Spatial search using discrete quantum walks

Neil Lovett, Daniel Mosby, Daniel Stockton, and Viv Kendon

School of Physics and Astronomy, University of Leeds, Leeds, LS2 9JT, UK
pynbl@leeds.ac.uk, V.Kendon@leeds.ac.uk

Draft: May 15, 2009

Abstract. We study the quantum walk search algorithm of Shenvi, Kempe, and Whaley [PRA 67 052307 (2003)] on data structures of one to two spatial dimensions, on which the algorithm is less efficient than in three or more spatial dimensions. Our aim is to understand why the quantum algorithm is dimension dependent whereas the best classical algorithm is not, and to find in more detail how the efficiency of the quantum algorithm varies with spatial dimension or accessibility of the data.

1 Introduction

The promise of quantum computers to provide fundamentally faster computation is dependent both on being able to build a quantum computer of a useful size, and on finding algorithms it can run that are faster than any classical algorithm. Since some of the most efficient classical algorithms are based on random walks, a natural place to look for faster quantum algorithms is to see if there is a faster quantum version of a random walk. This approach was pioneered by Ambainis et al. [2] and Aharonov et al. [3], who proved that a discrete time quantum walk on the line or cycle spread or mixed, respectively, quadratically faster than a classical random walk. Continuous time quantum walks were introduced by Farhi and Gutmann [4] and provide the same algorithmic speed up as discrete time quantum walks. We concentrate on the discrete time walk in this paper, since it is more convenient for numerical calculations.

One of the first algorithmic applications of quantum walks was searching of unsorted databases. Shenvi et al. [1] proved a discrete quantum walk can repeat the performance of Grover's search algorithm [5] by finding a marked item among a total of N in $O(\sqrt{N})$ steps. The quantum walk search is now a standard tool in the development of quantum algorithms, for reviews see Ambainis [6] and Santha [7].

In this work, we are interested in the case where the data to be searched is also physically restricted in how it can be accessed. For example, to read a particular item from a magnetic tape, it is necessary to wind through the tape until the correct position is reached. Data stored on a hard disk is arranged in concentric rings on spinning disks: the heads move sideways across the rings and then the item can be read within the next revolution of the disk. This is quicker than reading data on a magnetic tape, but still requires a significant amount of time per item. A classical search of data stored on tape has to work through the tape from one end to the other, testing each item in turn to see if it is the required one. In fact, this is the optimal classical strategy for any arrangement of unsorted data on any storage medium. Finding a particular item requires in the worst case all the items to be checked, and on average half of the items will be examined before the marked one is located. This gives a classical running time of $O(N)$.

The quantum walk search algorithm of Shenvi et al. [1] arranged the data to be searched as the nodes of a graph on which the quantum walk then propagated. Specifically, they used a hypercube (of dimension $\lceil \log_2 N \rceil$), for which the quantum walk can be solved analytically [8]. They proved their quantum walk search can equal the quadratic speed up over classical searching given by Grover's search algorithm [5] – this is known to be optimal [9]. Improvements by Potocek et al. [10] bring the actual running time of the quantum walk search algorithm very close to the optimal one.

The hypercube is a highly connected structure, which doesn't correspond to physically realistic storage media. Motivated by this observation, study of lower dimensional search began with Benioff [11], who considered the additional cost of the time taken for a robot searcher moving between different spatially separated data items. Aaronson and Ambainis [12] then produced a quantum algorithm that finds a marked item in $O(\sqrt{N})$ for data arranged on lattices of dimension greater than two, and $O(\sqrt{N} \log^{3/2} N)$ for a square lattice (dimension $d = 2$). Work by Childs and Goldstone [13, 14] and Ambainis et al. [15] found quantum walk algorithms with the same running time for $d > 2$ and a small improvement to $O(\sqrt{N} \log N)$ for $d = 2$. Tulsi [16] recently improved this again to $O(\sqrt{N \log N})$. It remains an important open problem whether the running time for $d = 2$ can be improved to $O(\sqrt{N})$.

In this work, we are interested in what happens between $d = 1$, where simple arguments show that no speed up is expected, and $d = 3$, where the quantum speed up is known to be optimal. Classical search algorithms show no such dimensional dependence. We are also interested in how important the symmetry of the database arrangement is for searching using quantum walks. For a quantum walk crossing a hypercube, or the special "glued trees" graph of [17], it is known that defects in the graph can seriously disrupt the efficiency of the quantum walk [18–21].

We use numerical studies of quantum walks to illuminate their behaviour on lower dimensional structures, to gain insight into the issues involved. A one-dimensional quantum walk on a line or cycle corresponds to the case of unsorted data stored on a magnetic tape. We begin by introducing the quantum walk on the line in section 2, showing how the coin toss operation can be varied to indicate the marked state. We then consider the quantum walk search algorithm of Shenvi et al. [1] on a two-dimensional Cartesian lattice in section 3, where the search algorithm works. In section 4, we show that applying the same quantum walk search strategy to the walk on the line fails to find the marked state even in linear time. We discuss our results and plans for future work in section 5.

2 Quantum walk on the line

A discrete time quantum walk on the line is defined in direct analogy with a classical random walk: there is a walker carrying a coin which is tossed each time step and the walker steps left or right according to the heads or tails outcome of the coin toss. We denote the basis states for the quantum walk as an ordered pair of labels in a "ket" $|x, c\rangle$, where x is the position and $c \in \{0, 1\}$ is the state of the coin. A unitary coin operator is used at each time step and then a shift operation is applied to move the walker to its new positions. The simplest coin toss is the Hadamard operator H, defined by its action on the basis states as

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

and the shift operation S acts on the basis states thus

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

The first three steps of a quantum walk starting from the origin are as follows

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

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 eq. (3), which causes some probabilities to be amplified or decreased at each timestep. This leads to the different behaviour compared to its classical counterpart: the position of a walker following a classical random walk on a line spreads out in a binomial distribution about its starting point. Both classical and quantum distributions are illustrated after 100 steps in fig. 1. It is clear in fig. 1 that the quantum walk spreads faster. Ambainis et al. [2] proved that the quantum walk spreads in $O(T)$ compared to a classical random walk which spreads in $O(\sqrt{T})$.

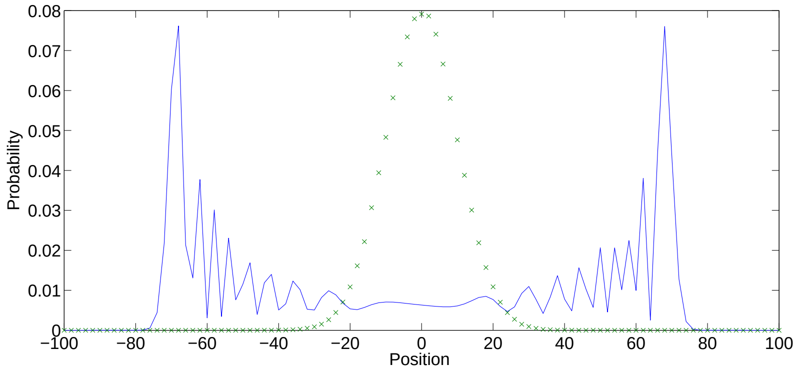

Fig. 1. Classical (crosses) and quantum (solid line) probability distributions for walks on a line after 100 timesteps. Only even positions are shown since odd positions are zero. Initial state for the quantum walk of $(|0\rangle + i|1\rangle)/\sqrt{2}$.

Adding some decoherence into this basic walk [18] actually helps for some applications. A nearly smooth "top hat" distribution can be obtained with a small amount of decoherence, which is useful for random sampling in Monte-Carlo simulations as an example. This also gives us a clue about how to use a quantum walk for searching. The "top-hat" is obtained after starting at a single location. Unitary dynamics are reversible, so if we run the quantum walk backwards from a uniform distribution on all points, we might expect it will go to being approximately located at a single point. Of course, it doesn't know which single point to converge on unless we mark it in some way: we do this with a different coin operator. If the coin operator allows the marked state to retain more probability than it passes on, this creates a bias in the walk at this point. This can be done with a biased Hadamard coin operator

$$H_\delta = \begin{pmatrix} \sqrt{\delta} & \sqrt{1-\delta} \\ \sqrt{1-\delta} & -\sqrt{\delta} \end{pmatrix}, \tag{4}$$

where $\sqrt{\cdot}$ is the positive root, and H_δ acts only on the coin state. The value of δ determines how much of the incoming probability is sent in each direction. Taking $\delta = 1$ gives the identity; the unbiased Hadamard operator eq. (1) corresponds to $\delta = 1/2$; and $\delta = 0$ gives the σ_z (spin flip) operator.

Since this search algorithm doesn't provide any speed up on the line, we next describe how it works on a two-dimensional Cartesian lattice. The behaviour of the quantum walk search on a line will then be compared in section 4.

3 Two-dimensional Cartesian lattice

A quantum walk in a higher dimension needs a larger coin, with one coin dimension for each choice of direction at the vertices. For the quantum walk search algorithm we need to use a symmetric coin operator based on Grover's diffusion operator,

$$G^{(D)} = \begin{pmatrix} \frac{2}{D} & \cdots & \frac{2}{D} \\ \vdots & \ddots & \vdots \\ \frac{2}{D} & \cdots & \frac{2}{D} \end{pmatrix} - I_n, \tag{5}$$

where I_n is the identity matrix and D is the size of the coin. In the case of a square lattice, $D = 4$ for the four choices of direction at each lattice site, and equation (5) reduces to

$$G^{(4)} = \frac{1}{2} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}. \tag{6}$$

We also need a different coin for the marked state. As we show later, it is optimal to invert the phase of the $G^{(4)}$ coin operator,

$$G^{(4)}_m = \frac{1}{2} \begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{pmatrix}. \tag{7}$$

To carry out the quantum walk search algorithm, we start with the quantum walker in an equal superposition of all the possible sites in the lattice, and the coin in an equal superposition of all directions,

$$|\psi(0)\rangle = \frac{1}{\sqrt{4N}} \sum_{x=1}^{N} \sum_{c=0}^{3} |x, c\rangle. \tag{8}$$

We use periodic boundary conditions at the edges of the $\sqrt{N} \times \sqrt{N}$ square lattice.

Figure 2 shows how the distribution of the walker evolves with time for a 20×20 lattice, i.e., $N = 400$. We are interested in how quickly the quantum walker finds the marked state: fig. 3 shows the probability of being at the marked state for each timestep as the walk proceeds. It has periodic behaviour with the first peak occuring at roughly $t = (\pi/2)\sqrt{N} \simeq 32$. The maximum probability for $N = 400$ is around 0.23. This can be increased as close to 1 as desired by standard amplifcation techniques (repeating the search a few times). Subsequent peaks occur somewhat later than $t = 2(\pi/2)\sqrt{N}$ etc., but we are interested in the oscillatory behaviour not for the precise timings of the peaks, but rather for the necessity of knowing when the first peak occurs in order to measure the walker's position at the optimum time. In fact, as can be seen in fig. 3, the peaks are quite broad, so even if a 50% error occurs in when to measure it only means a 50% lower probability of finding the marked state, this is only a constant extra overhead on the amplification. The maximum probability also varies with the size of the data set, the theoretical value of $O(1/\log N)$ from Ambainis et al. [15] is born out in our results in fig. 4.

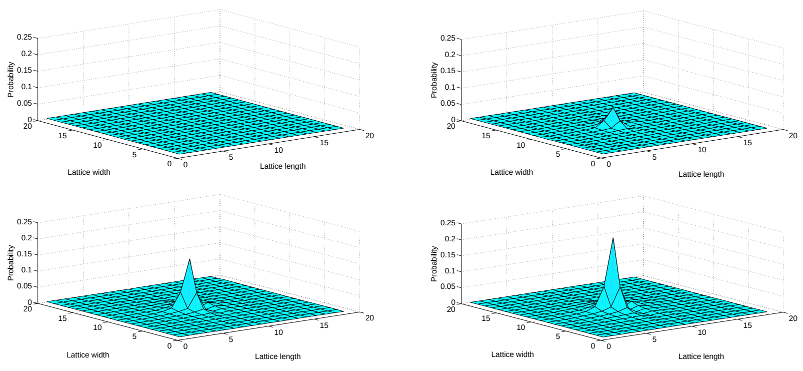

Fig. 2. Probability distribution of a discrete time quantum walk search on 400 vertices arranged in a 20×20 square with periodic boundary conditions, evolved for 0, 10, 20 and 32 timesteps. The marked vertex is at position 190.

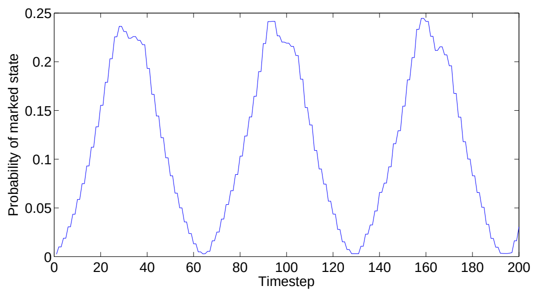

Fig. 3. Probability of marked state over 200 timesteps on a 20×20 grid with periodic boundary conditions. The marked vertex is at position 190.

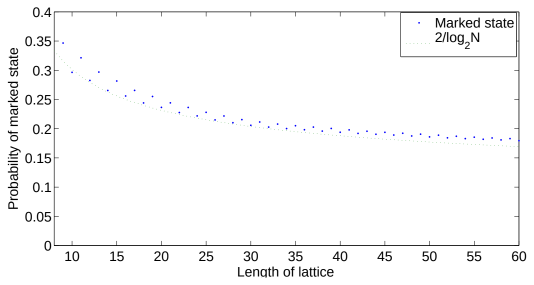

Fig. 4. Maximum of the first peak in the probability of being at the marked state for different sized data sets, using the optimal marked state coin in eq. (7) on a 2D lattice of size $\sqrt{N} \times \sqrt{N}$, plotted against $\sqrt{N}$ (blue circles). Also shown dotted $2/(\log_2 N)$ for comparison.

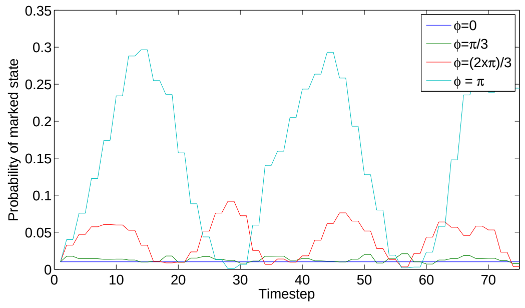

Fig. 5. Probability of marked state over 75 timesteps for $N = 100$ i.e., a 10×10 grid with marked state at 45, using the marked state coin in eq. (9) with $\phi = 0$, $\pi/3$, $2\pi/3$ and π.

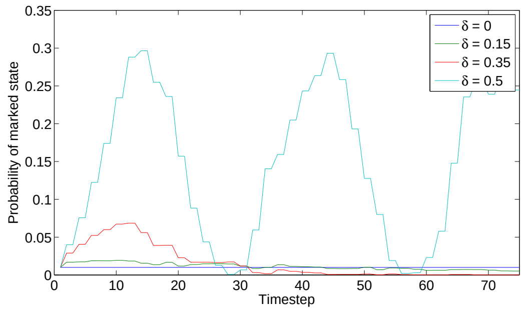

Fig. 6. Probability of marked state over 75 timesteps for $N = 100$ i.e., a 10×10 grid with marked state at 45, using the marked state coin in eq. (10) with $\delta = 0$, 0.15, 0.35 and 0.5.

To explore how the coin affects the search result, we can introduce a phase into the coin operator,

$$G^{(4)}_{\phi} = e^{i\phi} G^{(4)}, \tag{9}$$

where $0 \leq \phi \leq \pi$, or a bias,

$$G^{(4)}_{\delta} = \begin{pmatrix} 1-\delta & -\delta & -\delta & -\delta \\ -\delta & 1-\delta & -\delta & -\delta \\ -\delta & -\delta & 1-\delta & -\delta \\ -\delta & -\delta & -\delta & 1-\delta \end{pmatrix}, \tag{10}$$

where $0 \leq \delta \leq 0.5$. The standard $G^{(4)}$ coin operator corresponds to $\phi = 0$, and the marked coin operator used before, $G^{(4)}_m$, eq. (7), corresponds to $\phi = \pi$. Taking $\delta = 0$ makes the marked state coin into the identity operator, while $\delta = 0.5$ corresponds to $G^{(4)}_m$. (For $\delta > 0.5$, the operator would no longer be unitary.) These variations have been chosen to preserve the symmetry of the coin operator. Figures 5 and 6 shows the effect of varying the phase ϕ and the

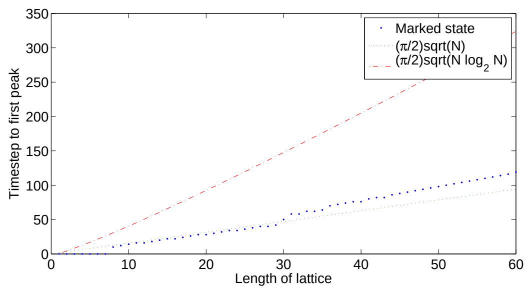

Fig. 7. Time step at which the first peak in the probability of being at the marked state occurs for different sized data sets, using the optimal marked state coin in eq. (7) plotted against $\sqrt{N}$. Also shown $(\pi/2)\sqrt{N}$ and $(\pi/2)\sqrt{N(\log_2 N)}$ for comparison.

bias δ. These figures show the largest probability of finding the marked state is for $\phi = \pi$ and $\delta = 0.5$, justifying our choice of the optimal marked state coin operator.

The key open question is the scaling with N for the 2D lattice. Our numerical evidence is intriguing, see fig. 7. The closest fit for the data up to $N = 32^2$ is $(\pi/2)\sqrt{N}$, but there is a jig near some powers of two that pushes it above this line by $N = 32^2$. At these low numbers, nothing firm can be said, but since $(\pi/2)\sqrt{N \log_2 N}$ is much higher, the deviation from $(\pi/2)\sqrt{N}$ may be smaller, say $\log\log N$.

4 Quantum walk search on the line

We now examine how the quantum walk search algorithm behaves for data arranged on a line. We consider both a line segment, and a loop (cycle) where periodic boundary conditions are applied at the ends. The loop is less physical (for most tape storage, the ends are far apart from each other), but allows us to investigate the behaviour of the algorithm without the edge effects the ends introduce. For the line segment, we use a reflecting boundary condition. This means we have to use a different coin at the edges, which has the effect of introducing two spurious marked states.

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

$$H_\delta^{(\text{sym})} = \begin{pmatrix} \sqrt{\delta} & i\sqrt{1-\delta} \\ i\sqrt{1-\delta} & \sqrt{\delta} \end{pmatrix}. \tag{11}$$

For $\delta = 0.5$, this reduces to

$$H^{(\text{sym})} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}. \tag{12}$$

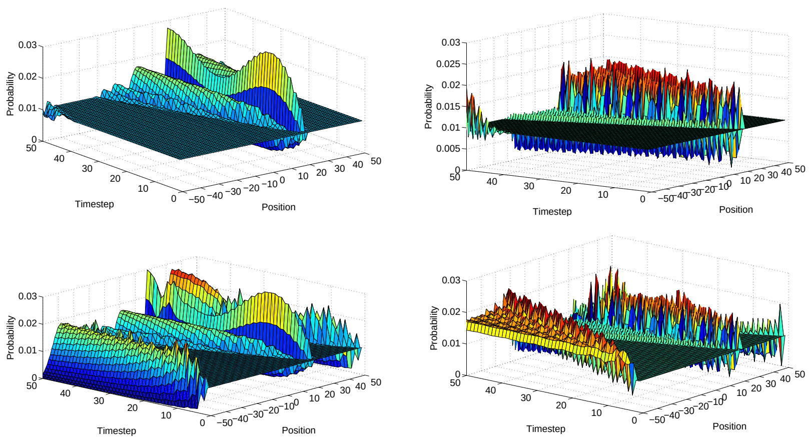

Fig. 8. Probability distribution of the quantum walk search of $N = 101$ items arranged on a line, after 50 time steps with marked state at position 20, for a symmetric coin and periodic boundary conditions (top left), Hadamard and periodic (top right), symmetric and reflecting (bottom left), and Hadamard and reflecting (bottom right).

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

$$|\phi(0)\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N} \frac{1}{\sqrt{2}} (|x, 0\rangle + i|x, 1\rangle), \tag{13}$$

and for the symmetric coin operator we use

$$|\phi(0)\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N} \frac{1}{\sqrt{2}} (|x, 0\rangle + |x, 1\rangle). \tag{14}$$

First we contrast the symmetric coin operator, eq. (12) with the standard Hadamard operator, eq. (1), fixing $\delta = 0$ for the marked state coin, giving a σ_z. In fig. 8 we see the differences between the two different boundary conditions (periodic and reflecting), each with the two different coin operators. We find that the symmetric coin operator gives a more smoothly varying probability distribution about the marked state and we can see oscillations in the probability of finding the marked state. Superficially, these results look similar to the square lattice. However, the square lattice has a peak probability at the marked state of around 0.3 for a 10×10 square lattice with $N = 100$. On a line with $N = 101$, the peak in the probability is only around 0.028, This is not significantly larger than the uniform distribution, which has a probability of 0.01 for any site. The Hadamard coin operator varies over a period of around seven time steps, regardless of the other parameters, and shows slightly higher

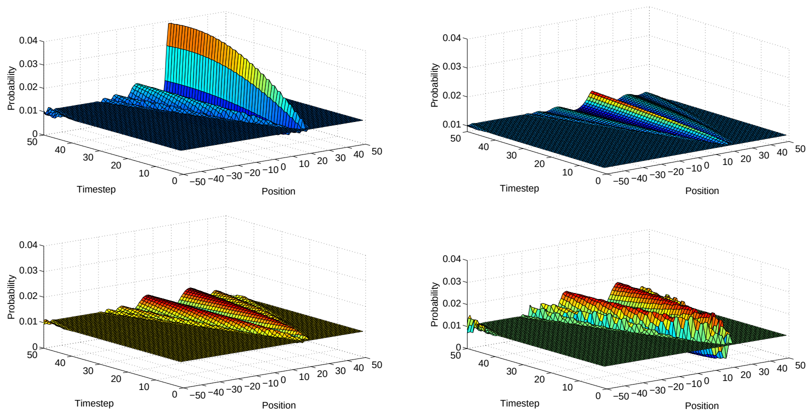

Fig. 9. Probability distribution of the quantum walk search of $N = 101$ items arranged on a line, after 50 time steps with marked state at position 20, for a symmetric coin and periodic boundary conditions with the marked coin operator given by eq. (11) with $\delta = 0.15$ (top left), $\delta = 0.45$ (top right), $\delta = 0.65$, (bottom left) and $\delta = 1$ (bottom right).

probability spreading out from the marked state in two soliton-like waves. This supports the case for symmetry being important for quantum walk searching. Reflecting boundary conditions also produce spreading soliton-like waves from the boundaries for both symmetric and Hadamard coins.

Concentrating on the case of a symmetric coin operator and periodic boundary conditions, we now consider what happens when δ is varied. For δ increasing from zero to a half, the period of the oscillations increases towards infinity as $\delta \to 0.5$, the value for which the marked coin operator becomes the same as the unmarked coin operator and the distribution remains uniform. Increasing δ above 0.5, we find that instead of finding the marked state, in effect it "un-finds" it with the probability of being in the marked state decreasing below the uniform distribution. This is due to the fact the coin is now biased in the wrong direction and so is giving away more and more probability instead of retaining it. Figure. 9 shows the variation in probability distribution with δ, compare with $\delta = 0$ for fig. 8 (top left). For the symmetric quantum walk with periodic boundary conditions, evolving for longer times with $\delta \simeq 0.45$ eventually results in a peak in the probability of the marked state approaching $2\pi/N$, but only after around $5N$ or more time steps, see fig. 10. This is obviously not useful, either in the size of the peak probability, which we would like to scale with $\log N$ as it does for the search in two dimensions, nor with the number of time steps, which far exceeds the classical worst case of N.

The quantum walk search algorithm used for data on a line is thus completely ineffective for the parameters we have considered: it does not find the marked state with significant probability even when run for as long as the worst case classical time of N steps for N items. Of course, we can easily specify a quantum

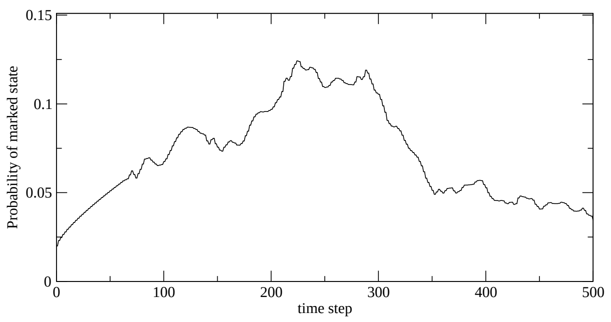

Fig. 10. Probability of the marked state for each time step for a line of $N = 50$ sites with periodic boundary conditions and $\delta = 0.45$, run for 500 time steps.

version of the classical algorithm that does find the marked state in N steps. Start in the state $|0, 1\rangle$ and use the identity as the coin operator everywhere except the marked state. This causes the walk to hop deterministically along the line. At the marked state, use σ_z for the coin operator. This will flip the coin from $|1\rangle$ to $|0\rangle$ and thus reverse the direction of the walker. If the position of the walker is measured after N steps, the current location allows you to work out where it turned round, and thus locate the marked state. This method uses only a single measurement. If you allow measurements at every step, then of course you can immediately find out if the walker has arrived at the marked state by testing the state of the coin.

5 Discussion and further work

We have investigated in some detail how the quantum walk search algorithm of Shenvi et al. [1] behaves on a two dimensional lattice, where it finds the marked state efficiently, and on the line, where it does not. Our numerical results suggest the $\sqrt{\log N}$ factor in the running time of the algorithm on the square lattice is not tight, we observe close to a full quadratic speed up, with an intriguing jig in the data that may be something more like a $\log \log N$ factor.

Future work will incude studying the quantum walk algorithm on structures of intermediate dimension between the line and the square lattice. We will start with a honeycomb lattice, which is still planar, but each lattice point has three directions rather than four. Another regular graph with three edges at each lattice point is the Bethe lattice. Since this has no loops, a quantum walk on a Bethe lattice should behave quite differently to on a honyecomb lattice. Intermediate structures with loops of different sizes can be studied to interpolate between the two cases.

The regularity and connectivity of the square lattice can also be varied by studying two dimensional percolation lattices. In these, any particular edge is present only with probability p. By varying p, we can see at what level of connectivity in the lattice the quantum walk search algorithm becomes inefficient.

A structure analogous to a hard drive can be constructed by connecting a concentric set of cycles by a line of edges. Searching the cycle cannot be done efficiently by a quantum walk, but the addition of one or more lines of connecting edges between sets of cycles may allow for some quantum speed up.

Acknowledgments: NL is funded by the Engineering and Physical Sciences Research Council. VK is funded by a Royal Society University Research Fellowship.

Bibliography

[1] Neil Shenvi, Julia Kempe, and K Birgitta Whaley. A quantum random walk search algorithm. *Phys. Rev. A*, 67:052307, 2003.

[2] Andris Ambainis, Eric Bach, Ashwin Nayak, Ashvin Vishwanath, and John Watrous. One-dimensional quantum walks. In *Proc. 33rd Annual ACM STOC*, pages 60–69. ACM, NY, 2001.

[3] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani. Quantum walks on graphs. In *Proc. 33rd Annual ACM STOC*, pages 50–59. ACM, NY, 2001.

[4] E. Farhi and S. Gutmann. Quantum computation and decison trees. *Phys. Rev. A*, 58:915–928, 1998.

[5] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28th Annual ACM STOC*, page 212. ACM, NY, 1996.

[6] Andris Ambainis. Quantum walks and their algorithmic applications. *Intl. J. Quantum Information*, 1(4):507–518, 2003.

[7] Miklos Santha. Quantum walk based search algorithms. In *5th Theory and Applications of Models of Computation (TAMC08), Xian, April 2008*, volume 4978, pages 31–46. LNCS, 2008.

[8] Christopher Moore and Alexander Russell. Quantum walks on the hypercube. In J. D. P. Rolim and S. Vadhan, editors, *Proc. 6th Intl. Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM '02)*, pages 164–178. Springer, 2002.

[9] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5): 151–152, 1997.

[10] V Potocek, A Gabris, T Kiss, and I Jex. Optimized quantum random-walk search algorithms. *Phys. Rev. A*, 79:012325, 2009.

[11] P. Benioff. Space searches with a quantum robot. *AMS Contempory Maths Series volume, Quantum Computation & Information*, 305, 2002.

[12] S. Aaronson and A. Ambainis. Quantum search of spatial regions. In *Proc. 44th FOCS*, page 200. IEEE, Los Alamitos, CA, 2003.

[13] Andrew Childs and Jeffrey Goldstone. Spatial search by quantum walk. *Phys. Rev. A*, 70:022314, 2004.

[14] Andrew M. Childs and Jeffrey Goldstone. Spatial search and the Dirac equation. *Phys. Rev. A*, 70:042312, 2004.

[15] A. Ambainis, J. Kempe, and A. Rivosh. Coins make quantum walks faster. In *Proc. 16th ACM-SIAM SODA*, pages 1099–1108. ACM, NY, 2005.

[16] Avatar Tulsi. Faster quantum walk algorithm for the two dimensional spatial search. *Phys. Rev. A*, 2008. To appear.

[17] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proc. 35th Annual ACM STOC*, pages 59–68. ACM, NY, 2003.

[18] Viv Kendon and Ben Tregenna. Decoherence can be useful in quantum walks. *Phys. Rev. A*, 67:042315, 2003.
[19] Hari Krovi and Todd A. Brun. Hitting time for quantum walks on the hypercube. *Phys. Rev. A*, 73(3):032341, 2006.
[20] Hari Krovi and Todd A. Brun. Quantum walks with infinite hitting times. *Phys. Rev. A*, 74(4):042334, 2006.
[21] Hari Krovi and Todd A. Brun. Quantum walks on quotient graphs, 2007.