Finding a Needle in a Haystack: From Baldwin Effect to Quantum Computation

Akira Imada
Brest State Technical University
Moskowskaja 267, Brest 224017 Republic of Belarus
akira@bsty.by

Abstract

If we want to break someone else's PIN (Personal Identification Number) of, say, an ATM (Automated Teller Machine), how many trials would be necessary when we want to be efficient? This is a sort of what we call *a-needle-in-a-hay-stack* problem. In 1987, in their wonderful paper, Hinton & Nowlan proposed a Genetic Algorithm with a needle being a unique configuration of 20-bit binary string while all other configurations being a haystack. What they proposed was to exploit a *lifetime learning* of individuals in their Genetic Algorithm, calling it the Baldwin effect *in a computer.* Since then there has been a fair amount of exploration of this effect, claiming, "This is a-needle-in-a-hay-stack problem, and we've found a more efficient algorithm than a random search." Some of them, however, were found to be the results of an effect of *like-to-hear-what-we-would-like-to-hear.* In this talk, we will try a bird's eye view on a few examples we have had so far, and how they were explored, including the approach by means of quantum computation which claims, "The steps to find a needle are $O(\sqrt{N})$ while those of exhaustive search by a traditional computer are $O(N)$ where N is the number of search points."

I. Introduction

Usually in an ATM (Automated Teller Machine) or in a private web page, we need to show our PIN (Personal Identification Number), and for the security reason we are allowed to try inputs just a couple of times, typically three times. So it might interest us how many times would be necessary if we are allowed to repeat a trial-and-error procedure until we reach the legitimate number? Though typically PIN is a 4-digit decimal, we now assume n-bit binary PIN. Let me show two really simple strategies, that is, (i) random search and (ii) exhaustive one-by-one search. In Fig. 1, results of both strategies as for the number of trials until we find a hidden and unique PIN, averaged after 100 runs, as a function of n. In both cases we see the number of trials explodes exponentially as number of bits increases. The motivation of this paper is to argue if we have more efficient strategies or not.

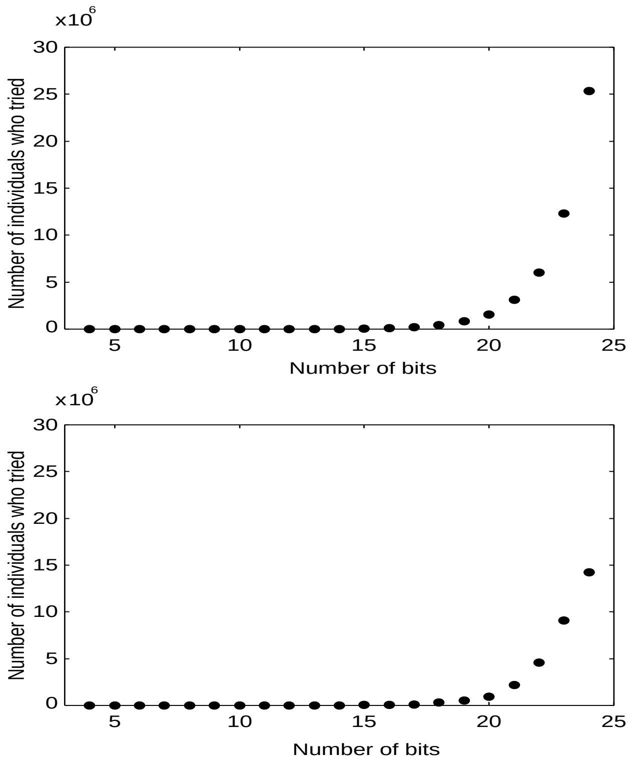

Fig. 1. The number of trials needed to find out a hidden and unique PIN, averaged after 100 runs, as a function of n. Top: A result by a random search. Bottom: An exhaustive one-by-one search.

II. Baldwin Effect

It was Hinton & Nowlan [1] who firstly proposed computational model of the Baldwin Effect. It was in 1987. The scenario was exactly the same as above mentioned PIN breaking. What they proposed was *a lifetime learning* of each of the individuals in an algorithm which based on a population search. That is, each individual has adaptive genes in addition to the original 0's and 1's. Let's denote it here as 9's. For example, the individual is like:

$$(01100900099110011901).$$

Then each individual is allowed to assign 0 or 1 at the location where 9's are, which is referred to as *life-time learning* of individuals when we assume those strings which includes 9's are genotype and strings in which all of the 9's are replaced either 0 or 1 are phenotype, and replacement will not affect genotype. Individuals are allowed to try this learning, say, 1000 times in their lifetime. If, by chance, phenotype matches to the needle, then the number of learnings before the success is the fitness of the individual. This might be called the Baldwin effect. It has attracted, and is still attracting, many computer scientists. In 2005, Mills & Watson [2] made a more deeply analytical approach to the issue. They analyzed in exactly the same universe as the one Hinton & Nowlan had explored almost two decades ago, while many others applied the model to a more sophisticated problems. Later, in 2006, Mills & Watson extended the study to a more complex but still a simple universe of *two needles* whose length is different with each other on a completely flat plateau [3]. What is the Baldwin effect more specifically? Mills & Watson clearly summarized, *"The Baldwin effect indicates that individually learned behaviors acquired during an organism's lifetime can influence the evolutionary path taken by a population, without any direct Lamarckian transfer of traits from phenotype to genotype. ... Our knowledge of modern genetics suggests that an organism's lifetime adaptations cannot influence the course of evolution because learned characteristics do not change ones own genes. ... The hypothesis appears very similar to Lamarck's disproved beliefs that an acquired trait is directly inherited by offspring."*

A. Discussion in this Section

Hinton & Nowlan showed that the number of genes whose value is 9 reduced over time of evolution. From an engineering point of view, however, this is not interesting any more. Why? Because it implies that we already knew the secret PIN if at least one of those individuals reached the needle in the first generation, though Mills & Watson [2] pointed out, *"It is implicit in Hinton & Nowlan's model that the organisms (phenotype) do not have to recognize their own success."*

The top graph in Fig. 2 with filled circles shows the effect of lifetime learning in the first generation. We can see enormous reduction of the number of individuals until the needle is firstly found. But, alas, if we count the total number of points which are visited by any of those individuals during their lifetime learning, then we realize the result is almost the same as the random search. The time needed to reach the needle is indeed almost the same in all those three cases; that is, random search, exhaustive search, and search by lifetime learning. In other words, when individuals are allowed a local search around where they locate, the number of individuals necessary to finally reach to the needle is much smaller than a simple random search, but total points explored by individuals are the same more or less.

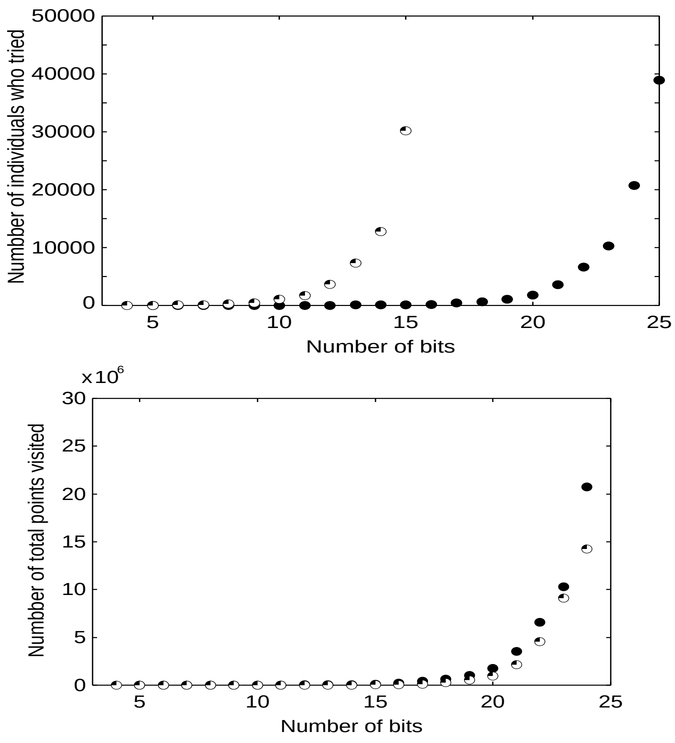

Fig. 2. Top: The total number of individuals until one of them reaches to the needle (filled circle). Bottom: The total number of points visited by any individuals during its lifetime learning until one of them reaches to the needle (filled circle). The same set of data of number of individuals who made random search shown in Fig. 1 is also in both figures for comparison purpose (partly filled circle). Note the scale of y-axis of the top figure was totally different from others. all data are average after 100 runs.

Anyway what we have to notice from the top graph of Figure 2 is that the complexity is still $O(N)$ despite claims of efficiency of the Baldwin effect if any.

III. Neutral Mutation on Intron

A. Reduced Even-n-Parity Problem

Even-n-parity problem is a classic benchmark problems in evolutionary computation research. Given any n-bit binary input, it returns 1 if the number of 1 of the input is even, otherwise it returns 0. Assume now we are constructing a population of candidate solutions of even-n-parity logic circuit only by using *XOR* and *EQ* Boolean functions, which is called a *reduced* even-n-parity problem. We can evaluate each of those candidate solutions by giving it all possible combinations of n-bit binary string from one to the next. When we check each result of whether the output is correct or not, we will notice that there are only three cases: (i) all of the results for those 2^n test cases are correct; (ii) a half of the results are correct, and (iii) all are incorrect.

The number of all possible candidates when we are allowed to use either of *XOR* and *EQ* gates from 1 up to N will be

$$\sum_{i=1}^{N} 2^i n^{i+1}.$$

The number of solutions are analytically given by Collins [4]. For $n = 12$, for instance, Collins wrote the number of solutions is 2.568×10^{134} while the number of all possible candidates is 1.315×10^{142} when we use up to 100 gates of either *XOR* or *EQ*. That is, the ratio of solutions in the search space is 0.00042%. So we might say the solutions are like *needles in a haystack*.

Then we have a very interesting discussion described in the following two subsections.

B. *Finding Needles in Haystacks is Not Hard with Neutrality?*

In 2002, Yu & Miller [5] applied *Cartesian Genetic Programming* (CGP) [6] to the even-n-parity problem.

Algorithm 1 (Candidate Creation in CGP)

```
(1) Create 100 gates of XOR or EQ at random.
(2) Select randomly one output gate out of those 100 gates created.
(3) From one gate to the next, set the two input connections to either of the output line previously set gate or one of the $n$ input lines at random.
```

Let me show an example, paraphrasing the one from Yu & Miller's paper [5], assume now we create 6 gates instead of 100 just for the sake of simplicity. The gate is numbered *a, b, c, d, e, f,* and the gate f is the output unit. Then a genotype will be

$$((EQ, 1, 2)^a (EQ, 3, a)^b (XOR, a, a)^c$$
$$(XOR, b, 3)^d (EQ, 4, c)^e (EQ, d, 4)^{f=out}).$$

The phenotype of this example genotype is

$$1 \ \text{EQ} \ 2 \ \text{EQ} \ 3 \ \text{XOR} \ 4 \ \text{EQ} \ 5.$$

Note that the 3rd gene c and the 5th gene e do not contribute to construct the phenotype, and hence these are called *introns*. As you might notice that this CGP representation inevitably includes introns. Mutation of the intron will have no effect on the phenotype and this is why the mutation is called *neutral*.

By running an evolutionary algorithm with one parent and 4 offspring, the authors showed this neutral mutation on the intron enhances the ability to search for the solutions. To be more specific, with the maximum number of gates being 100 and the maximum iteration being 10,000, carefully tuned 48 CGP runs out of 100 for $n = 12$ found a solution (48%), while a random search, any randomly generated 4,000,000 candidates for $n = 12$, were not solutions (0%).

C. *Finding Needles in Haystacks is Harder with Neutrality?*

Collins doubted the result of the paper by Yu & Miller in the above subsection. In his paper, he argued that 10,000 iterations with 4 offspring are a sample of at most 40,000 points in the search space [7]. The CGP Yu & Miller reported for $n = 12$ resulted in all *EQ* gates, and the solution density in the case of 12 parity with one type of gate is 0.003756%. Hence an expected success rate should be

$$1 - (1 - 0.00003756)^{40000} = 0.778$$

and as such, Yu & Miller's average success rate of 48% mentioned above might be said not to be a good one. He proposed a random sampling of candidates as follows.

Algorithm 2 (Randomly sampling non-CGP representations)

```
(1) Create 100 random gates either from XOR or EQ.
(2) Set the output to the 100-th gate.
(3) For each gate:
        (i) Set the type of the gate to either XOR or EQ at random.
        (ii) Set one of the input connection of the gate to the previous gate.
        (iii) Set the other input connection to a randomly selected input line.
(4) Repeat (3) up to N.
```

Then Collins wrote, "We recorded 0.0204% success from a sample of 10 million trials." Notice that this result was without an evolution where Yu & Miller's didn't find any solutions in 40 million trials.

D. *Discussion in this Section*

Collins' logic seems to be clear. However, what about the condition of *"the number of gates up to 100?"* Although he also admitted and wrote, *"This sampling* *method is also not 'fair' – it over samples short functions as well,"* the situation would be worse than that. Implication of Yu & Miller's paper is that the solutions will be no more possible to be found for n larger than 12. And the list in the Collins' paper regarding the density of the solutions in the search space was limited to $n < 16$. However, just a brief consideration shows it's really easy to find one of the simplest solutions even for $n = 16$ as shown in the Fig. 3 and 4. Or even for much bigger n. Those are simply the first and second trials on the paper by me as a human, not by an computer algorithm. Theoretically, Yu & Mille's method of creating initial population could produce both of these two simple solutions, because intron could make some gates inactive, but not so likely or rather few. Collins' way, on the other hand, will never create both of these two. Though it is frequently said that an evolutionary algorithm can create sophisticated solutions that a human could not, The above is of opposite case.

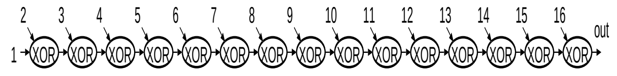

Fig. 3. An example of solutions of reduced even-16-parity found by a human not by a computer algorithm.

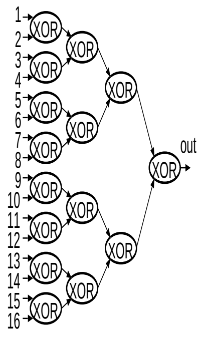

Fig. 4. Yet another example of solutions of reduced even-16-parity.

IV. Search by Quantum Random Walk

Back in 1965 Fyenman considered computers could not simulate some quantum mechanical phenomena [8], which led to his later proposal of a possibility to use quantum mechanics [9][10]. Since then the topic has attracted physicists' and computer scientists' interest and curiosity.

In 1997, Grover [11] reported an algorithm exploiting concepts in quantum mechanics to search for a needle in a haystack more efficiently than a traditional computation. As previously mentioned, we don't know essentially more efficient algorithm to search for a needle in a haystack than the one whose complexity is $O(N)$ where N is the number of points in the whole search space. For example, we can see the $O(N)$ complexity in Fig. 5 where we plot the data already shown in Fig. 1 as a function of number of points of the search space instead of the number of bits.

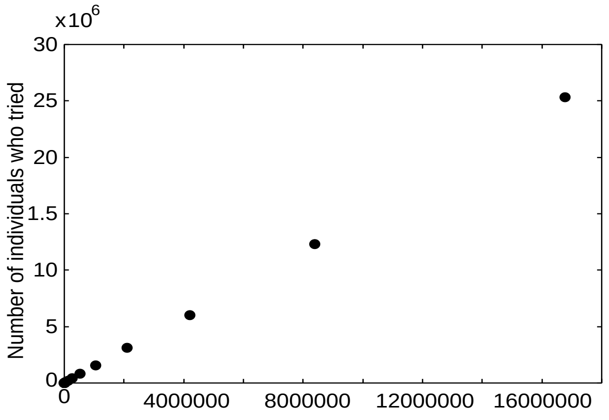

Fig. 5. Plots of the points already shown in the top of Fig. 1, but this time against the total number of points in the corresponding search space not against the number of bits.

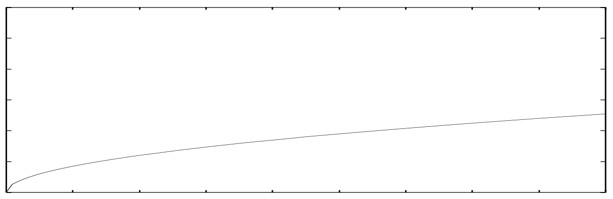

Fig. 6. A fictitious plot of the Grover's Quantum algorithm to search for a needle in a haystack.

Grover's proposed algorithm was proofed to be able to search for the needle with a complexity of $O(\sqrt{N})$ as shown in the fictitious graph in Fig. 6. However, practical quantum computer hardware has never existed yet. Up to now, all we have had are just a few toy implementation of quantum computers.[1] Nevertheless, arguments regarding the possibility of quantum algorithms

[1] At the moment I wrote this paper, I found an article in New York Times which read *"D-Wave, a start-up company in Burnaby, British Columbia, demonstrated 'the world's first commercial quantum computer' in February."* But many Physicists and Computer Scientists are still fishy about that. The headline was *"A Giant Leap Forward in Computing? Maybe Not. If a 'practical quantum computer' had been built and demonstrated, it would be a wonderful thing."* — New York Times on 8th March 2007 by Jason Pontin.

becomes more and more active these days. Concerning the topic of this paper – search for a needle – see, for example, Kempe's [15] wonderful introductory overview of *quantum random walks.* The author wrote, *"We will give a thorough introduction to the necessary terminology without overburdening the reader with unnecessary mathematics."*

A. Notations – A little intimidating though

While the state space of a general quantum system is modeled by wave functions in an infinite Hilbert space, the finite quantum system we need for quantum computing is sufficient to be modeled on a Hilbert space of finite dimensional complex vector with an inner product that are spanned by abstract wave functions [13].

Classical computer systems represent a single bit of information deterministically: the value is either a logic 0 or a logic 1. Quantum computer systems represent a single bit of information as what they call a *qubit*, which is a unit vector in a complex Hilbert space [12].

Here we use the *bra/ket* notation coined from the word *bracket*, which was introduced by Dirac [16] The *ket* symbol is denoted by $|x\rangle$ and the corresponding *bra* is denoted by $\langle x|$. The *ket* describes a quantum state and the corresponding *bra* is its complex conjugate [12].

In computer science domains, the *ket* and *bra* is a column and row vector respectively. For example, the orthonormal basis $\{|0\rangle, |1\rangle\}$ can be expressed like $\{(1,0)^T, (0,1)^T\}$ by using this *ket.* Any complex linear combination of two *kets* is also a *ket.* The inner product of two vectors is denoted by $\langle x|y\rangle$. Note that since $|0\rangle$ and $|1\rangle$ are orthonormal, $\langle 0|1\rangle = 0$. On the other hand, $|x\rangle\langle y|$ denotes the outer product of the vectors [12].

Then n bit binary string in the traditional computation can be denoted $|b_{n-1}b_{n-2}\cdots b_0\rangle$ where each b_j is either 0 or 1. Furthermore, we abbreviate this by using corresponding decimal number. For example $|12\rangle$ means $|1100\rangle$ [14].

B. Quantum Random Walk

Our needle search in a haystack is after all *searching the Boolean hypercube.* The first algorithm of this type was proposed by Shenvi et al. [17]. Shenvi starts his argument

> The discrete time random walk can be described by the repeated application of a unitary evolution operator U. This operator acts on a Hilbert space $H^C \otimes H^S$ where H^C is the Hilbert space associated with a quantum coin and H^S is the Hilbert space associated with the nodes of the graph ($\otimes$ is tensor products). The operator U can be written as
>
> $$U = S \cdot C$$
>
> where S is a permutation matrix which performs a controlled shift based on the state of the quantum coin, and C is a unitary matrix which "flips" the quantum coin [18].

Ambainis [19] clearly summarizes the algorithm.

> Let V be the set of vertices and E be the set of edges. We use basis states $|v, 3\rangle$ for $\forall v \in V$ and $e \in E$ such that the edge e is incident to the vertex v. One step of quantum walk now consists of the following:

Algorithm 3

(1) *For each v, perform a coin flip transformation C_v on the states $|v, e\rangle$*
(2) *Perform shift S defined as follows. If edge e has two vertices in its both ends, v and v', then $S|v, e\rangle = |v', e\rangle$ and $S|v', e\rangle = |v, e\rangle$.*

Ambainis [19] continues:

> As a coin flip C_v Grover's diffusion D_n is a good option, where D_n is $n \times n$ matrix in which all elements except for the diagonal ones are $2/m$, and the diagonal elements are all $(-1 + 2/m)$.
>
> In the Boolean hypercube, we have $N = 2^n$ vertices v_x indexed by n-bit strings. Two vertices v_x and v_y are connected by an edge if the corresponding strings x and y differ in exactly one place. The maximum distance between two vertices is $n = \log N$. Thus, Grover's algorithm could search this graph in $O(\sqrt{N \log N})$ steps. Shenvi et al. [17] showed how to search it by quantum walk in $O(\sqrt{N})$ steps. Consider now Hilbert space spanned by $|\mathbf{x}\rangle|i\rangle$ where $\mathbf{x} \in \{0,1\}^n$ corresponds to a vertex of hypercube and $i \in \{1, 2, \cdots, n\}$ corresponds to an edge.

Ambainis [19] summarizes the algorithm clearly as follows.

Algorithm 4

(1) *Generate the starting state*

$$\frac{1}{\sqrt{2^n n}} \sum_{x,i} |x\rangle|i\rangle.$$

(2) Perform $O(\sqrt{N})$ steps with each step consisting of

   (i) Apply D_n to $|i\rangle$ register if the vertex x is not the target. Otherwise Apply $-I$ (identity matrix).

   (ii) Apply shift S defined by

$$S : |x\rangle|i\rangle \quad \rightarrow \quad |x^i\rangle|i\rangle.$$

If there is no target items, then the algorithm stays in the starting state. If there is a single target vertex v_x, then, running this algorithm for $O(\sqrt{N})$ steps and measuring the state gives $|x, i\rangle$ for the target x and some i with a high probability.

Let me stop now since we are running out of space and time. Hoping to have a hot and useful discussion on this topic in the session at Elk.

References

[1] G. E. Hinton, and S. J. Nowlan "How Learning Can Guide Evolution." Complex Systems, Vol. 1, 1987, pp. 495–502.

[2] R. Mills, and R. A. Watson "Genetic assimilation and canalization in the Baldwin effect." In Proceedings of the European Conference on Artificial Intelligence, 2005, pp. 353-362.

[3] R. Mills, and R. A. Watson "On Crossing fitness Valleys with the Baldwin Effect." In Proceedings of Artificial Life X, 2006, pp. 493–499.

[4] M. Collins " Counting Solutions in Reduced Boolean Parity." In Workshop Proceedings of the Conference on Genetic and Evolutionary Computation, 2004, pp. 26–30.

[5] T. Yu, and J. Miller "Finding Needles in Haystacks Is Not Hard with Neutrality." In Proceedings of the European Conference on Genetic Programming, LNCS 2278, Springer, 2002, pp. 13–25.

[6] J. F. Miller, and P. Thomson "Cartesian genetic programming." In Proceedings of the European Conference on Genetic Programming, LNCS 1802, Springer, 2000, pp. 121–132.

[7] M. Collins "Finding Needles in Haystacks is Harder with Neutrality." In Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, 2005 pp. 1613 – 1618.

[8] R. Feynman, R. Leighton, and M. Sands "Lectures on Physics" Vol. III, Addison-Wesley, 1965.

[9] R. Feynman "Simulating physics with computers. International Journal of Theoretical." Physics Vol. 21, 1982, pp. 467–488.

[10] R. Feynman "Quantum mechanical computers." Foundations of Physics, Vol. 16, 1986, pp. 507–531.

[11] L. K. Grover "Quantum Mechanics helps in searching for a needle in a haystack." Physical Review Letter, Vol. 79, 1997, pp. 325–328.

[12] G. W. Greenwood "Finding Solutions to NP Problems: Philosophical Differences Between Quantum and Evolutionary Search Algorithms," In Proceedings of the Congress on Evolutionary Computation, 2001, pp. 815 - 822.

[13] E. Rieffel, and W. Polak "An Introduction to Quantum Computing for Non-Physicists." Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/9806078, 1998.

[14] L. Spector, H. Barnum, H. J. Bernstein, and N. Swamy "Finding a Better-than-Classical Quantum AND/OR Algorithm using Genetic Programming" In Proceedings, of the Congress on Evolutionary computation, 1999, pp. 2239–2246.

[15] J. Kempe "Quantum Random Walks – An Introductory Overview." Contemporary Physics, Vol. 44 Issue 4 (also preprint quant-ph/0303081), 2003, pp. 307-327.

[16] P. Dirac "The Principles of Quantum Mechanics." Oxford University Press, 4th edition, 1958.

[17] N. Shenvi, J. Kempe, and B. Whaley "A Quantum Random Walk Search Algorithm" Physical Review A, Vol 67 (also preprint quant-ph/0210064), 2003.

[18] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, "Quantum Walks On Graphs." In Proceedings of ACM Symposium on Theory of Computing, ACM (also preprint quant-ph/0012090), 2001, pp. 50–59.

[19] A. Ambainis "Quantum Walks and Their Algorithmic Applications " International Journal of Quantum Information, Vol 1 (also preprint quant-ph/0403120), 2003, pp. 507–518.