Effect of Quantum Decoherence on the Deutsch-Jozsa Algorithm

Luis Gabriel Q. del Rosario
Henry N. Adorna
luigidr97@gmail.com,ha@dcs.upd.edu.ph
Department of Computer Science (Algortihms & Complexity)
University of the Philippines Diliman
Diliman 1101 Quezon City, Philippines

ABSTRACT

Quantum computers have the potential to solve certain problems exponentially faster than classical computers, with one of the most simple examples being Deutsch and Jozsa's black box algorithm for determining whether a function $f : \{0,1\}^n \rightarrow \{0,1\}$ is *constant* or *balanced.* However, one major roadblock in the realization of the quantum computer is *decoherence*, or the loss of quantum information through coupling with the environment. Several methods have been proposed for incorporating decoherence in the study of quantum algorithms, one of which was introduced by Chuang et. al. and redefined by Brian De Jesus in 2014. This method, which had the characteristic of being easily applicable to different quantum algorithms, was used to find that the decoherence of the Deutsch-Jozsa algorithm is bounded by $\alpha < \frac{L}{3L-1}$, which, for large L, shows it is more tolerant than Shor's factoring algorithm and Grover's unstructured search algorithm. Moreover, it was found that even if the algorithm were to return the wrong answer 50% of the time, it would still be more efficient than its classical counterpart.

KEYWORDS

Deutsch-Jozsa Algorithm, Decoherence, Quantum Computing

1 INTRODUCTION

As classical computers become increasingly efficient over time, the components that make up these computers get smaller and smaller. Eventually, transistors may begin to exhibit quantum mechanical behaviour. The idea of quantum computation was brought about as a way of exploiting these quantum mechanical properties, such as *superposition* and *entanglement,* in order to solve certain computational problems [22] [7]. This unconventional model relies on the use of *qubits* instead of classical bits, which have the ability to exhibit these properties during computation. Developments in quantum algorithms have shown exponential speedup over classical algorithms for certain problems such as factoring and discrete logarithm [18].

As an example of how quantum computers could solve certain problems exponentially faster than classical computers, David Deutsch and Richard Jozsa introduced in 1992 a simple solution to a specific black box problem [6]. The Deutsch-Jozsa algorithm determined whether a function $f : \{0,1\}^n \rightarrow \{0,1\}$ is *constant,* meaning any input would result in only 0 or only 1, or *balanced,* meaning the probability of getting 0 or 1 is equal. Given an input size of 2, a classical, intuitive algorithm would require 2 queries, while the Deutsch-Jozsa algorithm would only require one. The exponential speedup is more evident as the number of inputs increases: a function with n inputs would require $2^{n-1}+1$ queries with a classical computer in order to determine with full certainty whether the function is constant or balanced, but still would need only one measurement with a quantum computer.

Research on quantum algorithms, however, assume a perfect quantum system that could hold information for long periods of time. The information stored in a qubit is extremely fragile, and could be lost once the qubit interacts with the environment. This phenomenon is known as *quantum decoherence,* and it is considered a limitation in the realization of the quantum computer.

Because of the apparent inevitability of decoherence, it is important to take it into consideration when designing or analyzing quantum algorithms. Multiple methods have already been proposed for determining and measuring the decoherence of some quantum algorithms [1] [19] [3] [5], however we will apply the methods introduced by Chuang et al. and refined by De Jesus for determining the decoherence of the Deutsch-Jozsa algorithm.

The flow of the paper is as follows: first, the problem being solved by Deutsch and Jozsa will be presented, followed by a classical algorithm for solving the problem. The quantum version of the algorithm will then be described and demonstrated. Afterwards, the concept of quantum decoherence will be introduced, and will be related to quantum algorithms through a measure proposed by Chuang et al. in 1995, and refined by De Jesus in 2014. The decoherence of the Deutsch-Jozsa algorithm will then be determined using De Jesus's method, and the results will be expounded on.

2 PRELIMINARIES

In this section, we go over some preliminaries needed to understand the basics of quantum computing and the quantum circuit model, which are needed to derive the decoherence of a quantum circuit. Information in this section is taken from textbooks on quantum computing [16] [17] and lecture notes from R. Jozsa [11].

2.1 Linear Algebra for Quantum Computing

The *Dirac notation*, or the *bra-ket notation*, was introduced by Paul Dirac to make algebraic computations easier for quantum physicists. A vector v in an finite n-dimension vector space is referred to as the *ket* vector, and is notated as

$$|\, v\rangle = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}.$$

Its conjugate-transform is called the *bra* vector, and is denoted as

$$|\, v\rangle^\dagger = \langle v\,| = \left(a_1^* \cdots a_n^*\right)$$

.

To simplify things, *bra* could be seen as the row vector, and *ket* the column vector. Using this notation, the inner product of two vectors ϕ and ψ could be easily notated as $\langle \phi \mid \psi\rangle$, and its outer product $|\,\phi\rangle\langle\psi\,|$.

In quantum mechanics, it is common to use the following vectors as an orthonormal basis for a 2-dimensional Hilbert space:

$$|\,0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |\,1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Furthermore, a *ket* vector in this vector space would be denoted as

$$|\,v\rangle = a|\,0\rangle + b|\,1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}.$$

When working with multiple vector spaces, it is helpful to combine them into a single, larger vector space. For this, the *tensor product* is used, which can be explicitly calculated using the *Kronecker product*.

Definition 2.1. Given an $m \times n$ matrix A and a $p \times q$ matrix B, the Kronecker product is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ & \ddots & \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$$

Definition 2.2. Given two vector spaces V and W of dimensions m and n respectively, the tensor $V \otimes W$ is a vector space of dimensions $m \times n$, and consists of linear combinations of the Kronecker products of the elements of each vector space.

Example 2.3. If we have two vectors $|\,\phi\rangle = a|\,0\rangle + b|\,1\rangle$ and $|\,\psi\rangle = c|\,0\rangle + d|\,1\rangle$, then its corresponding tensor product would be

$$|\,\phi\rangle \otimes |\,\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a\begin{pmatrix} c \\ d \end{pmatrix} \\ b\begin{pmatrix} c \\ d \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}.$$

$$|\,\phi, \psi\rangle = ac|\,00\rangle + ad|\,01\rangle + bc|\,10\rangle + bd|\,11\rangle$$

Example 2.4. The tensor product of two 2×2 matrices is

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \otimes \begin{pmatrix} p & q \\ r & s \end{pmatrix} = \begin{pmatrix} a\begin{pmatrix} p & q \\ r & s \end{pmatrix} & b\begin{pmatrix} p & q \\ r & s \end{pmatrix} \\ c\begin{pmatrix} p & q \\ r & s \end{pmatrix} & d\begin{pmatrix} p & q \\ r & s \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} ap & aq & bp & bq \\ ar & as & br & bs \\ cp & cq & dp & dq \\ cr & cs & dr & ds \end{pmatrix}.$$

Example 2.5. A vector $|\,v\rangle$ tensored with itself n times is notated as $|\,v\rangle^{\otimes n}$.

$$|\,v\rangle^{\otimes 2} = |\,v\rangle \otimes |\,v\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} a \\ b \end{pmatrix}$$

$$= \begin{pmatrix} a\begin{pmatrix} a \\ b \end{pmatrix} \\ b\begin{pmatrix} a \\ b \end{pmatrix} \end{pmatrix} = \begin{pmatrix} aa \\ ab \\ ba \\ bb \end{pmatrix}.$$

A linear mapping between two vector spaces V and W can be described as a function $A : V \longrightarrow W$. This operator may be described as *linear* if applying the function A to the whole tensor space yields the same result as applying it to each element of that space.

Definition 2.6. A linear operator between two vector spaces V and W is defined as a function mapping $A : V \longrightarrow W$ which satisfies the condition

$$A\left(\sum_i a_i\,|\,v_i\rangle\right) = \sum_i a_i A|\,v_i\rangle\,.$$

Example 2.7. The identity operator I on a vector space V is a linear operator, such that for all $|\,v\rangle \in V, I|\,v\rangle = |\,v\rangle\,.$

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Note that if the linear operator A is applied to every basis vector of V, and the result expressed as a linear combination of the basis vectors of W, it is possible to construct a matrix representation of A.

2.2 Postulates of Quantum Mechanics

The most basic unit of quantum information is a quantum bit, or *qubit*. It can be described as a unit vector in a 2-dimensional vector space:

$$|\,\psi\rangle = \alpha\,|\,0\rangle + \beta\,|\,1\rangle$$
$$|\,\alpha\,|^2 + |\,\beta\,|^2 = 1$$

Classical bits are represented by the presence of electricity, and thus may easily be assigned arbitrary values like 1 or 0. The quantum bit, however, is physically represented by quantum mechanical properties like electron spin or photon polarization, and thus the values assigned to it will also exhibit the same quantum phenomena.

Quantum systems with multiple qubits may be represented by a tensor product of all qubits in the system.

Example 2.8. A system of two qubits $\mid \phi\rangle$ and $\mid \psi\rangle$ can be described as

$$\mid \phi, \psi\rangle = \mid \phi\rangle \otimes \mid \psi\rangle = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \alpha \mid 00\rangle + \beta \mid 01\rangle + \gamma \mid 10\rangle + \delta \mid 11\rangle \, .$$

2.2.1 Superposition. Just like how the position of an electron within an atom is shrouded within a cloud of probability, the value of a qubit may exist in a superposition of states until it is observed.

The qubit $\mid \psi\rangle$ may be described as having a probability amplitude α of being in state $\mid 0\rangle$, and a probability amplitude β of being in state $\mid 1\rangle$. Likewise, given the quantum system in example 2.8, the outcome has a probability amplitude α of being in state $\mid 00\rangle$, amplitude β of being in state $\mid 01\rangle$, and so on.

2.2.2 State Evolution. Change in the state of a quantum mechanical system over time may be described by a *unitary* operation on the vector space of the state.

Definition 2.9. An operator U is unitary if $U^\dagger U = UU^\dagger = I$.

If an operator is unitary, the inner product is preserved. Moreover, the definition of a unitary operator implies the existence of its inverse, thus, we may say that unitary operations are *reversible.*

Some examples of unitary operators include the *Hadamard gate (H),* and the *Pauli matrices,* which together form a basis for a 2-dimensional vector space.

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

$$\sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$$

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = X$$

$$\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = Y$$

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = Z$$

Example 2.10. The Hadamard gate, H, puts a qubit in state $\mid 0\rangle$ into an *equal superposition.* By multiplying the Hadamard matrix to $\mid 0\rangle$, the resulting probability amplitudes all become equal.

$$H \mid 0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

For an operator to be applied to multiple qubits, it must be scaled to a higher order by getting its tensor product.

Example 2.11. Scaling the Pauli X gate to be applied to 2 qubits:

$$\sigma_x \otimes \sigma_x = \begin{pmatrix} 0\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 1\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ 1\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 0\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

2.2.3 Measurement. Once a qubit under superposition is observed, it collapses into one of the basis states of the vector space, with a probability distribution directly tied to the amplitudes α and β. Unlike other quantum operations, this is not unitary, and therefore not reversible; once the qubit is measured, the information about its superposition is destroyed. This may be generalized as follows:

Definition 2.12. The probability of measuring an outcome $j = 1, \ldots, n$ in an quantum system with an n-dimension vector space is given by $P(j) = \mid a_j \mid^2$, where a_j is the amplitude associated with outcome j.

It is also possible to only measure certain qubits in the quantum system. Consider the 2-qubit system $\mid \psi\rangle$ in equal superposition:

$$\mid \psi\rangle = \frac{1}{2} \mid 00\rangle + \frac{1}{2} \mid 01\rangle + \frac{1}{2} \mid 10\rangle + \frac{1}{2} \mid 11\rangle$$

Because it is in equal superposition, the probability of getting $\mid 0\rangle$ when measuring the 2nd qubit is 50% (likewise with $\mid 1\rangle$). Suppose that the 2nd qubit collapses to $\mid 0\rangle$. The quantum system then transforms into

$$\mid \psi\rangle' = \frac{1}{\sqrt{2}} \mid 00\rangle + \frac{1}{\sqrt{2}} \mid 10\rangle \, .$$

By removing the 2nd qubit, we get

$$\mid \psi\rangle' = \frac{1}{\sqrt{2}} \mid 0\rangle + \frac{1}{\sqrt{2}} \mid 1\rangle \, .$$

2.2.4 Entanglement. It is possible for the measurement of one qubit to directly affect the measurement of another. If this is the case, then we say the two qubits are *entangled.* Consider the two-qubit system

$$\mid \psi\rangle = \frac{1}{\sqrt{2}} \mid 00\rangle + \frac{1}{\sqrt{2}} \mid 11\rangle \, .$$

If we observe the 2nd qubit to be $\mid 0\rangle$ then the amplitude of the state $\mid 11\rangle$ is diminished entirely, and we are left with the term $\mid 00\rangle$, in which the 1st qubit will collapse to $\mid 0\rangle$ with a 100% probability. Likewise, if we get an outcome of $\mid 1\rangle$, then the 1st qubit will collapse to $\mid 1\rangle$ as well. Thus, the two qubits in the system are *entangled.*

There exists unitary operators that operate on multiple qubits. One example is the *Controlled-NOT* gate (or CNOT), which flips the state of the 2nd qubit only if the 1st qubit reads $\mid 1\rangle$.

Example 2.13. The Controlled-NOT Gate.

$$C_{NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

2.3 Quantum Circuit Model

A quantum algorithm consists of successions of quantum operators applied to a register of qubits. A *quantum circuit* is a visual representation of a quantum algorithm, which is the standard for how quantum algorithms are described and visualized (examples may be seen in Figures 1, 2 and 3).

Each qubit is represented by a horizontal wire with its initial state on the left. Quantum operators, or *gates,* are applied to the qubit, and are executed from left to right.

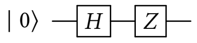

Figure 1: Quantum circuit with one Hadamard gate and one Pauli Z gate

Multiple qubits are represented with multiple wires. Controlled gates that span multiple qubits (such as the CNOT gate) are connected via a vertical wire, while single gates that span multiple qubits may either be separated or combined into one. A meter at the end of a wire denotes a measurement on the indicated qubit.

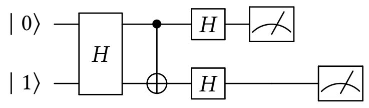

Figure 2: Two-qubit quantum circuit with two scaled Hadamard gates and one CNOT gate. Note that both versions of the scaled Hadamard gate are the same.

Qubits with similar circuitry may be merged. The circuitry of a qubit that is duplicated is indicated by a slash, followed by the number of times it is to duplicated, *n*.

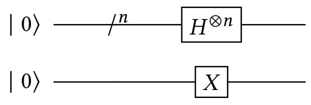

Figure 3: A quantum circuit with $n + 1$ qubits.

3 DEUTSCH-JOZSA ALGORITHM

3.1 The Deutsch Problem and its Classical Solution

The Deutsch problem is as follows. We are given a function $f : \{0,1\}^n \to \{0,1\}$, which is promised to be either of the following:

(1) *Constant;* e.g. the function will map to one value (either 0 or 1) no matter the input; or
(2) *Balanced;* e.g. the chances of getting 0 for any input is equal to the chances of getting 1.

An intuitive solution for this problem would be to query the function f for every input on $\{0,1\}^n$, which would require 2^n queries. However, it is possible to get the answer with only about half of the domain.

If the function is balanced, this implies that half of the domain of f will give an output of 1, and the other half will give 0. This means that if we gather the outputs of $2^{n-1} + 1$ of the possible inputs of f and see that the set of outputs contains both 0's and 1's, then we can conclude that the function is balanced. Otherwise, if the set contains all 0's or all 1's, then the function is constant. If we consider one step of the classical algorithm to be a query on f, we could determine the following:

Lemma 3.1. *The number of steps needed to solve the Deutsch-Jozsa problem is* $2^{n-1} + 1$

Lemma 3.2. *The time complexity for the classical solution the Deutsch-Jozsa problem with full certainty is* $O(2^n)$.

It is also possible to query random values of $\{0,1\}^n$ to f, as suggested by Deutsch and Jozsa, and guess with a bound of error whether the function is constant or balanced. After 2 queries on the function f, the probability of guessing the correct type of function will be about 1/2. However, if we want to solve the problem with full certainty, we need to invoke at least $2^{n-1} + 1$ function calls.

Suppose we have a function $f : \{0,1\}^3 \to \{0,1\}$ defined as $x_1 \oplus x_2 \oplus x_3$. This function is *balanced*, as is shown in the following mapping:

Input	Output
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

Let us also assume that when evaluating the function f, we select completely random input values one at a time. If we select the first two inputs to be 011 and 111, we get outputs 0 and 1, respectively, and thus, may already conclude that the function is balanced. However, if we select inputs 000 and 101, both outputs will be 0, and thus, it is inconclusive.

In the worst case, we may select values 001, 010, 100, and 111, which will all give outputs of 1. This is still inconclusive, and yet we have already used up half of the domain, or $O(2^{n-1})$. However, once we query any other input (like 101, for instance), we get an output of 0, and thus we may conclude that the function is balanced in $O(2^{n-1} + 1)$ steps. Since we are promised that the function is either constant or balanced, if we ever got an output of 1 in the last step, we may have conclude that the function is constant.

3.2 The Quantum Algorithm

Because the function f on its own is not unitary, it cannot be put directly onto the quantum circuit. Thus, a mechanism called a *quantum oracle* is required, which applies the black box function on a single output *ancilla bit*, while keeping the input itself intact. The quantum oracle U_f maps the input $|\, x^{\otimes n}, y\rangle$ to $|\, x^{\otimes n}, y \oplus f(x)\rangle$, ensuring a copy of x is left behind even after the function f is applied.

This algorithm, along with other quantum algorithms such as Shor's factoring algorithm, has been revisited by Cleve et. al. in 1997 to more clearly define the quantum circuit [4]. Given a quantum register of size $n+1$, the Deutsch-Jozsa algorithm is as follows (refer to Figure 4 for the Quantum circuit diagram):

(1) Initialize each qubit to $|\, 0\rangle$, and the ancilla bit to $|\, 1\rangle$.
(2) Apply the *Hadamard Gate* H to each qubit.
(3) Apply the oracle function U_f to each qubit.
(4) Apply H to each of the first n qubits.
(5) Perform a measurement on each of the n qubits.

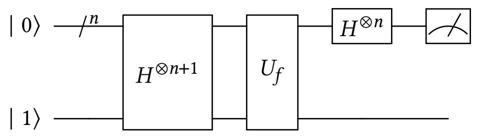

Figure 4: Quantum circuit for the Deutsch-Jozsa Algorithm

The transformation of a quantum register with n qubits throughout is algorithm is summarized below. First the Hadamard gate is applied to all qubits, which puts all n qubits in equal superposition, and the ancilla bit in state $|\, 0\rangle - |\, 1\rangle$:

$$\psi_1 : |\, 0\rangle^{\otimes n} |\, 1\rangle \xrightarrow{H^{\otimes(n+1)}} \sum_{x\in\{0,1\}^n} |\, x\rangle\,(|\, 0\rangle - |\, 1\rangle),$$

Next the oracle gate U_f is applied, which maps $|\, x^{\otimes n}, y\rangle$ to $|\, x^{\otimes n}, y \oplus f(x)\rangle$:

$$\psi_2 : \sum_{x\in\{0,1\}^n} |\, x\rangle\,(|\, 0\rangle - |\, 1\rangle) \xrightarrow{U_f} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |\, x\rangle\,(|\, 0\rangle - |\, 1\rangle)$$

Finally, applying the last Hadamard transform to all but the ancilla bit, the system evolves into

$$\psi_3 : \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |\, x\rangle\,(|\, 0\rangle - |\, 1\rangle)$$

$$\Big\downarrow H^{\otimes n}$$

$$\sum_{x,y\in\{0,1\}^n} (-1)^{f(x)\oplus(x\cdot y)} |\, y\rangle\,(|\, 0\rangle - |\, 1\rangle).$$

To determine whether the function is constant or balanced, we must look at the probability of measuring $|\, 0^{\otimes n}\rangle$:

$$P(|\, 0^{\otimes n}\rangle) = \left| \sum_{x\in\{0,1\}^n} \frac{(-1)^{f(x)}}{2^n} \right|^2 .$$

From here, it can be deduced that if the function is constant, the state will be $(-1)^{f(0^{\otimes n})} |\, 0^{\otimes n}\rangle\,(|\, 0\rangle - |\, 1\rangle)$, whereas if the function is balanced, the amplitude of the state $|\, 0^{\otimes n}\rangle$ will be zero in the first place. By expounding on these two possible cases, the following Lemma could be demonstrated:

LEMMA 3.3. *The Deutsch-Jozsa problem could be solved with certainty with only 1 measurement [4].*

Case 1: Constant. A constant function means that $f(x)$ will yield either only 0 or only 1 for all inputs x. This means that the term $(-1)^{f(x)}$ does not change with input x, which means that it will exhibit *constructive interference*. Getting its summation over all values of x will eventually yield a value of 1.

Case 2: Balanced. A balanced function means that for all inputs x, $f(x)$ will yield 0 half the time, and 1 otherwise. This means that for all values of x, the term $(-1)^{f(x)}$ will yield -1 half of the time, and 1 otherwise, meaning it will exhibit *destructive interference*. Getting its summation over all values of x will eventually yield a value of 0. Therefore, we could determine with full certainty whether the function is constant or balanced with only a single measurement.

3.3 Uses of the Algorithm

Although the algorithm was designed primarily to demonstrate the potential speedup of quantum computers over classical computers, it inspired others to design more sophisticated algorithms, like the Quantum Fourier transform and Shor's factoring algorithm. Nonetheless, the algorithm itself has been used as baselines for other solutions.

Nagata and Nakamura in 2015 used the algorithm as an intermediary for quantum key exchange [15]. Alice would send an $N+1$-qubit register in superposition to Bob, who would then apply the transformation U_f on the register and return the N qubits. Alice would then be able to determine whether the shared function was constant or balanced.

Samir Lipovaca in 2009 was able to use the Deutsch-Jozsa algorithm to partition an array and apply the specified function to each partition [13].

Stephan Gulde et. al. in 2003 implemented the Deutsch-Jozsa algorithm on an ion-trap quantum computer in order to test the system's reliability [10]. The evolution of the quantum system throughout the algorithm was determined by measuring the probability of detecting the qubit in a $D_{5/2}$ electronic state.

4 QUANTUM DECOHERENCE

Research on quantum algorithms assume a perfect quantum system that could hold information for long periods of time. However, the information stored in a qubit is extremely fragile, and could be lost once the qubit interacts with the environment. This phenomenon is known as *quantum decoherence,* and it is a limitation in the realization of the quantum computer. While developments in preventing environment-induced decoherence focus on isolating the qubits, it was also shown that decoherence may occur even when said qubits are perfectly isolated [9]. This therefore shows the importance of taking decoherence into consideration when designing quantum algorithms.

While the study of Gulde et. al. showed favorable results, it was stated that there were slight deviations between the actual result and what was expected. The authors attributed this deviation to the decoherence of the quantum system.

One other study done by Cao et. al. in 2018 highlights a possible imperfection of the Deutsch-Jozsa algorithm. By considering the actual implementation of the quantum oracle, it was shown that the results of it being applied to a qubit in superposition are inconsistent with the assumptions made in the design of the original algorithm [2]. If the physical realization of the oracle function U_f is faulty, the algorithm could yield imperfect results. This inconsistency could be one possible cause of decoherence for the implementation of the algorithm.

4.1 Background and History

Heinz-Deiter Zeh enumerated and defined some forms of decoherence [21]. The first one is decoherence due to measurement or observation. The second one is labelled as virtual decoherence, which happens in the microscopic level. The third type of decoherence is real decoherence, which is caused by the qubit's interactions with the environment. Real decoherence is unavoidable and non-unitary.

Zurek's study on decoherence in 1991 revolved around the flaws of Neils Bohr's Copenhagen interpretation of quantum mechanics and took into consideration Hugh Everett III's many worlds interpretation [22]. Bohr argued that a classical measuring device is needed to collapse and measure a quantum state, thus implying a fine line between quantum and classical computing. On the other hand, Everett insisted that the entire universe is continuously evolving via tha Schrödinger equation, and implied that the boundary between classical and quantum computing might have been more inclusive than previously thought. This dillema is also known as the *quantum measurement problem.* Zurek showed how, using Von Neumann's hypothesized process of nonunitary reduction, the addition of the environment in the simple quantum measurement system could lead to a loss of excess information (decoherence) which allows the quantum state to be interpreted classically.

Another study on the quantum measurement problem was done by Eric Galapon in 2016 [9]. It was shown that, given a simple measurement system consisting of a probe and a pointer, there exists a finite time to achieve exact decoherence and orthogonality without coupling with the environment. While other studies of decoherence made use of the environment-induced decoherence theory (EIDT), the system studied here had no correlation with the environment whatsoever and only consisted of one internal degree of freedom. It was determined that there exists a *decoherence time* $\Delta\tau_D$ when the system loses all coherence, and an *orthogonality time* $\Delta\tau_O$ when all pointer states become mutually orthogonal. By comparing these times, we could see that $\Delta\tau_O > \Delta\tau_D$, and so for $\Delta\tau_D < \Delta\tau < \Delta\tau_O$, the measurement reading would have an ambiguous outcome.

The apparent inevitability of quantum decoherence presents a major roadblock in the realization of the quantum computer. Many error-correcting methods have been studied in order to circumvent erroneous quantum computers, some of which have been summarized by Kempe in 2007 [12]. Some methods hold similarities with classical error-correcting approaches, such as hamming codes and bit allocation. Others deal with the actual realization of a decoherence-free quantum computer in order to avoid errors altogether.

A study by Flores and Galapon in 2016 explored a scheme to maintain entanglement between two qubits by adding additional qubits [8]. By considering the exact evolution of multiple qubits in a common reservoir, it was shown that increasing the number of qubits N preserves the entanglement of two initially-entangled qubits. One possible error correction scheme would be to add a qubit before the decoherence time of the isolated system, $\Delta\tau_D$, is reached.

4.2 Decoherence Measures for Quantum Algorithms

Due of the apparent inevitability of quantum decoherence, researchers have been taking this into consideration in the design and analysis of quantum algorithms. Some methods and papers are listed below.

A perturbative approach introduced by Azuma [1] assumes that each qubit in each step of the algorithm interacts independently with the environment, giving an error of σ_z with a probability p. The limit of this is then taken as the number of qubits n reaches infinity, thus providing an upper bound for the amount of error an algorithm could have without losing vital information.

Another method proposed by Walls and Milburn [20] is a master-equation approach to determine the effect of dissipation on a macroscopic level. Here they explore the use of Markovian master equations based on the effects of damping on a harmonic oscillator in order to determine the relationship between the decoherence and the quantum state of the system.

Utsunomiya et. al. analyzed the decoherence of some quantum algorithms by studying the collective, rather than individual, interactions between the qubits and its environment [19]. The proposed method of analyzing different basis states in order to determine the relaxation rate of the algorithm was executed on the Deutsch-Jozsa algorithm, as well as Grover's algorithm for data search. It was shown that minor changes in the sequence of gates (as well as the choice of initial states of the qubits) could help avoid unstable (*superradiant*) states, favor stable (*subradiant*) states, and decrease the effect of collective decoherence altogether.

Chuang et. al., along with Shor himself, explored the effects of decoherence on Shor's factoring algorithm [3]. By introducing the environment as a separate qubit $| \epsilon\rangle$ (defined as the "degrees of freedom from the environment"), the quantum states of Shor's factoring algorithm could be rewritten as:

$$| \psi_1\rangle = \frac{1}{\sqrt{L}} \sum_{a=0}^{L-1} | a, 0\rangle \times | \epsilon\rangle$$

$$| \psi_2\rangle = \frac{1}{\sqrt{L}} \sum_{a=0}^{L-1} | a, x^a modN\rangle \times | \epsilon_a\rangle$$

Because there is no adverse effect from decoherence for the second state, the study focused on the effects on the first state, and thus the following reduced density matrix could be obtained, which could be applied to any quantum register of length L:

$$\rho_{red} = \sum_{a=0}^{(L-1)'} \sum_{a=0}^{(L-1)'} [1 - \beta_{aa'}] \mid a\rangle \langle a' \mid$$

As long as a and a' are diagonal to the pointer basis, the term $(1 - \beta_{aa'})$ will remain constant, and thus may be approximated as follows:

$$1 - \beta_{aa'} \approx \exp[-\xi(a \text{ XOR } a')],$$

where ξ is a decoherence parameter that will depend on the quantum computer itself. This measurement will result in a probability distribution where originally impossible outcomes would have some nonzero probability, and the originally nonzero probabilities will have a smaller value.

Furthermore, the approximation may be simplified to $\delta_{aa'} + (1 - \delta_{aa'})\beta$, with $\beta = 0$ representing complete coherence, and $\beta = 1$ representing a state of complete decoherence.

Chuang et al. defined the value α as the amount of information lost per qubit per operation, assuming an environment that is Markovian in nature, and a relationship between α and β that is established as $\beta \approx 1 - e^{n_{op}\alpha L}$, where n_{op} is the operation count of the quantum algorithm. This is used to further simplify the definition of decoherence in a quantum algorithm.

Naturally, decoherence would increase the number of times needed to run the program in order to get the desired result. The required number of trials needed to get a desirable result would be increased from $O(L)$ to $O(L/(1 - \beta))$. In terms of α, the number of trials would be:

$$n_{trial} = O\left(\frac{L}{e^{n_{op}\alpha L}}\right)$$

In computing for this, the value for L could be derived from the probability P of getting the correct answer. For Shor's factoring algorithm, the probability of getting a factor for N for each execution is $O(logN)$ for a perfect system (i.e. a system with no decoherence). Introducing decoherence will raise that number to $O\left(\frac{logN}{1-\beta}\right)$.

In order for a quantum algorithm to be advantageous over its classical counterpart, the value n_{trial} must not be greater than the performance of the classical algorithm. Thus, these two values are compared. For Shor's algorithm, this comparison is as follows:

$$n_{trial} \sim \frac{L}{1 - L^2\alpha}$$

$$\frac{L}{1 - L^2\alpha} \leq 2^{L^{1/3}}$$

From here, the value α may be obtained in terms of L:

$$\alpha < \frac{1}{L^{8/3}}$$

Due to its simplicity as compared to other approaches, the method introduced by Chuang et. al. was chosen by De Jesus for his paper, where he quantified and analyzed the decoherence of some quantum algorithms. Moreover, its parameters, such as the number of trials needed to get to a solution, are already being used in the analysis of computational complexity.

De Jesus described α as the *unit decoherence* of the quantum algorithm, which acts as a boundary for the algorithm will perform better than its classical counterpart. An alternative method of analysis was proposed, which was deemed more suitable for computational analysis, and was applied to some well-established quantum algorithms such as Shor's factoring algorithm and Grover's search algorithm. It was also shown that distributed counterparts of quantum algorithms could potentially increase the tolerance for decoherence [5].

The step-by-step description on deriving α is as follows:

(1) Acquire the following values:
   - n_{qop}, the number of operations in the qubit level,
   - $n_{op(\text{classical})}$, the number of operations needed to solve the problem using the fastest known classical solution, and
   - P, the probability of getting the correct result at the end of the algorithm (in terms of L).

(2) Compute for $n_{trial} = O\left(\frac{e^{n_{qop}\alpha}}{P}\right)$

(3) Compare n_{trial} and $n_{op(\text{classical})}$ $\left(n_{trial} < n_{op(\text{classical})}\right)$.

Note that instead of defining n_{op} as a function of L, De Jesus chose to manually *count* the number of quantum operations by looking at the actual quantum circuit. Thus, the relationship between β and α is changed to $\beta = 1-e^{\alpha n_{qop}}$, and the number of trials to $e^{n_{qop}\alpha}/P$. This changes $\alpha = \frac{1}{L^{5/3}}$ for Shor's algorithm.

5 RESULTS AND ANALYSIS

5.1 Decoherence of the Deutsch-Jozsa Algorithm

Here we apply the aformentioned method to the Deutsch-Jozsa algorithm.

THEOREM 5.1. *For the Deutsch-Jozsa algorithm, the allowable decoherence per operation per qubit α is bounded by* $\frac{L}{3L-1}$.

Proof. First, we find $n_{op(classical)}$. This is given in Lemmas 3.2 and 3.1.

$$n_{op(classical)} = 2^{L-1} + 1 = O\left(2^{L}\right) \tag{1}$$

Next, we consider the quantum algorithm. The sequence of operations is to apply a Hadamard gate to all L qubits, then apply the oracle gate U_f (which operates on all qubits), then finally, another Hadamrd gate to all but one qubit. Thus:

$$n_{qop} = L + L + (L-1) = 3L - 1 \tag{2}$$

It was stated in Lemma 3.3 that only one measurement was necessary to get the answer with certainty. Thus:

$$P = 1 \tag{3}$$

We then compute for n_{trial} as follows:

$$n_{trial} = O\left(\frac{e^{\alpha n_{qop}}}{P}\right) = O\left(e^{\alpha(3L-1)}\right) \tag{4}$$

To get a relationship between α and L, we compare $n_{trial} < n_{op(classical)}$:

$$e^{\alpha(3L-1)} < 2^{L} < e^{L}$$

$$\alpha(3L-1) < L$$

$$\alpha < \frac{L}{3L-1} \tag{5}$$

$\square$

5.2 Boundary of Input Size N

The value α may also be used to derive the largest input that a certain quantum computer can take without decoherence. In a zero-temperature environment, the off-diagonal terms of the density matrix may be reduced by the amount [3]

$$\alpha \sim \frac{\mu^2\eta}{2\pi}\left[-C - \frac{\pi^2}{4} + log\left(\frac{\Delta}{\Lambda}\right)\right].$$

Theorem 5.2. *For the Deutsch-Jozsa algorithm, the largest input size is bounded by* $\frac{\mu^2\eta}{3\mu^2\eta - 2\pi}$.

Expressing equation (5) in terms of L, we get the relation

$$L < \frac{\alpha}{3\alpha - 1}. \tag{6}$$

It is already established that the register size L is equal to the length of the input string. With N being the size of the input string, and by substituting α with the decrease in the off-diagonal term of a zero-temperature system, we get

$$L < \frac{\frac{\mu^2\eta}{2\pi}}{3\frac{\mu^2\eta}{2\pi} - 1}$$

$$N < \frac{\mu^2\eta}{3\mu^2\eta - 2\pi} \tag{7}$$

5.3 Implementation

In order to apply the acquired unit decoherence α to the algorithm, a simulator was implemented using the NumPy package of Python. The program takes the function f as its input, and by breaking down the quantum circuit into its basic matrix forms, determines whether the given function was balanced or constant. Although one could easily solve the problem by looking at the input file, the aim is to effectively simulate the evolution of the quantum state of the system when running the algorithm; this includes the assumption that the quantum computer has restricted access to the function itself.

A helper function was designed in order to implement the quantum oracle U_f, which takes a function mapping for f as its input and returns a matrix version of U_f. The design draws inspiration from [14]. The sample output may be seen in Figure 5, while a snippet of the Python code for the oracle function may be seen in Figure 6.

```
CASE 1: BALANCED
CASE 2: BALANCED
CASE 3: CONSTANT
CASE 4: CONSTANT
CASE 5: BALANCED
```

Figure 5: Sample output of the Deutsch-Jozsa simulator

```
def generate_oracle(f_map, num_qubits):
    U = np.zeros((2**num_qubits, 2**num_qubits))

    for input_state in range(2**num_qubits):
        input_string = input_state >> 1
        output_qubit = (input_state & 1)^(f_map[input_string])
        output_state = (input_string << 1) + output_qubit
        U[input_state, output_state] = 1
    return U}
```

Figure 6: Python code of oracle function generator

5.3.1 Adding Decoherence. Because the Oracle function requires access to the function f, is may be considered as the weakest point of the algorithm, and therefore could be the most prone to decoherence. Moreover, by the definition of the problem, the function should either all map to a single value, or the output generated should be evenly split between 1 and 0. Therefore, if the function mapping passed onto `generate_oracle()` contains one single error, the final output will lose confidence. This loss in confidence could be attributed to the algorithm's decoherence.

To test this, the input to the program was manipulated to include decoherence. The register size was set to 6 qubits, with a 5-bit function f. The following inputs were passed:

(1) Constant-0 function (all inputs map to 0)
(2) Constant-0 function with 3 errors
(3) Balanced function (constant probability)
(4) Balanced function with 4 errors
(5) Balanced function (scattered probability)
(6) Balanced function with 4 errors

Constant-0 Function (Figures 7 and 8). Without decoherence, the probability of getting $|\,0^{\otimes n}\rangle$ is at 100%, meaning there is full confidence in getting the right result. However, by modifying 3 out of 32 bits in the function mapping, the probability of getting the right answer diminishes to almost 70%, while the probability of getting other states increased ever so slightly.

Balanced Function 1 (Figures 9 and 10). Similar to the previous test case, the probability of getting $|\,0^{\otimes n}\rangle$ is at 0%, meaning there is no chance of error. Once decoherence was introduced, the amplitude of $|\,0^{\otimes n}\rangle$ increased ever so slightly, indicating a loss in certainty.

Balanced Function 2 (Figures 11 and 12). Although the probability distribution without decoherence seems scattered and chaotic, the amplitude of $|\,0^{\otimes n}\rangle$ is still zero, meaning there is no chance of error. Modifying 4 bits, however, increases the amplitude to about 7%.

5.3.2 Error. The most notable effect of decoherence on a quantum algorithm is the decrease in probability of getting the correct answer. For the Deutsch-Jozsa algorithm, it is the discrepancy in the amplitude of $|\,0^{\otimes n}\rangle$. We expect the amplitude to be 1 if the function is constant, and 0 if it is balanced. Thus, if the amplitude is anything in between 0 and 1, we can attribute it to decoherence, and define the discrepancy as the *error* of the system.

Definition 5.3. The error ϵ of the quantum system after running the Deutsch-Jozsa algorithm is $P(|\,0^{\otimes n}\rangle)$ if the function is balanced, and $1 - P(|\,0^{\otimes n}\rangle)$ if it is constant.

To get a relationship between ϵ and α, data had to be taken from a single input that consistently lost coherence per test run. A constant 6-bit function passed onto the program was made to slowly lose 1 bit of coherence per test run, and the error ϵ was measured per run. A graph was generated from the result (see Figure 13).

5.4 Analysis

It is understood through its derivation that if the quantum algorithm's decoherence exceeds α, it is just as efficient as its classical counterpart, if not less efficient. In the case of the Deutsch-Jozsa algorithm, once the value of α exceeds $\frac{L}{3L-1}$, the quantum algorithm only becomes just as efficient as its brute force counterpart.

A graph of the function $\alpha(L)$ was generated (see Figure 14). It was found that the limit of the unit decoherence α reaches the value $\frac{1}{3}$ as the register size L increases.

Once α reaches a certain value, it is only just as efficient as its classical counterpart. Meaning, with enough decoherence, the time complexity of the quantum algorithm would approach the time complexity of the classical algorithm; in the case of the Deutsch-Jozsa algorithm, $O(1)$ would eventually become $O(2^n)$. Looking at Figure 14, this occurs when α approaches 1/3, for very large registers.

The vertical line in Figure 13 indicates the boundary for α for very large registers; any value to the left of the line yields better efficiency than its classical counterpart. The maximum error ϵ for a system exhibiting the maximum allowable amount of decoherence is roughly 55.56%. This means that at $\alpha \approx \frac{1}{3}$, the algorithm will arrive at the wrong result half of the time, and yet it will still be more efficient than its classical counterpart.

At a glance, the value of α for this algorithm would seem more tolerant to larger register sizes than those of Shor's factoring algorithm and Grover's search algorithm [5]. The values are tabulated in Table 1. A graph comparing the α functions of the 3 algorithms could be found in Figure 15.

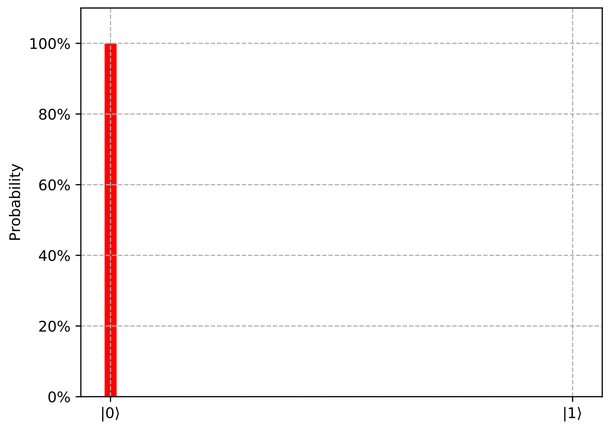

Figure 7: Probability distribution of a Constant-0 function without decoherence.

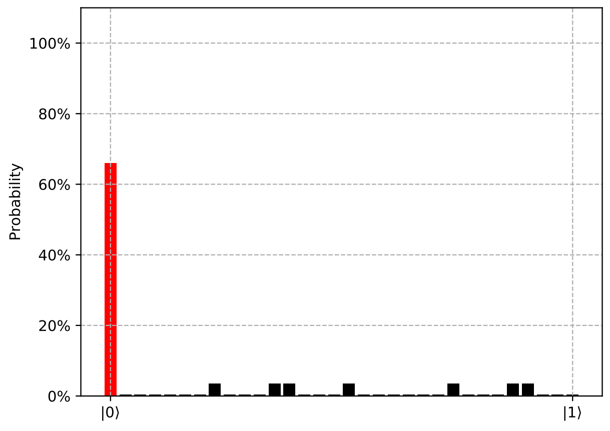

Figure 8: Probability distribution of a Constant-0 function with 9% decoherence. Notice how the probability of measuring $|\,0\rangle$ is diminished while the probabilities of measuring other states increase.

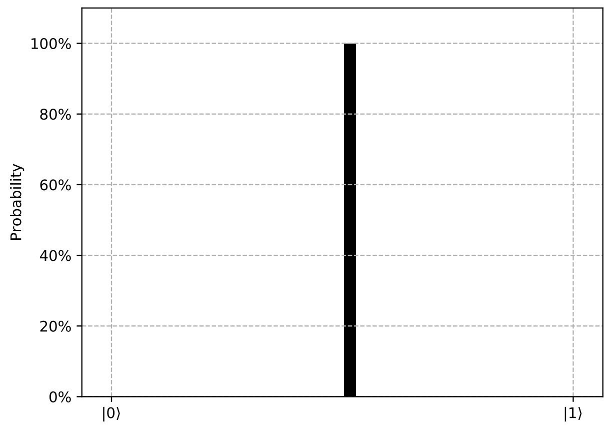

Figure 9: Probability distribution of a balanced function without decoherence.

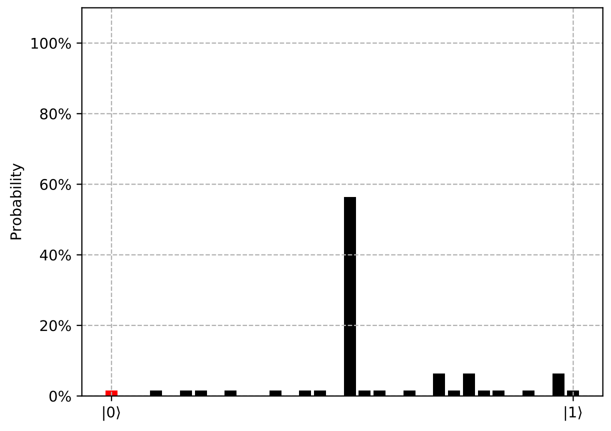

Figure 10: Probability distribution of a balanced function with 12.5% decoherence. Note the presence of the red bar at $|0\rangle$.

Table 1: α boundaries of the 3 quantum algorithms.

Algorithm	*Deutsch-Jozsa*	*Shor*	*Grover*
α **Boundary**	$\frac{L}{3L-1}$	$\frac{1}{L^{5/3}}$	$\frac{1}{2^{L/2}}$

With a register size of 1 qubit, the tolerance for decoherence is much greater for Shor's algorithm than it is for the Deutsch-Jozsa algorithm. However, Shor's factoring algorithm requires much more than 1 qubit in order to be effective; the amount of qubits needed depends on the size of the number being factored [5]. Increasing the register size to 4, the Deutsch-Jozsa algorithm becomes more tolerant.

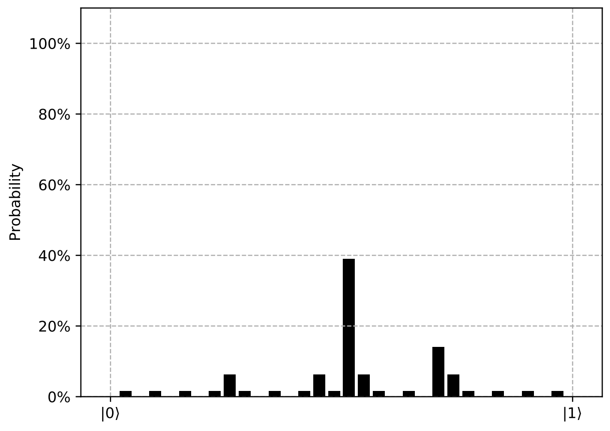

Figure 11: Probability distribution of a balanced function without decoherence. Despite the seemingly chaotic probability distribution, the probability of measuring $|0\rangle$ is still 0.

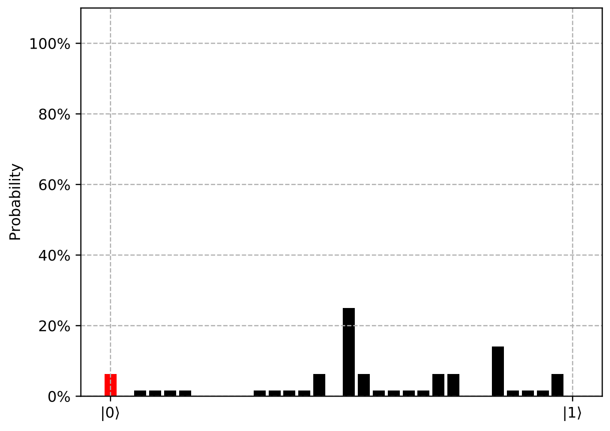

Figure 12: Probability distribution of a balanced function with 12.5% decoherence.

Similarly, Grover's algorithm requires a register size L which depends on the size of the search space, and thus is impractical for registers of size 1 or 2. With a register size of 4, the Deutsch-Jozsa algorithm also becomes more tolerant.

Moreover, as the register becomes infinitely larger, the α functions of Shor's and Grover's algorithms approach 0; meaning there approaches a register size where there is virtually no tolerance for decoherence. The $O(L^3)$ running time of Shor's factoring algorithm [5] would become $O(e^{L^{-1/3}})$ even with very little decoherence. On the other hand, the α function of the Deutsch-Jozsa algorithm approaches $\frac{1}{3}$ as L becomes infinitely larger, meaning there will always be some tolerance for decoherence.

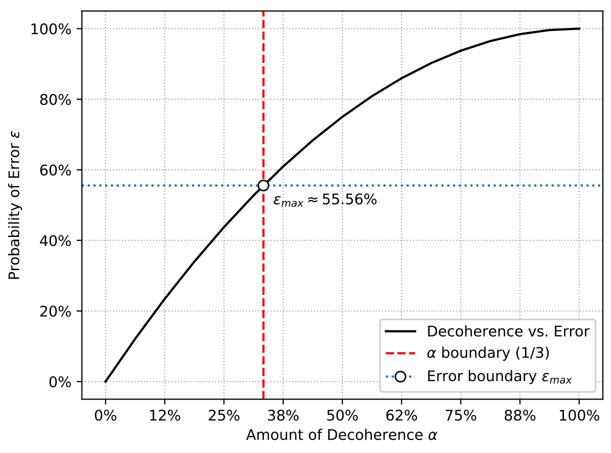

Figure 13: Probability of getting a wrong measurement, with respect to the amount of decoherence in the system. Note that anything to the left of the dashed line will be *faster* than a classical algorithm, while everything to the right will be slower or just as efficient.

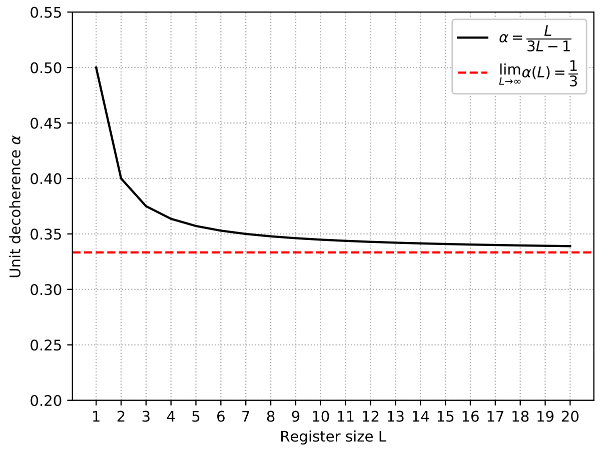

Figure 14: Visualization of the decoherence function α. Note the limit of overall decoherence, 1/3.

6 CONCLUSION

Applying the method proposed by Chuang et. al. and De Jesus, we were able to obtain the unit decoherence α for the Deutsch-Jozsa algorithm. Moreover, we defined the error ϵ of the quantum system, and derived the largest amount of error allowable for the algorithm to be more efficient than its classical counterpart. We were also able to obtain the largest input size N based on the realization of the quantum computer. The results are tabulated in Table 2.

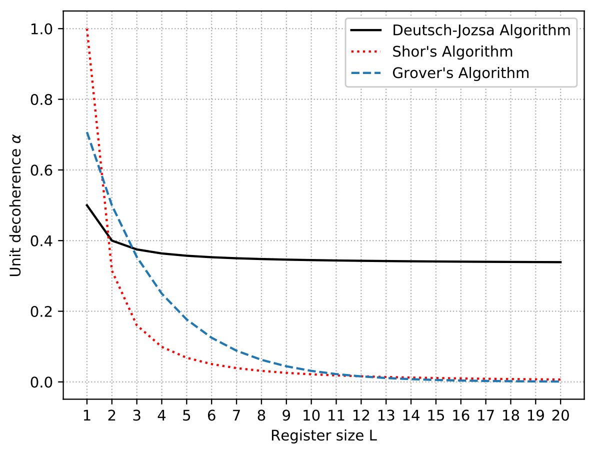

Figure 15: Visualization of the decoherence function α of the Deutsch-Jozsa algorithm over Shor's and Grover's algorithms.

Table 2: Summary of results.

Decoherence	$\alpha < \frac{L}{3L-1}$
Maximum Error	$\epsilon < 55.56\%$
Largest Input Size	$N < \frac{\mu^2 \eta}{3\mu^2 \eta - 2\pi}$

As the register size L increases, the value α approaches $\frac{1}{3}$, meaning that the Deutsch-Jozsa algorithm can still tolerate some amount of decoherence even on infinitely large inputs. It could also be said that the Deutsch-Jozsa algorithm is more tolerant to decoherence than Shor's factoring algorithm and Grover's search algorithm, due to the latter two approaching 0 on large L.

These results imply that even if the algorithm returns the wrong answer half of the time, it will still be more efficient than its classical counterpart. It is still advantageous to repeatedly run the algorithm again and again until the correct answer is obtained than it is to execute the brute-force classical algorithm, so long as the error does not exceet 55.56%.

Further research recommendations include applying the method to other quantum algorithms in order to construct a larger database of decoherence values, which could eventually lead to the establishment of a hierarchy based on α. Experimentation could also be done with an actual quantum computer in order to find any correlations between the values. Finally, error-correcting methods, such as those proposed in [12], as well as procedures for preserving coherence, like the one proposed by Flores and Galapon [8], may be applied to the algorithm in order to see how it affects the unit decoherence α.

ACKNOWLEDGEMENTS

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

L. del Rosario would like to thank the DOST-SEI for supporting his studies through their undergraduate scholarship.

REFERENCES

[1] Hiroo Azuma. 2002. Decoherence in Grover's quantum algorithm: Perturbative approach. *Phys. Rev. A* 65 (Apr 2002), 042311. Issue 4. https://doi.org/10.1103/PhysRevA.65.042311

[2] Z. Cao, J. Uhlmann, and L. Li. 2018. Analysis of Deutsch-Jozsa Quantum Algorithm. *IACR Cryptology ePrint Archive* 2018 (2018), 249.

[3] I. L. Chuang, R. Laflamme, P. W. Shor, and W. H. Zurek. 1995. Quantum Computers, Factoring, and Decoherence. *Science* 270, 5242 (1995), 1633–1635. https://doi.org/10.1126/science.270.5242.1633 arXiv:http://science.sciencemag.org/content/270/5242/1633.full.pdf

[4] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. 1998. Quantum Algorithms Revisited. *Proceedings: Mathematical, Physical and Engineering Sciences* 454, 1969 (1998), 339–354. http://www.jstor.org/stable/53169

[5] Brian Kenneth de Jesus. 2014. Quantum Decoherence and Computational Complexity. (May 2014).

[6] D. Deutsch and R. Jozsa. 1992. *Rapid Solution of Problems by Quantum Computation*. Technical Report. Bristol, UK, UK.

[7] Richard P. Feynman. 1982. Simulating physics with computers. *International Journal of Theoretical Physics* 21, 6-7 (1982).

[8] M.M. Flores and E.A. Galapon. 2015. Two qubit entanglement preservation through the addition of qubits. *Annals of Physics* 354 (2015), 21 – 30. https://doi.org/10.1016/j.aop.2014.11.011

[9] Eric A. Galapon. 2016. Internal one degree of freedom is sufficient to induce exact decoherence. *EPL (Europhysics Letters)* 113, 6 (2016), 60007. http://stacks.iop.org/0295-5075/113/i=6/a=60007

[10] Stephan Gulde, Mark Riebe, Gavin P. T. Lancaster, Christoph Becher, Jürgen Eschner, Hartmut Häffner, Ferdinand Schmidt-Kaler, Isaac L. Chuang, and R. Blatt. 2003. Implementation of the Deutsch–Jozsa algorithm on an ion-trap quantum computer. *Nature* 421 (2003), 48–50.

[11] R. Jozsa. 2014. Quantum Computation Prerequisite Material. https://people.maths.bris.ac.uk/~csxam/teaching/qc2018/QCPrerequisites.pdf

[12] Julia Kempe. 2007. *Approaches to Quantum Error Correction*. Birkhäuser Basel, Basel, 85–123. https://doi.org/10.1007/978-3-7643-7808-0_3

[13] Samir Lipovaca. 2009. Using the Deutsch-Jozsa algorithm to determine parts of an array and apply a specified function to each independent part. (08 2009).

[14] E. Malykh. 2017. quantum-python: Small quantum algorithms framework written in Python with examples. https://github.com/meownoid/quantum-python.

[15] Koji Nagata and Tadao Nakamura. 2015. The Deutsch-Jozsa Algorithm Can Be Used for Quantum Key Distribution. *OALib* 2, 8 (2015), 1–6. https://doi.org/10.4236/oalib.1101798

[16] M. A. Nielsen and I. L. Chuang. 2011. *Quantum Computation and Quantum Information: 10th Anniversary Edition* (10th ed.). Cambridge University Press, New York, NY, USA.

[17] R. Portugal. 2013. *Quantum Walks and Search Algorithms*. Springer Publishing Company, Incorporated. 195–214 pages.

[18] P. W. Shor. 1994. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (SFCS '94)*. IEEE Computer Society, Washington, DC, USA, 124–134. https://doi.org/10.1109/SFCS.1994.365700

[19] Shoko Utsunomiya, Cyrus P. Master, and Yoshihisa Yamamoto. 2007. Algorithm-based analysis of collective decoherence in quantum computation. *J. Opt. Soc. Am. B* 24, 2 (Feb 2007), 198–208. https://doi.org/10.1364/JOSAB.24.000198

[20] D. F. Walls and G. J. Milburn. 1985. Effect of dissipation on quantum coherence. *Phys. Rev. A* 31 (Apr 1985), 2403–2408. Issue 4. https://doi.org/10.1103/PhysRevA.31.2403

[21] H. D. Zeh. 2003. *Basic Concepts and Their Interpretation*. Springer Berlin Heidelberg, Berlin, Heidelberg, 7–40. https://doi.org/10.1007/978-3-662-05328-7_2

[22] Wojciech H. Zurek. 2003. Decoherence and the transition from quantum to classical – REVISITED. arXiv:arXiv:quant-ph/0306072