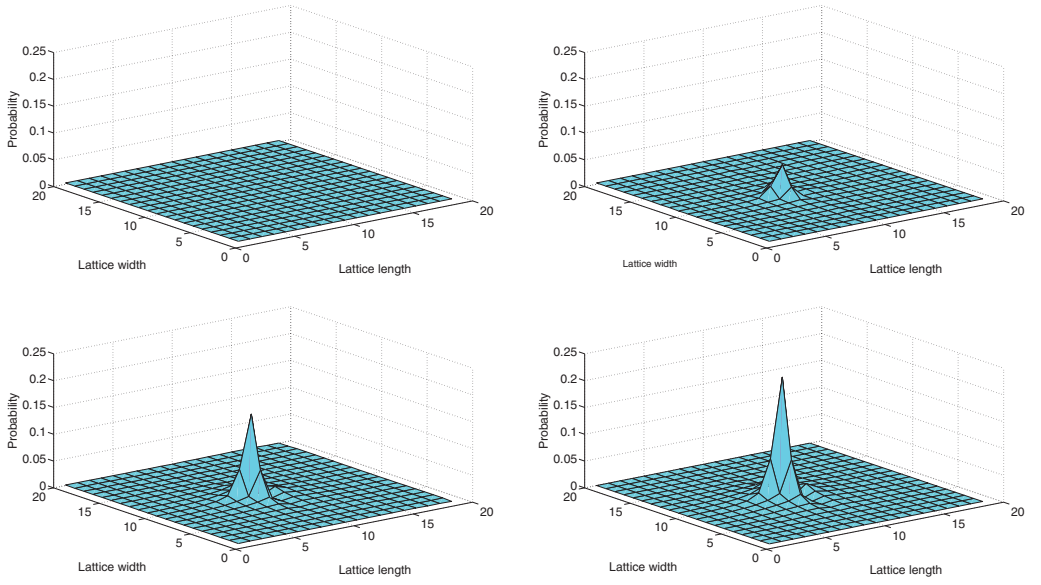


Fig. 3. Probability distribution of a discrete-time quantum walk search on 400 vertices arranged in a 20×20 square with periodic boundary conditions, evolved for 0, 10, 20 and 32 time steps. The marked vertex is in the centre of the lattice.

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 $n = 2, 3, 4.....$

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 of no interest: we want to know when the first peak occurs. Ideally, we would measure the walker at the precise timing of the maximum in the first peak, but this is not strictly necessary. In fact, as Figure 4 shows, the peaks are quite broad, so even if an error occurs in choosing when to measure, it only means a slightly lower probability of finding the marked state, and 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 time steps ($t = (\pi/4)\sqrt{N} \simeq 16$), 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 sections, 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 find a specific item from a set of many, we must consider how likely it is that the walker coalesces at the marked state. The maximum probability of the walker at the marked state, that is, the maximum value of the first peak, varies with the size of the dataset (for the two-dimensional Cartesian lattice). In this case, the theoretical value of $O(1/\log_2 N)$ from Ambainis *et al.* (2005) is numerically confirmed in our results

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

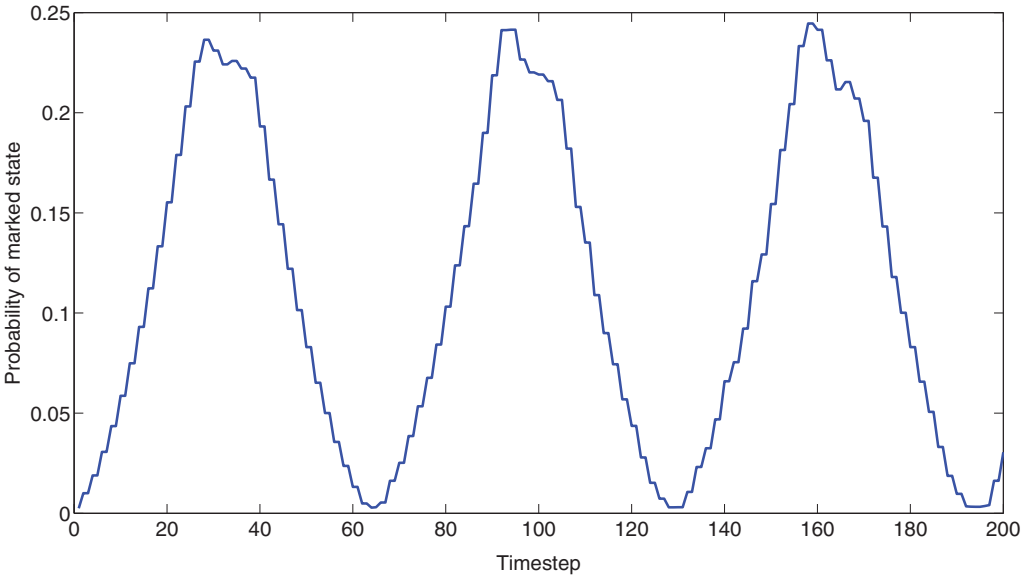


Fig. 4. Probability of finding the quantum walker at the marked state over 200 time steps on a 20×20 grid with periodic boundary conditions, with the same grid as in Figure 3.

in Figure 5 with a small prefactor of just over 2. The second factor we are interested in is the number of time steps it takes to reach this maximum probability. The scaling of the time to find the marked state with the size N of the dataset for the two-dimensional Cartesian lattice is shown in Figure 6. We see a scaling of $O(\sqrt{N})$ here, also with a prefactor of 2. This scaling is less than the $O\sqrt{N \log N}$ given by Ambainis *et al.* (2005). This is because the peaks of the probability are broad, in a similar fashion to that found in Ambainis *et al.* (2013), so we take an average of the time when the probability is more than 70% of the maximum.

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 two-dimensional Cartesian lattice, the maximum probability scales as $O(1/\log_2 N)$. Hence, 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 Brassard (2002). This makes the total algorithmic complexity $O(\sqrt{N \log N})$ for the two-dimensional lattice, which agrees with the recent results in Tulsi (2008) and Magniez *et al.* (2009). However, these scalings are not the same for all graph structures. In particular, on a cubic lattice, the maximum probability scales as a constant value $O(1)$, so only a constant number of amplification steps are needed to bring the probability to ≈ 1, and hence the total algorithmic complexity is just $O(\sqrt{N})$.

3. Dimensionality

In this section, we investigate how the spatial dimension of the database arrangement affects the searching algorithm. We already know, as noted in Section 2, that the basic

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

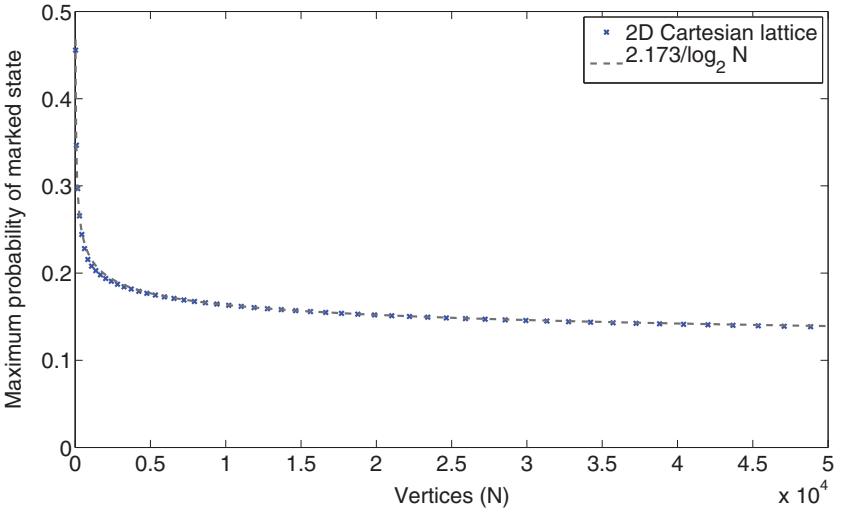


Fig. 5. Plot showing the maximum of the first peak in the probability of being at the marked state for different sized data sets, using the optimal marked state coin in Equation (9) on a two-dimensional lattice of size $\sqrt{N} \times \sqrt{N}$, plotted against N (crosses). Also shown is the closest fit to our data, which is $2.173/\log_2 N$ (dashes).

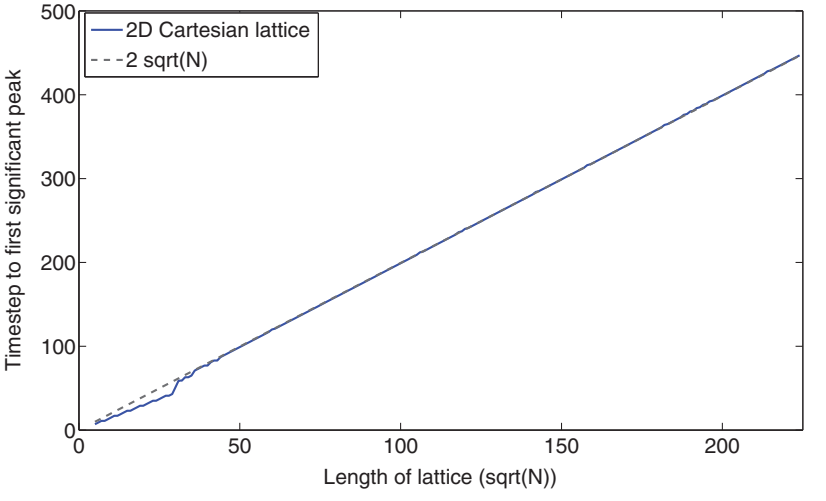


Fig. 6. Plot showing the time step at which the first peak in the probability of being at the marked state occurs for different sized data sets, using the optimal marked state coin in Equation (9), plotted against $\sqrt{N}$. Also shown is the closest fit to our data, which is $2\sqrt{N}$.

scaling of the algorithm is heavily dependent on 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 change for structures of differing spatial dimension. We are interested here in how this basic scaling changes when the spatial dimension is altered. We will investigate this in two ways: first by introducing a simple form of tunnelling, which allows us to interpolate between structures of varying spatial dimension, and then

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

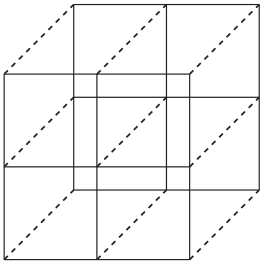


Fig. 7. Basic unit of the structure we use to interpolate between the two-dimensional Cartesian and three-dimensional cubic lattices. The solid lines represent the fixed, normal edges and the dashed lines represent the edges we set to be tunnelling.

by using lattices of varying height (one dimension to two dimensions) and depth (two dimensions to three dimensions).

3.1. *Tunnelling operator*

In this section, we describe a modified coin operator, which we will use in the search algorithm to gradually vary the connectivity of the structures studied. In particular, 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 set of two-dimensional Cartesian lattices into a cubic lattice by introducing connecting links between the lattices – see Figure 7.

In order to achieve the quantum walk dynamics we require, we need to use a different coin operator. The only condition on this operator is that it must be unitary, so we can 'design' a new coin operator that 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 for the Grover coin, Equation (4), 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 Figure 7, the solid edges are normal fixed edges creating a two-dimensional lattice, and the dashed edges are tunnelling edges, which convert the two-dimensional lattice to a cubic one. The general matrix for the desired coin operator would be as follows:

$$T_{d,t} = \begin{pmatrix} a & b & b & \dots & b & c & c & c & \dots & c \\ b & a & b & \dots & b & c & c & c & \dots & c \\ . & . & . & \dots & . & . & . & . & \dots & . \\ . & . & . & \dots & . & . & . & . & \dots & . \\ b & b & b & \dots & a & c & c & c & \dots & c \\ c & c & c & \dots & c & e & f & f & \dots & f \\ c & c & c & \dots & c & f & e & f & \dots & f \\ . & . & . & \dots & . & . & . & . & \dots & . \\ . & . & . & \dots & . & . & . & . & \dots & . \\ c & c & c & \dots & c & f & f & f & \dots & e \end{pmatrix} \tag{10}$$

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

where the blocks of '*abbb*...' are $d-t$ square and the blocks of '*efff*...' are t square. 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, so

$$\left(T_{d,t}\right)\left(T_{d,t}\right)^{\dagger} = \mathbb{I}_d, \tag{11}$$

must hold, where $\left(T_{d,t}\right)^{\dagger}$ is the hermitian conjugate of the general matrix. We can solve the five equations formed as a consequence in terms of just the degree of the lattice d, the number of tunnelling edges t and the tunnelling parameter c, giving

$$a = b - 1 \tag{12}$$

$$b = \frac{1 + \sqrt{1-(d-t)tc^2}}{d-t} \tag{13}$$

$$e = f - 1 \tag{14}$$

and

$$f = \frac{1 - \sqrt{1-(d-t)tc^2}}{t}. \tag{15}$$

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 no more than half the degree, that is, $t \leqslant d/2$. By using vertices of degree six with two tunnelling edges, we have a set of basic two-dimensional Cartesian lattices, which gradually become a cubic lattice as in Figure 7. In this case, setting $c = 0$, we obtain

$$T_{6,2} = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 1 & 0 & 0 \\ 1 & 1 & -1 & 1 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \tag{16}$$

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

$$T_{6,2} = \frac{1}{3}\begin{pmatrix} -2 & 1 & 1 & 1 & 1 & 1 \\ 1 & -2 & 1 & 1 & 1 & 1 \\ 1 & 1 & -2 & 1 & 1 & 1 \\ 1 & 1 & 1 & -2 & 1 & 1 \\ 1 & 1 & 1 & 1 & -2 & 1 \\ 1 & 1 & 1 & 1 & 1 & -2 \end{pmatrix}. \tag{17}$$

These choices of c represent the extremes of the operator when the structure would be either a basic two-dimensional Cartesian lattice, Equation (16), or a cubic lattice, Equation (17). Any other values of c where $0 \leqslant c \leqslant 2/d$ would give varying strengths of tunnelling across the tunnelling edges.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

3.2. *Interpolating between the two-dimensional and three-dimensional lattices using the tunnelling operator*

We used the tunnelling operator introduced in the previous section to carry out a numerical study of how the algorithm is affected by the change in spatial dimension. We used the operator to interpolate between a Cartesian lattice (2D) and a cubic lattice (3D). In this case, we interpolated between a set of two-dimensional Cartesian lattices and a fully connected cubic lattice by using vertices of degree six with two tunnelling edges with the correct connectivity – see Figure 7.[†]

The search algorithm we use is the same as in Shenvi *et al.* (2003), but the tunnelling requires us to make a small modification to the initial state. In their original algorithm, the initial state is chosen to be a uniform superposition over all the vertices, so no state is favoured over any other. The distribution over the edges at each vertex is also uniform, so that without a marked state, the quantum walk dynamics from this initial state is stationary. In order to obtain the same properties, we still start the walker in a uniform superposition over all vertices, but the distribution over the edges must be altered 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:

$$(d-t)\alpha + tp\alpha = \frac{1}{\sqrt{N}}, \tag{18}$$

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 1. In this way, the tunnelling probability matches the proportion of the initial state placed on the tunnelling edges. It is easy to check that this initial state gives a probability distribution, where there is no marked state, that is periodic over two time steps. 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 time steps means it will give rise to the required dynamics.

We ran the algorithm on a three-dimensional cubic lattice where the edges linking the 'slices' of two-dimensional lattices together are tunnelling – Figure 7 shows a basic unit in the lattice, where 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 that make the structure three dimensional, thereby interpolating between the two-dimensional Cartesian lattice and the three-dimensional cubic lattice.

The maximum probability of the marked state shown in Figure 8 changes from the $1/\log_2 N$ scaling in the two-dimensional case to the constant $O(1)$ scaling as soon as the additional edges have even a small weighting attached to them. At low tunnelling strengths, the probability drops 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

[†] We have also performed the same investigation interpolating between the one-dimensional line and the two-dimensional lattice, and found a quantitatively similar result. However, the results were more difficult to analyse because of the search algorithm failing as the structure approached a line, so we will not present them here.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

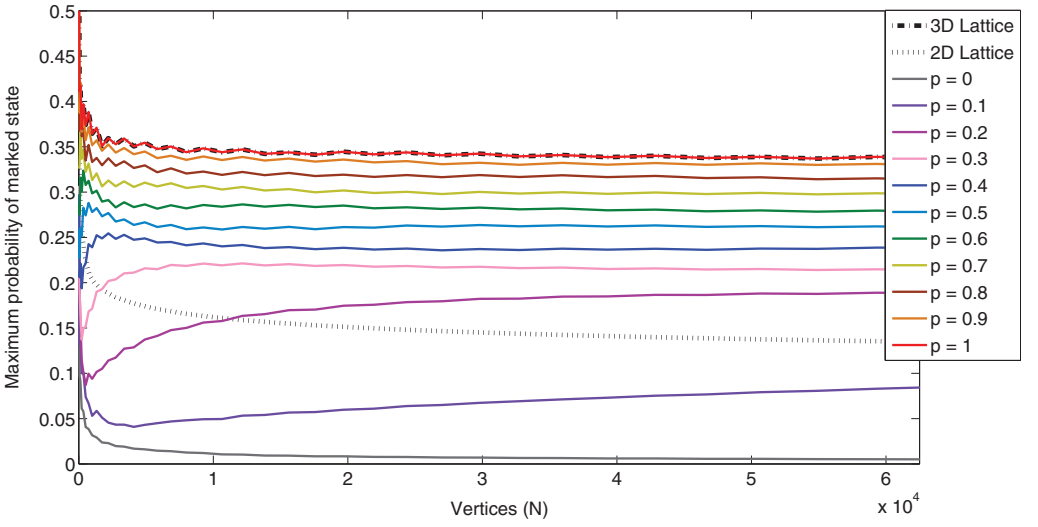


Fig. 8. Plot showing how the maximum probability of the marked state varies with both the size of the lattice and the tunnelling strength as a two-dimensional lattice is gradually changed into a three-dimensional Cartesian lattice.

constant for any tunnelling strength, and just have varying prefactors. Figure 9 shows how this prefactor to the scaling of the maximum probability of the marked state changes as we increase the tunnelling probability. Note that in determining the prefactor, we only used the data for $N > 40{,}000$ since at that point all the probabilities seem to have converged on a constant prefactor. The sharp drop at low tunnelling probabilities is probably due to the fact that the scaling has not reached the constant value since we can only simulate up to a fixed lattice size.

Note also that when the probability of tunnelling is zero, the scaling does not match that of the basic two-dimensional lattice. At $p = 0$, the structure is in effect a collection of unlinked two-dimensional lattices. The initial state is still spread across all these individual lattices, and because of the connectivity of the structure, only the amplitude in one of the lattices (the one containing the marked state) is able to coalesce on the marked state. This means that the scaling is just reduced by a constant factor, as shown in Figure 8.

The time to find the marked state follows a similar behaviour. Figure 10 shows how the scaling of the time to find the marked state varies with both the size of the lattice and the tunnelling strength. The basic scaling of the time to find the marked state is $O(\sqrt{N})$ in both two and three dimensions. 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. Figure 11 shows how this prefactor varies with the tunnelling strength. Note that at very low tunnelling probabilities, specifically, $p <\approx 0.3$, the scaling of the time to find the marked state does not follow the same behaviour. This is because, as mentioned previously, the probability at the marked state has not yet reached a constant value. If we were able to run the algorithm on much larger sized lattices, we should find the probability of the marked state stabilising, and thus

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

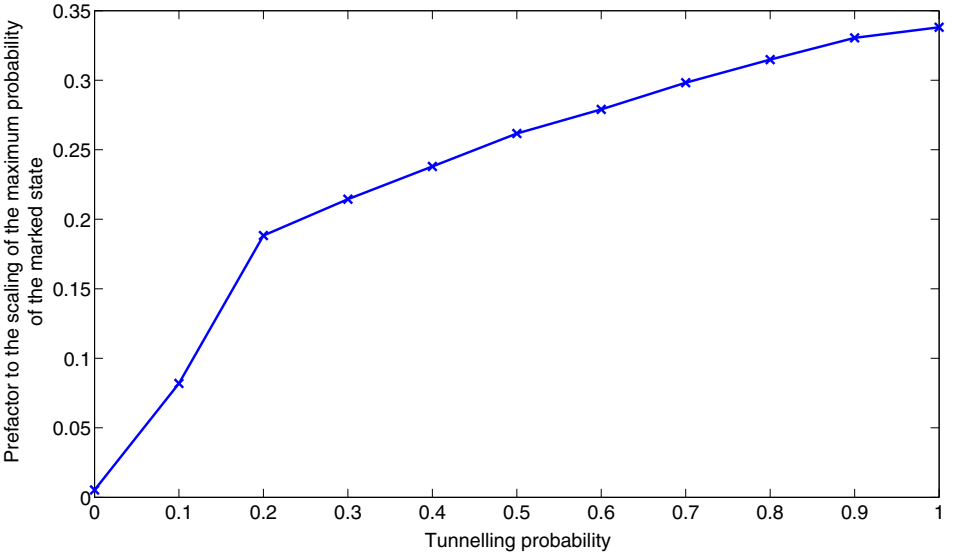


Fig. 9. Plot showing how the prefactor to the scaling of the maximum probability of the marked state, which was obtained from the data in Figure 8, varies with the tunnelling strength as a two-dimensional Cartesian lattice is gradually changed into a three-dimensional lattice. Note that the range of N used to obtain this data was from 40,000 upwards in Figure 10.

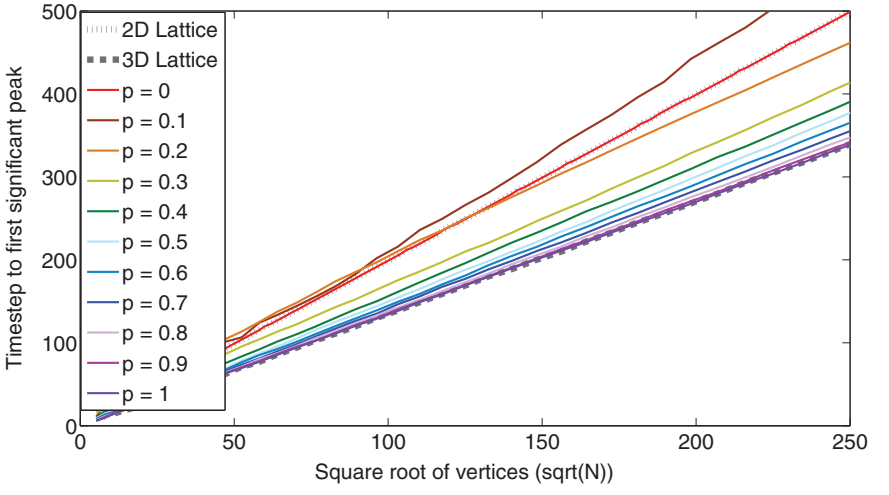


Fig. 10. Plot showing how the time to find the marked state varies with both the size of the lattice and the tunnelling strength as a two-dimensional lattice is gradually changed into a three-dimensional Cartesian lattice.

the time to find the marked state matching the quadratic speedup in scaling. Figure 11 shows the expected trend to the scaling of the prefactors to the time to find the marked state.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

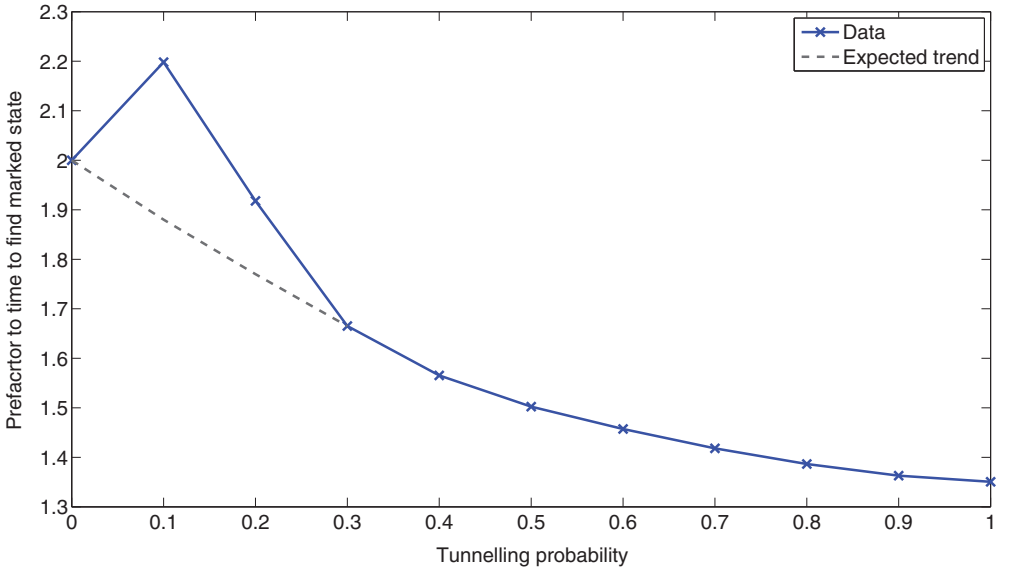


Fig. 11. Plot showing how the prefactor to the scaling of the time to find the marked state, which was obtained from the data in Figure 10, varies with the tunnelling strength as a two-dimensional lattice is gradually changed into a three-dimensional Cartesian lattice.

3.3. *Interpolating between the two-dimensional and three-dimensional lattices by varying the depth*

In this section, we discuss lattices of varying depth to provide a different avenue of investigation of the dependence on spatial dimension. At low depths, a three-dimensional lattice can be viewed in effect as a two-dimensional lattice. We are interested in how the scaling of the probability of the marked state and the time to find it changes as the lattice depth is gradually increased, eventually becoming a fully symmetric three-dimensional lattice[‡].

We will first show how the maximum probability of the marked state varies for a fixed width and height of lattice (30×30) and varying depth. Figure 12 shows how this probability varies, along with the equivalent scaling for the fully cubic three-dimensional lattice, which shows that the scaling matches at roughly $l = 15$, where l is the depth of the lattice.

We will now fix the number of vertices in the lattice and alter the width, height and depth of the lattice for each run of the search algorithm. Figure 13 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 two-dimensional logarithmic scaling

[‡] As with the results of the tunnelling operator, we have also investigated this interpolation between the one-dimensional line and the two-dimensional lattice and found a quantitatively similar behaviour, but, again due to the algorithm failing as the structure approaches the line, the behaviour can be seen much more clearly in the case of interpolating between two and three dimensions.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

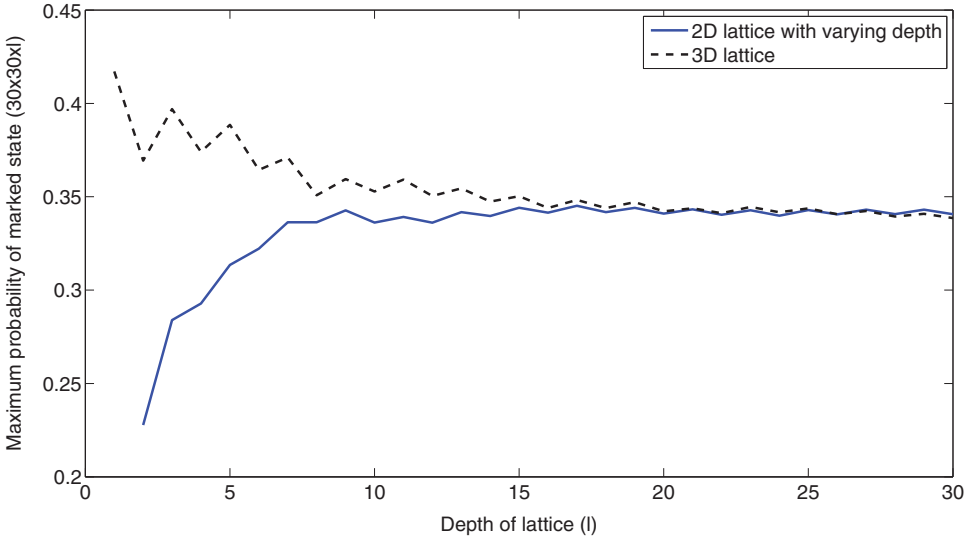


Fig. 12. The maximum probability of the marked state as the depth of the three-dimensional lattice is increased from one layer to a perfect cube of 30x30x30 sites (solid line). The maximum probability of the marked state for cubic lattices of varying size is shown for comparison (dashed line).

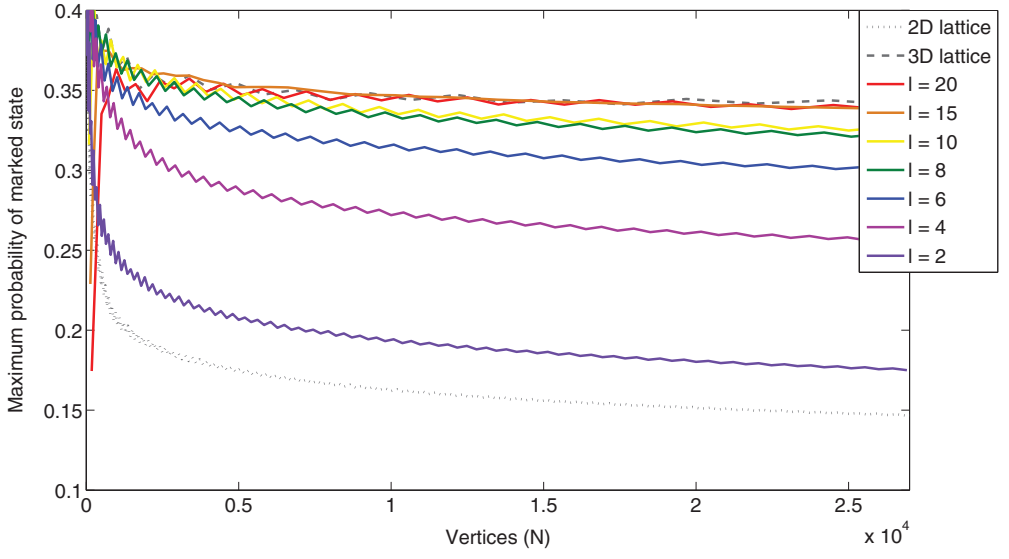


Fig. 13. Plot showing how the maximum probability of the marked state varies as a two-dimensional lattice is gradually increased in depth (l) to become a three-dimensional cubic lattice. We maintain the same number of vertices in each case.

to the constant scaling of the cubic lattice. This is in contrast to the almost instantaneous change in scaling when we interpolate 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. However, the prefactor to this scaling does decrease as we increase

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

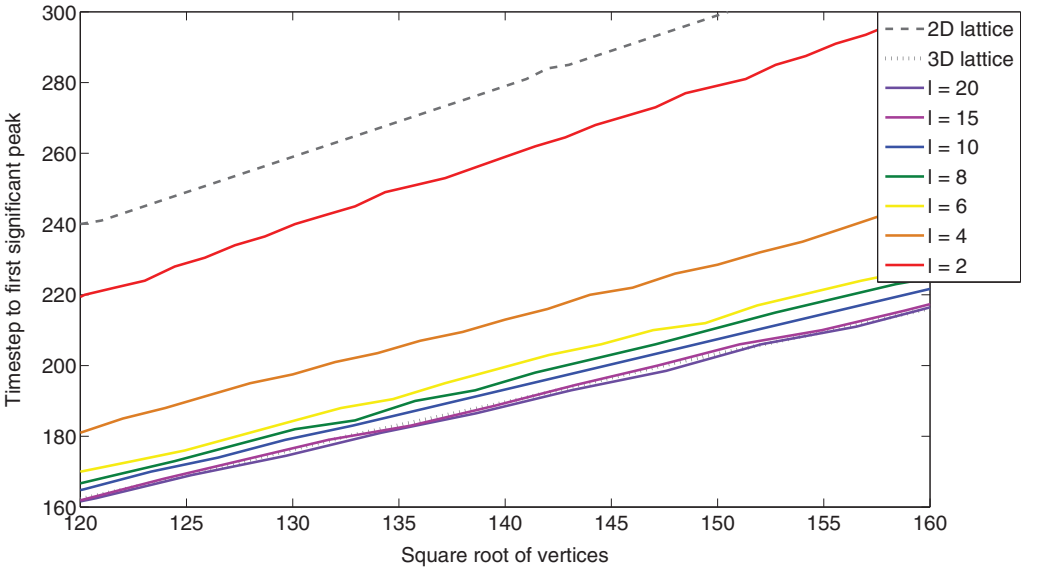


Fig. 14. Plot showing how the time to find the marked state varies as a two-dimensional lattice is gradually increased in depth (l) to become a three-dimensional cubic lattice. We maintain the same number of vertices in each case.

the depth, changing from that of the basic two-dimensional lattice to almost match that of the cubic lattice, even for lattices of modest depth.

4. Connectivity

In this section we investigate how important the connectivity of the database arrangement is for the searching algorithm. As previously discussed, the basic scaling of the algorithm is heavily dependent on the spatial dimension of the structure in question. Here, we carry out a numerical study of how the connectivity in a specific spatial dimension affects the prefactors to this scaling. Although it is unlikely that the run time of the search algorithm on a two-dimensional 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.

We use the tunnelling coin operator introduced in the previous section, 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 two-dimensional Cartesian lattice, with $d = 4$. 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 Lovett *et al.* (2011) by considering an interpolation between lattices with fixed degree. We will begin by introducing the structures we wish to perform the search algorithm upon, and then present our results for both two- and three-dimensional structures with varying connectivity. We will show that the prefactors to the scaling of the algorithm for both

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

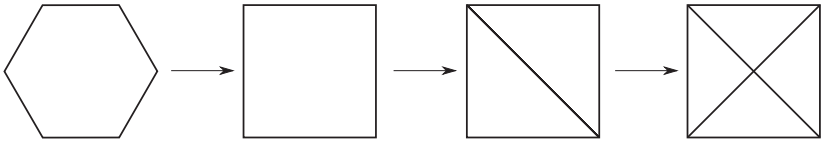


Fig. 15. The two-dimensional lattices we interpolate between using the tunnelling matrix. We change gradually from a hexagonal lattice, $d = 3$, to a two-dimensional Cartesian lattice with diagonals included, $d = 8$. The diagrams just show the building blocks of each lattice. Note that in the case of the two-dimensional Cartesian lattice with diagonals, there is no vertex at the central point where the edges cross.

the maximum probability of the marked state and the time to find the marked state are dependent on the connectivity of the underlying structure.

4.1. *Two-dimensional structures*

Using the tunnelling matrix we introduced earlier, we ran the search algorithm on two-dimensional lattices ranging from $d = 3$, a hexagonal lattice, through to $d = 8$, a Cartesian lattice with diagonals added as shown in Figure 15, and for varying lattice sizes from 6^2 (36) vertices up to 250^2 (62500) vertices. As in Figure 15, we gradually changed the degree of the structure that we performed the search algorithm on. This was split into intermediate steps: first from the two-dimensional hexagonal lattice ($d = 3$) to the square lattice ($d = 4$); then from the square lattice to the triangular lattice ($d = 6$); and finally to the more highly connected Cartesian lattice with diagonals ($d = 8$). We spread the walker in the same way as in Equation (18) to ensure we distribute the state evenly based on the tunnelling strength of the edges.

Figure 16 shows how the time to find the marked state varies with both the size of the lattice and the connectivity. The plot shows 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 make fits to each of the data sets in Figure 16 to determine the prefactor to the scaling of the time to find the marked state. Figure 17 shows how this prefactor to the scaling changes with the degree of the underlying structure being searched. Note that although there is no specific scaling here, there does seem to be a symmetry effect for integer degree, though less strong than might have been expected.

Figure 18 shows how the maximum probability of the marked state varies with both the size of the dataset and the connectivity of the structure. The plot shows 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 probability of success close to unity. Figure 19 shows how this prefactor to the scaling of $O(1/\log_2 N)$ varies with the degree of the structure being searched. The 'dips' in the scaling seem counter-intuitive, but appear to arise from the dynamics of the walk on these structures when 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

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

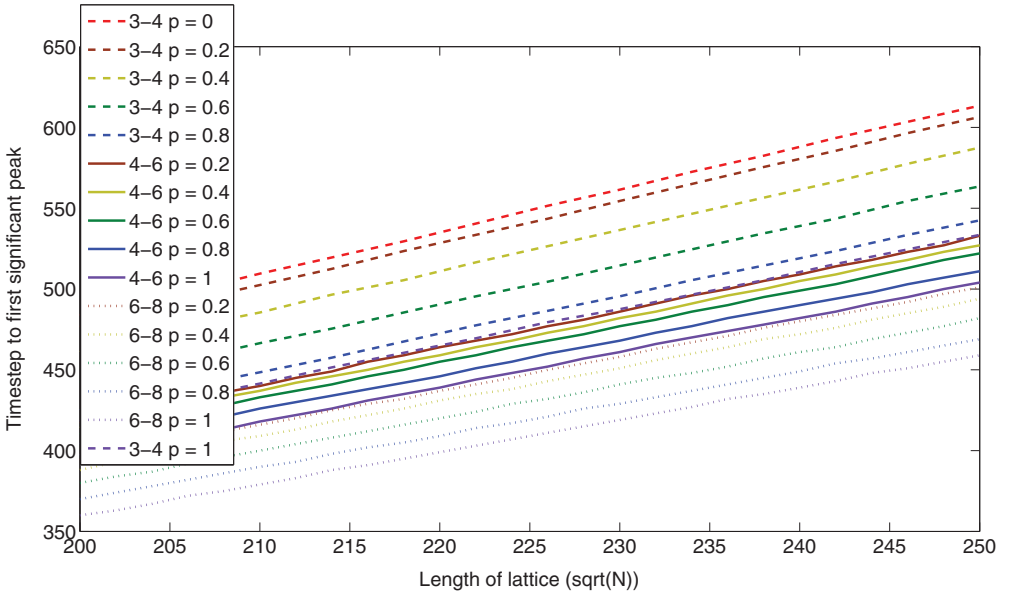


Fig. 16. Plot showing how the time to find the marked state varies with both the size of the lattice and the connectivity in two dimensions. It is clear that as the connectivity of the structure increases, the time to find the marked state decreases. Note that this is a zoomed in plot showing only larger lattices sizes – the data for $\sqrt{N} < 200$ have been omitted for greater clarity.

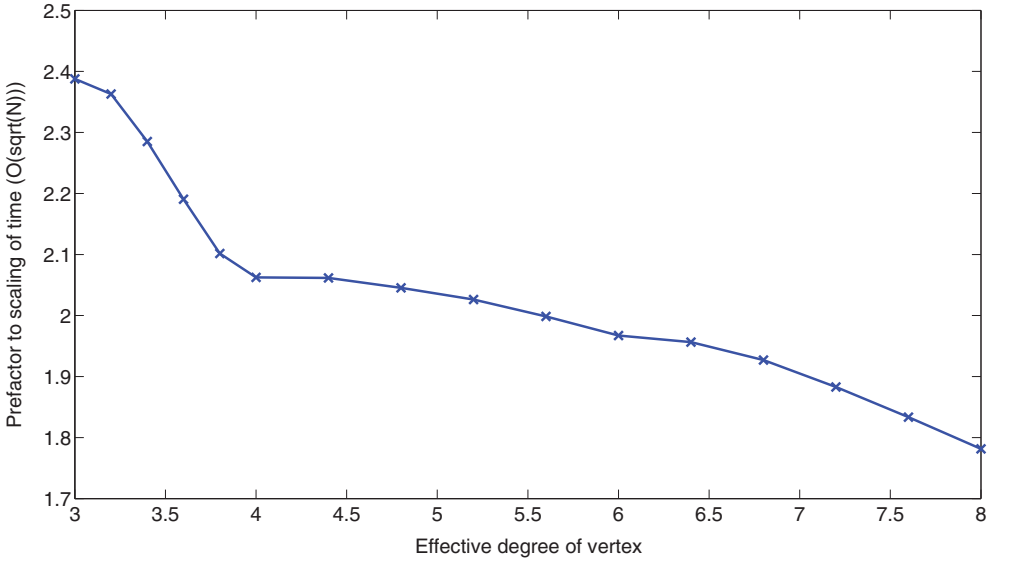


Fig. 17. Plot showing how the prefactor to the scaling obtained from the data shown in Figure 16 of the time to find the marked state of $O(\sqrt{N})$ changes with the degree of the two-dimensional structure being searched.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

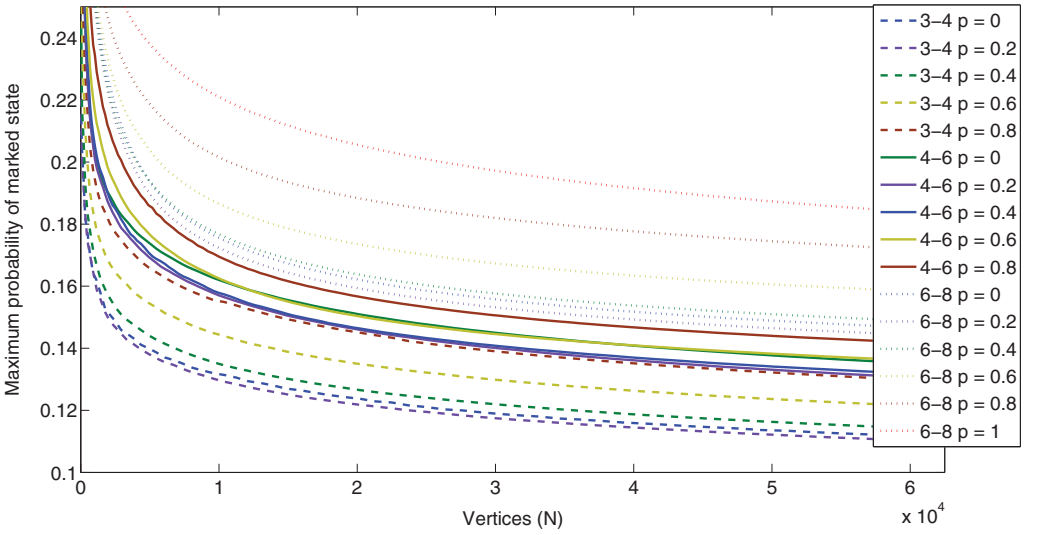


Fig. 18. Plot showing how the maximum probability of the marked state varies with both the size of the lattice and the connectivity in two dimensions. In general, as the connectivity of the structure increases, the maximum probability of the marked state also increases.

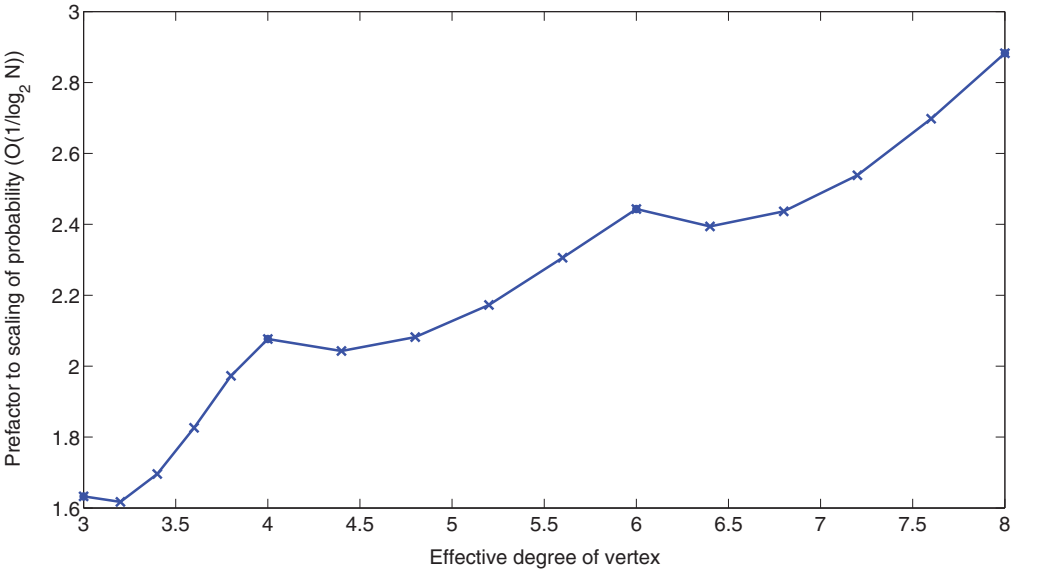


Fig. 19. Plot showing how the prefactor to the scaling, which was obtained from the data in Figure 18, of the maximum probability of the marked state of $O(1/\log_2 N)$ changes with the degree of the two-dimensional structure being searched.

a specific vertex in the graph, as opposed to an equal superposition, and allowed it to propagate outwards in order to determine its dynamics. In this case, we are interested in the basic dynamics of the walker, so the lattice has no marked vertex. We define the

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

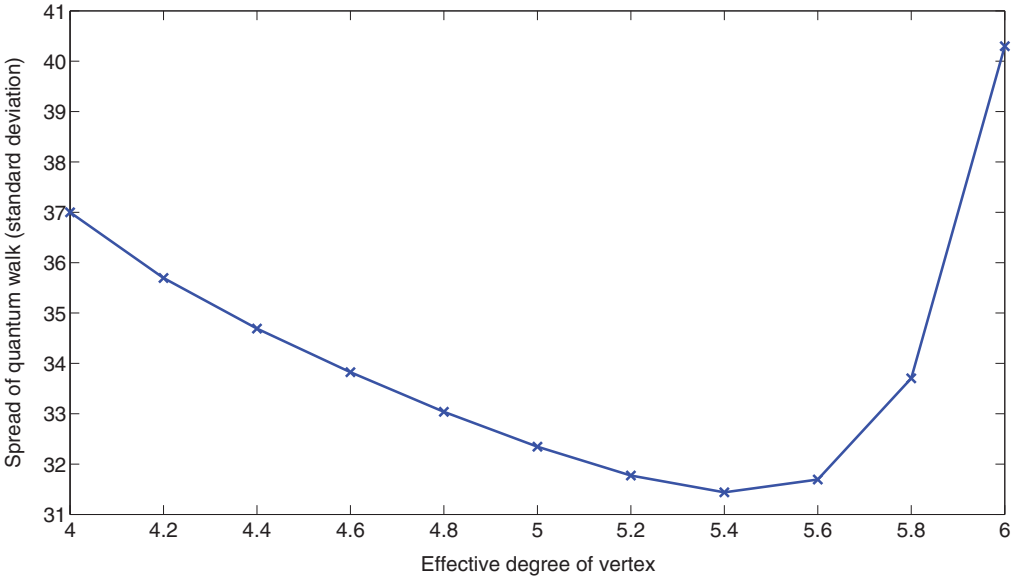


Fig. 20. Plot showing how the spreading of the quantum walk, characterised by Equation (19), changes with the degree of the two-dimensional structure being searched. In this case, we show the dynamics of the spread of the quantum walk on a two-dimensional Cartesian lattice ($d = 4$) being changed into a triangular lattice ($d = 6$)

spread of the walker as

$$\langle r \rangle = \sum_{i=1}^{N} p_i s_i, \tag{19}$$

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 20 shows this spreading on a two-dimensional Cartesian lattice when it is gradually turned into a triangular lattice as in Figure 15. We found at low tunnelling strengths, where the symmetry breaking is most obvious, that the spread $\langle r \rangle$ decreased. As the tunnelling strength was increased, the quantum walk was able to recover and $\langle r \rangle$ increased back to the value of the original lattice, before further increasing as the tunnelling strength reached its maximum value, that is, 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 found 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. Because of the periodic boundary conditions imposed in the searching algorithm, we found slightly different behaviour, which is related to the extra interference effects. We found similar results for all the lattices studied in both two and three dimensions.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

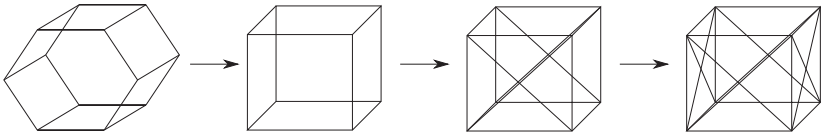


Fig. 21. The three-dimensional lattices we interpolate between using the tunnelling matrix. We change gradually from a three-dimensional hexagonal lattice, $d = 5$, through to a cubic lattice with diagonals added on the faces, $d = 14$. The diagrams only show the building blocks of each of the lattices. Note that vertices are only present at the eight corners of the cubic structures; there are no vertices where the edges cross.

4.2. *Three-dimensional structures*

We now consider three-dimensional lattices and use the tunnelling matrix to study structures ranging from $d = 5$, a three-dimensional hexagonal lattice, through to $d = 14$, a cubic lattice with additional diagonals added as shown in Figure 21.

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 but split it into intermediate steps as shown in Figure 21. We first changed from the three-dimensional hexagonal lattice ($d = 5$) to the cubic lattice ($d = 6$), then from the cubic lattice to one with diagonals added to one face only ($d = 10$), and finally to a cubic lattice with diagonals added to 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 22 shows 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, and thus the efficiency of the algorithm increases. As the time to find the marked state scales as $O(\sqrt{N})$, we made fits to each of these to obtain the prefactor to the scaling of the time to find the marked state. Figure 23 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 as a constant, *viz*. $O(1)$, so this scaling does not affect the complexity of the algorithm in the same way as in the two-dimensional case. However, note that this constant probability value does affect how many times we must run the algorithm to ensure we have the correct result. Figure 24 shows that, in general, as the connectivity of the structure being searched increases, the maximum probability of the marked state also increases.

The closer we can get this prefactor to unity, the fewer times we need to run the algorithm. Figure 25 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' in the scaling as in the two-dimensional case, and they can be explained in the same way.

5. Substrate disorder

So far we have only considered perfect, regular lattices with no defects. We now consider a simple form of noise (disorder) and are interested in how this affects the efficiency of the search algorithm. Previous work by Keating *et al.* (2007) has highlighted the effect

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

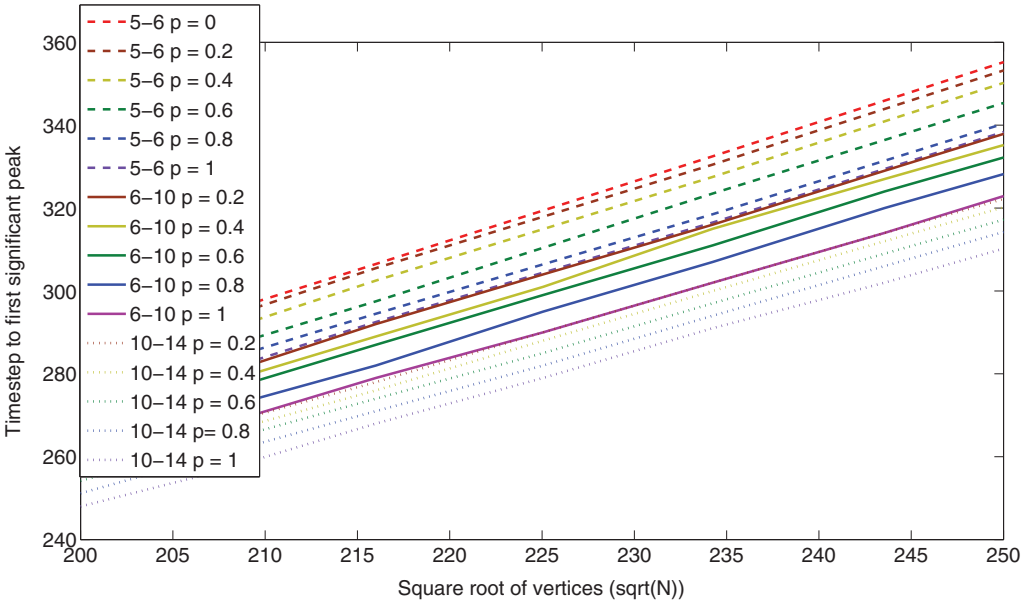


Fig. 22. Plot showing how the time to find the marked state varies with the size of the lattice and varying connectivity in three dimensions. It is clear that as the connectivity of the structure increases, the time to find the marked state decreases. Note that this is a zoomed in plot showing large lattices sizes – the data for $\sqrt{N} < 200$ have been omitted for greater clarity.

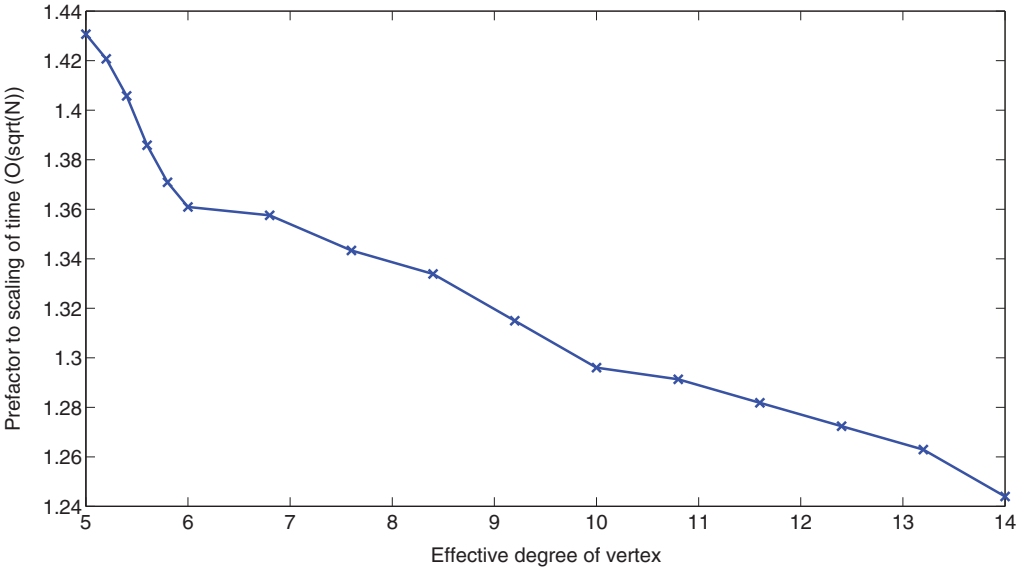


Fig. 23. Plot showing how the prefactor to the scaling, which was obtained from the data shown in Figure 22, of the time to find the marked state of $O(\sqrt{N})$ changes with the degree of the three-dimensional structure being searched.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

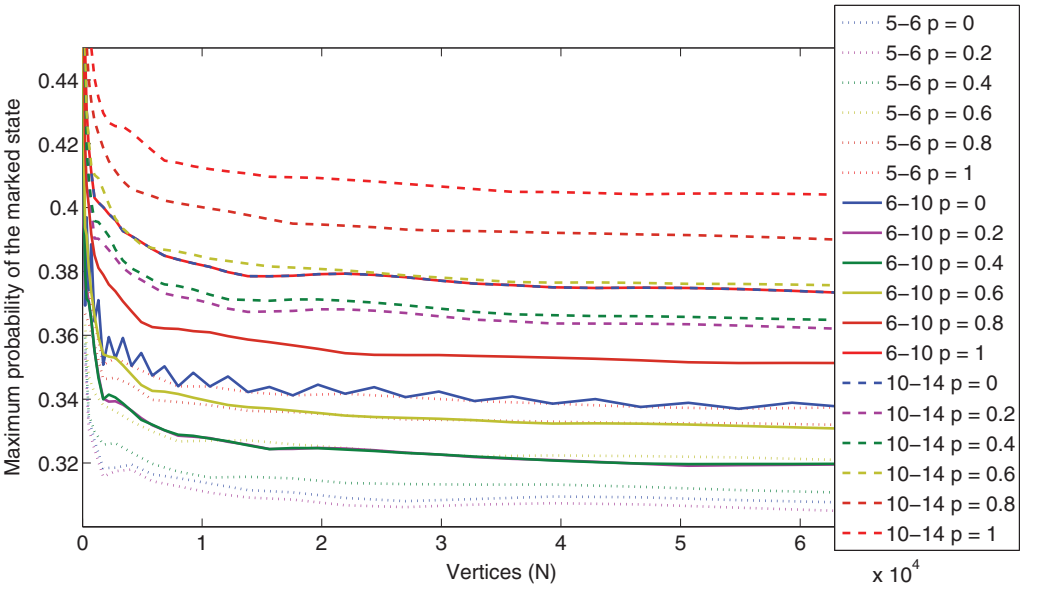


Fig. 24. Plot showing how the maximum probability of the marked state varies with the size of the lattice and the connectivity in three dimensions. In general, as the connectivity of the structure increases, the maximum probability of the marked state also increases.

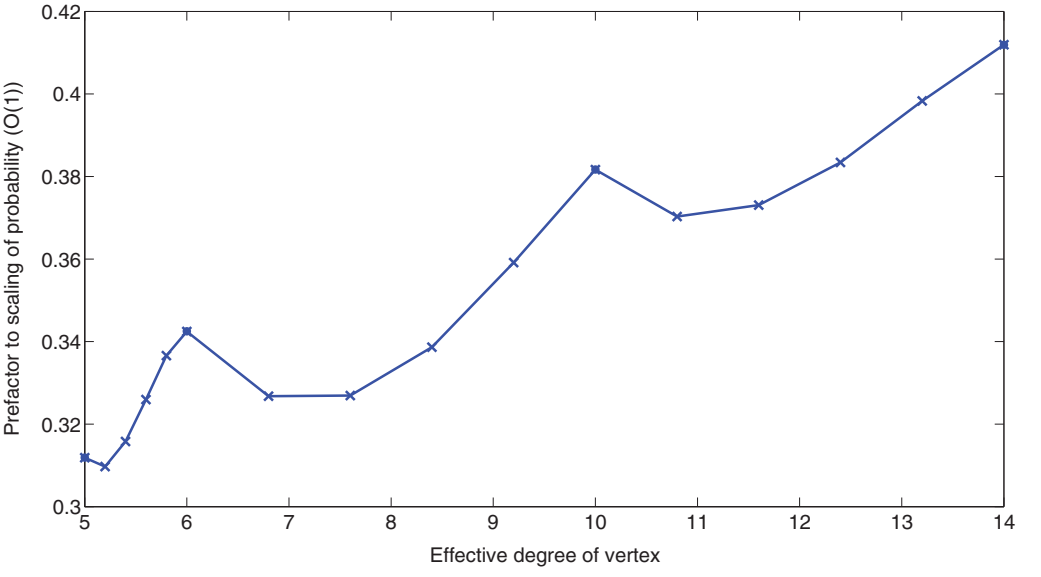


Fig. 25. Plot showing how the prefactor obtained from the data in Figure 24 of the maximum probability of the marked state changes with the degree of the three-dimensional structure being searched.

of Anderson localisation in continuous-time quantum walks and Krovi and Brun (2006a, 2006b, 2007) have shown how defects and a lack of symmetry in discrete-time quantum walks can have an impact on the spreading of the walk. Both these factors suggest that

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

the search algorithm will fail as soon as any level of disorder is introduced into the lattice. However, in contrast to these results, a study of the transport properties of discrete-time quantum walks on one- and two-dimensional percolation lattices was recently presented in Leung *et al.* (2010) and showed that the spreading of a discrete-time quantum walk on a two-dimensional percolation lattice follows a fractional scaling, that is, $\langle r \rangle \propto T^{\alpha}$ where $\langle r \rangle$ is the spread of the quantum walk and T is the number of time steps. This seems to contradict the previous work in the continuous-time context. Moreover, Abal *et al.* (2009) investigated how the quantum walk search algorithm performs in the presence of decoherence: specifically, for phase errors in the coin operator. In the current paper, we assume that we have a quantum computer with error correction available, so we are not interested in these errors. Instead, we are interested in any disorder that could be present in an imperfect data structure. Our aim is 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. Because of the computational time required for averaging over many lattices, we have only considered site percolation lattices here, though we expect a qualitatively similar behaviour in lattices with edge percolation.

5.1. *Percolation lattices*

A percolation lattice is a lattice, for example, a two-dimensional Cartesian lattice, that has vertices (site percolation) or edges (bond percolation) that are randomly missing. The probability p of a vertex or edge existing determines the amount of disorder present in the lattice. As the probability increases, we reach a point p_c where the structure changes from a set of smaller, unconnected pieces into one larger piece that is almost all connected. At probabilities $p \geqslant p_c$, there will, in general, be a path from one side of the lattice to the other. Note that this is only the case for structures with dimension two or more since it is 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, that is, $p_c = 1$. Figure 26 shows an example of a two-dimensional bond percolation lattice with various probabilities of an edge existing. There is clearly a path from one side of the lattice to the other for probabilities greater than or equal to the critical percolation probability $p_c = 0.5$. This particular percolation threshold only applies to bond percolation on a two-dimensional square lattice. Although site and bond percolation lattices exhibit similar behaviour, the critical percolation probabilities differ: for site percolation, $p_c = 0.5928...$. (Djordjevic 1982; Gebele 1984). For other lattices, the critical probabilities depend on their structure, and there are many efficient numerical methods for calculating them (Djordjevic 1982; Gebele 1984). We are only interested here in two- and three-dimensional lattices, and we summarise their critical percolation probabilities in Table 2.

It is fairly obvious that the properties of the lattice change significantly at the critical percolation threshold. For 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). Because of their transport properties, percolation lattices are

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

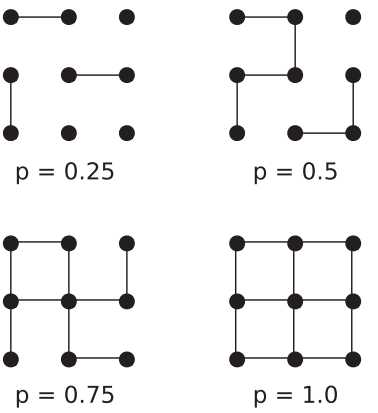


Fig. 26. An example of a two-dimensional bond percolation lattice with different levels of disorder determined by the probability of an edge existing. The critical percolation probability, $p_c = 0.5$, for bond percolation clearly shows a path from one side of the lattice to the other.

Table 2. *Summary of critical percolation probabilities for two and three-dimensional lattices*

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

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 Stauffer and Aharony (1992).

5.2. *Search algorithm on percolation lattices*

We use 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, 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 because the walker only has to coalesce on the marked state over a small piece of the lattice. In order to cope with this and obtain useful averages over many percolation lattices, we used the harmonic mean to average the time to find the marked state. If the algorithm failed, we set the reciprocal of

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

the time to find the marked state to zero. We also set the probability of the marked state to be zero if the algorithm failed. We note here that although the maximum probability of the marked state does not reach the same value as in the basic lattice, it does still vary periodically in the same way as in Figure 4

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 Kempe (2003b) 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

$$G^{perc}_{1,2,4} = \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} & 0 & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 1 & 0 \\ \frac{2}{3} & \frac{2}{3} & 0 & -\frac{1}{3} \end{pmatrix}, \tag{20}$$

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 or 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 before, and assign a binary number to each edge, with the value depending on whether an edge is present or not. Hence, for the example in Equation (20), we would get 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, but we will stick as closely as we can 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.

5.3. *Two-dimensional percolation lattices*

In this section we discuss our initial results for the quantum walk search algorithm on two-dimensional site percolation lattices. Figure 27 shows how the maximum probability of the marked state varies with both the size of the dataset and the percolation probability. As we might expect intuitively, as the percolation probability drops and the structure becomes less connected, the maximum probability of the marked state decreases.

Note that the scaling of the maximum probability initially maintains the logarithmic scaling of the basic two-dimensional lattice before eventually reverting to the scaling of the line, *viz.* $1/N$, at lower percolation probabilities. This change in scaling in the case of site percolation seems to occur at probabilities p below about 0.65, which is not significantly higher than the critical percolation threshold. This is as expected since at the critical threshold, the structure has, in general, a single path from one side to the other, so it is effectively a one-dimensional 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, there is a change in

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

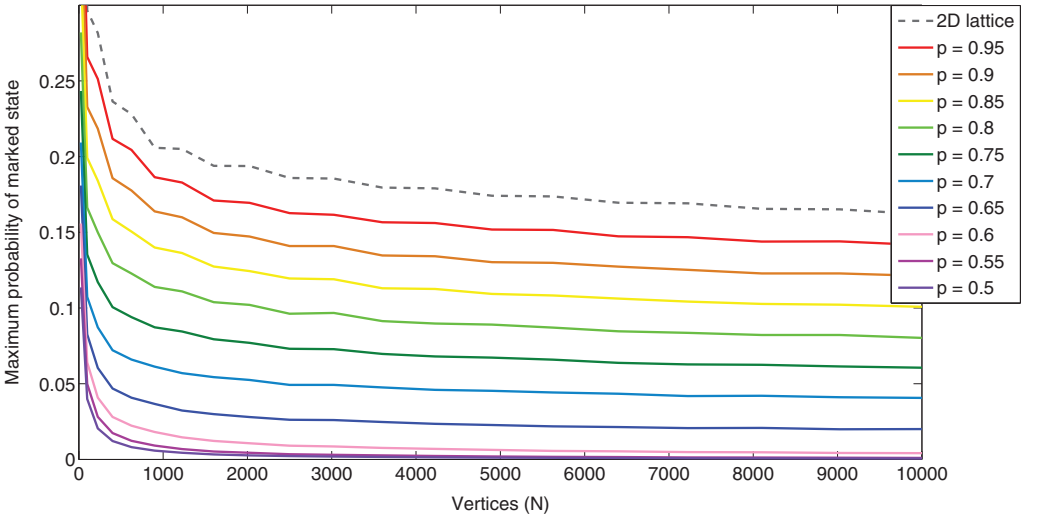


Fig. 27. Plot showing how the maximum probability of the marked state varies with the size of the dataset and percolation probability in two dimensions. We also include the same plot for a fully connected two-dimensional lattice (dashed line).

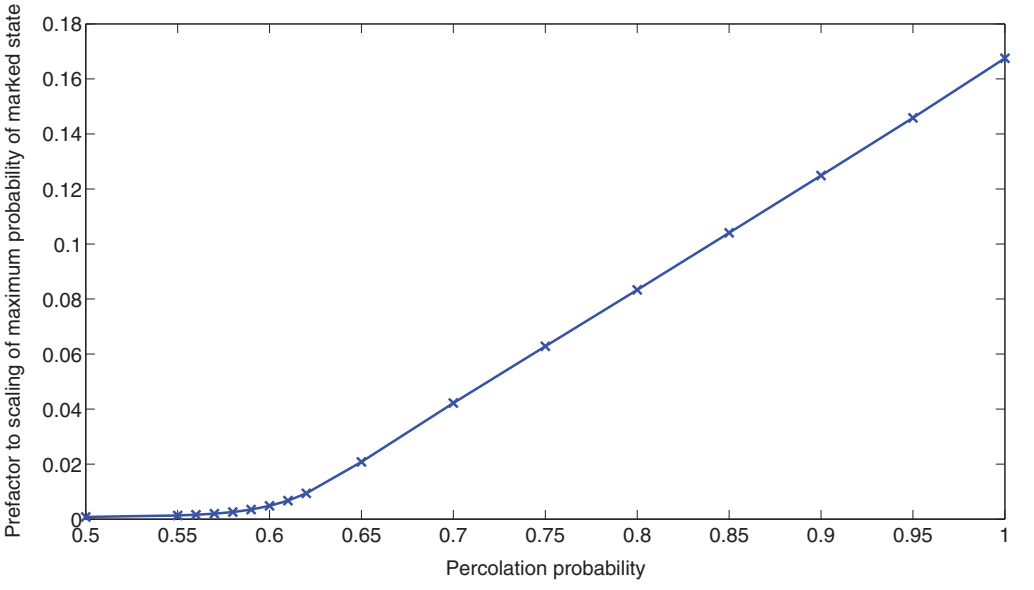


Fig. 28. Plot showing how the prefactor to the scaling of the maximum probability of the marked state, which was obtained from the data in Figure 27, varies with the size of the dataset and the percolation probability for site percolation in two dimensions. We find this prefactor by fitting the data in Figure 27 to the theoretical scaling of $1/\log N$ for each value of percolation probability. Only values of $N > 4{,}000$ were used for the fit.

the prefactor to the scaling of the maximum probability of the marked state. Figure 28 shows this prefactor using a logarithmic scale. Note that in order to extract the prefactor to the scaling here, we had to fit the data to the theoretical value of $1/\log N$. We only

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

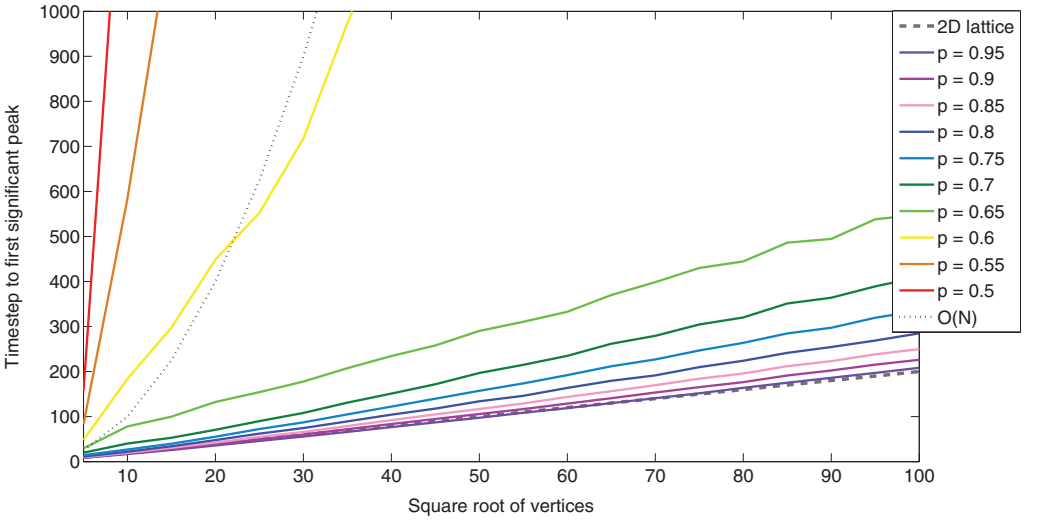


Fig. 29. Plot showing how the time to find the marked state varies with the size of the dataset and the percolation probability for site percolation lattices in two dimensions. We also include the same plot for a fully connected two-dimensional lattice (dashed line).

used values of $N > 4{,}000$ from the data in Figure 27 when extracting the prefactor for each value of percolation probability. 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. Note also that after investigation on a finer scale, 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 two-dimensional lattice to a classical linear scaling as p is reduced. Figure 29 shows the time to find the marked state for site percolation. When $p = 0.6$, the scaling of the time to find the marked state is very similar to the classical run time of $O(N)$. The kinks in this scaling (and the other percolation probabilities) are just from averaging over many percolation lattices. Given more time, a larger number could have been run, and thus a smoother scaling obtained. In addition, when the percolation probability is at or lower than the critical probability, the time to find the marked state quickly becomes extremely high (approaching infinity if the lattice is sufficently disconnected). Hence, only a small number of data points can be plotted in Figure 29, and this makes the plots look almost linear. 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 Leung *et al.* (2010), 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

$$T \propto N^{\alpha}, \tag{21}$$

where T is the time to find the marked state and N is the size of the dataset. We used the same analysis as in Leung *et al.* (2010) to establish how the scaling of the time to find

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

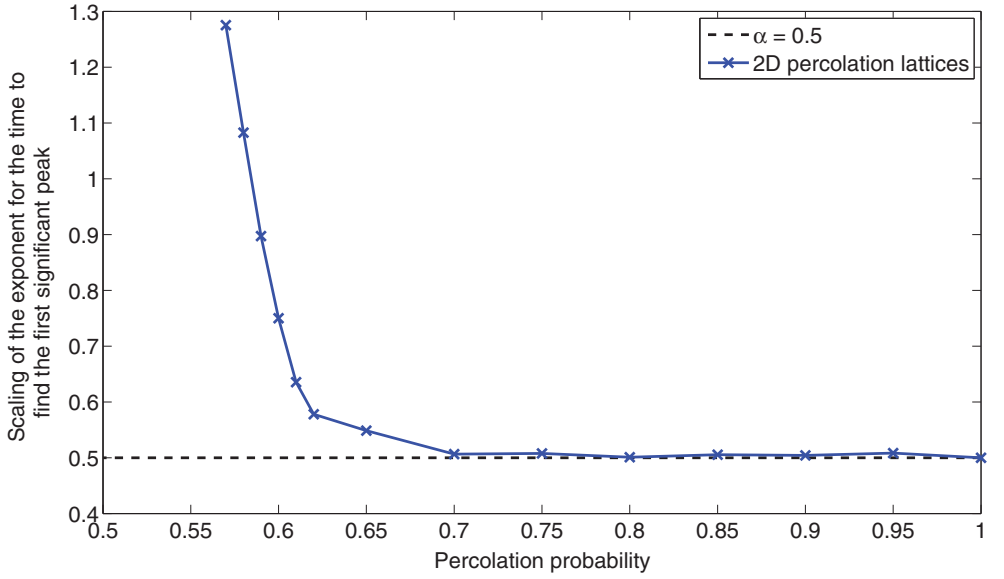


Fig. 30. Plot showing how the exponent α to the scaling of the time to find the maximum probability of the marked state, which was obtained from from the data in Figure 29, varies with the size of the dataset and the percolation probability for site percolation in two dimensions. The line $\alpha = 0.5$ is also included to indicate the lower bound of the algorithm (dashed line).

the marked state varies with the percolation probability. Figure 30 shows how the value of the coefficient α varies as the level of disorder is increased.

The plot shows that the quadratic speedup is maintained, that is, $\alpha \approx 0.5$, for percolation probabilities p greater than about 0.65. Below this probability, the quantum speed up disappears gradually and ends up at the classical run time when $p = p_c$. This is for the same reason as in the scaling of the maximum probability of the marked state: at the critical threshold, the structure is effectively a line, and below the critical threshold, the algorithm fails since the marked state is probably in a disconnected region. Note that the coefficient α is not exactly the 0.5 expected for a quadratic speed up. This is probably because percolation lattices are random in nature, and we only average over a finite number. If we averaged over more lattices, we would see a more constant scaling of the coefficient at $\alpha = 0.5$, that is, a full quadratic speed up.

5.4. *Three-dimensional percolation lattices*

We will now turn to three-dimensional site percolation lattices. We use the same analysis as in the two-dimensional case. Figure 31 shows how the maximum probability of the marked state varies as the percolation probability is decreased. As in the two-dimensional case, 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, $p_c = 0.3116....$ Figure 32 shows how the prefactor to this scaling of the maximum

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

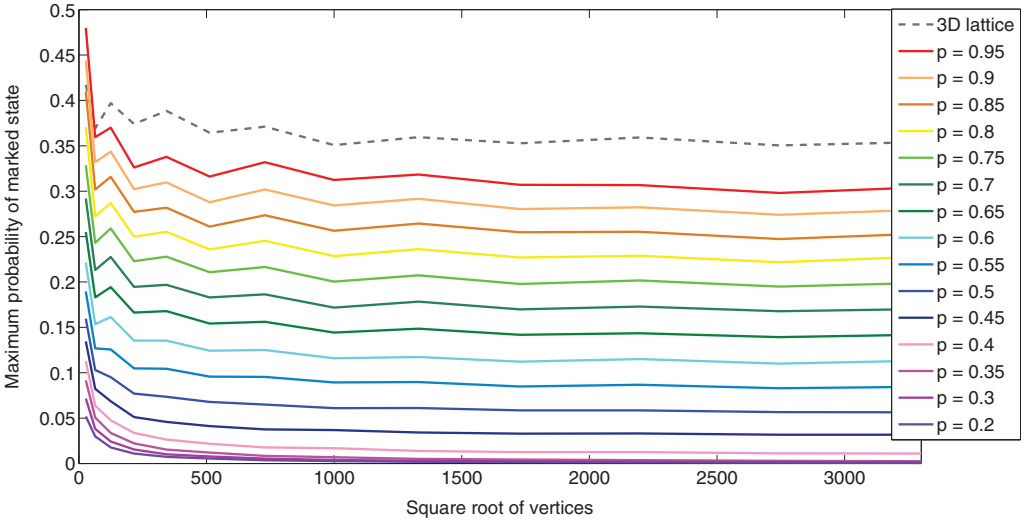


Fig. 31. Plot showing how the maximum probability of the marked state varies with the percolation probability in three dimensions. We also show the same plot for a fully connected three-dimensional lattice (dashed line).

probability varies with the percolation probability. In the same way as in the two-dimensional case, there is an almost linear scaling of the prefactor once the percolation probability has passed the critical threshold. The scaling here does not seem to be as close as in the two-dimensional case. This is probably because in the case of three-dimensional percolation lattices, many more combinations of lattice 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 for two-dimensional percolation lattices. Figure 33 shows how the time to find the marked state varies with the percolation probability. Figure 34 shows that, as in the two-dimensional case, the scaling coefficient α gradually changes from the quadratic speed up to the classical run time. Note again 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 $p = p_c$. As in the two-dimensional case, the coefficient is not exactly 0.5. This can be explained in the same way as for the two-dimensional percolation lattices, and averaging over more lattices should give a constant value for the coefficient α.

6. Discussion

In this paper we have discussed various factors that affect the efficiency of the quantum walk search algorithm. We introduced a simple form of tunnelling that allowed us to modify the substrate used as the database arrangement, and then used it to interpolate between structures with varying dimensionality and degree. We found that although the dependence on the spatial dimension of the underlying substrate is strong, it is not the only factor that affects the efficiency of the algorithm. We also found secondary dependencies

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

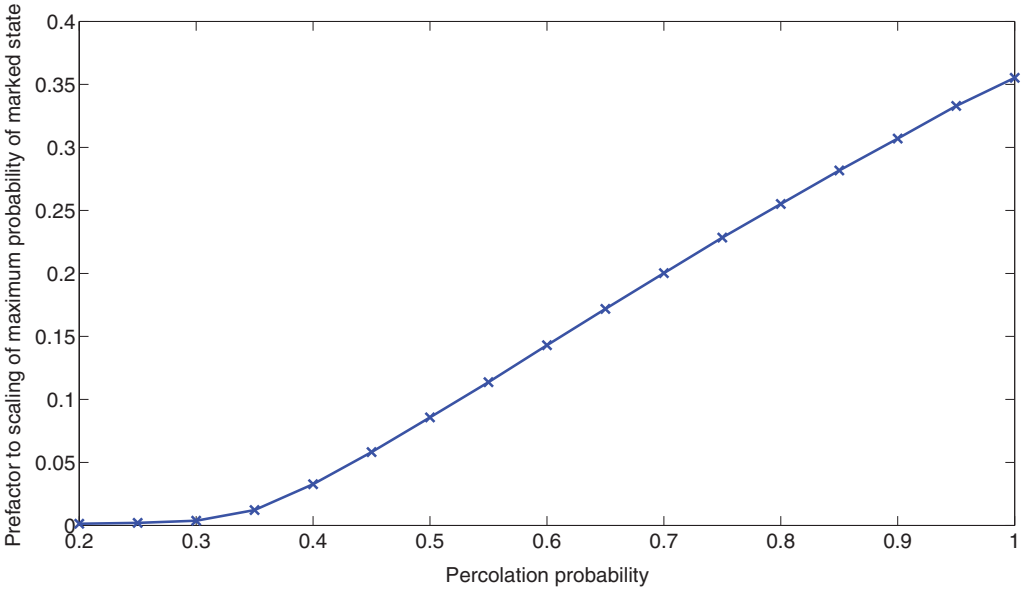


Fig. 32. Plot showing how the prefactor to the scaling of the maximum probability of the marked state, which was obtained from the data in Figure 31, varies with the size of the dataset and the percolation probability for site percolation in three dimensions.

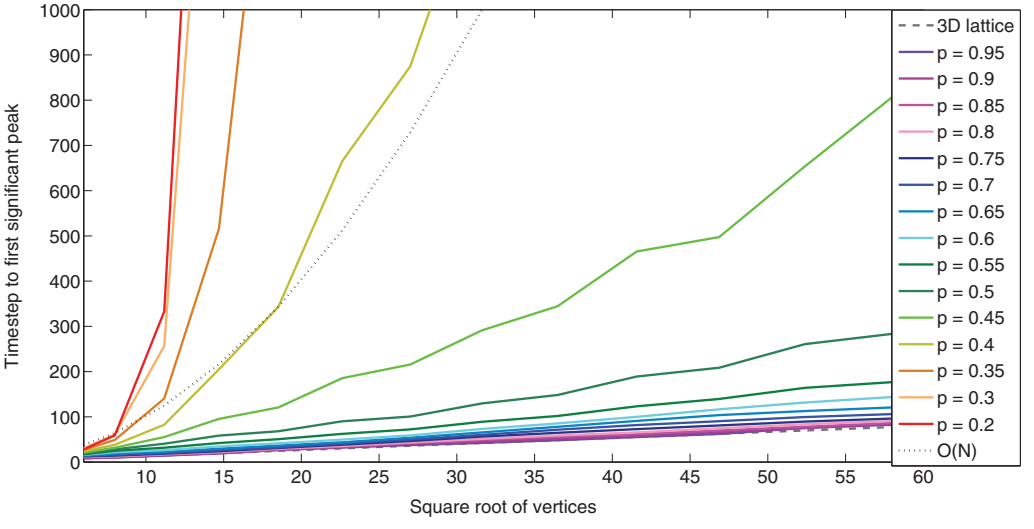


Fig. 33. Plot showing how the time to find the marked state varies with the size of the dataset and the percolation probability for site percolation lattices in three dimensions. We also show the same plot for a fully connected three-dimensional lattice (dashed line).

on the connectivity and symmetry of the structure. We then used percolation lattices to model disorder in the lattice in a simple way. In this case we found that the algorithm is able to maintain the quantum speed up even in the presence of non-trivial levels of disorder. In the following sections we will discuss our findings for each factor in turn.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

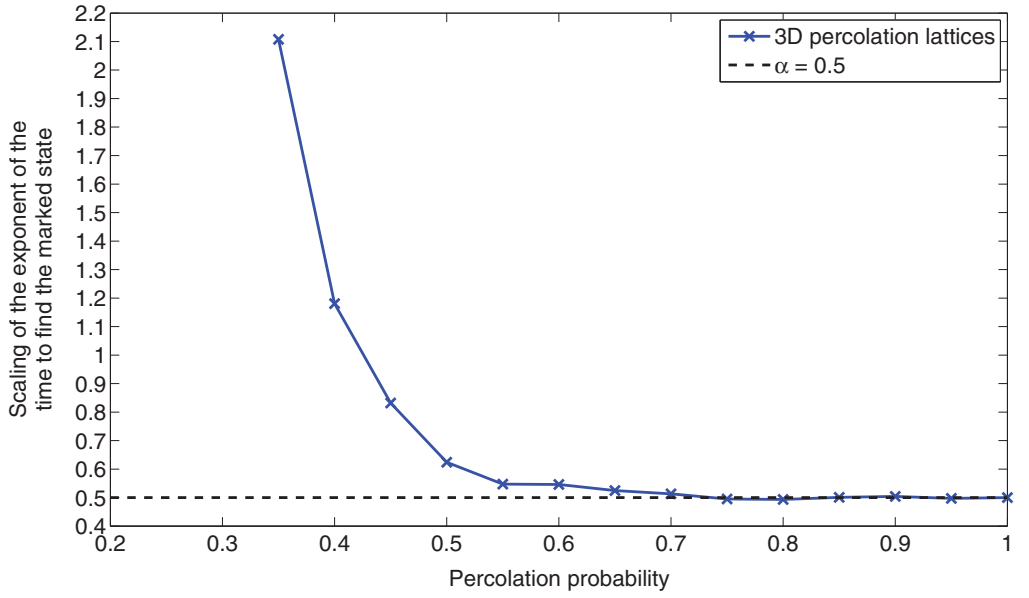


Fig. 34. Plot showing how the exponent α to the scaling of the time to find the maximum probability of the marked state, which was obtained from the data in Figure 33, varies with the size of the dataset and the percolation probability for site percolation in three dimensions. The line $\alpha = 0.5$ is also included to indicate the lower bound of the algorithm (dashed line).

6.1. *Dimensional dependence*

We have shown two different ways in which we can interpolate between structures of differing spatial dimension. First, we used a tunnelling operator to vary specific edges of a lattice, which enabled 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 is 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 found that the prefactor to the scaling of the probability varies with the strength of the tunnelling edges, increasing as the tunnelling strength increases. The basic scaling of the time to find the marked state is not affected by the change in dimensionality, but the prefactor to the scaling decreases as the tunnelling strength increases, and thus the algorithm becomes more efficient.

We also considered the case of lattices with varying height or depth, such as a three-dimensional lattice with fixed width and height but 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 two-dimensional Cartesian lattice. Suprisingly, in this case there is a gradual change in scaling in the maximum probability of the marked state. At low 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 depth increases to become equal to that of the other dimensions.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

6.2. *Connectivity*

We showed how the search algorithm is affected by varying connectivity in regular lattices. We used our simple model of tunnelling to interpolate between structures such as the square lattice ($d = 4$) and the triangular lattice ($d = 6$). With this model, we were 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, *viz.* $O(\sqrt{N})$, is not affected by the increase in connectivity, but the prefactor to this scaling decreases as the connectivity of the structure being searched increases. This is because of 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 found that the additional connectivity does not affect the basic scaling of $O(1/\log_2 N)$ in the two-dimensional case. It is only if we move to three spatial dimensions that the walker can find the marked state with a constant probability $O(1)$. However, in general, the prefactors to this scaling in both two and three dimensions increase as the connectivity of the structure increases. Again, this increases the efficiency of the algorithm since it may not have to be repeated as many times. We also found that the probability of the marked state does not increase uniformly with the additional connectivity. The prefactor in the scaling decreases and then recovers before increasing as the tunnelling strength increases. This is due to the dynamics of the quantum walk on a structure with some broken symmetry, that is, 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 since we found that the spread of the quantum walk also fell 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 was by no means exhaustive, and it would be interesting to investigate it further in the future.

6.3. *Substrate disorder*

We studied both two- and three-dimensional percolation lattices as a way to model disorder in the quantum walk search algorithm. We were interested in how the algorithm performs with increasing disorder and used percolation lattices as a random substrate for the database arrangement we wished to search.

We found that in both the two- and three-dimensional cases, 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 has dropped to the critical threshold, this scaling changes to that of the line, *viz.* $1/N$. This is as expected since at this point the structure is effectively a line. Also, the prefactor to the scaling of the maximum probability

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

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 found that as the disorder increases, the time to find the marked state also increases. Surprisingly, however, the quadratic speed up is maintained for a non-trivial level of disorder, before gradually reverting to the classical run time of $O(N)$ as the disorder reaches the critical percolation threshold. This seems to match the results of Leung *et al.* (2010), 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 results of Krovi and Brun (2006a, 2006b, 2007), which highlight the effect of localisation on the quantum walk when defects are introduced into the substrate.

Both of 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 because 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 further investigation of this result may be required.

Acknowledgments

We would like to thank Jiannis Pachos and Noah Linden for useful and interesting discussions.

References

Aaronson, S. and Ambainis, A. (2003) Quantum search of spatial regions. In: *Proceedings of the 44th annual IEEE symposium on foundations of computer science (FOCS)* 200–209.

Abal, G., Donangelo, R., Marquezino, F. L. and Portugal, R. (2010) Spatial search in a honeycomb network. *Mathematical Structures in Computer Science* **20** 999–1009.

Abal, G., Donangelo, R., Marquezino, F. L., Oliveira, A. C. and Portugal, R. (2009) Decoherence in Search Algorithms. In: *Proceedings of the 29th Brazilian computer society congress (SEMISH)* 293–306.

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

Ambainis, A. (2003) Quantum walks and their algorithmic applications. *International Journal of Quantum Information* **1** 507–518.

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

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

Ambainis, A., Backurs, A., Nahimovs, N., Ozols, R. and Rivosh, A. (2013) Theory of quantum computation, communication and cryptography. *Lecture Notes in Computer Science* **7582** 87–97.

Ambainis, A., Kempe, J. and Rivosh, A. (2005) Coins make quantum walks faster. In: *Proceedings of the 16th annual ACM–SIAM symposium on discrete algorithms (SODA)* 1099–1108.

Benioff, P. (2002) Space searches with a quantum robot. In: Quantum Computation and Information. *Contemporary Mathematics* **305** 1–12.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

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

Brassard, G., Høyer, P., Mosca, M. and Tapp, A. (2002) Quantum amplitude amplification and estimation. In: Quantum Computation and Information. *Contemporary Mathematics* **305** 53–74.

Buhrman, H. and Spalek, R. (2004) Quantum verification of matrix products. In: *Proceedings of the 17th annual ACM–SIAM symposium on discrete algorithms (SODA)* 880–889.

Childs, A. M. (2009) Universal computation by quantum walk. *Physical Review Letters* **102** 180501.

Childs, A. M. and Goldstone, J. (2004a) Spatial search by quantum walk. *Physical Review A* **70** 22314.

Childs, A. M. and Goldstone, J. (2004b) Spatial search and the Dirac equation. *Physical Review A* **70** 42312.

Djordjevic, Z. V., Stanley, H. E. and Margolina, A. (1982) Site percolation threshold for honeycomb and square lattices. *Journal of Physics A: Mathematical and General* **15** L405.

Farhi, E. and Gutmann, S. (1998) Quantum computation and decision trees. *Physical Review A* **58** 915–928.

Gebele, T. (1984) Site percolation threshold for square lattice. *Journal of Physics A: Mathematical and General* **17** L51.

Gottlieb, A. D. (2005) Convergence of continuous-time quantum walks on the line. *Physical Review E* **72** 47102.

Grimmett, G., Janson, S. and Scudo, P. F. (2004) Weak limits for quantum random walks. *Physical Review E* **69** 26119.

Grover, L. K. (1996) A fast quantum mechanical algorithm for database search. *Proceedings of the 28th annual ACM symposium on theory of computing (STOC)* 212–219.

Hein, B. and Tanner, G. (2009) Quantum search algorithms on the hypercube. *Journal of Physics A: Mathematical and Theoretical* **42** 085303.

Hein, B. and Tanner, G. (2010) Quantum search algorithms on a regular lattice. *Physical Review A* **82** 12326.

Inui, N., Konishi, Y. and Konno, N. (2004) Localization of two-dimensional quantum walks. *Physical Review A* **69** (5) 052323.

Keating, J. P., Linden, N., Matthews, J. C. F. and Winter, A. (2007) Localization and its consequences for quantum walk algorithms and quantum communication. *Physical Review A* **76** 12315.

Kempe, J. (2003a) Quantum random walks hit exponentially faster. In: Arora, S., Jansen, K., Rolim, J. D. P. and Sahai, A. (eds.) Approximation, Randomization, and Combinatorial Optimization – Proceedings of RANDOM 2003. *Springer-Verlag Lecture Notes in Computer Science* **2764** 354–369.

Kempe, J. (2003b) Quantum random walks: an introductory overview. *Contemporary Physics* **44** 307–327.

Kendon, V. (2003) Quantum walks on general graphs. *International Journal of Quantum Information* **4** 791–805.

Kendon, V. (2007) Decoherence in quantum walks – a review. *Mathematical Structures in Computer Science* **17** 1169–1220.

Kendon, V. and Tregenna, B. (2003) Decoherence can be useful in quantum walks. *Physical Review A* **67** 42315.

Kendon, V., Leung, G., Bailey, J. and Knott, P. (2011) Fractional scaling of quantum walks on percolation lattices. *Journal of Physics: Conference Series* **286** 012053.

Krovi, H. and Brun, T. A. (2006a) Hitting time for quantum walks on the hypercube. *Physical Review A* **73** 32341.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

Krovi, H. and Brun, T. A. (2006b) Quantum walks with infinite hitting times. *Physical Review A* **74** 42334.

Krovi, H. and Brun, T. A. (2007) Quantum walks on quotient graphs. *Physical Review A* **75** 62332.

Krovi, H., Magniez, F., Ozols, M. and Roland, J. (2010) Finding is as easy as detecting for quantum walks. *Automata, Languages and Programming* 540–551.

Leung, G., Knott, P., Bailey, J. and Kendon, V. (2010) Coined quantum walks on percolation graphs. *New Journal of Physics* **12** 123018.

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

Lovett, N., Everitt, M., Trevers, M., Mosby, D., Stockton, D. and Kendon, V. (2011) Spatial search using the discrete time quantum walk. *Natural Computing* **11** (1) 23–35.

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

Magniez, F. and Nayak, A. (2005) Quantum complexity of testing group commutativity. In: Caires, L., Italiano, G. F., Monteiro, L., Palamidessi, C. and Yung, M. (eds.) Automata, Languages and Programming – Proceedings 32nd International Colloquium, ICALP 2005. *Springer-Verlag Lecture Notes in Computer Science* **3580** 1312–1324.

Magniez, F., Nayak, A., Richter, P. C. and Santha, M. (2009) On the hitting times of quantum versus random walks. In: *Proceedings of the 20th annual ACM–SIAM symposium on discrete algorithms (SODA)* 86–95.

Magniez, F., Nayak, A., Roland, J. and Santha, M. (2007) Search via quantum walk. In: *Proceedings of the 39th annual ACM symposium on theory of computing (STOC)* 575–584.

Magniez, F., Santha, M. and Szegedy, M. (2005) Quantum algorithms for the triangle problem. In: *Proceedings of the 16th annual ACM–SIAM symposium on discrete algorithms (SODA)* 1109–1117.

Marquezino, F. L., Portugal, R. and Abal, G. (2010) Mixing Times in Quantum Walks on Two-Dimensional Grids. *Physical Review A* **82** 042341.

Moore, C. and Russell, A. (2002) Quantum walks on the hypercube. In: Rolim, J. D. P. and Vadhan, S. (eds.) Proceedings of 6th international workshop on randomization and approximation techniques in computer science (RANDOM 2002). *Springer-Verlag Lecture Notes in Computer Science* **2483** 164–178.

Nayak, A. and Vishwanath, A. (2000) Quantum walk on the line. DIMACS Technical Report 2000-43. Arxiv preprint arXiv:0010117.

Patel, A. and Rahaman, Md. A. (2010) Search on a Hypercubic Lattice through a Quantum Random Walk I: $d > 2$. *Physical Review A* **82** 032330.

Patel, A., Raghunathan, K. S. and Rahaman, Md. A. (2010) Search on a Hypercubic Lattice through a Quantum Random Walk II: $d = 2$. *Physical Review A* **82** 032331.

Potoček, V., Gábris, A., Kiss, T. and Jex, I. (2009) Optimized quantum random-walk search algorithms on the hypercube. *Physical Review A* **79** 12325.

Reitzner, D., Hillery, M., Feldman, E. and Bužek, V. (2009) Quantum searches on highly symmetric graphs. *Physical Review A* **79** 12323.

Santha, M. (2008) Quantum walk based search algorithms. In: Agrawal, M., Du, D., Duan, Z. and Li, A. (eds.) Theory and Applications of Models of Computation – Proceedings 5th International Conference, TAMC 2008. *Springer-Verlag Lecture Notes in Computer Science* **4978** 31–46.

Santos, R. A. M. and Portugal, R. (2010) Quantum Hitting Time on the Complete Graph. *International Journal of Quantum Information* **8** (5) 881–894.

Shenvi, N., Kempe, J. and Whaley, K. B. (2003) A quantum random-walk search algorithm. *Physical Review A* **67** 52307.

Stauffer, D. and Aharony, A. (1992) *Introduction to percolation theory*, Taylor and Francis.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press

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

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

Tulsi, A. (2008) Faster quantum-walk algorithm for the two-dimensional spatial search. *Physical Review A* **78** 12310.

Underwood, M. S. and Feder, D. L. (2010) Universal quantum computation by discontinuous quantum walk. *Physical Review A* **82** 042304.

Zalka, C. (1999) Grover's quantum searching algorithm is optimal. *Physical Review A* **60** 2746–2751.

https://doi.org/10.1017/S0960129518000051 Published online by Cambridge University Press