Discrete Quantum Walks and Search Algorithms

J.C. Adcock

10/4/15

Abstract

Recently a number of discrete quantum walk based algorithms have been produced [1–6]. These are closely related to, or are specific cases of, structured search applied using a quantum walk. The formalism of discrete quantum walks has clear classical parallel in the form of Markov chains. Similarly, search via quantum walk has similar parallels in classical structured search. This report will review the field of search via discrete quantum walk and give a technical overview of the mathematical methods and structure.

1 Introduction

Search algorithms are of fundamental interest to computer science and have been studied thoroughly in the classical domain. Generally speaking, search algorithms aim to solve the problem of identifying a single "marked" element x^* (or subset of elements $x^* \in \mathcal{M}$) in some set of $x \in \mathcal{X}$. Using naive random sampling, one expects to find such an element in $O(N/M)$ queries, where $||\mathcal{X}|| = N$ and $||\mathcal{M}|| = M$. However, Grover's seminal 1996 paper "A fast quantum mechanical algorithm for database search", showed that an unstructured search for a single marked item could be completed in $O(\sqrt{N})$ queries to an oracle function, a square root speed up compared to classical approaches. This opened to the door to a new area of study and approach to search problems and subsequent work has lead to numerous developments such as a more general quantum search [7, 8].

This report aims to introduce discrete quantum walk based quantum searches, whilst reviewing in detail a number of current results and identify regions of interest for further study and research. This section is split in to two parts. Section 1.1 will briefly review the current literature give an overall feel for the research landscape, whilst section 1.2 will give an introduction to Markov chains (which provide the basis for our quantum walks) and classical search algorithms. Discrete time quantum walks will be discussed in Section 2 and searching via discrete quantum walk will be discussed in Section 3. A summary of main results and concluding remarks will then be made in Section 4.

1.1 Brief Historical Reivew

The first hint that quantum walks on graphs may produce effective search algorithms came from the discovery that discrete quantum walks on finite graphs (first defined in 2000 [9]) can have exponentially faster hitting times (time taken to get from one node to another node). This was shown first for continuous time decision trees [10], then for discrete walks on graphs such as the hypercube, and 'glued tree' graphs [11–13]. Naturally this seems to imply that searching using a quantum walk would be a fruitful avenue to pursue. This was first realised in context of the element distinctness problem (which can be reduced to search) by Ambainis [1], who found an optimal [14] $O(N^{\frac{2}{3}})$ algorithm for the problem, (improving the best quantum algorithm by a fractional power) by implementing quantum walks on Johnson graphs. This work utilised a type of search via quantum walk to produce it's result, and opened up enquiry in to search via quantum walk.

After this seminal work, Childs [15] and Shenvi [16], arrived at discrete quantum walk search algorithms for the case of the d-dimensional lattice (complexity $O(\sqrt{N}Poly(Log(N)))$) and hypercubes (complexity $O(\sqrt{N})$) respectively. Though these have larger complexities than Grover's algorithm, these are of use if a cost is associated in grovers algorithm of moving between sites where data is stored, which is likely in a physical implementation. The quantum walk formalism subverts this, efficiently traveling around the data structure. Meanwhile, the field of quantum walks on graphs was continually explored with results such as [17,18] exploring the effect of coins and initial states.

A major breakthrough was made in 2004 when Szegedy [2,19] prescribed a way to generate quantum walks using the classical Markov chains they correspond to directly. Szegedy prescribed a method to produce these walks provided the Markov chain has a symmetric, ergodic, reversible transition matrix P (these terms are definied in section 2). This approach generalises Grover search as well as Ambainis' previous results [1,18] to a larger space of quantum walks. Due to that fact that its construction is based around classical random walks, it was possible for Szedegy to use properties of the classical Markov process, such as the eigenvalue gap, to analyse the complexity of the algorithm. With this, Szegedy presented an algorithm for the related problem of detecting if the set of marked elements contains members. This runs with complexity $O(\phi_0 + \frac{1}{\sqrt{\delta\epsilon}}(\phi_1 + \phi_2))$. Here, ϕ_0 is the cost of sampling at random from the data, ϕ_1 is the cost of propagating the quantum Markov process once, ϕ_2 is the cost of checking that the current element is marked. The number of marked elements is $M = \epsilon N$ and δ is the eigenvalue gap of the Markov process' transition matrix P. There is a similar classical algorithms for implementing a search of this type is of complexity $O(\phi_0 + \frac{1}{\delta\epsilon}(\phi_1 + \phi_2))$, which is essentially given by Algorithm 1 with $n = 1$. In this paper Szegedy also found the important result that hitting times on quantum walks are quadratically faster than their classical counterparts. These facts give hope for a quantum search (not just detection) with complexity $O(\phi_0 + \frac{1}{\sqrt{\delta\epsilon}}(\phi_1 + \phi_2))$.

Ambainis' original search algorithm on Johnson graphs has a minimum complexity of $O(\phi_0 + \frac{1}{\sqrt{\epsilon}}(\frac{\phi_1}{\sqrt{\delta}} + \phi_2))$, which is comparable to classical Algorithm 1 with $n = \frac{1}{\delta}$ with a square root speed up in ϵ and δ (though similarly it is possible to "set $n = 1$" in this algorithm, which achieves the same complexity as Szegedy's detection algorithm. It should be noted that Ambainis' algorithm reduces to Grovers for the complete graph (which exists in the set of Johnson graphs). This complexity is slightly better that that of Szegedy's detection algorithm, however the restriction to Johnson graphs reduces the utility of this approach to less general search and applications, for example group commutativity can not be tackled using only Johnson graphs [20]. Moreover, the algorithm will only find a single marked element. These facts hint that better complexities for search may be attainable on more general graphs.

Magniez, Nayak, Roland and Santha (MNRS) [3] improved upon this work by producing a scheme where a Szegedy style Markov chain quantum walk is used to produce a reflection operator for an amplitude amplification scheme. This combines elements of Ambainis' previous work with that Grover's search, providing a more general graph based search. The MNRS algorithm is valid for a larger class of Markov chains (those which are reversible and ergodic) and improves has an improved complexity of $O(\phi_0 + \frac{1}{\sqrt{\epsilon}}(\frac{\phi_1}{\sqrt{\delta}} + \phi_2))$ for the problem of finding a single marked element. This type of search is equivalent to a quantum version of Algorithm 1 with $n = \frac{1}{\delta}$. This classical version has complexity $O(\phi_0 + \frac{1}{\epsilon}(\frac{\phi_1}{\delta} + \phi_2))$.

Recently, Krovi et. al. have discovered an algorithm that extends the MNRS algorithm to the case of multiple marked elements, again encompassing a larger class of Markov chains. The only condition they require is that the Markov chain be reversible. They also show that the problem of detecting marked vertices is equally as hard as searching (and finding) them for quantum walk search algorithms.

1.2 Searching on a graph

It is easy to imagine a data set which has underlying structure such that random search has more complexity. For example if it is known that x^* come from a known process or distribution, then an algorithm could become more efficient by using this information in the algorithm. One way of doing this would be to simply alter the probability distribution used to randomly sample the elements. Another common approach is to represent the data structure as a weighted graph (or Markov chain), where a data point is chosen according to some distribution and the Markov process iterated until a marked element is found, or for a set number of steps until a new starting point is chosen. Algorithm 1, given below, is an example of a general graph based search for some Markov process P corresponding to the graph $G(E, V)$.

Clearly properties of the algorithm depend on the values of the parameters t, n and the probability distribution used to sample x. These will effect the likelihood that a marked element is found an also the overall efficiency of the algorithm. For example, a large value of t is needed in order for the algorithm to converge, however if t is too large it will cause a large delay if there exists no marked element in the set $\mathcal{M}$.

So far we have talked about searching using a Markov process P, but can we use the properties of the Markov chain P to infer properties of the search algorithm? Some definitions and results will help us here.

Definition 1. *An n-state Markov chain is a description of a classical probabilistic process with no memory. It can be represented by an $n \times n$ square matrix P such that the rows sum to 1 (P is stochastic) and the components p_{xy} corresponds to the probability of transition from state x to state y.*

Algorithm 1: Search on a Graph for one marked element using Markov Process

```
Initialise x ∈ 𝒳 with some random distribution
for i = 1 to t do
    if x ∈ ℳ then
        Output x, stop
    else
        Iterate Markov Process n times: x = xP^n
    end
end
Output 'No marked element found'
```

Clearly P must be connected if we wish to find all possible marked elements. Another obviously desirable property is that of irreducibility - that every state can be reached from every other state. Less obviously, any irreducible chain has a stationary vector state π such that $\pi P = \pi$, where π is a distribution over the states $x \in \mathcal{X}$ with magnitude 1. This is a result of the Perron-Frobenius theorem [21]. Other properties that will be of use when examining the discrete quantum walk construction of Ambainis, Szegedy and others [1,2] include ergodicity - a Markov chain is ergodic if it is irreducible and also aperiodic - and reversibility - by defining $\pi_x p_{xy} = \pi_x(p^*_{yx})$ where $(p^*_{yx}) = P^*$ is the reverse of P, we say if P is reversible if $P = P^*$. Clearly if $P = P^T$, then $P^* = P$. Yet another important concept is that of an eigenvalue gap. if a Markov chain P is ergodic, then it has a single eigenvalue with magnitude 1. The eigenvalue gap δ_P of P is then the difference the between the largest and second largest eigenvalues.

This eigenvalue gap can be used to evaluate the mixing (and hitting) times of classical Markov processes - the hitting time for a marked element M of a symmetric ergodic Markov process P, $H(P, M)$ is $O(\frac{1}{\delta})$. Moreover, Szegedy showed that for symmetric, ergodic Markov processes the quantum hitting time for a marked element was always of order the square root of the classical hitting time [2].

Two important but distinct cases of Algorithm 1 occur when, for a search on an ergodic Markov chain, $n = 1$ and $n = \frac{1}{\delta}$, the inverse of the eigenvalue gap. It is easy to see why the case where $n = 1$ is an effective algorithm, since it checks at every step of the Markov process for a marked element. The complexity of such at approach can be found to be $O(\phi_0 + \frac{1}{\delta\epsilon}(\phi_1 + \phi_2))$, where ϕ_i are defined as in section 1 but are the equivalent classical counterparts. Since the reciprocal eigenvalue gap is a measure of the hitting time, one expects that letting the system propagate for $n = \frac{1}{\delta}$ steps may provide marked elements with fewer checks. Such an algorithm will have complexity $O(\phi_0 + \frac{1}{\epsilon}(\frac{\phi_1}{\delta} + \phi_2))$.

2 Discrete time Quantum Walks

Discrete quantum walks can be naively implemented by replacing each state of a classical undirected graph with a quantum state and using a "coin" ancillary state to define the direction that the quantum walk will go in. Clearly for unitarity we must have the graph be d-regular and the quantum walk algorithm will be $UC...CUC\left|x\right\rangle\left|c\right\rangle$, where C is an operator that takes the coin state $\left|c\right\rangle$ of a walker state $\left|x\right\rangle\left|c\right\rangle$ in to equal super-positions of all the possible coin states unitarily, whilst U moves the walker in one step defined by it's coin ancillary state. One example of this is given by a discrete quantum walk on an infinite line (composed of states $\left|-\infty\right\rangle .. \left|-1\right\rangle, \left|0\right\rangle, \left|1\right\rangle .. \left|\infty\right\rangle$) produced by using the two dimensional Hadamard as a coin operator C. This is equivalent to a walk on the undirected graph of infinite vertices in a line connected only to their nearest neighbour.

$$C\left|0\right\rangle\left|+\right\rangle = \frac{1}{\sqrt{2}}(\left|0\right\rangle\left|+\right\rangle + \left|0\right\rangle\left|+\right\rangle)$$

$$C\left|0\right\rangle\left|-\right\rangle = \frac{1}{\sqrt{2}}(\left|0\right\rangle\left|-\right\rangle - \left|0\right\rangle\left|+\right\rangle)$$

$$U\left|n\right\rangle\left|-\right\rangle = \left|n-1\right\rangle\left|L\right\rangle$$

$$U\left|n\right\rangle\left|+\right\rangle = \left|n+1\right\rangle\left|R\right\rangle$$

It is clear to see that the coin operator can have a large effect on the dynamics, altering the kinds of interference that occurs during the walk. For example, the Hadamard coin shown above produces very unbalanced distributions over the states, with positive states constructively interfering and negative ones destructively interfering. Good coins have a large and balanced mixing effect, or in other words, are far away from the identity. Examples of good coins include the "Grover diffusion operator", $G = -H^{\otimes N} U H^{\otimes N}$ (where U inverts the phase of the state $|0\rangle$) which appears in Grover's algorithm, and the discrete quantum Fourier transform. It is worth noting that the condition of unitarity and using the same C at each operation means that this approach can only work for random walks on d-regular graphs - a small subset of possible configurations.

The above, coin based, approach follows that of [9] and has been shown to produce exponentially faster hitting times on regular graphs than the classical counterpart [11–13, 18, 22]. We are interested in general Markov processes which can have differently weighted transition probabilities, corresponding to the weights of a directed weighted graph, so require an extension of discrete quantum walks to these processes.

One way to do this was proposed by Szegedy in 2004 [2,19] and does so by acting on the space of the edges of a weighted graph G corresponding to the process P with N states. This space of edges corresponds to the space $\mathcal{C}^N \otimes \mathcal{C}^N$. The reason for this can be seen by comparison with the coin based walk above - we have simply replaced the combined space of the state and coin with the space of edges, which are equivalent (a state consisting of two neighbouring states is replaced with a state corresponding to the transition between them). For a Markov process P, we may define states

$$|\psi_j\rangle := \sum_{i=1}^{N} \sqrt{P_{ij}} \, |i\rangle \, |j\rangle$$

This state corresponds to the directional edge of Markov chain that takes state i to state j (which occurs with classical probability P_{ij}. Notice that $|\psi_j\rangle$ is normalised since P is stochastic. Clearly we have a unique state for each possible transition of P, thus our states correspond to the edges of P. If we define $S = \sum_{i,j=1}^{N} |i\rangle \, |j\rangle \, \langle j| \, \langle i|$ to "swap" the states then

$$\Pi := \sum_{i=1}^{N} |\psi_j\rangle \, \langle \psi_j|$$

is the projector on to the subspace spanned by $|\psi_j\rangle$. The unitary W_P that will implement the quantum Markov process given by P is given by

$$W_P = S(2\Pi - I)$$

which is a reflection about the subspace spanned by $\{|\psi_j\rangle\}$. the swap operator S ensures that the quantum states, which correspond to the directed edges of the graph, are correctly identified with respect to a walk on the vertices of the graph, and show the direction the walk (on the vertices) has moved in that step. It is easy to see that this is very close to the Grover diffusion operator $G = -H^{\otimes N} U H^{\otimes N} = 2 \, |+\rangle \, \langle +| - I$. Here, however, we have reflection taking in to account those states connected to the j^{th} vertex of the chain. This can be seen as starting from from the edge (i, j) (the one-way edge from vertex i to vertex j) we have transitions P_{ij} in $|\psi_j\rangle := \sum_{i=1}^{N} \sqrt{P_{ij}} \, |i\rangle \, |j\rangle)$ instead of about all possible states (given by $|+\rangle$). Clearly then the Grover coin walk corresponds to a walk of this kind on an equally weighted, fully connected graph (the Markov process corresponding to a uniformly random system). This illustrates the connection between discrete quantum walks and unstructured search. The next section will focus on algorithms that utilise this walk formalism and its connection to produce novel quantum search algorithms. Whilst it appears that this is close to a Grover's style search algorithm, Szegedy published only an algorithm for finding if the set of marked elements contains any elements, as the more general Markov chain diffusion operator can not be used naively (merely replacing the Grover diffusion operator) to construct a search algorithm.

3 Searching With Quantum Walks

A natural way to extend the definition of quantum walk given above to search is to simply modify the Markov transition matrix P used above to only contain transitions to the sought, marked elements, but not away from them (whilst keeping the new transition matrix stochastic by putting $p'_{ii} = 1$ for marked elements i). We call this the absorbing walk P'. To search using a quantum walk, we iterate $W_{P'}$ and after the quantum hitting time $\propto \frac{1}{\sqrt{\delta}}$, which is of order the square root of the classical hitting time [2], a measurement should produce a single marked element with good probability. Hence $\frac{1}{\sqrt{\delta\epsilon}}$ steps are needed for a set of $k = \epsilon N$ marked elements.

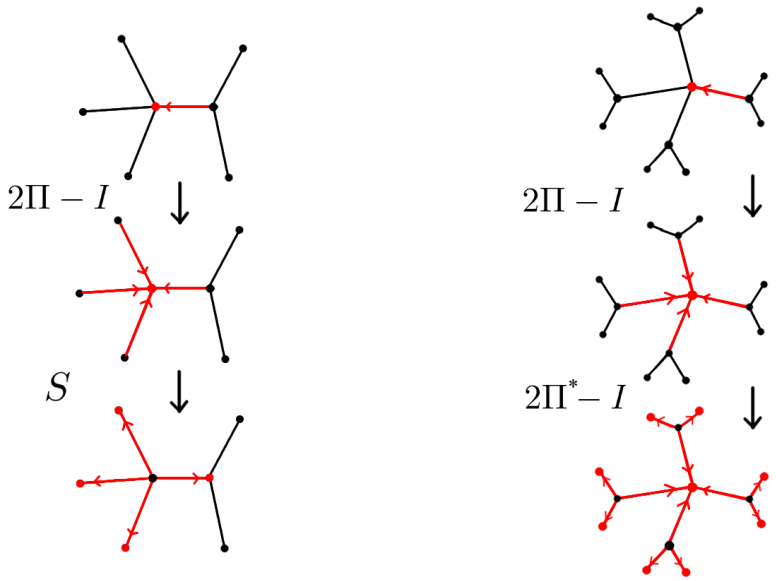

Figure 1: A Schematic showing the difference of the original Szegedy quantum walk [**?**] (left), and the walk employed by Magniez et. al. (MNRS) in their quantum walk based search algorithm [3] (right). Starting with the state $|i\rangle\,|j\rangle$ (the state corresponding to the directed edge from i to j), the Szegedy walk corresponds to to spreading the left-hand state $|i\rangle$ over the neighbours of state $|j\rangle$, then employing a swap operator to make sure the directionality of the walk is properly conserved. Notice that the walks is equivalent to a walk on vertices if the "receiving" state of the transition is considered. Instead of the swap operator, the MNRS Walk implements a spreading over the right-hand, $|j\rangle$ states around the left-hand *keti* states, implementing two steps of the walk and circumventing one of the walk steps. This second step corresponds to a step of the reverse quantum walk W_{P^*}.

This is a quantum analogue of the Algorithm 1 with $n = 1$ and thus has cost $O(\phi_0 + \frac{1}{\sqrt{\delta\epsilon}}\phi_1)$. Here we have no checking cost as no checking operation is performed, we merely expect that after $n = \frac{1}{\delta}$ steps of the walk, we should have been "trapped" on a marked element (as once these have been reached on the walk, there is no way for the walk to move away from them). Here we are not guaranteed to find a marked element upon measurement, and we can not know the algorithm has indeed produced one. Moreover in the implementation of this kind of walk, where the transition probabilities are modified depending on whether the vertex is marked or not seems require that we know which vertices are marked, so that we can alter the transition probabilities of the Markov chain

A more inspired way to use the quantum walk machinery of Szegedy for search is to try to use it's relation to Grover's algorithm, in which the Grover diffusion operator and checking "oracle" are used to "rotate" (in state space) towards an initial state towards the sought elements. A naive way of doing this would be to simply replace the Grover diffusion operator G with the Quantum walk operator W_P. Unfortunately this does not succeed for general Markov processes, as the iterated Grover step no longer correspond to a constant rotation toward the marked item - it varies from iteration to iteration. However, it is possible to recast Ambainis' search algorithm as an instance of Grover's algorithm with some power of W_P replacing G. This is based on a Lemma proved by Ambainis in [1] where he proves that constraints on Grover's G can be relaxed whilst still converging on to a marked element - an important result in the study of Grover's algorithm. However, this freedom is not enough to insert arbitrary random walks as the diffusion in Grover's algorithm, hence Szegedy left the implementation of search on his quantum walks an open problem.

The MNRS algorithm [3] improved on this by using a a modified version of Szegedy's quantum walk to generate reflection operators that can be used in an amplitude amplification scheme (generalized Grover's algorithm for many marked elements, see [7] and for the exact scheme used [8]). Thus the structure of the more general quantum walk can be used to produce operators which are compatible with the structure of Grover's algorithm, whilst still producing a result in $O(\phi_0 + \frac{1}{\sqrt{\epsilon}}(\frac{\phi_1}{\sqrt{\delta}} + \phi_2))$.

The MNRS quantum walk is similar to that of Szegedy, but rather than swapping the indices at each operation in order to the preserve the directionality of the walk, a reversed time step of P (labelled P^*) is taken. This produces the equivalent of two walk steps, however with an intermediate step that does not correspond to the pure walk (see Figure 1). This is done to produce the desired eigenvalue properties of the

walk.

In Grover's algorithm, we reflect our initial state $|\phi\rangle$ alternately over $|\phi^{\perp}\rangle$ and $|\xi\rangle$, where $|\xi\rangle$ is the desired, marked element. This produces a rotation on the the marked element after $\sqrt{N}\frac{\pi}{4}$ iterations. Similarly, setting $|\xi\rangle = |\pi\rangle$, the stationary distribution of W_P, and alternately implementing reflections about $|\mu^{\perp}\rangle$ and $|\pi\rangle$ rotates on to the $|\mu\rangle$ after $O(\frac{1}{\epsilon})$ iterations. Unfortunately implementing a reflection about $|\pi\rangle$ is difficult (whereas rotations about $|\mu^{\perp}\rangle$ can be done with a phase flip oracle with cost ϕ_2 (checking cost). This is resolved by the fact that as the stationary distribution of P, $|\pi\rangle$ is the only eigenvector of P with eigenvalue 1 [3]. Thus implementing a phase estimation algorithm on W_P allows all of the states which produced non zero estimate of the phase to have their phase flipped. This is equivalent to an approximate version of a reflection about $|\pi\rangle$, and allows the Grover-like scheme utilising the quantum walk to be implemented with cost $O(\phi_0 + \frac{1}{\sqrt{\epsilon}}(log(\frac{1}{\sqrt{\epsilon}})\frac{\phi_1}{\sqrt{\delta}} + \phi_2))$. This is possible as in order to do phase estimation, we do not need to use the checking oracle, but must iterate $\frac{1}{\delta}$ steps of the walk. Using an amplitude amplification scheme of Hoyer et. al. [8] this can be reduced to the claimed complexity of $O(\phi_0 + \frac{1}{\sqrt{\epsilon}}(\frac{\phi_1}{\sqrt{\delta}} + \phi_2))$.

This new approach led to the development of the algorithm of Krovi et. al. [6] who use a Markov chain $P(s)$, which is a superposition of the absorbing (P') and standard (P) Markov chain such that $P(s) = (1-s)P + P'$. Their result extends Szegedy's original detection algorithm to the problem of Finding the elements provided an initial guess to the stationary distribution of $P(s)$ can be supplied. This is a similar notion to knowing how many marked elements must be found, since $P(s)$ depends on the absorbing walk P'. Similarly to the MNRS algorithm, phase estimation is then used to find an approximate Grover reflection operator, with an amplitude amplification providing the conversion to the marked element.

4 Concluding Remarks

Aside from the obvious applications, graph based search algorithms can be applied (or shown to be equivalent) to several more abstract problems. For example and as previously mentioned, search on Johnson graphs can be used to solve the element distinctness problem. We will now list various problems that have been recently improved by quantum graph search (and their complexities in terms of queries to an oracle) [23].

Johnson walk based problems have been studied the most due to the early work completed by Ambainis. Some quantum algorithms based on Johnson graph search that beat their classical counterparts include:

- Element Distinctness: This can be solved optimally with quantum walk search with query complexity $O(n^{\frac{2}{3}})$ [1]
- Matrix Product Verification: This can be solved with quantum walk search in time $O(n^{\frac{5}{3}})$ [5]
- Restricted Range Associativity: This can be solved with quantum walk search in time $O(n^{\frac{5}{4}})$ [4]
- Finding Triangles on a Graph: This can be solved with quantum walk search in time $O(n^{\frac{13}{10}})$ [3]

There is also the Group Commutativity problem which requires more general graphs.

- Group Commutativity: This can be solved with quantum walk search in time $O(n^{\frac{2}{3}}\log(n))$ [3]

This is almost certainly an incomplete list as the results are still relatively new and progress is continually being made in this field. Clearly graph based search is a powerful tool, and can be used to great effect to produce new, faster than classical quantum algorithms for a variety of problems.

References

[1] A. Ambainis. Quantum walk algorithm for element distinctness. *eprint arXiv:quant-ph/0311001*, October 2003.

[2] Mario Szegedy. Quantum speed-up of markov chain based algorithms. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 32–41, Washington, DC, USA, 2004. IEEE Computer Society.

[3] F. Magniez, A. Nayak, J. Roland, and M. Santha. Search via Quantum Walk. *eprint arXiv:quant-ph/0608026*, August 2006.

[4] Sebastian Dörn and Thomas Thierauf. The quantum query complexity of algebraic properties. In *Fundamentals of Computation Theory*, volume 4639 of *Lecture Notes in Computer Science*, pages 250–260. Springer Berlin Heidelberg, 2007.

[5] H. Buhrman and R. Spalek. Quantum Verification of Matrix Products. *eprint arXiv:quant-ph/0409035*, September 2004.

[6] H. Krovi, F. Magniez, M. Ozols, and J. Roland. Finding is as easy as detecting for quantum walks. In *Proceedings of 37th International Colloquium on Automata, Languages and Programming*, pages 540–551, 2010.

[7] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp. Quantum Amplitude Amplification and Estimation. *eprint arXiv:quant-ph/0005055*, May 2000.

[8] P. Hoyer, M. Mosca, and R. de Wolf. Quantum Search on Bounded-Error Inputs. *eprint arXiv:quant-ph/0304052*, April 2003.

[9] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani. Quantum Walks On Graphs. *eprint arXiv:quant-ph/0012090*, December 2000.

[10] Edward Farhi and Sam Gutmann. Quantum computation and decision trees. *Phys. Rev. A*, 58:915–928, Aug 1998.

[11] Andrew M. Childs, Edward Farhi, and Sam Gutmann. An example of the difference between quantum and classical random walks. *Quantum Information Processing*, 1(1/2):35–43, April 2002.

[12] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman. Exponential algorithmic speedup by quantum walk. *eprint arXiv:quant-ph/0209131*, September 2002.

[13] J. Kempe. Quantum Random Walks Hit Exponentially Faster. *eprint arXiv:quant-ph/0205083*, May 2002.

[14] D. Q. Shi, M. Ionescu, J. McKinnon, W. M. Chen, and S. X. Dou. Relationship between orientation of CeO_2 films and surface morphology. In *Advances in Cryogenic Engineering ICMC*, volume 614 of *American Institute of Physics Conference Series*, pages 519–524, May 2002.

[15] A. M. Childs and J. Goldstone. Spatial search by quantum walk. *Phys. Rev. A*, 70(2):022314, August 2004.

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

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

[18] A. Ambainis, J. Kempe, and A. Rivosh. Coins Make Quantum Walks Faster. *eprint arXiv:quant-ph/0402107*, February 2004.

[19] M. Szegedy. Spectra of Quantized Walks and a $\sqrt{\delta\epsilon}$ rule. *eprint arXiv:quant-ph/0401053*, January 2004.

[20] F. Magniez and A. Nayak. Quantum Complexity of Testing Group Commutativity. *eprint arXiv:quant-ph/0506265*, June 2005.

[21] Oskar Perron. Zur theorie der matrices. *Mathematische Annalen*, 64(2):248–263, 1907.

[22] H. Krovi and T. A. Brun. Hitting time for quantum walks on the hypercube. *Phys. Rev. A*, 73(3):032341, March 2006.

[23] M. Santha. Quantum walk based search algorithms. *ArXiv e-prints*, August 2008.