An Alternative Method of Analysis of Quantum Decoherence

Brian Kenneth A. de Jesus
Henry N. Adorna
bkdejesus@up.edu.ph
ha@dcs.upd.edu.ph
Department of Computer Science (Algorithms & Complexity)
University of the Philippines Diliman
Diliman 1101 Quezon City, Philippines

ABSTRACT

Quantum computing has gained great interest due to its speedup over classical computing. Classical Computing has been limited to it's hardware as well as the algorithms it can utilize, whereas quantum computers take advantage of quantum properties of its corresponding physical implementations. This is not without some disadvantages, one of which we focus on which is quantum decoherence. Various models of decoherence were studied and one was selected that is most suitable for computational analysis. The paper presents three findings (1) An alternative method of analysis is proposed and compared to existing analysis. (2) A demonstration of the selected method on comparison of distributed vs. non-distributed quantum algorithms. (3) A comparison of two different algorithms on the same problem. From these it is demonstrated that quantum decoherence is a significant factor in the efficiency of quantum algorithms.

KEYWORDS

Quantum Algorithms, Decoherence, Dihedral Subgroup Problem, Quantum Search, Prime Factorization

1 INTRODUCTION

The rise in significance of quantum computing is due to the fundamental limits of computer miniaturization. Microprocessors gets smaller and smaller through advances in technology and it is only a matter of time that the limit is reached. At that point the system will behave quantum mechanically rather than the current macroscopic behavior.

Rather than avoid this change in behavior, it is possible that the properties of quantum particles are exploited and used for computational purposes. Quantum computing is a computational model that is inspired by natural behavior of quantum systems. Unlike classical computing, however, quantum computation should be able to account for the physical limitations of the quantum machine. Perhaps it is more elegantly stated by David Deutsch, "What computers can or cannot compute is determined by the laws of physics alone, and not by pure mathematics."[8]

One such limitation is decoherence. This paper explores the relationship between decoherence and quantum complexity. We follow the idea that time and space complexity are defined in terms of problem size. It answers the basic question:
What is the allowable decoherence per operation per qubit, given an algorithm and a specific register size?
The inverse can also seen, given an incremental improvement in the decoherence rate, what improvement can we expect in the maximum allowable register size for the algorithm to be efficient?

In the study of qubits and quantum computation, it is generally assumed that the system is perfectly isolated from the environment. This assumption, cannot be physically realized as no system can be truly isolated from the environment. At the quantum level, even uncontrollable factors such as cosmic rays can affect the quantum state of the system. This process of interaction, or coupling, of the quantum system with the environment is called *decoherence*. This section discusses the various concepts of decoherence, starting with general concepts (from references [13] [24] [3]) to various papers on its analysis.

Quantum computation exploit quantum phenomenon such as entanglement and superposition to be able to perform better than classical algorithms. By interacting with the computational system in an undesired way, the environment basically applies a measurement on the system. This unintentional measurement causes the superposition to diminish and affect the entanglement of the states as well.

2 PRELIMINARIES

We provide the relevant concepts focusing on Quantum Circuits as well as the relevant algorithms. The primary purpose is to establish conventions but it is also to provide background to the reader. The items in this chapter are derived from reference books on Quantum Computing [13] [17] [24] [7].

2.1 Quantum Circuits

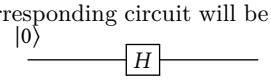
Quantum Circuits is the quantum analogue of classical circuits. Similar to classical circuits, it uses gates with input and output registers. One primary difference, of course, is that qubits are used instead of bits. Another is that due to quantum mechanical limitations, there is no fan-in nor fan out of wires.

Each quantum gate corresponds to a matrix operator in which the a qubit passing through a gate is equivalent to a matrix operator on the qubit.

Definition 2.1 (Hadamard operator[17])**.**

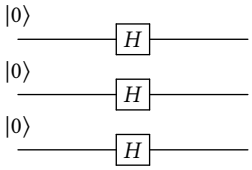
$$\text{Hadamard Operator} \qquad H_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Example 2.1. The corresponding circuit will be the following:

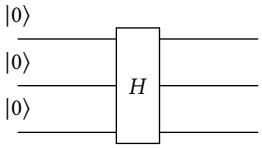


Quantum circuits are read from left to right.

Example 2.2. For multiple qubits, each qubit will represent a wire passing through operators. For example, $H|000\rangle$ will be represented by:



Or for conciseness,



Finally, when a measurement is to be done, a measurement symbol is added to the horizontal wire.

Figure 1: Quantum Circuit Symbol for Measurement

2.2 Algorithmic Problems

In this paper, all problem sizes will be standardized set to the variable L, which will correspond to the length of the register. For classical this will be the number of bits and for quantum, the number of qubits required to encode the problem. This is to standardize the comparison of algorithms and the effects of decoherence.

2.2.1 Standard Method for Solving the Hidden Subgroup Problem. The standard method of solving the Hidden Subgroup problem is a generalization of previous research including the work of Shor and Simon's algorithms. The term was coined in the paper by Gringi, et. al. [11] in their study of non-Abelian Hidden subgroup problem. The approach of this paper is to study the general algorithm and present it on how it applies on the sub-problems of the Hidden Subgroup Problem.

Algorithm:

The algorithm assumes that the elements of the group G can be encoded into different strings, and therefore can be represented by a register of length L, where $| G | = 2^L$.

(1) Using Hadamard gates, initialize an L-qubit register M with an equal superposition if all possible elements.
(2) Apply the function $f(a)$ on M and store the output in a different register N (We assume N is big enough to store the output information).
(3) Measure register N to reduce it to only one value. The values in M will be reduced exactly one coset of H that corresponds to the value in N. This is because $f(a)$ is constant on the cosets of H. N is then discarded.
(4) Apply Fourier transform to M to get some property on the set. (The type of Fourier transform will vary from problem to problem)
(5) Measure register M to obtain an element of the generating set for H
(6) The entire algorithm may be repeated to obtain all elements of H.

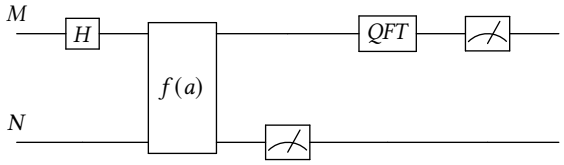


Figure 2: Quantum Circuit for the Standard Method

The quantum circuit for the standard method is shown in Figure 2. The actual function $f(a)$ and the *QFT* may depend on the problem.

2.2.2 Shor's Algorithm. Prime factorization is a seemingly simple problem in to find the non-trivial (1 and itself) factors of a given a number N, represented by a register of length L, where $N = O(2^L)$. Classically, the fastest known algorithm is the number field sieve by Lenstra, et al and runs at $O(e^{L^{-1/3}})$ [5]. The quantum inspired Shor's algorithm runs in $O(L^3)$ [20].

To solve prime factorization, Shor's algorithm reduces the problem to finding the smallest r such that $x^r \equiv 1 \bmod N$, where x is an arbitrary number that is relatively prime to N. If r is even, then we can find the factors of N by factoring the binomial:

$$x^r \equiv 1 \bmod N \tag{1}$$

$$x^r - 1 \equiv 0 \bmod N \tag{2}$$

$$(x^{r/2} + 1)(x^{r/2} - 1) \equiv 0 \bmod N \tag{3}$$

From this we can see that either $(x^{r/2} + 1)$ or $(x^{r/2} - 1)$ is a factor if N, which then can efficiently be verified classically. If the algorithm returns an odd r or trivial factors 1 or N, we simply find another x relatively prime to N, and rerun the algorithm.

The problem of finding r is efficiently done by the Standard Method. We can interpret that problem as the Hidden Subgroup Problem where the group is $\mathbb{Z}_m$, and the function is $f(a) = x^a \bmod n$. The HSP finds the generator r of the hidden set H.

In Shor's paper, the Quantum Fourier Transform used is simply termed as "Quantum Fourier Transform". The more

precise term would be Cyclical Quantum Fourier Transform, as pointed out by Lomont[15]. This is because this QFT is applicable only to cyclic groups, which includes $\mathbb{Z}_n$. The Cyclical Fourier Transform is shown in Equation 4 and the circuit diagram is shown in Figure 3 [13].

$$F_L = \frac{1}{\sqrt{N}} \sum_{j,k=0}^{N-1} e^{\frac{2\pi ijk}{N}} |k\rangle \langle j| \tag{4}$$

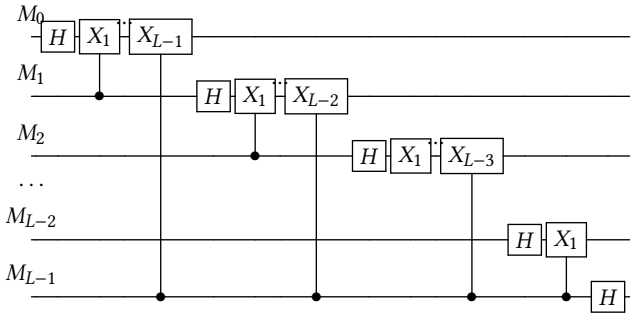


Figure 3: Quantum Circuit for the Cyclic Fourier Transform

2.2.3 Grover's Algorithm. Grover's Algorithm is a novel approach to an unstructured database search, as it improves the optimal classical linear search by quadratic speedup. Its applicability in multiple problems make it another baseline algorithm in the field of quantum computing. It has been generalized by Boyer, et. al. [4], to be able to search any problem space, but the core of the algorithm was defined by Grover[12]. We restate the problem here:

Problem: Given an indexed table T with size 2^L, whose ith element is t_i such that $0 \leq i \leq 2^L - 1$, and a unique element K, find the index j such that $t_j = K$.

The algorithm is designed with an oracle $f(i)$ in mind, wherein it accepts an index i and returns 1 if $t_i = K$, and returns 0 otherwise.

$$f(i) = \begin{cases} 1 & \text{if } t_i = K \\ 0 & \text{otherwise} \end{cases}$$

Algorithm:

(1) Using Hadamard gates, initialize an L-qubit register with an equal superposition if all possible indexes i.
(2) Grover Iterate: Incrementally increase the probability amplitude of the correct index j by performing the following steps:
   (a) Apply the sign changing operator (V_f). Using the oracle, only the amplitude of the index corresponding to the searched item K will be inverted.
   (b) Apply the inversion about the average operator (D_n). The effect is that amplitude of the correct index will increase.
(3) Steps 1 and 2 are repeated $\lceil \frac{\pi}{4}\sqrt{2^L} \rceil$ times to maximize the probability of measuring the correct value of the index. This is commonly referred to as the Grover Iterate.
(4) The register is observed and the index value is verified. If the index is incorrect, the algorithm is repeated.

The sign changing operator (V_f) and the inversion about the average (D_n) operators are given by these:

$$V_f = \frac{1}{\sqrt{2^L}} \sum_{n=0}^{2^L-1} (-1)^{f(i)} \tag{5}$$

$$D_n = \begin{pmatrix} \frac{2}{2^L} - 1 & \frac{2}{2^L} & \cdots & \frac{2}{2^L} \\ \frac{2}{2^L} & \frac{2}{2^L} - 1 & \cdots & \frac{2}{2^L} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{2^L} & \frac{2}{2^L} & \cdots & \frac{2}{2^L} - 1 \end{pmatrix} \tag{6}$$

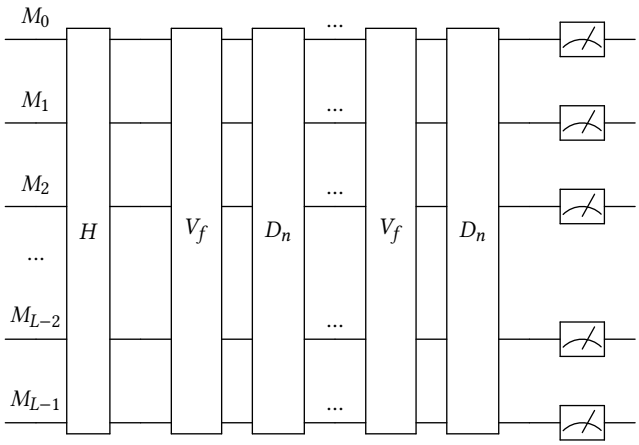


Figure 4: Quantum Circuit for Grover's Algorithm

The quantum circuit for Grover's algorithm is given by Figure 4.

2.2.4 Distributed Grover's Algorithms. Although Grover's algorithm has already been determined to be optimal[25], for practical reasons there are studies that venture into subdividing a large search problem, one of the most recent is a paper by Exman and Levy[10]. We are to discuss this section based on their paper, changing a few naming conventions.

The idea is to divide the search data into P partitions, and running Grover's algorithm for each one in parallel. To determine which subsystem contains the searched item, a classical method if searching a 1 bit among P bits representing the result of each partition.

If the database size of a monolithic Grover's search is 2^L, then for this distributed algorithm, the database size will $2^L/P$, requiring each partition to have a register size of $L - \lg P$. Since they are run in parallel, the running time is the same as the running time of one subsystem. Given the size of the subsystem, we can determine that the running time is

$$n_{op} = \lceil \frac{\pi}{4}\sqrt{2^L/P} \rceil$$

$$n_{op} \approx 2^{L/2}\sqrt{P} \tag{7}$$

The running time of determining the subsystem is considered to be much smaller than the first part of the algorithm, and thus the total running time is dependent on the first part of the algorithm.

2.2.5 Ettinger and Høyer's Algorithm for the Dihedral Hidden Subgroup Problem. The Dihedral Hidden Subgroup Problem (DSP) is the Hidden Subgroup Problem applied to Dihedral Groups. Ettinger and Høyer also employed the Standard Method using a different Fourier Transform that applies to Dihedral Groups, namely the QFT over $\mathbb{Z}_n \times \mathbb{Z}_2$.

The standard method is modified to separate the x and y components of the group element and put them in separate registers. After the QFT of the x element, Hadamard gate is applied to the y element. Figure 5 shows the circuit model.

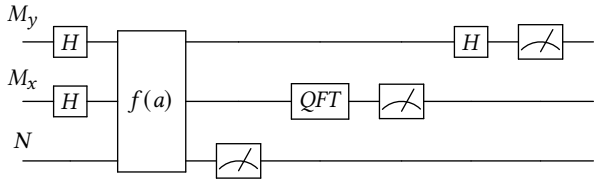


Figure 5: Quantum Circuit for Ettinger and Høyer's Algorithm for DSP

3 DIFFERENT APPROACHES TO ANALYSIS OF DECOHERENCE

After the discovery of Shor's and Grover's algorithms, the physical study of decoherence is being applied to quantum algorithms as well. The following are some research, theoretical and experimental, on how decoherence affects computation.

A study by W. G. Unruh, computes that the time taken in a quantum calculation must be less than the thermal time scale, $\frac{\hbar}{k_B T}$, where $\hbar$ is Planck's constant, k_B is the Boltzmann constant, and T is the temperature[22].

It has been also determined that the rate of decoherence is proportional to the square of the size of the register This has also been verified recently thru experimentation [21] [18].

The rate of decoherence is also affected by the separation of the quantum states of the register [23]. If there are more varied values within the superposition, there is a bigger tendency for the register to change values.

Decoherence is dependent on the technology used. As an example, photons have a low decoherence rate at low energies but may have difficulty exhibiting entanglement. Ion traps and superconductors have relatively high decoherence rates [19].

Another factor of decoherence is the interaction with the external system, called the heat bath. It is the noise that affects the state of the system to give undesirable computational results.

It is worth mentioning that there are multiple ways to correct decoherence. Error correcting methods has been proposed [1] [6] as well as other novel ideas such as qubit recycling[16]. At this point of the study we focus on the effects of decoherence and not yet on a method for error correction.

Studying the effects of decoherence will require some mathematical model. As decoherence has multiple factors, researchers tend to focus on one or two factors for simplicity. All results in the previous section have different models for decoherence. The author recommends a simple method to deal with the quantum mechanical details in an abstract manner, and focus on the computational complexity aspect. The three papers mentioned below follow this paradigm.

A peturbative approach assumes a small deviation (e.g. slight change in probability amplitudes) and applies the error throughout the algorithm to determine the overall effect on the solution. This will determine the upper bound of the acceptable unit error without affecting the results of the computation [2].

A Markovian Master Equation approach can also be used to determine the relationship between the current values of the register and the decoherence of the system at that point in time [23].

Chuang, et al. [5] altogether abstracted away decoherence by defining α, which is defined to be the coherence lost per operation per qubit. The number of trials from the resulting decoherence is computed based on the fully coherent probability P and the decoherence factor α. The register size L and the number of operations at full coherence $n_o p$ are included as parameters.

In terms of computation, the primary concern of the study of algorithms is the complexity of the algorithm. Two primary factors are how the problem size affects time complexity as well as space complexity. The goal of this research will be to find some relationship between problem size and decoherence. Because of this, the method of Chuang, et al. will be used and will be be described in detail.

4 CHUANG, ET. AL. MODEL OF DECOHERENCE

The model of Chuang, et. al. [5], has been chosen for the following reasons: (1) It's parameters focus on problem size and number of operations, both already being used for analysis of complexity. (2) It's simple approach can easily be applied to various algorithms. (3) Speed of computation and material type is abstracted away and thus the study is not limited to specific technologies.

In the paper the model of decoherence is in the form of adding the environment as extra qubits in the system. Equation (7) of the paper shows this.

$$\left|\tilde{\psi}_2\right\rangle = \frac{1}{\sqrt{L}} \sum_{a-0}^{L-1} \left|a, y^a \bmod N\right\rangle \bigotimes |\varepsilon\rangle$$

The study focuses on the effects of decoherence on the first register M and obtain the reduced density matrix.

$$\rho_{red} = \sum_{a=0}^{L-1} \sum_{a=0}^{L-1} (1 - \beta_{aa'}) \left|a\right\rangle \left\langle a'\right|$$

This equation can apply to any register of length L and will not be dependent on the algorithm to be used.

The term $(1-\beta_{aa'})\,|a\rangle\,\langle a'|$ will be nonzero if a and a' are of different values, so the equation can be simplified if it is assumed if $(1-\beta_{aa'})$ is constant. It can be approximated by a constant error we define to be $\mathcal{E}$.

$$1-\beta_{aa'} \approx \exp\left[-\mathcal{E}\left(a \text{ XOR } a'\right)\right]$$

The analysis is then further simplified to instead of using $\beta_{aa'}$, $\beta = 0$ is used to representing a state of complete coherence and $\beta = 1$ representing a state of complete decoherence. The methodology is described in the following section.

To represent the qualitative effect of decoherence, α is defined as the coherence lost per bit in a single logic operation. This variable will represent, in an abstract manner, the type of technology used with respect to maintaining coherence. For conciseness, in this text we will refer to α as *unit decoherence*.

It is assumed that the effect of the environment has a Markovian character and the relationship between α and β is established with $\beta \approx 1 - e^{-n_{op}\alpha L}$. The required number of trials with decoherence (which we label as n_{trial}) is calculated by:

$$n_{trial} = O(L/(1-\beta))$$

Which we can formulate in terms of α:

$$n_{trial} = O(L/(e^{-n_{op}\alpha L}))$$

The numerator L is derived from the reciprocal of the probability of the correct answer, given perfect operation (no decoherence). Decoherence ultimately affects the probability of getting the correct result. Since getting incorrect results can be expected due to the probabilistic nature of quantum computers, it is natural to simply rerun the program. This is how decoherence affects the computational complexity of the algorithm.

The limit in which we tolerate decoherence is when the running time of the algorithm is worse than the best known classical algorithm. The reasoning for this benchmark is that there is no reason to use quantum technology if it is not better than the corresponding classical algorithm. The comparison is specific to the algorithm, and not the problem. Since we are using the best known classical algorithm as a benchmark, advances in that field of research will affect the comparison.

From the paper we can derive a step by step description on how we find the relationship between the unit decoherence and problem size.

Consistent with the entire text, all algorithms are to be assumed to use the same register size, L. This will make it easier to compare the relationship between L and α.

(1) Obtain information on the algorithms to be used.
   - n_{op}, the number of operations to complete the algorithm, in terms of the register size L.
   - P, the probability of obtaining the correct result at the end of the algorithm, in terms of the register size L
   - $n_{op(classical)}$, the fastest known classical algorithm in terms of the register size.

(2) Compute for n_{trial}, the number of trials to complete the algorithm, taking account for decoherence.

$$n_{trial} = O\left(e^{n_{op}\alpha L}/P\right) \tag{8}$$

(3) As per our criteria, compare n_{trial} and $n_{op(classical)}$. This equation will determine the relationship between L and α.

$$n_{trial} < n_{op(classical)} \tag{9}$$

A summary of the results by [5] is shown in Table 1. IT shows the step by step computation from number of operations to the unit decoherence α.

Register Size	L
n_{op}	L^2
P	$O\left(\frac{1}{L}\right)$
n_{trial}	$O\left(Le^{L^3\alpha}\right)$
$n_{op(classical)}$	$O\left(2^{L^{1/3}}\right)$
α Condition	$\alpha < \frac{1}{L^{8/3}}$

Table 1: Decoherence Analysis of Chuang, et. al. Shor's Algorithm

5 PROPOSED CHANGE IN ANALYSIS

We suggest a slight modification of the methodology to get more accurate results. Since α is the decoherence lost per bit per operation, instead of obtaining β by the product $n_{op}L$, we look into the actual circuit used and *count* the actual number of operations at the qubit level, which we define to be n_{qop}. The factor β is therefore changed as well as the formula for n_{trial}:

$$\beta = 1 - e^{\alpha n_{qop}} \tag{10}$$

$$n_{trial} = \frac{e^{\alpha n_{qop}}}{P} \tag{11}$$

If applicable, we now apply this new methodology to the past algorithms we discussed.

5.1 Shor's Algorithm

Using this alternative analysis, we show the following:

THEOREM 5.1. *For Shor's Algorithm, given a problem size L, the allowable decoherence per operation per qubit α is bound by $\alpha < \frac{1}{L^{5/3}}$.*

PROOF. The number of operations on a qubit can be derived from the quantum circuit. Each gate will correspond to a matrix operation and therefore the number of operations is simply the total number of gates of the circuit. Gates with two inputs will count as two operations. If we refer to the design of the Cyclical Quantum Fourier Transform Circuit (Figure 3), we can compute n_{qop}. The first wire will contain

$L-1$ Phase Shift Gates and 1 Hadamard gate, while the last wire only contains a Hadamard gate. In general, wire x will contain $L-x$ Phase Shift Gates and 1 Hadamard Gate. Shift gates operate on two qubits, and thus counts as 2 operations per qubit.

$$n_{qop} = \sum_{x=1}^{L} 2(L-x)+1$$
$$n_{qop} = 2\sum_{x=1}^{L} L - 2\sum_{x=1}^{L} x + \sum_{x=1}^{L} 1$$
$$n_{qop} = 2L^2 - 2\frac{(L)(L-1)}{2} + L$$
$$n_{qop} = 2L^2 - L^2 - L + L$$
$$n_{qop} = L^2$$

We then proceed with computing n_{trial} and α. We summarize the straightforward computation using the rest of the methodology as follows:

$$n_{qop} = L^2$$
$$P = O\left(\frac{1}{L}\right) \tag{12}$$
$$n_{trial} = O\left(Le^{L^2\alpha}\right) \tag{13}$$
$$\alpha < \frac{1}{L^{5/3}} \tag{14}$$

□

What counts for the discrepancy? The formula $\beta = e^{\alpha n_{op} L}$ assumes that for every operation, all qubits will be involved. In the Cyclic QFT for example, it runs in $O(L^2)$ steps, but for each step, only one or two qubits are involved. As decoherence is a physical limitation, the design of the actual circuit may help in obtaining actual results. As we have seen less operations per qubit, we can expect less decoherence in the results, and therefore a higher tolerance for decoherence given a certain problem size L.

5.2 Ettinger and Høyer's Algorithm for the Dihedral Hidden Subgroup Problem

In this section we analyze the algorithm used in the paper by Ettinger and Høyer [9]. We cite the Theorem as stated in the paper, with the conventions adapted to this paper.

Theorem 5.2. *(Ettinger, Høyer) Let $f : D_N \to R$ be a function that fulfills the dihedral subgroup promise with respect to H. There exists a quantum algorithm that given f, uses $\Theta(L)$ evaluations of f and outputs a subset $X \subseteq D_N$ such that X is a generating set for H with probability at least $1 - \frac{2}{2^L}$.*

The approach is to use an abelian subgroup within the dihedral subgroup and solve the HSP for the abelian subgroup. For this, the QFT is used and therefore the number of operations will be the same as what we get from the analysis of Shor's algorithm.

The computation for Ettinger and Høyer's algorithm [9] will be similar to the analysis for Shor's algorithm, since it also used the Cyclic Fourier Transform for $\mathbb{Z}_n$. The $\mathbb{Z}_2$ component of the algorithm is a constant time so it can be removed from this analysis.

Theorem 5.3. *For Ettinger and Høyer's Algorithm for the Dihedral Hidden Subgroup Problem, given a dihedral group D_{2^L}, the allowable decoherence per operation per qubit α is bound by $\alpha < \frac{1}{2L}$.*

Proof. The DHSP for a problem of size L (or $N = 2^L$) will have the same number of gates in the QFT part, but the probability of getting the Hidden Subgroup is $1 - \frac{1}{2^L}$. The following equations show the calculations based on the proposed methodology.

$$n_{qop} = L^2$$
$$P = O\left(1 - \frac{1}{2^L}\right) = O\left(\frac{2^L - 1}{2^L}\right) \tag{15}$$
$$n_{trial} = O\left(\left(\frac{2^L}{2^L - 1}\right)\left(Le^{L^2\alpha}\right)\right)$$

We remove lower order terms and cancel out exponential orders in 2 and e.

$$n_{trial} = O\left(e^{L^2\alpha}\right) \tag{16}$$

As the benchmark is $n_{op(classical)} = O\left((2^{L/2})\right)$, α will have the following limit:

$$\alpha < \frac{1}{2L} \tag{17}$$

□

It is worth noting that the overall algorithm is still exponential in running time, but the quantum section is polynomial. In addition, the probability of getting the correct generating subgroup is significantly higher than that of Shor's algorithm.

5.3 Kuperberg's Algorithm for the Dihedral Hidden Subgroup Problem

Kuperberg improved upon the performance of Ettigner and Høyer's algorithm. The running time is sub-exponential but the number of qubits is also sub-exponential. We use his first algorithm in [14] for analysis with respect to decoherence.

Theorem 5.4. *(Kuperberg) There is a quantum algorithm that finds a hidden reflection in the dihedral group $G = D_N$ (of order $2N$) with time and query complexity $2^{O(\sqrt{\log N})}$*

$2^{O(\log N)}$ translates to $2^{\sqrt{L}}$ in terms of the problem size L. We perform the same method of analysis and arrive at the following result:

Theorem 5.5. *For Kuperberg's Algorithm for the Dihedral Hidden Subgroup Problem, given a dihedral group D_{2^L}, the allowable decoherence per operation per qubit α is bound by $\alpha < \frac{L/2}{4^{\sqrt{L}}}$.*

Proof.

$$n_{op} = 2^{\sqrt{L}}$$
$$P = 1 - \frac{1}{e^{2^{m/3-1}}}$$

To simplify, we can actually assume a probability close to 1.

$$P \approx 1 \tag{18}$$

Since the amount of space used is $O(2^{\sqrt{L}})$, the number of trials with decoherence becomes:

$$n_{trial} = O\left(e^{\alpha 4^{\sqrt{L}}}\right) \tag{19}$$

Having the same benchmark which is $(2^{L/2})$, α will have an exponential limit as follows:

$$\alpha < \frac{L/2}{4^{\sqrt{L}}} \tag{20}$$

□

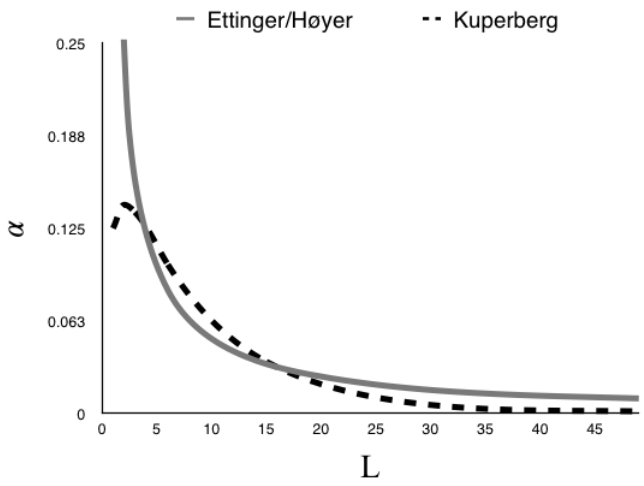


Figure 6: Comparison of Kuperberg's and Ettinger/Høyer's Algorithms (α/L)

We can compute the boundary in which this happens by equating the results of the α

$$\frac{L/2}{4^{\sqrt{L}}} = \frac{1}{2L}$$
$$4^{\sqrt{L}} = L^2$$

Which is satisfied by $L = 16$ and $L = 4$. For $L > 16$ and $L < 4$, Ettinger/Høyer's algorithm will have higher tolerance for decoherence than Kuperber'g algorithm. This demonstrates that different algorithms yield different tolerance for decoherence. We can see that for large L, Ettinger/Høyer's algorithm can have higher decoherence rates.

5.4 Grover's Algorithm and Distributed Grover's Algorithm

In this section, we demonstrate how distributed computing decreases the effect of decoherence. We use Grover's algorithm and a recent distributed as comparison.

Theorem 5.6. *For Grover's Algorithm, given a database size of 2^L, the allowable decoherence per operation per qubit is bound by $\alpha < \frac{1}{2^{L/2}}$.*

Proof. For the analysis of Grover's algorithm, we turn to the results of Zalka [25] wherein he calculated the optimal number of iterations that will maximize the probability of observing the correct solution. We use the variable n_{op} for this text and the value is: [1]

$$n_{op} = \lceil \frac{\pi}{4}\sqrt{2^L} \rceil \tag{21}$$

Also included in his results is the probability of getting the correct solution, which we call P.

$$P \approx sin^2\left(\frac{2T}{\sqrt{2^L}} + \frac{1}{\sqrt{2^L}}\right) \tag{22}$$

Where T is the number of iterations. To get the optimal probability we combine equations 21, 22.

$$P \approx sin^2\left(\frac{2T}{\sqrt{2^L}} + \frac{1}{\sqrt{2^L}}\right)$$
$$P \approx sin^2\left(\frac{2(\frac{\pi}{4}\sqrt{2^L})}{\sqrt{2^L}} + \frac{1}{\sqrt{2^L}}\right)$$
$$P \approx sin^2\left(\frac{\pi}{2} + \frac{1}{\sqrt{2^L}}\right)$$
$$P \approx cos^2\left(\frac{1}{\sqrt{2^L}}\right)$$

To determine the unit decoherence, we analyze the number of operations in Grover's algorithm. The number of iterations is mostly dependent on the number of Grover iterates, which is Equation 21. The initialization and operations within the iterate uses much less time and can be ignored.

Unlike what we determined in the Cyclic QFT, the Grover Iterate operates on all qubits per iteration, and therefore Equation 13 applies. Substituting 21 in 13 brings:

$$n_{trial} = O\left(\frac{e^{\frac{\pi}{4}2^{L/2}L\alpha}}{\cos^2\left(\frac{1}{2^{L/2}}\right)}\right)$$

For large values of L we can assume that the cos term approaches 1. And to simplify the equation, we focus on the term with the highest order, which is the super exponential part. The formula is then reduced to:

$$n_{trial} = O\left(e^{2^{L/2}\alpha}\right) \tag{23}$$

Using the linear search as the classical comparison $(O(2^L))$, we are left with the inequality:

$$2^{2^{L/2}\alpha} < 2^L$$

Finding α we get:

$$\alpha < \frac{1}{2^{L/2}} \tag{24}$$

□

[1] We change the conventions to to be consistent with our own

Compared to Shor's algorithm, the unit decoherence α exponentially decreases with the increase of the number of Qubits in Grover's algorithm. The primary reason is that while the effect of decoherence is the same for both algorithms, they have different speedups versus their classical counterpart.

Grover's algorithm shows only a quadratic speedup and therefore less tolerable to decoherence before it gets less feasible versus a classical linear search.

Analysis of Distributed Grover's algorithm works similarly. Based on the proposed algorithm by Exman and Levy[10], the problem is divided into P partitions. We quickly perform the same calculations but using equation 7 as the problem size. For this we are only interested in the decoherence in one partition as we expect the solution to be in one partition only. We presents the results here:

THEOREM 5.7. *For Exman and Levy's Algorithm for Distributed unstructured Search, given a database size of 2^L, divided into P partitions, the allowable decoherence per operation per qubit is bound by $\alpha < \frac{\sqrt{P}}{2^{L/2}}$.*

PROOF. If we divide the database size of 2^L into P partitions, we will have $2^L/P$ data per partition, which can be indexed by a register of size $L - \ln P$. Preprocessing of dividing the problem and post processing of selecting the correct partition with the correct answer ($O(\ln P)$) are both considered faster than the main algorithm, and thus will not have an effect on the running time. The rest of the calculations follow the same steps as the analysis for Grover's algorithm.

$$P \approx \cos^2\left(\frac{1}{\sqrt{2^L/P}}\right)$$

$$n_{trial} = O\left(\frac{e^{\alpha(L-\ln P)2^{L/2}/\sqrt{P}}}{\cos^2\left(\frac{1}{\sqrt{2^L/P}}\right)}\right)$$

$$n_{op(classical)} = O\left(2^L\right)$$

$$\alpha < \frac{\sqrt{P}}{2^{L/2}}$$

□

The unit decoherence is only quadratically improved by dividing the problem into sub-problems. Figure 7 shows the relationship. The overall decoherence of the entire system will increase, as there are more qubits than the monolithic approach. The difference is that due to distribution, only the partition with the correct solution needs to have the proper solution, and thus the decoherence in other systems will not matter. While it helps in avoiding decoherence, it is more attributed to the reduction of the problem size rather than the distributed nature of the algorithm.

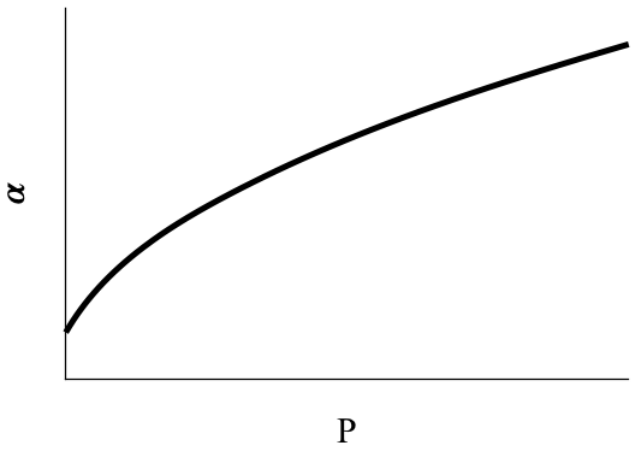


Figure 7: Relationship between the unit decoherence (α) and number of partitions (P)

6 CONCLUSION AND FUTURE WORKS

The general approach for decoherence is to leave it as an implementation problem and ignore the issue from a computational standpoint. However, the nature of quantum mechanics makes it very difficult to do so and we suggest that decoherence not be ignored. We establish decoherence as a parameter in the analysis of quantum algorithms. Here we summarize our results.

Using the modified technique employed in [5], a similar analysis has been done and it shows that Shor's algorithm may potentially be more tolerant to decoherence as previously known. We compare the results of Chuang, et. al. in Table 2. In the study of decoherence, the Quantum Circuit, that closely models actual implementation, is a viable basis for analysis.

We explore the effect of decoherence on distributed computing. We use a recent but straightforward solution by Exman and Levy [10] dividing the problem into P partition and later probing the results. The communication and final processing is considered faster and thus the more critical section of the algorithm is the Grover Iterate. We present the results here and see that dividing the problem into partitions will allow for quadratic improvement with respect to P. This is a demonstration on how distributed computing can reduce the effects of decoherence.

We compare two algorithms of the same problem and observe that they behave differently with respect to decoherence. This is similar to a time and space complexity but for this, we determine the effect of decoherence on problem size. From this we hypothesize that there may be instance that one algorithm may be faster than the other depending on the problem size. Also that for certain decoherence levels (e.g. certain technologies), one algorithm may be preferable to another.

	Chuang, et al [5]	Proposed Method
Register Size	L	L
n_{op}	L^2	N/A
n_{qop}	N/A	L^2
P	$O\left(\frac{1}{L}\right)$	$O\left(\frac{1}{L}\right)$
n_{trial}	$O\left(Le^{L^3\alpha}\right)$	$O\left(Le^{L^2\alpha}\right)$
$n_{op(classical)}$	$O\left(2^{L^{1/3}}\right)$	$O\left(2^{L^{1/3}}\right)$
α Condition	$\alpha < \frac{1}{L^{8/3}}$	$\alpha < \frac{1}{L^{5/3}}$

Table 2: Comparison of Modified Technique in the Analysis of Shor's Algorithm

Algorithm	GBBHT [12] [4]	Distributed [10]
Problem Size	Database size of 2^L	2^L divided into P partitions
Register Size	L	$L - \lg P$
n_{op}	$2^{L/2}$	$2^{L/2}/\sqrt{P}$
P	$O\left(\cos^2\left(\frac{1}{2^{L/2}}\right)\right)$	$O\left(\cos^2\left(\frac{1}{\sqrt{2^L/P}}\right)\right)$
n_{trial}	$O\left(\frac{e^{\frac{\pi}{4}2^{L/2}L\alpha}}{\cos^2\left(\frac{1}{2^{L/2}}\right)}\right)$	$O\left(\frac{e^{\alpha(L-\ln P)2^{L/2}/\sqrt{P}}}{\cos^2\left(\frac{1}{\sqrt{2^L/P}}\right)}\right)$
$n_{op(classical)}$	$O\left(2^L\right)$	$O\left(2^L\right)$
α Condition	$\alpha < \frac{1}{2^{L/2}}$	$\alpha < \frac{\sqrt{P}}{2^{L/2}}$

Table 3: Comparative Analysis of Distributed versus non-Distributed Grover's algorithm

Algorithm	Ettinger/ Høyer [9]	Kuperberg [14]
Register Size	L	L
n_{op}	L^2	$2^{\sqrt{L}}$
P	$1 - \frac{2}{2^L} \approx 1$	$1 - e^{-2^{m/3-1}} \approx 1$
n_{trial}	$O\left(e^{L^2\alpha}\right)$	$O\left(e^{\alpha 4^{\sqrt{L}}}\right)$
$n_{op(classical)}$	$O\left(2^L/2\right)$	$O\left(2^L/2\right)$
α Condition	$\alpha < \frac{1}{2L}$	$\alpha < \frac{L/2}{4^{\sqrt{L}}}$

Table 4: Comparison of Decoherence of Ettinger/Høyer vs. Kuperberg's Algorithms

For further study, the method may be applied to other algorithms, creating a hierarchy of algorithms with respect to decoherence. We may also look into other methods of analysis and consider other parameters such as size of entanglement and separation of states.

ACKNOWLEDGEMENTS

B. de Jesus would like to thank department of Computer Science of the College of Engineering, UP Diliman.

H. Adorna would like to thank the support from DOST-ERDT research grants, the semirarar Mining Dorp.Professorial Chair for Computer Science of the College of Engineering, UP Diliman; RLC grants from UPD-OVCRD.

REFERENCES

[1] Aharonov, D., Ben-Or, M. (2008). Fault-Tolerant Quantum Computation With Constant Error. SIAM Journal on Computing 38:4, 1207-1282.
[2] Azuma, H. (2002). Decoherence on Grover's quantum algorithm: perturbative approach. Phys. Rev. A 65, 042311.
[3] Bacon, D. (2001). Decoherence, Control, and Symmetry in Quantum Computers (Thesis), University of California
[4] Boyer, M., Brassard, G., Høyer, P., Tapp, A. (1996). Tight bounds on quantum searching. Fourth Workshop on Physics and Computation, (pp. 36-43).
[5] Chuang, I., Laflamme, R., Shor, P., Zurek, W. (1995). Quantum Computers, Factoring and Decoherence. Science Vol 270.
[6] Chuang, I., Yamamoto, Y. (1995). A Simple Quantum Computer. Phys. Rev. A 52, 3489–96.
[7] Nielsen, M., Chuang, I. (2010) Quantum Computation and Quantum Information, Cambridge University Press.
[8] Deutsch, D. (1997). The Fabric of Reality (Chapter 5), Penguin Press.
[9] Ettigner, M., Hoyer, P. On Quantum Algorithms for Noncommutative Hidden Subgroups. LANL e–preprint quant-ph/9807029, 1998.
[10] Exman, I., Levy, E. (2012) Quantum Probes Reduce Measurements: Application to Distributed Grover Algorithm, http://arxiv.org/abs/1208.3905
[11] Gringi, M. et al (2004) Quantum Mechanical Algorithms for the Non-Abelian Hidden Subgroup Problem. Combinatorica, (137-154)
[12] Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. Proceedings of 28th ACM STOC, (pp. 212-219).
[13] Gruska, D. (1999). Quantum Computing. McGraw-Hill International (UK) Limited.
[14] Kuperberg, G. (2003) A subexponential-time algorithm for the dihedral hidden subgroup problem, quant-ph/0302112.
[15] Lomont, C. The Hidden Subgroup Problem - Review and Open Problems, 2004.
[16] Martin-Lopez, E., Laing, A., Lawson, T., Alvarez, R., Zhou,X., O'Brien, J. (2012). Experimental realization of Shor's quantum factoring algorithm using qubit recycling. Nature Photonics (online).
[17] McMahon, D. (2008). Quantum Computing Explained. John Wiley & Sons, Inc. 359-372.
[18] Monz, T. , Schindler, P. , Barreiro, J. , Chwalla, M., Nigg, D., Coish, W., Harlander, M., Hansel, W., Hennrich, M., Blatt, R. (2011). 14-Qubit Entanglement: Creation and Coherence. Physical review Letters PRL106.
[19] Paraoanu, G. (2011). Quantum Computing: Theoretical versus Practical Possibility. Phys. Perspect. 13.
[20] Shor, P. (1997). Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing V26, N5, 1474-1483.
[21] Turchette, Q., Myatt, C., King, B., Sackett, C., Kielpinski, D., Itano, W., Monroe, C., Wineland, D. (2000). Decoherence and decay of motional quantum states of a trapped atom coupled to engineered reservoirs. Time and Frequency Division, National Institute of Standards and Technology.
[22] Unruh, W. G. (1995). Maintaining coherence in Quantum Computers. Physical Review A 51.

[23] Walls, D. F., Milburn, G. J. (1984). Effect of Dissipation on Quantum Coherence. Phys. Rev. A Vol 31, No.4.
[24] Williams, C. (2011). Explorations In Quantum Computing. Springer-Verlag London Limited.
[25] Zalka, C. Grover's quantum searching algorithm is optimal. Phys. Rev. A60 (1999) 2746 (quant-ph 9711070)
[26] Zeh, H. (1995) Decoherence, basic concepts and interpretation. Chapter 2 for Decoherence and the Appearance of a Classical World in Quantum Theory (D. Giulini et al., Springer 2003).