Depolarizing Channel on the Deutsch-Jozsa and Quantum Counting Algorithm using Qiskit

Pio L. Fortuno, III
Luis Gabriel Q. del Rosario, Henry N. Adorna
plfortuno@up.edu.ph,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

A novel method based on the Qiskit[1] Python framework is used on the Deutsch-Josza and quantum counting algorithms. This method uses a classical method to randomly place *depolarizing channels* in the form of X, Y, or Z quantum gates inside the quantum circuit, randomizing the locations for each trial. This method bypasses the quantum circuit implementation of the depolarizing channel by implementing the effect of the channel should it take affect on the circuit, instead of using its superposition. The method has shown that the superposition of measurements of the algorithm does not necessarily correspond to the actual results of each trial of the algorithm, as well as the effects of decoherence with respect to error.

KEYWORDS

Quantum Algorithms, Deutsch-Josza, Quantum Counting, Qiskit, Quantum Decoherence, Depolarizing Channel

1 INTRODUCTION

Moore's law states that the overall processing power of computers will double every two years. For classical computers, it means shrinking the transistors used so chips can store more of them and have more processing power. But as transistors get smaller, they may exhibit behavior such as quantum tunneling. The idea of using these quantum mechanical behaviors for computational power emerged[9].

Quantum computing differs from classical computing by using *qubits* instead of classical bits. Like bits, qubits can store information such as a 0 or 1, but can also store a value in-between as a *superposition*. Quantum algorithms can manipulate these qubits using *quantum gates*, changing their states, allowing for computations with lower time complexity compared to classical algorithms[11, 13], such as the *quantum counting algorithm*.

Some studies about quantum computing assume a perfect quantum system with no *decoherence*. Decoherence is an environmental factor that affects quantum systems, negating some potential power of quantum algorithms[15]. Studies made on decoherence discover the effect of decoherence on algorithms and their robustness [6, 8], or find methods to limit the effects of decoherence[10].

This paper will find the effects of decoherence on the *quantum counting algorithm* using a novel method. This method uses the Python framework Qiskit[1], used for simulating quantum algorithms.

In Section 2, we provide necessary concepts and notions related to the objective of the paper. We present in Section 3 the methodology used in the experiments and its implementation. Finally we present the results of the investigation and experiments in Section 4. We provided in the appendix the graphs and plots of the simulations done in the experiments.

2 PRELIMINARIES

The following sections use *bra-ket notation*, otherwise known as *Dirac notation*. This notation uses $\langle bra|$ and $|ket\rangle$ to represent a row and column vector respectively.

2.1 Quantum bits

Quantum bits, or *qubits* for short, are the fundamental unit of information to quantum computation and information, much like *bits* are the fundamental unit of information for classical computation and information. Mathematically, qubits are a state, like how a bit is either 0 or 1. Qubits may be in the state $|0\rangle$ or $|1\rangle$, but may also be in a *linear combination* or *superposition*:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Numbers α and β are complex numbers, and represent the state of the qubit in a complex vector state. The states $|0\rangle$ and $|1\rangle$ are the *computational basis states*, and form the orthonormal basis for the vector space.

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}, \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix}\alpha\\\beta\end{bmatrix}$$

A qubit can be measured, like a classical bit, but the information examined would not be the quantum state of the qubit. When examined, the qubit collapses into either a 0 or 1, with probabilities $|\alpha|^2$ and $|\beta|^2$ respectively. The probabilities must sum to one, so $|\alpha|^2 + |\beta|^2 = 1$. Once a qubit is measured, the information regarding its superposition is lost.

A system with multiple qubits are similar. A two qubit system has four *computational basis states*, $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. The two qubits can also exist in a superposition of those four states, with the state vector being

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

The measurement result of the two qubit system where $x = 00, 01, 10$, or 11 is with probability $|\ \alpha_x\ |^2$. Similarly with a single qubit, the sum of all probabilities must equal 1. Probabilities of states may also be normalized if a subset of qubits are measured. For a two qubit system, the probability of measuring a 0 for the first qubit is $|\ \alpha_{00}\ |^2 + |\ \alpha_{01}\ |^2$, leaving the post-measurement state as

$$|\ \psi'\rangle = \frac{\alpha_{00}\ |\ 00\rangle + \alpha_{01}\ |\ 01\rangle}{\sqrt{|\ \alpha_{00}\ |^2 + |\ \alpha_{01}\ |^2}}$$

The post-measurement state is *re-normalized* to satisfy the condition that the probabilities must equal 1.

An important two qubit state is the *Bell state*.

$$\frac{|\ 00\rangle + |\ 11\rangle}{\sqrt{2}}$$

The Bell state has the property where when one qubit is measured, the state of the second qubit will always be correlated to the first qubit. This allows information processing beyond the possibilities of a classical system.

2.2 Computations

Quantum computers are built using *quantum circuits* and *quantum gates* to manipulate information, like classical computers. Quantum gates are defined as matrices which manipulate the vector state of qubits. Gates may encompass any amount of qubits, with their matrix size increasing accordingly.

Quantum circuits are diagrams read left to right, where each horizontal line represents a *wire*. The wire is not necessarily a physical wire, but can be thought of as a qubit going through a passage of time. Vertical lines between a wire and a gate is a control, where the effects of the gate are determined by the attached wire.

Common Quantum Gates and symbols:

Hadamard	$-\boxed{H}-$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$
Pauli-X	$-\boxed{X}-$	$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$
Pauli-Y	$-\boxed{Y}-$	$\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$
Pauli-Z	$-\boxed{Z}-$	$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$
Measurement		Projection onto $\|\ 0\rangle$ and $\|\ 1\rangle$
n qubits	$-\!\!\!\!/^{n}-$	wire containing n qubits

2.3 Quantum oracle

The *oracle* or *black box* is a subroutine in some quantum algorithms which represents a function. While an oracle may appear to know the solutions to the problem, it can only *recognize* a solution. Mathematically, the oracle is simply a function whose inner workings are unknown. However, implementing the oracle is possible using quantum gates.

2.4 Deustch-Jozsa Algorithm

2.4.1 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 2.1. *The number of steps needed to solve the Deutsch-Jozsa problem is* $2^{n-1}+1$

Lemma 2.2. *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.

2.4.2 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 [7], as shown in Figure 1. Given a quantum register of size $n+1$, the Deutsch-Jozsa algorithm is as follows:

(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.

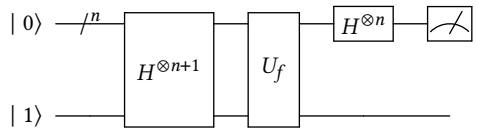

Figure 1: Quantum circuit for the Deutsch-Jozsa Algorithm

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

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

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

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 2.3. *The Deutsch-Jozsa problem could be solved with certainty with only 1 measurement [7].*

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.

2.5 Quantum Fourier Transform (QFT)

The *discrete Fourier transform* takes an input vector of complex numbers, $x_0, ..., x_{N-1}$ where N is the length of the vector, and outputs a vector of complex numbers $y_0, ..., y_{N-1}$, defined by

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N}$$

The *quantum Fourier transform* is the same transformation, except it works on an orthonormal basis $|\, 0\rangle, ..., |\, N-1\rangle$ with the action

$$|\, j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |\, k\rangle$$

2.6 Quantum Phase Estimation

The quantum Fourier transform is the key to *phase estimation*, which is used in many quantum algorithms. It estimates an unknown ϕ on a unitary operator U with an eigenvector $|\, u\rangle$ with eigenvalue $e^{2\pi i\phi}$. The estimation assumes there are oracles capable of preparing the state $|\, u\rangle$ and performing the controlled-U^{2^j} operation for suitable non-negative integers j.

This algorithm uses two registers. The first register contains t qubits initially state $|\, 0\rangle$. The number t itself depends on the desired accuracy and the desired probability of success

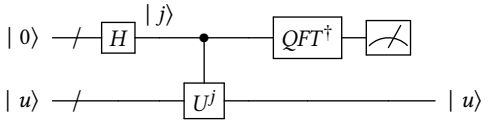

Figure 2: Schematic of the phase estimation procedure

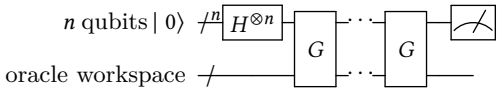

Figure 3: Circuit for Grover's Algorithm. There are $O(\sqrt{N})$ Grover iterations. The oracle may use work qubits for its implementation, but analysis only involves the n qubit register.

of the procedure. The second register contains as many qubits necessary to store $| u\rangle$.

The circuit begins by applying a Hadamard gate to the first register, then a series of controlled-U operations on the second register, with U raised to successive powers of two. The state of the first register will be

$$\frac{1}{2^{t/2}}(|0\rangle + e^{2\pi i 2^{t-1}\phi}|1\rangle)(|0\rangle + e^{2\pi i 2^{t-2}\phi}|1\rangle)...(|0\rangle + e^{2\pi i 2^{0}\phi}|1\rangle)$$

$$= \frac{1}{2^{t/2}}\sum_{k=0}^{2^t-1} e^{2\pi i\phi k}|k\rangle$$

Afterwards, the inverse quantum Fourier transform is applied to the first register. After the transform, the state of the first register is measured, giving an estimate of ϕ. The circuit for the phase estimation procedure can be seen in Figure 2.

2.7 Grover's Algorithm [11]

Grover's algorithm, also known as the *quantum search algorithm*, searches for the index of an item in an N item search problem. The algorithm uses an *oracle* which marks which index contains a solution. For convenience, it is assumed that $N = 2^n$, so the index can be stored in n bits, and the search problem has exactly M solutions, with $1 \leq M \leq N$. The circuit is shown by Figure 3.

The *Grover iteration*, denoted G, has four steps:

(1) Apply the oracle O. $|x\rangle \rightarrow (-1)^{f(x)}|x\rangle$
(2) Apply the Hadamard transform $H^{\otimes n}$
(3) Perform a conditional phase shift, where every computational basis state except $|0\rangle$ receives a phase shift of -1. $|x\rangle \rightarrow -|x\rangle$ for $x > 0$
(4) Apply the Hadamard transform $H^{\otimes n}$

The Grover iteration G (circuit in Figure 4) may be written as $G = (2|\psi\rangle\langle\psi| - I)O$, where ψ is the starting vector and O is the oracle operation. The effect of G can be seen as the effects of these normalized states, where $\sum_x'$ indicates a sum over all x which are solutions to the search problem, and $\sum_x''$ indicates a sum over all x which are not solutions to the search problem.

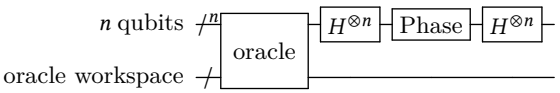

Figure 4: Circuit for the Grover iteration

$$|\alpha\rangle \equiv \frac{1}{\sqrt{N-M}}\sum_x{}''|x\rangle$$

$$|\beta\rangle \equiv \frac{1}{\sqrt{M}}\sum_x{}'|x\rangle$$

The initial state ψ can be expressed as

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle$$

The effect of the Grover iteration G is that the oracle O performs a *reflection* about the vector $|\alpha\rangle$ in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$. Afterwards, $2\langle\psi||\psi\rangle - I$ performs a reflection in the same plane defined by $|\alpha\rangle$ and $|\beta\rangle$ about the vector $|\psi\rangle$. This shows that the state $G^k|\psi\rangle$ remains in the space spanned by $|\alpha\rangle$ and $|\beta\rangle$ for all k.

The rotation angle can also be shown. Let $cos(\theta/2) = \sqrt{(N-M)/N}$, so that $|\psi\rangle = cos(\theta/2)|\alpha\rangle + sin(\theta/2)|\beta\rangle$. The two reflections of G transform $|\psi\rangle$ to

$$G|\psi\rangle = cos\frac{3\theta}{2}|\alpha\rangle + sin\frac{3\theta}{2}|\beta\rangle$$

Continued application of G is

$$G^k|\psi\rangle = cos(\frac{2k+1}{2}\theta)|\alpha\rangle + sin(\frac{2k+1}{2}\theta)|\beta\rangle$$

Repeated application of G brings the state vector close to $|\beta\rangle$, which produces with high probability that one of the measured outcomes is a solution to the search problem when observed in the computational basis.

In the $|\alpha\rangle, |\beta\rangle$ basis, the Grover iteration can be written as

$$G = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \quad (1)$$

where θ is a real number in the range 0 to $\pi/2$.

Rotating the initial state of the system $arccos\sqrt{M/N}$ radians takes the system to $|\beta\rangle$. Repeating the Grover iteration to the closest integer to $arccos\sqrt{M/N}$ (where halves are round down) rotates ψ to within the angle $\theta/2 \leq \pi/4$ of $|\beta\rangle$. Knowing the number of solutions M helps decide how many Grover iterations to implement, which the quantum counting algorithm is for.

2.8 Quantum Counting Algorithm

2.8.1 Counting Problem. What is the number of solutions, M, to an N item search problem? Here, the value and indexes of solutions is not needed, only the quantity. This has applications such as knowing the M for Grover's algorithm, speeding up solutions to NP-complete problems, as well as finding if a problem has an existing solution.

2.8.2 Classical Solutions. The classical solution iterates through the entire N item list, determining the number of solutions M after $\Theta(N)$ consultations.

2.8.3 Quantum Solution. The quantum algorithm to the counting problem is an implementation using Grover iterations and the quantum phase estimation[5, 13]. It finds the number of solutions M to a search problem by using phase estimation on a Grover iteration G to find its corresponding eigenvalues $e^{i\theta}$ and $e^{i(2\pi-\theta)}$ based on equation 1. It is also assumed that the oracle has been augmented to expand the size of the search space to $2N$, allowing $sin^2(\theta/2) = M/N$.[1] This algorithm requires $\Theta(\sqrt{N})$ oracle calls.

The circuit estimates θ to m bits of accuracy, with a success probability of at least $1-\epsilon$. The first register contains $t \equiv m + \lceil log(2 + 1/2\epsilon) \rceil$ qubits, and the second register contains n qubits[2], enough to implement the Grover iterations on the augmented search space.

The circuit has these steps:

(1) Initialize both registers t and n to $|0\rangle$.
(2) Apply to both registers the Hadamard transform.
(3) Apply controlled-G operators on the second register according to the first register, 2^n times.
(4) Apply the inverse quantum Fourier transform to the first register.
(5) Measure the first register for an estimation of θ.

Controlled-G operators are commutative can be reorganized to have the Grover iterations go from the 0th to the 2^{n-1}th (known as the *ascending order* algorithm, as shown in Figure 5), or from the 2^{n-1}th to the 0th (known as the *descending order* algorithm, as shown in 6). Without *decoherence*, these two algorithms are equivalent[12].

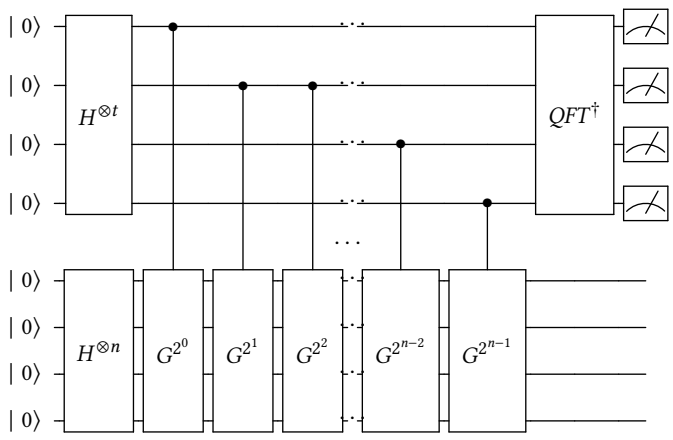

Figure 5: Representation of the quantum counting circuit in the Qiskit Textbook[3]. This is the *ascending-order* quantum counting circuit.

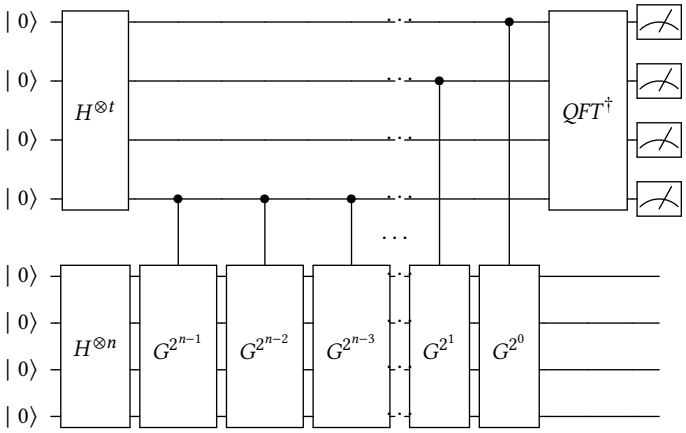

Figure 6: *Descending-order* quantum counting circuit.

2.9 Decoherence

Quantum computers, like classical computers, have physical components and are subjected to the environment around them. The environment could affect the physical representation of a qubit, the effect being known as *decoherence* or *quantum noise*[13]. Studies have been done to model the robustness of algorithms against decoherence, such as Azuma[4] on Grover's algorithm. Other studies have simulated the effect of decoherence such as Obenland and Despain[14] which uses a third state to affect the circuit in an ion state quantum computer. Other studies try to limit the effect of decoherence such as Flores and Galapon [10] which use additional qubits to preserve entanglement.

Most studies on decoherence are on either Grover's algorithm or Shor's algorithm. Hasegawa and Yura[12] worked on a theoretical analysis on the quantum counting algorithm.

2.9.1 Depolarizing Channel[13]. The *depolarizing channel* is a type of decoherence where a single qubit with state ρ has a probability p to be *depolarized*, meaning it is replaced by a mixed state $I/2$, while it has a probability $1-p$ to be unaffected.

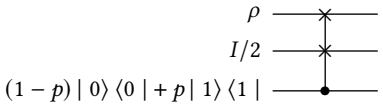

Figure 7: Circuit implementation of the depolarizing channel

Quantum circuits simulating the depolarizing channel use two 'environment' qubits to implement a controlled switch gate, with the idea that the third qubit, with the state $|0\rangle$ with probability $1-p$ and state $|1\rangle$ with probability p acting as a control if the mixed state $I/2$ is mixed with the qubit. The circuit can be seen in Figure 7.

The depolarizing channel can be parametrized as shown, for any arbitrary ρ, and $\mathcal{E}(\rho)$ as the quantum state:

$$\mathcal{E}(\rho) = (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z) \tag{2}$$

[1] Some references have $sin^2(\theta/2) = M/2N$, depending on how many qubits they consider the second register to have.
[2] Some references use n+1 as the number of qubits in the second register.

Figure 8: Implementation of the depolarizing channel on a chosen arbitrary qubit ρ, done after any arbitrary gate or operation G. The σ gate is one of the X, Y, or Z gates, chosen at random.

This is interpreted as the state ρ being unaffected with probability $1-p$, and with state ρ undergoing either an X, Y, or Z gate, with equal probability $p/3$.[13]

3 IMPLEMENTATION AND METHODOLOGY

The quantum circuits used are modified versions of the Qiskit textbook versions of the Deutsch-Josza algorithm[2] and the quantum counting algorithm[3]. Two versions of the quantum counting circuit are used for analysis, ascending order, and descending order. Both circuits both have $t = 4$ and $n = 4$, as well as the oracle pointing to 5/16 solutions existing in the item list. For calculations, let the second register use n[3, 12], instead of $n+1$ qubits of other calculations[13].

3.1 Decoherence Model

For modelling decoherence, the depolarizing channel is used[12]. However, this implementation does not use the environmental controlled swap qubits in Figure 7. This implementation uses a classical algorithm to insert a depolarizing channel to qubits after a gate operation. This depolarizing channel is either an X, Y, or Z gate chosen at random. This model bypasses the probability qubit by predetermining whether the ρ cubit is affected or not. This allows the effect of an applied depolarizing channel on a quantum circuit. The depolarizing channel will only be applied to qubits during the Grover iteration step of the circuit, as that step has exponentially more chance of error than other steps[12]. The circuit implementation can be seen in Figure 8.

Two types of experiments were conducted. The first has a set number of depolarizing channels being placed in the quantum algorithms. The second has a p chance of manifesting a depolarizing channel for each possible location.

This novel method is used for its simplicity and implementation compared to other methods.

3.2 Calculating Upper Bound of Decoherence Alpha

The decoherence model by De Jesus[8] is used due to its simplicity.

3.2.1 *Deutsch-Josza Algorithm.*

Theorem 3.1. *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 2.2 and 2.1.

$$n_{op(classical)} = 2^{L-1} + 1 = O\left(2^L\right) \tag{3}$$

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 Hadamard gate to all but one qubit. Thus:

$$n_{qop} = L + L + (L-1) = 3L - 1 \tag{4}$$

It was stated in Lemma 2.3 that only one measurement was necessary to get the answer with certainty. Thus:

$$P = 1 \tag{5}$$

We then compute for n_{trial} as follows:

$$n_{trial} = O\left(\frac{e^{\alpha n_{qop}}}{P}\right) = O\left(e^{\alpha(3L-1)}\right) \tag{6}$$

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

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

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

$$\alpha < \frac{L}{3L-1} \tag{7}$$

□

3.2.2 *Quantum Counting Algorithm.*

Theorem 3.2. *For the Quantum Counting Algorithm, given register sizes t and n where $t+n=L$, the allowable decoherence per operation per cubit α is bound by $\alpha < \frac{\ln \frac{8n^2}{c\pi^2}}{(t+n)(2t+n+(2^n-1)(n+1))}$*

Proof. De Jesus decoherence model uses L as its register size, but the quantum counting algorithm uses two registers of t qubits and n qubits. Thus, $L = t + n$.

- For n_{op}, we have $2t + n + (2^n - 1)(n+1)$:
  - Hadamard gates to t and n qubits $\Rightarrow t + n$
  - $2^n - 1$ Grover iterations affecting $n+1$ qubits each $\Rightarrow (2^n - 1)(n+1)$
  - QFT on t qubits $\Rightarrow t$
- For P, we have $8/\pi^2$[5].
- For $n_{op(classical)}$, we have n^2.
- Afterwards, we compute for n_{trial}.

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

$$n_{trial} = O(\pi^2 e^{(t+n)(2t+n+(2^n-1)(n+1))\alpha}/8) \tag{8}$$

Comparing with $n_{op(classical)}$

$$n_{trial} < n_{op(classical)}$$

$$O(\pi^2 e^{(t+n)(2t+n+(2^n-1)(n+1))\alpha}/8) < n^2 \tag{9}$$

$$c\pi^2 e^{(t+n)(2t+n+(2^n-1)(n+1))\alpha}/8 < n^2$$

$$\alpha < \frac{\ln \frac{8n^2}{c\pi^2}}{(t+n)(2t+n+(2^n-1)(n+1))} \tag{10}$$

□

Substituting $t = 4$ and $n = 4$, and assuming $c = 1$:

$$\alpha < \frac{\ln \frac{128}{\pi^2}}{696}$$

$$\alpha \lesssim 0.00368 \tag{11}$$

3.3 Deutsch-Josza Algorithm

This experiment creates a randomized 4 bit oracle for each trial. Depolarizing channels are applied with probability p to each qubit in the algorithm after their respective operations. The value of p starts with 0 and increases by 1% every 1000 trials. This was done for both *constant* and *balanced* oracle functions.

3.4 Quantum Counting Algorithm

3.4.1 First Experiment. The quantum counting algorithm with register sizes $t = 4$ and $n = 4$ can only have decoherence of $\alpha \lesssim 0.368\%$. For this implementation, 1 error is $1.\overline{33}\%$ possible decoherence (15 possible errors in the first register, and 60 possible errors in the second register). Since an implementation of a decoherence level within the bounds of α is not possible, the algorithm is expected to not give the correct number of solutions, with the correct measurements and solutions being shown in Table 1.

Four setups are as follows, which order the circuit is in, paired with the register with placed depolarizing channels. For brevity in later sections, the term used for the setup in placed in (parenthesis).

(1) Ascending order, first register 't', (ascending first) Figures 11 and 12
(2) Descending order, first register 't', (descending first) Figures 13 and 14
(3) Ascending order, second register 'n', (ascending second) Figures 15 and 16
(4) Descending order, second register 'n', (descending second) Figures 17 and 18

Each setup undergoes three tests, determining how many depolarizing channels are placed. Setups with the first register undergo a test with 1, 4 (one-fourth), and 7 (half) depolarizing channels. Setups with the second register undergo a test with 1, 15 (one-fourth), and 30 (half) depolarizing channels. Each test undergoes 50 trials, every trial randomizing the location and gate of the depolarizing channels. Average measurement probability of the test and count of estimated number solutions of each trial are taken.

3.4.2 Second Experiment. The second experiment applies depolarizing channels with probability p to each qubit in the Grover Iteration portion of the algorithm after their respective gates. The value of p starts from 0 and increases by 1% every 20 trials. The chance for incorrect readings was measured for each run. This was done for both the *ascending* and *descending* algorithms. This obtains the relationship between error ϵ and decoherence boundary α.

Table 1: Measured values from the Quantum Counting algorithm and the corresponding estimated number of solutions. Bold is the expected result. Expected results have an asterisk* in Figures 11 to 18

Measured Value	Estimated M
0000	16.0
0001, 1111	15.4
0010, 1110	13.7
0011, 1101	11.1
0100, 1100	8.0
0101, 1011	**4.9**
0110, 1010	2.3
0111, 1001	0.6
1000	0.0

4 RESULTS

4.1 Deutsch-Josza

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 4.1. 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.

During actual execution, the error ϵ could be measured by getting the percentage of incorrect readings of the quantum algorithm.

To get a relationship between ϵ and α, the effect of decoherence had to be simulated. For each test run, there was a probability p of placing a depolarizing circuit after a gate operation (including after the oracle), with the value of p increasing by 1% for every 1000 runs. The amount of incorrect readings ϵ was measured for each run. This was done for both *constant* and *balanced* functions, and could be seen in Figures 9 and 10.

The vertical line in Figures 9 and 10 indicate the boundary for α for very large registers; any value to the left of the line yields better efficiency than its classical counterpart.

For constant functions, the amount of error ϵ when α is at its maximum is roughly 63.37%. This means that even at max decoherence, the algorithm will only get the correct answer less than half of the time, and yet it will still be more efficient than its classical counterpart.

On the other hand, balanced functions show more leniency when it comes to error. Even if the system has completely lost decoherence, the range of error values only lie between $0-20\%$, with the intersection value being approximately 12.97%. This low error range is due to how the Deutsch-Jozsa problem is defined; we can infer from the algorithm description that the final reading of the quantum register would only be either

$|0\rangle$ or $|1\rangle$, and nothing else. Hence, the deciding factor of the algorithm is that if the final reading is $|0\rangle$, the given function is "constant". This means that any other reading between $|0\rangle$ and $|1\rangle$ would lead to "balanced", thus the solution space for balanced functions is much larger than that of constant functions, which leads to the lower error values.

4.2 Quantum Counting

4.2.1 First Experiment. Before trials were run, the assumption was that measurements other than the correct 0101 and 1011 would appear. Trials with a single error would usually still have the correct answer, while trials with more errors will have a more even distribution of measurements. Another assumption was that the computed number of solutions would follow the proportions of the average measurement chance, with the pairs seen in Table 1.

Across most setups, 0101 and 1011 were the most likely measurement on average for the single error test. However, the corresponding number of solutions 4.9 was not the considerably most common occurrence for all setups except for the *ascending second* with one error. Remarkably, that was the only case where the algorithm output the correct answer in over 50% of the trials.

A trend noticed for the setups with errors on the *first* register is that the measurement chance of each measurement is close to the mean, but with a bias towards measurements near the two correct measurements 0101 and 1011. The bias seems to be reduced with more errors introduced. The solutions the algorithms give are usually 4.9, 8.0, and 11.1 with similar chances while being above the mean, especially with only one error introduced. Introducing more errors causes other solutions to appear more often.

A trend noticed for the setups with errors on the *second* register is that with more errors, measurements seem to gather closer to the center measurement 1000 on the *ascending* algorithm, while the opposite happened on the *descending* algorithm. Following the measurement and solution pairings on Table 1, the *ascending* algorithm had more computed solutions on the lower range (0.0 to 8.0), while the *descending* algorithm had more computed solutions on the higher range (15.4 and 16.0).

It seems that of all setups, the *ascending second* algorithm is the most resistant to decoherence, at least with only 1 error introduced. It is the only setup which obtained the correct solution with over 50% certainty. When more errors are introduced, both *ascending first* and *descending first* algorithms give solutions close to the correct solution.

4.2.2 Second Experiment. The vertical line in Figures 19 and 20 indicate the boundary for α for when the registers $t = 4$ and $n = 4$. Any value to the left of the line yields better efficiency than its classical counterpart.

It is seen for both the *ascending* and *descending* algorithms have a sharp increase in decoherence up to around 90% when the amount of decoherence α increases from zero. The error percentage when alpha is at 0.368% is around 12.6% for both the ascending and descending quantum counting algorithms, showing that the algorithms are very sensitive to low changes in decoherence.

Furthermore, the error of the *descending* algorithm increases faster than the error of the *ascending* algorithm. This follows the description of Hasegawa and Yura[12].

5 FINAL REMARKS

The used method for simulating decoherence is easily implementable on Qiskit, and may show more insights on the effects of decoherence on quantum algorithms. The method shows in the quantum counting algorithm that the superposition of measurements across all trials does not necessarily follow the actual results of each trial. The method also shows some trends in the algorithm when the decoherence level is high.

This method also suggests that use of the Deutsch-Josza algorithm is much faster than its classical counterpart for low decoherence. It also suggests that the quantum counting algorithm is extremely sensitive to decoherence and would not be as reliable as the classical solution.

With respect to the study of Hasegawa and Yura[12], the results confirm their conclusions on decoherence on the first and second registers. Namely, when the errors are on the first register, the ordering of the Grover iterations seem independent from the results. When errors are on the second register, it is shown that the *ascending order* algorithm is more robust than the *descending order* algorithm.

Further use of this method may be used to compare to other implementations of decoherence such as using auxiliary qubits to simulate the depolarizing channel. Other options are to change the oracle of the algorithm used, expand the algorithm with bigger register sizes, or using this method on different quantum algorithms.

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.

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

P. Fortuno would also like to thank the DOST-ASTI COARE services for helping run simulations for this paper.

REFERENCES

[1] Héctor Abraham et al. 2019. Qiskit: An Open-source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2562110

[2] Héctor Abraham et al. 2020. Deutsch-Jozsa Algorithm. https://qiskit.org/textbook/ch-algorithms/deutsch-jozsa.html.

[3] Héctor Abraham et al. 2020. Quantum Counting. https://qiskit.org/textbook/ch-algorithms/quantum-counting.html.

[4] 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

[5] G. Brassard, P. Høyer, and A. Tapp. 1998. Quantum Counting. arXiv:quant-ph/9805082 [quant-ph]

[6] 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

[7] 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

[8] Brian Kenneth de Jesus. 2014. Quantum Decoherence and Computational Complexity.

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

[10] 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

[11] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. arXiv:quant-ph/9605043 [quant-ph]

[12] J. Hasegawa and F. Yura. 2005. Theoretical Analyses of Quantum Counting against Decoherence Errors. arXiv:quant-ph/0503202 [quant-ph]

[13] M. A. Nielsen and I. L. Chuang. 2010. *Quantum Computation and Quantum Information* (10th anniversary ed.). Cambridge University Press, New York, NY, USA.

[14] Kevin M. Obenland and Alvin M. Despain. 1998. Simulating the Effect of Decoherence and Inaccuracies on a Quantum Computer. arXiv:quant-ph/9804038 [quant-ph]

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

A APPENDIX: GRAPHS

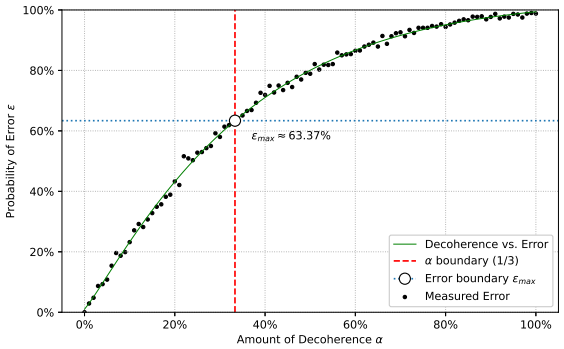

Figure 9: Error vs. Decoherence for the Deutsch Jozsa algorithm (Constant).

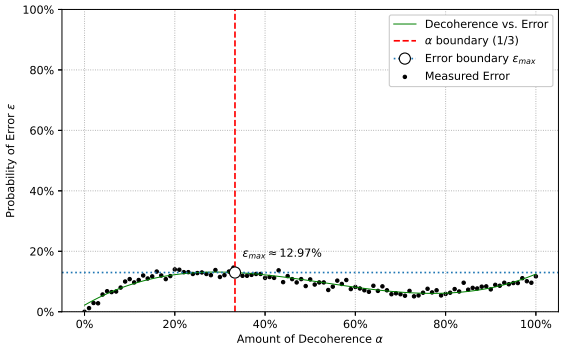

Figure 10: Error vs. Decoherence for the Deutsch Jozsa algorithm (Balanced).

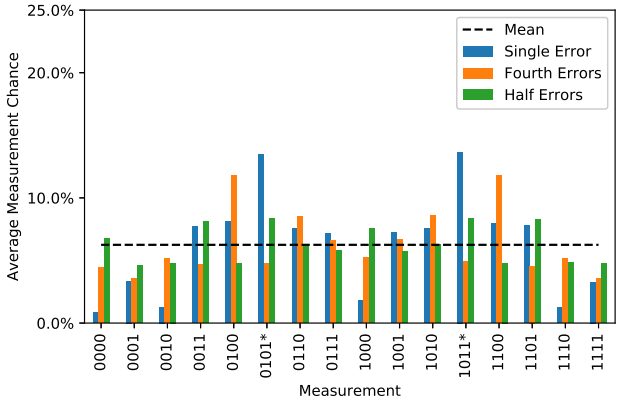

Figure 11: Average measurement for the *ascending* quantum counting algorithm with decoherence on the *first* register.

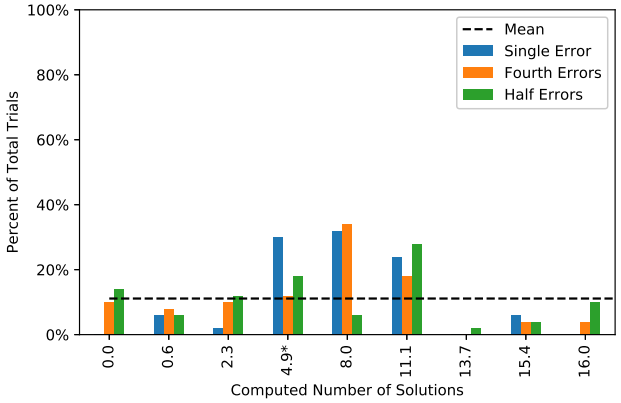

Figure 12: Counted results for the *ascending* quantum counting algorithm with decoherence on the *first* register.

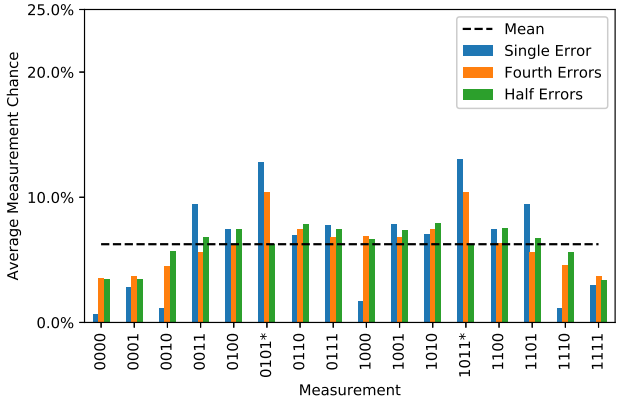

Figure 13: Average measurement for the *descending* quantum counting algorithm with decoherence on the *first* register.

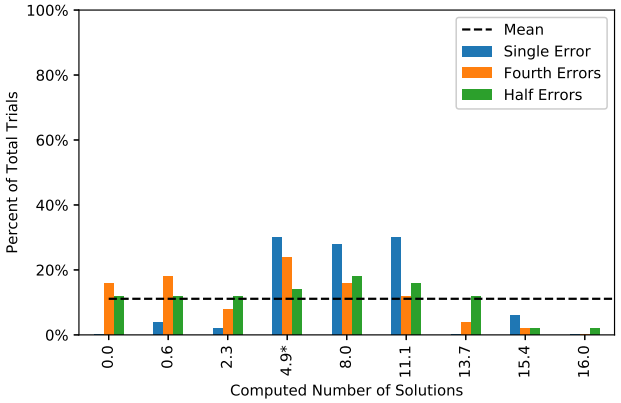

Figure 14: Counted results for the *descending* quantum counting algorithm with decoherence on the *first* register.

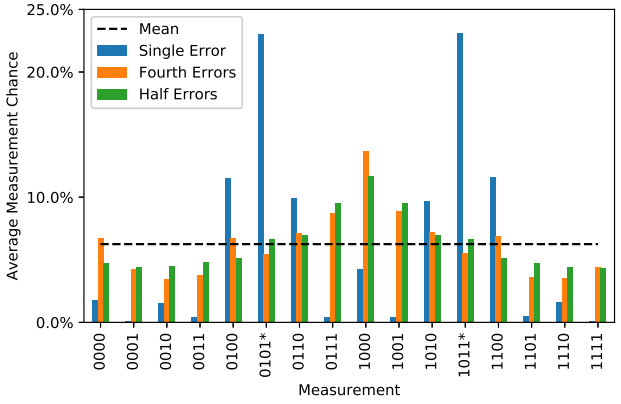

Figure 15: Average measurement for the *ascending* quantum counting algorithm with decoherence on the *second* register.

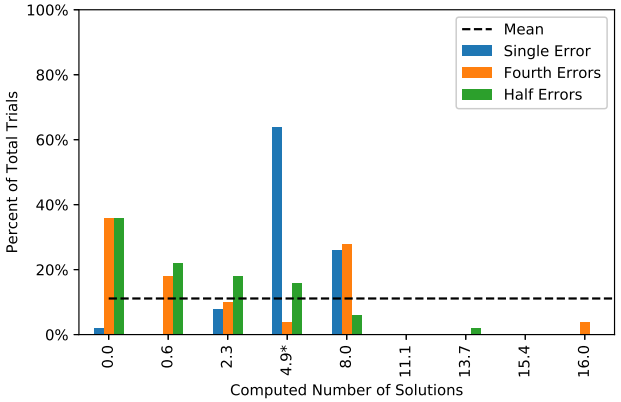

Figure 16: Counted results for the *ascending* quantum counting algorithm with decoherence on the *second* register.

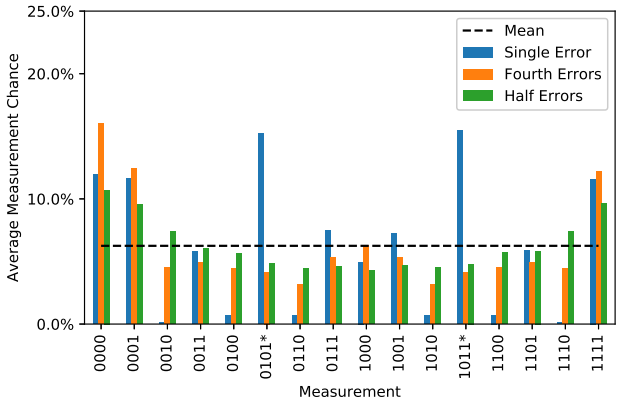

Figure 17: Average measurement for the *descending* quantum counting algorithm with decoherence on the *second* register.

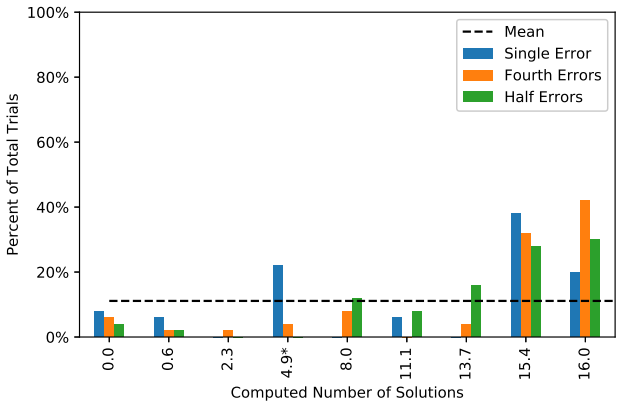

Figure 18: Counted results for the *descending* quantum counting algorithm with decoherence on the *second* register.

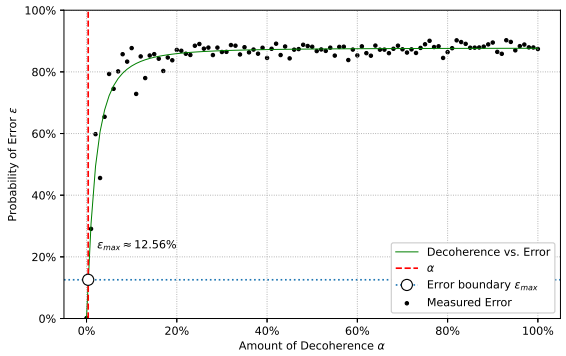

Figure 19: Error vs. Decoherence for the *Ascending* Quantum Counting Algorithm.

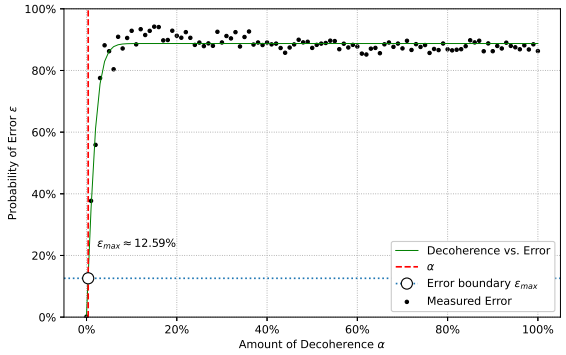

Figure 20: Error vs. Decoherence for the *Descending* Quantum Counting Algorithm.