Biomedical Data Science
2024, VOL. 4, NO. 2, 39-50
DOI: 10.47297/wspbdsWSP2752-630504.20240402

Experimentation of Noise on Quantum Monte Carlo Tree Search

Michelle Wang

Robert Louis Stevenson, Pebble Beach, California

ABSTRACT
In 2023, CNN News reported that nearly 800, 000 deaths or cases of permanent disability in the U.S. are linked to diagnostic errors annually, with some caused by mistakes in Artificial Intelligence (AI). The accuracy and accountability of AI in healthcare have long been questioned, and the application of Quantum Computing (QC) could improve the efficiency, accountability, and accuracy of decision trees.

KEYWORDS
Artificial Intelligence;Quantum Computing;Monte Carlo Tree Search

Research Objective—This paper tests the effects of noise in quantum computing on a popular type of decision tree search, the Monte Carlo Tree Search (MCTS), to determine the best noise model for implementation in AI healthcare.
Methods—·Simulation: A simulation was constructed to estimate the ground state energy of a one-dimensional quantum harmonic oscillator using the Quantum Monte Carlo (QMC) method. QMC: The Variational Monte Carlo (VMC) method combined with MCTS was implemented, using inferential statistics through random sampling to approximate the ground state of a quantum system. MCTS: The Monte Carlo Tree Search algorithm specializes in decision-making in complex problems, searching for the optimal solution through random samples and building a tree from its outcomes. Results：The program was run on the Qiskit simulator, testing three different noise models: Readout Error, Bit-Flip with Tensor Product, and Amplitude Damping Error. Results indicated that the Readout Error model, at a low error rate of 0.0001, produced outputs closest to the no-noise model, suggesting the highest accuracy. The Bit-Flip and Amplitude Damping models, at a high error rate of 0.999999999, showed significant differences from the no-noise model, indicating lower accuracy.

1 INTRODUCTION & BACKGROUND

1.1 Research Significance

On July 19 2023, CNN News headline strikes: "Diagnostic errors linked to nearly 800,000 deaths or cases of permanent disability in US each year" [1]. Compared to the 370,000 deaths or permanent disability in 2022, the number in 2023 has significantly increased [2]. Part of these deaths are caused by the mistakes of Artificial Intelligence (AI) [3]. Although the use of AI has been implemented since the 1970s in areas such as interpreting findings and recommending treatments, it has long been questioned in its accuracy and accountability [4]. A key role in AI programs are possibility trees (decision trees). These trees are coded in AI to assist healthcare professionals in determining the patient's symptoms, test results, medical background, and current conditions to execute a diagnosis. In the treatment stage, the possibility tree determines the stage of the disease, provides a list of treatment options, and an estimated prediction of the condition.

One way of improving the accuracy and accountability of possibility trees is using quantum computing (QC). QC uses qubits that serve like bits in classical computers and outperforms classical computers. Using QC, possibility trees are able to improve in efficiency, accountability, and accuracy. However, in QC, the factor of noise, which is disturbance from environments such as influences from other particles or the Earth, can cause errors in the final results of these simulations.

This paper tests the effects of noise on a type of widely popular possibility tree search, the monte carlo tree search (MCTS), to determine the best noise model that is able to be used with MCTS and be implemented in AI healthcare.

1.2 Background

Possibility trees, also known as decision trees, are a visual representation used to predict, analyze, and implement to select an optimal solution in a scenario. Possibility trees are built of nodes (decision points), with each node branching out to several nodes and each iteration improving from prior iterations. During its selection process of the best node, possibility trees calculate the probability of the outcome and multiply that value to the connected branches while considering potential uncertainties, consequences, and outcomes with a variety of conclusions.

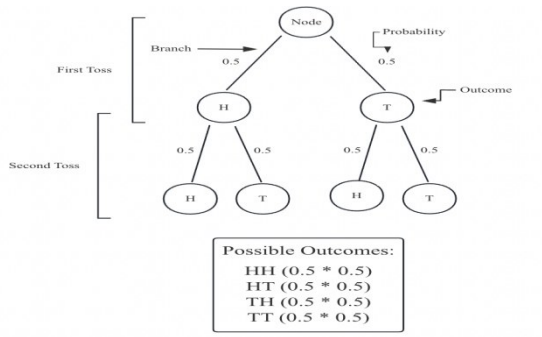

Figure 1 A basic overview of what a possibility tree looks like

The calculation of the final possibility includes multiplying the possibility values of each branch. In this example of coin toss, there are two tosses and the probability of getting Heads or Tails is both 50% (0.5). There are four possible combinations of the two tosses, each with a 25% (0.25) chance producing one of them as the final outcome.

Current applications of possibility trees include medicine, game theories, predictions in research, and mainly artificial intelligence (AI). Through the use of possibility trees, AI is able to play against an individual and hone their skills at the same time, challenging critical thinking and imagination. The AI AlphaGO is a prime example of the use of possibility trees. Its huge success led to further enhancement of the algorithm and branched out to different variations. Gaming companies compete to build interactive NPCs (non-playable characters), where the NPC is able to respond to a player based on the player's input. These companies implement possibility trees in training their AI to better adapt to a player's interest and produce the finest results. Continuous improvements are being tested on possibility tree algorithms with many aspects of possibility trees still being enhanced: timeliness, its exponential growth in the possibilities, uncertainty, assumptions, and bias.

The major struggle with the possibility tree is determining the uncertainty of an outcome and deciding which node to choose from two similar outcomes. In an attempt to improve possibility trees and this uncertainty, Quantum Computing (QC) has been applied and achieved huge accomplishments such as encryption choices and advances in chemistry. QC uses qubits that serve like bits in classical computers. Though there are limitations, QC outdoes classical computers in

many areas: efficiency, complexity, simplicity, security, simulations, and more. QC enables possibility trees to expand beyond their current achievements on classical computers and reach a more accurate conclusion.

	Classical Computers	Quantum Computers
Accuracy	Uses bits (0 and 1) to solve problems – not very accurate	Uses qubits ($\|0\rangle$ and $\|1\rangle$) to solve problems – more accurate
Limitations	Solves smaller scaled problems Use for quick computation that does not require a lot of complexity	Solves larger scaled problems (such as factoring large numbers) Use for complex computation
Speed	Fast until the problem gets to big (factoring a large number might take years to solve)	Faster (factoring a large number takes seconds to solve)
Uncertainty	One input → same output Biased probability	Deals better with unbiased probability

Figure 2 Comparison on classical computers and quantum computers based on accuracy, limitations, speed, and uncertainty

One way of implementing QC into a possibility tree is through the Monte Carlo method [5]. Monte Carlo estimates the values of an unknown quantity using inferential statistics and does this by random sampling [6]. The core of the algorithm is to create an unbiased group of samples from a large group of possibilities to let the simulation evolve randomly. Quantum Monte Carlo (QMC) is essentially just Monte Carlo but in the quantum world. Currently, variations of QMC exist and are applied to many understandings of complex problems: medicine, quantum systems, nuclear physics, and behaviors of particles.

A variation of Monte Carlo is the Monte Carlo Tree Search (MCTS), an algorithm that specializes in decision making in complex problems and searches for the optimal solution through random samples and building a tree from its outcome. It is applicable to any domain that can be modeled by a tree and is heavily implemented in Artificial Intelligence (AI) and in games like GO. The tree searches for the optimal solution by choosing a node, expanding from that node, subjecting it to simulation, and examining if it is the most favorable result within the interaction. The tree is built by the previous results of nodes and examines the next possible moves; each possibility becoming more accurate as the tree expands. Though there are limitations to the accuracy of MCTS, its development over the years has enhanced the tree. This diversification helps diversify the possibilities and applications to specific problems, enabling faster searching of a complex issue.

Applications of MCTS in the quantum world are heavily applied in calculating energy states. Experimentations with electrons and particles are often tested on different simulations. However, not all results produced are accurate. This applies to anything that involves QC. The factor of noise, which is disturbance from environments such as influences from other particles or the Earth, can cause errors in the final results of these simulations. Minimizing noise can be especially difficult on large programs, causing huge differences in the final outputs. In order to test out the effect noise has on MCTS, this paper uses three different noise models and analyzes their effect on MCTS and their accuracy on a one-dimensional quantum harmonic oscillator using a Qiskit simulator.

2 METHODS

2.1 Simulation

To simulate a simple Monte Carlo Tree Search, a simulation is built to estimate the ground state energy (lowest state for that particle) of a one-dimensional quantum harmonic oscillator

(something that swings back and forth in a one dimensional plane) using a trial wave function.

The mathematical representation of Quantum Harmonic Oscillator can be described using the Schrödinger equation with its Hamiltonian form written as:

$$\widehat{H} = \underbrace{-\frac{h^2}{2m}\frac{d^2}{dx^2}}_{\text{Kineti}} + \underbrace{\frac{1}{2}m\omega^2 x^2}_{\text{Poten}}$$

Figure 3 Mathematical representation of Quantum Harmonic Oscillator [7]

Ĥ - Hamiltonian operator (total energy) h - reduced Planck constant
m - mass of particle ω - angular frequency x - position of particle

2.2 Quantum Monte Carlo (QMC)

QMC implements quantum computing to solve complex problems in the quantum world such as particle interactions. There are two methods of QMC: Variational Monte Carlo (VMC) and Diffusion Monte Carlo (DMC). This paper implements VMC along with MCTS.

VMC is a computational method that combines quantum mechanics with statistical sampling to approximate the ground state of a quantum system [9]. In order to describe the quantum state of a system, a trial wave function is chosen and its parameters can be adjusted to fit the energy of the system [5]. VMC is chosen in this paper because it permits fast computation of ground state energy and is less computationally heavy. The algorithm of VMC is mostly random sampling through Monte Carlo techniques to stabilize the energy or until convergence criteria is met.

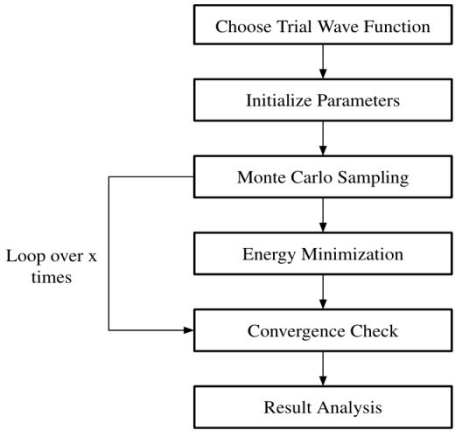

Figure 4 Overview of how Monte Carlo functions

Steps 3-5 are looped over x amount of times before it terminates and checks for the final result. Monte Carlo does these sampling by random to create the most unbiased outcome it can possibly give [8].

The key properties when looking at VMC are the mean local energy, which is the total amount of energy divided by number of samples, and the uncertainty, which is from the variance (represented as σ^2, it is the range of energy distribution and gets smaller as better wave functions are implemented) and the standard error of the mean. The mathematical representation of calculating

uncertainty is: $E_{VMC} \pm \frac{\sigma}{\sqrt{M}}$, where E_{VMC} is the energy, σ is the standard deviation, and M is the number of samples [9].

2.3 Monte Carlo Tree Search (MCTS)

The MCTS is a heuristic search algorithm that belongs in the Monte Carlo class. It aims to create decision processes through random sampling by building a tree using nodes similar to that of a probability tree. These nodes are used to calculate possibilities, expanding until the optimal (highest possibility) node is reached.

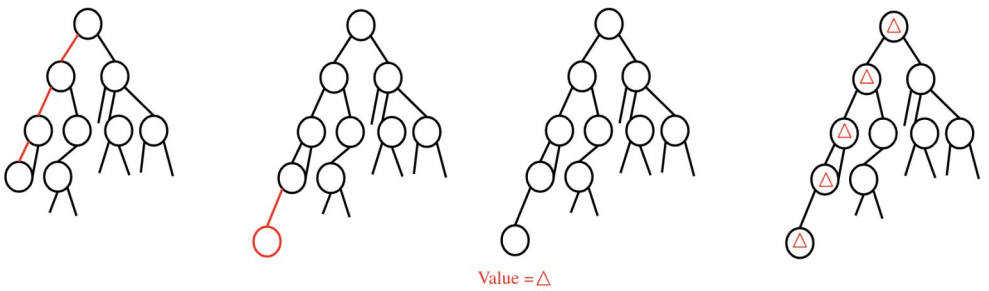

Figure 5 The four different steps of MCTS

From left to right, Descension (selection), Expansion, Simulation, Backpropagation. The red parts are to highlight what is done each step. From selection, a path is selected and expanded on. Then that node gets assigned a value Δ and is run through the simulation to check for its accuracy and record its value [10].

There are four major steps in the MCTS algorithm: descension (selection), expansion, simulation, and backpropagation [9] [11].

1. *Descension (selection)* – MCTS needs to explore and expand itself in order to find the best child node. In order to balance the expansion and exploration on a given node, the following formula is implemented:

$$UCB1\,(k,p) = \underbrace{E[win|k,p]}_{\text{Expand}} + \underbrace{C\sqrt{\frac{2ln\,(n_{\,parent(k)})}{n_{\,k}}}}_{\text{Explore}}$$

Figure 6 The left side (expand) is how good a player's chances are to win from that position.

The right side (explore) is a combination of the number of games a player played and the number of games the person before the player (parent) played and tries to balance between the two values. This formula is able to maximize the expansion while increasing the chance of the node being explored again. If the explored portion of the formula is zero, the algorithm has to pick the node it has never picked before, thus expanding the tree.

2. *Expansion* – A certain number of child nodes are added based on the possible actions. Set n_k = 0 and w_k = 0 to start the expansion.

3. *Simulation* – After the tree reaches a node that has never been explored, it will simulate the game randomly to keep moving forward. The node will be at a certain state and pick random nodes

from both players to simulate the game until the game ends. If the current player wins, $\Delta = +1$, otherwise $\Delta = 0$.

4. *BackPropagation* - BackPropagation passes information recursively to parent nodes. It updates the tree by the following formula:

Given simulation results Δ, for each k:

$n_{k\text{-new}} = n_{k\text{-old}} + 1$

$w_{k,1\text{-new}} = w_{k,1\text{-old}} + \Delta$

Basically it updates the number of wins in the game to the fraction ($w_{k,1} / n_k$).

In order to know when it is best to terminate the tree, the highest expected win ($E[win|k]$) and the most visited node is taken [10].

Each node in MCTS stores important information that helps determine its final outcome. For instance when at a point in the game (k), a node can be represented by the formula $w_{k,p} / n_k$ where n_k is the number of games that included that node and $w_{k,p}$ is the number of games the player has won. If the number on the node is ⊠ it means it player one has four wins out of the five games played involving that node [12].

The Upper Confidence Bound (UCB) is part of a larger group of strategies called the Multi-armed-bandit that is aimed to create unbiased probability while returning the highest reward to solve a problem [13]. UCB's goal is to prioritize nodes that have been less visited during a simulation to give that node a chance of producing a better result. There are two parts of the equation to UCB which includes exploration and exploitation [13]:

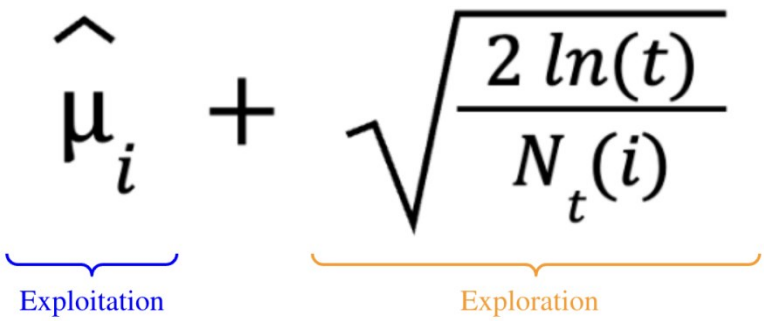

Figure 7 The exploitation is $\hat{\mu}_i$, representing the sample mean

The second half of the equation is exploration, where $N_t(i)$ is the number of times the node is visited. The lower this number is, the less times this node has been visited, thus, if a node produces a high number given the equation, it will be prioritized to be explored in the next iteration.

The UBC is implemented in MCTS to ensure that all nodes receive an equal chance in the random sampling and is how the tree explores and expands.

2.4 Noise in Quantum Computing

In a quantum system, there exists high numbers of errors, such as the instability of qubits and unintentional state changes. Noise is what causes these instabilities in a quantum system. Noise affects electrons through its environment or any uncontrolled variables. The vast amount of variables that can affect a quantum system is difficult to control, therefore, noise is an important inclusion during quantum simulations.

Readout (Measurement) error - Readout error is error caused by two main factors. One is the mistakes during the measuring of the quantum state (normally caused by noise). The second is the

extensive time of measurement can lead to interactions with its environment thereby increasing error chances. Readout errors affect the decoherence of quantum states, disrupting properties that make superpositions and quantum computations possible [14] [15]. It often limits the overall performance and final accuracy of the model.

Bit-flip Noise with tensor product - Bit-flip Noise is a type of error where the state vector of a qubit flips from |0⟩ to |1⟩ or vice versa. Tensor product uses two vectors and combines them into one with an expanded dimensionality. When the two are combined together, certain qubits experience independent Bit-Flip, while the other remains unaffected, which is represented through the tensor product.

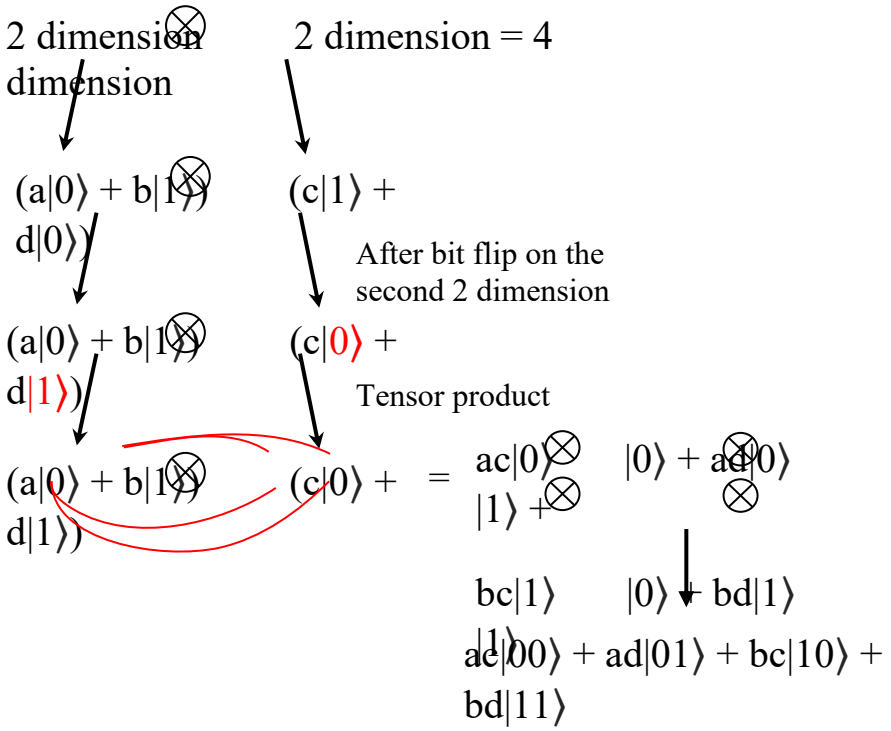

Figure 8 Bit-Flip with Tensor product in action

The two dimensions are combined using tensor products to create four dimensions. The state vectors are flipped on the second qubit after implementing Bit-Flip and the rest is multiplied out to form the final four dimensions.

Amplitude Damping Error - an error that causes the quantum state to decay from a high energy state to ground state. This type of error happens when storing superpositions for an extensive time, disturbing its final value and causes inaccurate results. Amplitude damping error takes three parameters: the parameter amplitude, the population of |0⟩ state at equilibrium (default: 0), and canonical kraus [16].

3 RESULTS

The program is run on an oscillator frequency of 1.0 with the particle's mass as 1.0. The calculation for kinetic energy, potential energy, wave function, MCTS are all kept constant. The program contains 100 trials each with 3048 nodes and a step size of 0.1. There are two error

probability rates that were tested: 0.0001 (small error rate) and 0.999999999 (big error rate). All other variables are kept constant except for the modification on noise.

In order to find the noise model that had the least effect on the simulation, the cluster ranges on both the x and y axis are compared to the no noise model in addition to its scatteredness in the cluster.

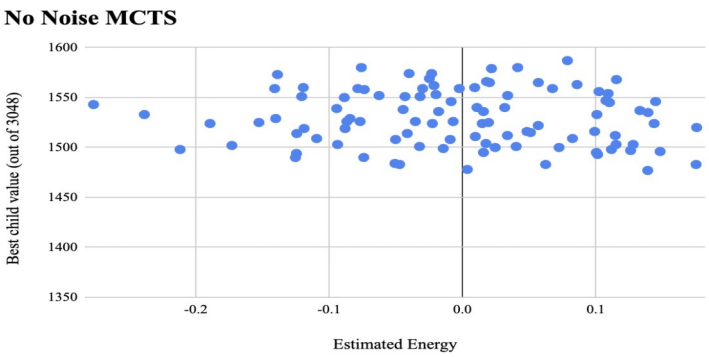

Figure 9 The No Noise Model

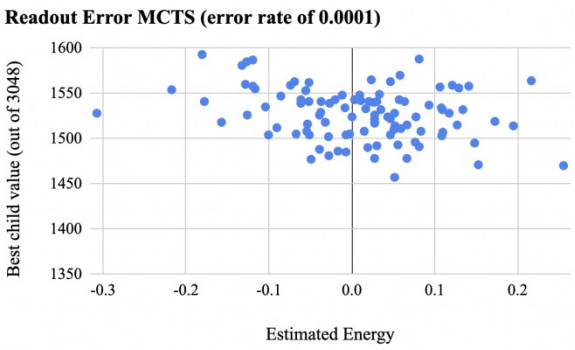

Figure 10 Readout Error with error rate of 0.0001

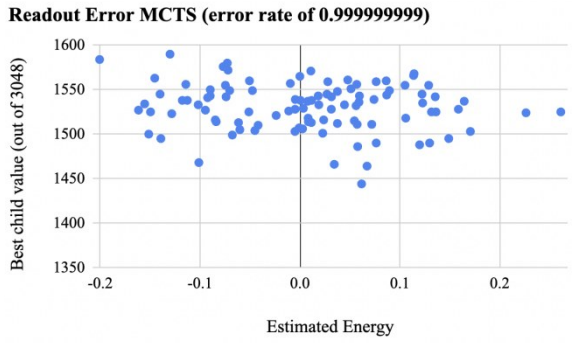

Figure 11 Readout Error with error rate of 0.999999999

The graph is a scatter plot with the estimated ground state energy as the x-axis and the best node (out of 3048) on the y-axis.

From the no noise model, the cluster of points mostly centered around an estimated energy of -0.1 to 0.1 and a best child node value of 1480 to 1570 with the dots a bit scattered around that range.

The output cluster that is closest to the no noise model in the error rates of 0.0001 is the readout

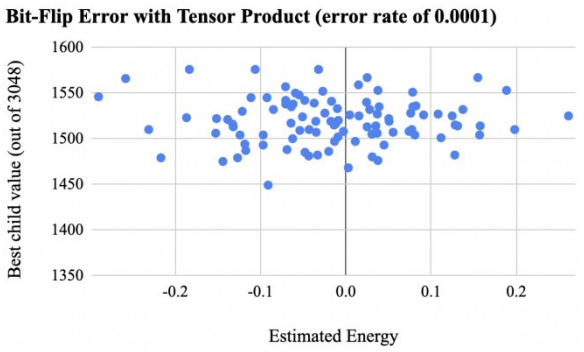

Figure 12 Bit-Flip using Tensor Product with error rate of 0.0001

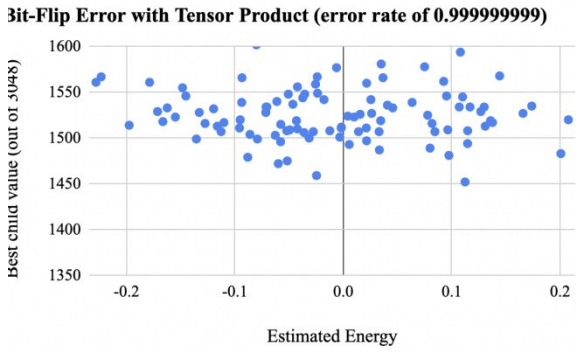

Figure 13 Bit-Flip using Tensor Product with error rate of 0.999999999

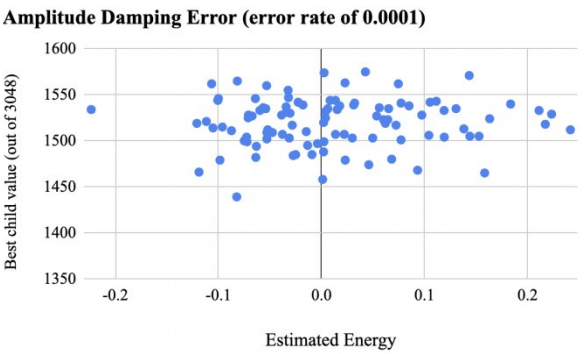

Figure 14 Amplitude Damping Error with error rate of 0.0001

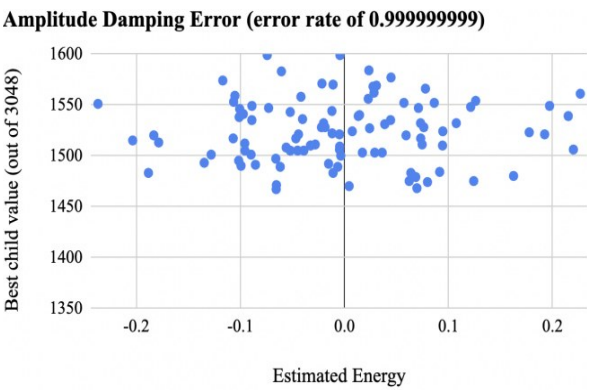

Figure 15 Amplitude Damping Error with error rate of 0.999999999

error. Its cluster is similar to the no noise model, with a -.07 to 0.1 estimated energy value and best child node value of 1470 to 1550. From the graph, the readout error with an error rate of 0.0001 has data closer in output to one another compared to the no noise model. Its output is the closest to the no noise model despite its difference in data. This implies that it yielded the most accurate result out of all noise models compared.

The output cluster that is farthest to the no noise model in the error rates of 0.999999999 is the Bit-flip with Tensor product and the Amplitude Damping Error. The Bit-flip with Tensor product had a range of -1.5 to 0.1 on the estimated energy and a 1500 to 1560 on the best child value. The Amplitude Damping model had a range of -1.2 to 0.1 on the estimated energy and a 1500 to 1550 on the best child value. Both of the models (Amplitude damping and Bit-Flip with tensor product) are more scattered on its graph compared to other noise models, implying its accuracy is the lowest among all other noise models compared to the no noise model.

4 Model Evaluation

When extracting information from a qubit, the measurements can be affected by errors such as readout errors. Readout error achieved the highest accuracy compared to other models mainly because it did not affect the measurement as much, therefore the information extracted is similar to that of the no noise model. Since the error rate is small (0.0001), it did not affect the final output as much as the other models.

Bit-flip error achieved a low accuracy because it enlarges the error by flipping the state of a qubit. With the high error rate (0.999999999), there is a loss in the information and instability of outputs as can be seen in figure 13 where the dots are more scattered. Amplitude Damping also achieved a low accuracy because energy 'escapes' from the model. When the error rate is tuned up high, the energy ground state goes from high to low, making the qubits unstable and scattered as seen in figure 15. The readout error did not produce as inaccurate of a result as the other two noise models mainly because the measurement tuned might still have a chance that it performs as well as the no noise and also that it does not ensure a guaranteed large loss of information as the Bit-flip with tensor product in certain scenarios.

There are some points on the scatter plots where either the ground state energy became very large or small, or where the best child node value was far from the rest of the data. These 'outliers' implied that the noise model greatly affected the simulation compared to the rest of the simulation run. The noise model affected the qubit or its measurements that led to a loss of information, decoherence, or error propagation that altered its final output.

5 DISCUSSION

5.1 Program

The program is coded in Python on the Qiskit simulator. It mainly consists of the MCTS simulation and the different noise that it was tested on with two qubits. The different types of noise models are implemented within the functions of the simulation. The basic code of the simulation is as follows [7]:

Pseudo-code of the MCTS Simulation without Noise
function trial_wave (position x, spread of the wave function)
return value of the wave function at position x
function local_energy (position x, spread of the wave function)
kinetic_energy equation

potential_energy equation
return kinetic_energy + potential_energy
class MCTS
constructor
function mcts (root, noise model, number of shots, number of iterations)
selection, expansion, simulation, backpropagation
selection
while node is child
select the node that has the highest UCB
expansion
if the node visit > 0
run the node through MCTS class and create a new child
set node to newly created child
simulation
simulate MCTS
backpropagation
while node is ran
number of visit on node + 1
set node to parent node
best_action = node with the highest UCB

There are still many improvements that can be made on the program, such as the methods for calculating MCTS and the wave functions. MCTS is constantly being improved on with faster and more accurate algorithms that can be implemented in the future.

5.2 Applications

This MCTS simulation can be applied in many future areas in the study of both noise and quantum probability. It gives an insight on the optimal noise model on limited qubits and what quantum MCTS deals with in real hardware such as IBM quantum. This simulation can later be enlarged and run on better computers to better predict predictions.

In the future, more comparisons between different noises will be implemented, in addition to running the simulation on real hardware such as IBM quantum with more qubits. Heavier oscillation of noise will also be tested with different energy levels. Additionally to improving the MCTS tree, there will be more involvement in ways of improving the uncertainty of the tree when there are multiple equal options to choose from. Better and faster algorithms can be combined with MCTS to achieve an even faster and more efficient way of random sampling while eliminating bias. There is much to the future of MCTS and noise.

To enhance possibility trees in healthcare, more tree searches will be explored using quantum computing. These tree searches will be tested on a specific area of healthcare, such as cancer, to minimize misdiagnosis.

6 Conclusion

This research looked at the effect of three different noise models (Readout error, Bit-Flip with Tensor Product, Amplitude Damping error) in quantum computing to determine its effect on MCTS. All code were run on the Qiskit Simulator and were compared to the no noise model. A simulation of the calculation of the ground state energy on a one-dimensional quantum harmonic oscillator using MCTS was executed to test the three noise models. The final outputs determine its probability value

in MCTS but also its accuracy in determining the ground state energy. Among the three different noise models, the output cluster that is closest to the no noise model in the error rates of 0.0001 is the readout error, making it the optimal model for MCTS.

References

[1] McPhillips D. (2023, July 19). Diagnostic errors linked to nearly 800,000 deaths or cases of permanent disability in US each year, study estimates. CNN. https://www.cnn.com/2023/07/19/health/diagnosis-error-study/index.html

[2] Kounang N. (2022, December 16). More than 7 million incorrect diagnoses made in US emergency rooms every year, government report finds. CNN. https://www.cnn.com/2022/12/15/health/hospital-misdiagnoses-study/index.html

[3] Davenport T., & Kalakota R. (2019). The Potential for Artificial Intelligence in Healthcare. Future Healthcare Journal, 6 (2), 94–98. https://doi.org/10.7861/futurehosp.6-2-94

[4] Hall K., & Fitall E. (2020). Artificial Intelligence and Diagnostic Errors. Psnet. ahrq. gov. https://psnet. ahrq. gov/perspective/artificial-intelligence-and-diagnostic-errors

[5] Kanno, Shu & Nakamura, Hajime & Kobayashi, Takao & Gocho, Shigeki & Hatanaka, Miho & Yamamoto, Naoki & Gao, Qi. (2023). Quantum computing quantum Monte Carlo with hybrid tensor network toward electronic structure calculations of large-scale molecular and solid systems

[6] OpenCourseWare, MIT . 2017. "6. Monte Carlo Simulation." Www.youtube.com. May 19, 2017. https://www.youtube.com/watch?v=OgO1gpXSUzU&t=290s.

[7] Clark, Bryan. 2023. "Quantum Monte Carlo. " Illinois. edu. 2023 https://clark. physics. illinois. edu/Tutorials/VMCIntroTutorial/index.html.

[8] PACK, QMC. 2021. "2 - Introduction to Quantum Monte Carlo - QMC Workshop 2021." Www.youtube.com. October 21, 2021. https://www.youtube.com/watch?v=aOm3ilUNNU8&t=1310s.

[9] X. Zhou, Y. Feng and S. Li, "A Monte Carlo Tree Search Framework for Quantum Circuit Transformation," 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), DiegoSan, CA, USA, 2020, pp. 1-7.

[10] OpenCourseWare, MIT. 2018. "Advanced 4. Monte Carlo Tree Search." Www.youtube.com. October 26, 2018. https://www.youtube.com/watch?v=xmImNoDc9Z4&t=1878s.

[11] C. B. Browne et al., "A Survey of Monte Carlo Tree Search Methods," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 1, pp. 1-43, March 2012, doi: 10.1109/TCIAIG.2012.2186810.

[12] P. Wang, M. Usman, U. Parampalli, L. Hollenberg and C. Myers, "Automated Quantum Circuit Design With Nested Monte Carlo Tree Search" in IEEE Transactions on Quantum Engineering, vol. 4, no. 01, pp. 1-20, 2023. doi: 10.1109/TQE.2023.3265709

[13] math, ritvik. 2020. "Best Multi-Armed Bandit Strategy? (Feat: UCB Method)." Www.youtube.com. October 5, 2020. https://www.youtube.com/watch?v=FgmMK6RPU1c&t=510s.

[14] Beisel, Martin, Johanna Barzen, Frank Leymann, Felix Truger, Benjamin Weder, and Vladimir Yussupov. 2022. "Configurable Readout Error Mitigation in Quantum Workflows" Electronics 11, no. 19: 2983. https://doi. org/10.3390/electronics11192983

[15] Kim, Jihye, Byungdu Oh, Yonuk Chong, Euyheon Hwang, and Daniel K Park. 2022. "Quantum Readout Error Mitigation via Deep Learning." New Journal of Physics 24 (7): 073009. https://doi.org/10.1088/1367-2630/ac7b3d.

[16] Qiskit Contributors, "Qiskit: An Open-source Framework for Quantum Computing", 2023, doi: 10.5281/zenodo.2573505