Introduction to Randomized Compiling

Danesh Morales Hashemi

PHYS 468 Final Project

Waterloo, Ontario, Canada, Fall 2024

Abstract

Quantum computers have the potential of outperforming classical computers by reducing the computational complexity of certain problems, such as factorizing large numbers via Shor's algorithm [1]. However, quantum computers are not perfect; physical quantum computers can be inaccurate due to noise in the environment, State Preparation And Measurement (SPAM) errors, crosstalk, etc [2].

Randomized Compiling (RC) is an error suppression protocol proposed by J. Emerson and J. Wallman in 2015 [3]. RC works by tailoring any coherent and non-Markovian error channels into stochastic error channels (such as the Pauli channel) without altering the logical circuit, and keeping the depth of the circuit unchanged [3]. Having a stochastic error channel will yield a higher probability of obtaining the correct output since stochastic channels have a lower worst-case error rate than coherent channels [4].

The goal of this project is understand the RC protocol step-by-step and explain why it works. We will test how RC suppresses noise in the IBM 127-qubit quantum computer `ibm_brisbane`, by exploring the effectiveness of RC on quantum circuits with varying parameters, such as: number of qubits, circuit depth, and high or low entropy. We will observe that RC works best for circuits with high entropy rather than low entropy. Finally, we will conclude that RC is a highly scalable efficient protocol as the classical resources needed to run RC are very small regardless of the number of qubits and depth of the circuit [3].

Table of Contents

List of Figures

1 Introduction

1.1 Background

The following definitions and notations will be used throughout the project.

Definition 1.1 (Easy Gates)**.** [3] The easy gate set is the group generated by $C = \{\mathbf{P}_2, S\}$ defined as

$$\mathbf{C} = \{I, X, Y, Z, S, S^\dagger, SX, S^\dagger X\} \tag{1.1}$$

where $\mathbf{P}_2 = \{I, X, Y, Z\}$ is the one-qubit Pauli gate set, and S is the phase flip gate defined as

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{1.2}$$

Definition 1.2 (Hard Gates)**.** [3] The hard gate set is defined as

$$\mathbf{G} \ = \{H, CX, T\} \tag{1.3}$$

where H is the Hadamard gate, CX is the CNOT gate, and T (also known as $\pi/8$ gate) is defined as

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \tag{1.4}$$

Definition 1.3 (Total Variation Distance)**.** [4] The TVD between two given probability distributions $\mathcal{P}, \mathcal{Q}$ is

$$d_{\mathrm{TV}}(\mathcal{P}, \mathcal{Q}) = \frac{1}{2} \sum_{x \in X} |\mathcal{P}(x) - \mathcal{Q}(x)| \tag{1.5}$$

TVD equal to 1 implies that the probability distributions fully overlap, whereas a TVD equal to 0 means that there is no overlap whatsoever.

Definition 1.4 (Pauli Channel)**.** [3] The $n-$qubit Pauli channel is defined as

$$\mathcal{E}(\rho) = \sum_{P \in \mathcal{P}_n} p_P P \rho P \tag{1.6}$$

where $P \in \mathcal{P}_n$ represents a n-fold tensor product of Pauli matrices acting on the density matrix ρ, and satisfies

$$\sum_{P \in \mathcal{P}_n} p_P = 1 \tag{1.7}$$

Definition 1.5 (Pauli Twirl)**.** [5] The Pauli twirl is a process that transforms an arbitrary quantum channel into a stochastic Pauli channel by averaging the channel over the Pauli group as follows:

$$\mathcal{E}_{\text{Twirled}}(\rho) = \frac{1}{|\mathcal{P}_n|} \sum_{P \in \mathcal{P}_n} P\mathcal{E}(P\rho P)P \tag{1.8}$$

1.2 Randomized Compiling

1.2.1 Introduction

Imagine you start with a two-qubit system in the state $|00\rangle$. Then, you create a quantum circuit, and use your initial state as the input. After running the circuit, your initial state has been mapped to the state $|00\rangle \mapsto |\psi\rangle$, where

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{1.9}$$

In theory, after measuring the state, there is a 50/50 chance that your resultant state is in the $|00\rangle$ or $|11\rangle$ state; however, you notice something peculiar. You repeat this procedure a 1000 times, and you realize that out of 1000 measurements, you have 398 $|00\rangle$ states and 602 $|11\rangle$ states. You repeat the experiment thousands of times more, and indeed there is approximately a 40/60 probability of measuring $|00\rangle$ and $|11\rangle$ respectively using the same circuit. You can see that the noise in the system is affecting your output.

However, something occurs to you. You try running the experiment with the same initial state, but with a different circuit, such that the output $|\psi\rangle$ is the same. Now you notice something else: after thousands of measurements, now the probabilities have flipped, you have a 60/40 chance of measuring $|00\rangle$ or $|11\rangle$ respectively. The noise is affecting the new circuit differently. You realize that if you average over the two circuits used, then you will get roughly a 50/50 chance of measuring the desired states, matching with the ideal probability.

This is the main idea behind Randomized Compiling. RC works by tailoring any coherent and non-Markovian error channels into stochastic Pauli error channels without altering the logical circuit, and keeping the depth of the circuit unchanged [3]. We start with a noisy quantum circuit, and modify it several times such that the logical circuit remains the same, take measurements for each circuit, then average all the results. But, how many circuits do we need to generate and average? And how do we generate them to ensure we are tailoring the noise into a stochastic Pauli channel and we are able to suppress the noise as much as possible?

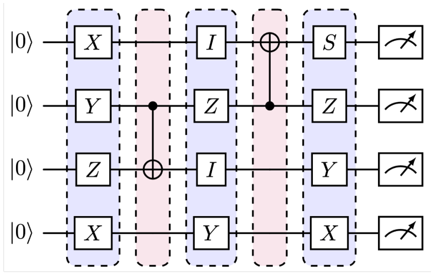
Unfortunately, the proofs to the above answers are beyond the scope of this course and project, but if you are interested, please refer to [3], [4]. Even though we are not going to prove mathematically that RC works in this project, in the next section we will go over the RC procedure step by step and try to explain why RC works.

1.2.2 RC Procedure [3]

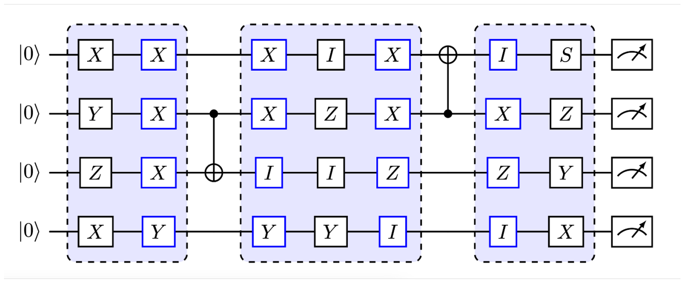
First, let's start with a primitive gate set that generates universal circuits. Then, we split the universal gate set into easy and hard gates. Note that we have already defined the easy gate set in Definition 1.1 and hard gate set in Definition 1.2, and they indeed form a universal set of quantum gates; therefore, we can rewrite any circuit in terms of easy and hard gates. As a quick reminder, the easy gate set is $\mathbf{C} = \{I, X, Y, Z, S, S^\dagger, SX, S^\dagger X\}$, and the hard gate set is $\mathbf{G} = \{H, CX, T\}$. After defining our easy and hard gate sets, RC works as follows:

(1) Define the desired quantum circuit. (2) Rewrite the circuit as alternating cycles of easy and hard gates. (3) Add random gates from the one-qubit Pauli set around the hard gate cycles, such that the resultant circuit is logically equivalent. (4) Compile the easy gates. (5) Run the compiled circuits and measure the results. (6) Repeat steps (3) - (4) - (5) for multiple randomizations. (7) Add up all the results and normalize them. (8) Calculate the total variation distance (TVD) with respect to the ideal circuit.

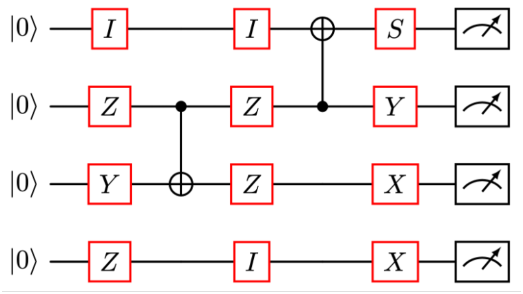
The following figures will visually explain steps (2) - (3) - (4).



(2) Circuit with alternating easy and hard gates



(3) Adding random Paulis around the hard gate cycles



(4) Result of compiling the Paulis with the easy gates

Figure 1: Steps (2) - (3) - (4) for Randomized Compiling

The circuits in step (2) and (4) are equivalent; that is, their unitary matrix representation is the same. Repeating this process several times will have a similar effect as applying the Pauli twirl to the noisy channel, without affecting the original circuit. The more randomizations we use, the closer the effect is to a Pauli twirl. Therefore, from Definition 1.8, our noisy channel will approximately be a stochastic Pauli channel (it will exactly be a Pauli Channel as the number of randomizations goes to infinity, but only ≈ 20 randomizations are needed RC for to work [4]). Having a stochastic error channel will yield a higher probability of obtaining the correct output since stochastic channels have a lower worst-case error rate than coherent channels [4].

2 Experiment

2.1 Introduction

From Section 1.2.2, we can see that in order to apply the RC protocol we need: (1) A desired circuit. (2) A set of randomized circuits, logically equivalent to the original, which we will use to average over its results and see the effect of RC. (3) A quantum computer to measure and collect our results.

There is no problem with (1) as we can just choose our own circuit. For (2), we will use TrueQ [6], a Python library which has a function `randomly_compile` where you can input any circuit, number of randomizations n, and returns a collection of n randomly compiled circuits, logically equivalent to the original, which we will run in the quantum computer. For (3), we will run our experiments in the IBM 127-qubit quantum computer `ibm_brisbane`. We will use the Qiskit Python library to run our circuits in the physical quantum computer.

2.2 Challenges

The biggest challenge of this project is that there is no direct compatibility between the TrueQ and Qiskit circuits. Thankfully, TrueQ circuits have the attribute `to_qiskit()` which converts a TrueQ circuit into a Qiskit circuit. Similarly, Qiskit does not have good visualization tools for showing RC results, but we can translate the Qiskit results into TrueQ results using the TrueQ `Results` function and then use the TrueQ `Visualization` tools.

2.3 Experiment Steps

From Sections 2.1 and 2.2, these are the following steps we need to follow to complete our experiment:

1. Define our desired circuit in TrueQ.
2. Use the `randomly_compile` function to generate a collection of circuits.
3. Translate the TrueQ circuits into Qiskit circuits using the `to_qiskit()` function.
4. Run the circuit cillection in the `ibm_brisbane` quantum computer and save the results.
5. Translate the Qiskit results into TrueQ results using the `Results` function.
6. Use the `Visualization` tools in TrueQ to plot our results.

3 Results

In this section, we will test the RC protocol on quantum circuits with varying parameters, such as: number of qubits, circuit depth, and high or low entropy. Additionally, we will show that the number of randomizations needed to make RC work is independent of circuit depth and number of qubits.

4 Conclusions

1. We tested the ability of RC to suppress noise on the IBM 127-qubit quantum computer, `ibm_brisbane`, and evaluated its effectiveness on quantum circuits with different parameters.
2. We observed that RC is most effective on circuits with high entropy compared to low entropy circuits.
3. We concluded that the number of randomizations needed to make RC work is independent of circuit depth and number of qubits.

References

[1] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, January 1996. URL http://arxiv.org/abs/quant-ph/9508027. arXiv:quant-ph/9508027.

[2] Avimita Chatterjee, Koustubh Phalak, and Swaroop Ghosh. Quantum Error Correction For Dummies, April 2023. URL http://arxiv.org/abs/2304.08678. arXiv:2304.08678.

[3] Joel J. Wallman and Joseph Emerson. Noise tailoring for scalable quantum computation via randomized compiling, June 2016. URL http://arxiv.org/abs/1512.01098. arXiv:1512.01098.

[4] Akel Hashim, Ravi K. Naik, Alexis Morvan, Jean-Loup Ville, Bradley Mitchell, John Mark Kreikebaum, Marc Davis, Ethan Smith, Costin Iancu, Kevin P. O'Brien, Ian Hincks, Joel J. Wallman, Joseph Emerson, and Irfan Siddiqi. Randomized compiling for scalable quantum computing on a noisy superconducting quantum processor, May 2021. URL http://arxiv.org/abs/2010.00215. arXiv:2010.00215.

[5] Joseph Emerson. *Theory of Quantum Systems: From Mathematical Foundations to Experimental Methods with Applications to Quantum Computing*. 2021.

[6] Stefanie J. Beale, Kristine Boone, Arnaud Carignan-Dugas, Anthony Chytros, Dar Dahlen, Hillary Dawkins, Joseph Emerson, Samuele Ferracin, Virginia Frey, Ian Hincks, David Hufnagel, Pavithran Iyer, Aditya Jain, Jason Kolbush, Egor Ospadov, José Luis Pino, Hammam Qassim, Jordan Saunders, Joshua Skanes-Norman, Andrew Stasiuk, Joel J. Wallman, Adam Winick, and Emily Wright. True-Q, June 2020. URL https://zenodo.org/records/3945250.