Milestone

Hayden Coffey, Kanghee Park, Saurabh Kulkarni

April 22, 2022

1 Discuss Key Insight

Now that we have been working on this project for some time, we have been able to clearly observe that simulation is only practical for trivial circuits. While this is well documented, actually observing the explosion in execution time has reinforced our understanding of a need for a predictive model or a heuristic to predict circuit fidelity. This observation has directed us towards examining the use of machine learning for predicting circuit fidelity measurements.

We've found that TKET is a well-documented, modular framework that plays nicely with Qiskit. While we are still exploring its usage, it may provide an alternative to Qiskit which we could use as another comparison to our baseline model.

2 Build on Literature Survey

2.1 Noise Simulation

Research on noise simulation is not as relevant as we originally thought. While these papers describe efficient noise simulation and achieve speedups, they are not nearly fast enough for deployment into a developer tool like we are targeting. This has lead to us pursuing a machine learning based approach instead. We have found a paper on using neural networks for fidelity measurement, but it appears to be from an outside journal the university does not have access to.

2.2 Qubit Mapping/Routing

Qubit mapping/routing papers helped to provide us with context to what the compiler is doing under the hood, but there are enough frameworks already existing (Qiskit,TKET) that we don't really need to implement these ourselves and doing so would add unnecessary complications to the project.

2.3 Variational Algorithms

Variational algorithms for computing fidelity measurements are not feasible for our use case with the current state of the technology. Maybe in the future when every laptop has an onboard quantum processor.

2.4 Distance Metrics

Papers on various measurements of quantum state distance have been the most relevant for our work. We discuss some new metrics we are incorporating into our design in this report.

2.5 Characterization Data Collection

Characterization data collection research and crowd funding data collection would be a project in of itself. We are operating with the assumption that this data is provided to us by IBM or whichever back-end we are running on.

3 Discuss Your Design

Our code is available on GitHub.

We have a set of .qasm files that we benchmark and produce metrics for using IBMQ back-ends and Qiskit. We plan to implement the analysis tools that are currently implemented using Qiskit alongside TKET. We would like to see if TKET can provide different information or results.

3.1 Simulation Baseline

Our baseline fidelity estimation is provided in *est.py*. The baseline reads in a QASM file and calculates PST, TVD, Entropy, and the number of compiler inserted SWAP gates via simulation for each compatible IBMQ fake back-end. An aggregate metric, named "Fitness", is used to sort the results and the top n results are reported where n can be configured. Below is an example output from the program for $n = 10$:

```
qasm/Noise_Benchmarks/QFT-5B.qasm 3.409527(s) +++++++++++++++
Backend Name        PST       TVD       Entropy   Swaps     Fitness
fake_lima           0.873     0.168     2.555     6         0.853
fake_ourense        0.874     0.173     2.586     8         0.710
fake_belem          0.797     0.238     2.756     7         0.656
fake_manila         0.843     0.218     2.601     9         0.612
fake_athens         0.773     0.278     2.842     9         0.529
fake_london         0.719     0.298     2.940     8         0.516
fake_burlington     0.726     0.312     2.960     9         0.481
fake_essex          0.657     0.378     3.108     7         0.473
fake_bogota         0.638     0.424     3.094     9         0.390
fake_quito          0.615     0.426     3.140     10        0.354
```

The header of the table details which circuit we are evaluating alongside how long the simulation took, 3.41 seconds in this case for $QFT - 5B$. The "Fitness" metric used to sort the results is defined as follows:

$$\begin{aligned} Fitness &= \frac{\alpha PST}{\beta TVD + \gamma Entropy + \delta Swaps} \\ \alpha = 1, \beta = 1, \gamma &= \frac{1}{10}, \delta = \frac{1}{10} \end{aligned} \tag{1}$$

We can tune the hyper-parameters α, β, γ, and δ to scale the impact of each metric on the fitness score. Finding optimal values for these parameters as well as incorporating more sophisticated metrics into the fitness calculation is an area of the project we are actively working on. We discuss some of the more advanced metrics we plan to incorporate later in the report.

3.2 Predictive Model

Since our end goal is a tool that developers could use to estimate circuit fidelity while writing their code, our system needs to be able to provide meaningful metrics within a practical time frame. While simulation may work for trivial circuit sizes, the execution time explodes exponentially with number of qubits. We are currently working on training a machine learning model with circuit and machine characterization information to see if we can do the computational work ahead of time and provide users with a much faster oracle they can query.

We are still actively working on this model and are generating data-sets from simulation runs that can be used to train it. Our first data-set design includes IBM basis gate counts, machine ID, average node degree, and machine qubit size. While conducting this initial data generation, we have also found new values/metrics that could be meaningful inputs to the model. We will discuss these metrics later in the report as we go over design revisions we have planned.

3.3 VSCode Extension

We have begun the process of setting up the boiler plate code for a VSCode extension. We believe that a VSCode extension would be an effective front-end for the tool due to its large developer base and its extensive plugin support. However, this work is our lowest priority as we focus on developing an effective oracle for circuit fidelity estimation. Once the back-end is reasonably complete, time permitting, we will use this extension as a way to convey information to the user.

4 Evaluate the Design and Show Preliminary Results

4.1 Design

As we have discussed previously, the largest flaw with the baseline is the reliance on simulation to collect data. This prevents the tool from providing meaningful results to the user within a reasonable time. While simulation results can provide us with a control we can compare with, for our system to be effective, the estimation must be capable of being performed within a reasonable time frame for non-trivial circuits.

Additionally, we are primarily relying on simplistic fidelity metrics such as PST and TVD. For our final design we would like to provide more sophisticated metrics in addition to these. While examining the literature, we found more options such as L2 distance and Hellinger distance. Although they have many problems representing the distance between probability distributions, they could be used as additional values for estimating fidelity.

In regards to the ML model and its training data-sets, we believe our inputs could be improved. Including details such as circuit depth, operation density, measurement density, retention lifespan, quantum area, and entanglement variance alongside the gate noise characterization data would provide a much stronger characterization of circuit/machine behavior. The simulation is the current bottleneck for us estimating fidelity, so any characterization metrics that can be determined statically with reasonable computation could be meaningful inputs. We have found a QASM benchmark suite named QASMBench that includes code for calculating these metrics, making the collection of these data points more feasible.

4.2 Preliminary Results

We have previously shown an example of our baseline results, but to further demonstrate the difficulties with simulation we provide the results for $qaoa10_depth2$. Some details to note, going from a 5 qubit to a 10 qubit circuit has increased our execution time from around 3.4 seconds to about 11 seconds. While there are other factors that can impact simulation time outside of number of qubits, we have found simulation time to rapidly increase to the point of taking upwards of 12 hours for each individual back-end for a 20 qubit circuit. Next, we have found some back-ends produce faulty results in simulation. Our tool must be able to account for these outliers. The back-end $fake\_rueschlikon$ seems to be particularly prone to producing outlier results as we can see it appears to vastly outperform the other models. Our fitness metric does default to 0 in the case that the denominator is 0, but outliers can still cause the calculation to run awry. Revising our fitness calculation and adding some form of outlier detection to the tool would allow for us to filter these results out.

```
qasm/Noise_Benchmarks/qaoa10_depth2.qasm 11.039076(s) +++++++++++++++
Backend Name        PST       TVD       Entropy   Swaps     Fitness
fake_rueschlikon    1.000     0.001     -0.000    0         1024.105
fake_melbourne      0.408     0.592     2.832     4         0.320
fake_almaden        0.417     0.583     2.783     6         0.285
fake_singapore      0.440     0.560     2.861     9         0.252
fake_cairo          0.771     0.241     1.322     33        0.210
fake_boeblingen     0.496     0.504     2.513     18        0.194
fake_guadalupe      0.666     0.334     1.743     32        0.180
fake_poughkeepsie   0.415     0.585     2.874     15        0.175
fake_johannesburg   0.274     0.726     3.682     17        0.098
fake_tokyo          0.262     0.738     3.745     20        0.084
```

In addition to our simulation baseline results, we do have data-sets containing simulation results for some of our benchmark QASM files across the different model back-ends available through Qiskit. We are using these data-sets to train a neural network model to predict metrics such as PST, TVD, Entropy, and Swaps. Below is an example excerpt from one such data-set:

```
id rz, sx, x, cx, rst, msre, u3, u2, u1, Machine, ADeg, NumQubit, PST, TVD, Entropy, Swaps
0, 46,  8, 0, 20, 0,   5,    -1, -1, -1,  2,      3.2,  5,      0.773, 0.241,2.814,  9
0, 46,  8, 0, 20, 0,   5,    -1, -1, -1,  3,      3.2,  5,      0.822, 0.202,2.705,  10
0, 46,  8, 0, 20, 0,   5,    -1, -1, -1,  5,      3.2,  5,      0.662, 0.382,3.092,  9
0, -1, -1,-1, 20,-1,   5,     0,  8, 30,  7,      3.2,  5,      0.742, 0.287,2.961,  8
0, -1, -1,-1, 20,-1,   5,     0,  8, 30, 11,      3.2,  5,      0.639, 0.391,3.076,  7
0, 46,  8, 0, 20, 0,   5,    -1, -1, -1, 18,      3.2,  5,      0.85,  0.194,2.582,  8
0, -1, -1,-1, 20,-1,   5,     0,  8, 30, 19,      3.2,  5,      0.733, 0.289,2.918,  8
```

As we have previously discussed in our design evaluation, we believe that the data-set could be revised to include more meaningful input metrics. We are in the process of generating revised data-sets including these metrics as we train an initial ML model on our existing data-sets. We hope to have our simulation baseline, an initial predictive model, and a revised model as comparison points on our final report.

Our initial neural network predicts each simulation measurement from our input features detailed above. Some issues with our initial approach are as follows. The current input features do not consider the circuit structure. Additionally, there are also some features close to cheating, such as machine number (which machine we are running on), which limit how well this model can generalize. The model and feature still needs a lot of improvement; however, our inital findings give us hope that some approximate metric values can be predicted using this approach. Below is a sample of the results we obtained for estimating TVD:

idx	TVD(Actual)	TVD(Predicted)
318	0.32490423	0.347656
210	0.432537	0.466797
191	0.10520383	0.057617
366	0.43971738	0.460937
217	0.27837223	0.239258
92	0.6015685	0.626953
95	0.44383398	0.460937
368	0.44828677	0.424805
220	0.21073438	0.239258
197	0.22616467	0.215820

5 Discuss Evaluation Methodology (Benchmarks, Metrics)

5.1 Benchmarks

We are evaluating our tool on a number of QASM files. The circuits available to us range in size from 1 - 1000 qubits. While this gives us quite a range to explore, simulating a 20+ qubit circuit is computationally expensive and impractical for this project. These larger circuits provide cases where simulation techniques will no longer be viable for the tool and an alternative such as a predictive model or heuristic are necessary. Realistically, we will likely not be able to obtain results for circuits larger than 20 qubits, as our current baseline numbers are those obtained via simulation, and the cost of simulation exponentially increases with number of qubits.

5.1.1 Noise and SWAP Benchmarks

We have been using the QASM files supplied from Assignment 4 for preliminary testing of the tool as they are small enough to evaluate through simulation. However, qft_n20 from the SWAP benchmarks has been particularly difficult to gather simulation data for (collecting data for the various IBMQ fake back-ends takes 32+ hours).

5.1.2 QASMBench

QASMBench is an open source collection of QASM files collected for the purpose of benchmarking NISQ machines and simulation techniques. This collection is broken up into small (2-5 qubit), medium (6-15 qubit), and large (15+ qubit) circuits. While we have not thoroughly explored these circuits at the time of writing, we believe this collection would provide benefit our evaluation by allowing us to explore a wide variety of circuit sizes.

5.2 Metrics

5.2.1 Aggregate Fitness

Currently we are ranking the back-ends off of PST, TVD, Entropy, and SWAP gate counts as shown in Equation 1. While this appears to be working for the most part, we have also observed that faulty simulations can disrupt our rankings with outlier data points. Outlier detection and revising Equation 1 can reduce the impact of these models on our output data.

As we have described previously, metrics we are in the process of collecting and integrating into our system include L2 distance and Hellinger distance. Although it is difficult to say that these two represent the distance of probability as well as compared to TVD, their computation is not difficult and it is worth providing as additional reference.

5.3 Model Training / Data-Set Generation

Our model aims to estimate the metrics we obtain through simulation, ideally sacrificing some accuracy for an exponential speedup. As a result, we can still use our fitness metric to rank our back-ends, substituting the simulation provided values for those predicted by the model. We believe that our current training data-sets are too simplistic and do not effectively characterize the circuits. We are revising these sets to include input values such as: Operation Density, Measurement Density, Retention lifespan, Quantum Area, and Entanglement Variance. The QASMBench benchmark suite includes python code that can determine these values for a Qiskit circuit, which greatly streamlines the process of integrating these values into our current system.

5.4 Capturing Probability Distribution Information

Besides the numerical values we use to calculate fitness, providing some estimation of the probability distribution of the circuit on a given machine with noise would be a valuable output. However, this is looking to be one of the more challenging aspects to effectively estimate without simulation. We may be able to incorporate a system to present this data to the user for small circuits across a variety of back-ends via simulation, but we believe designing a model to predict these distributions to be non-trivial. As a result, given the remaining time left for this project and our current task priorities, we do not expect to have this implemented for complex circuits by the final report.

6 Summary / Final Steps

In summary, our baseline implementation is complete with our first model training on our generated data-sets as we work on generating more sophisticated data to train a revised model for estimating circuit fidelity. Our primary areas of focus as we wrap up the project are as follows:

1. Data-set generation (Nearing Completion): Revising our existing framework for creating training data-sets to include new metrics. This includes outlier detection/pruning. Revising Equation 1 could also fall under here.

2. ML Model Design (In Progress): Implementing an intelligent neural network layout to predict our chosen metrics.

3. Front end Data Visualisation (Early Stages): Expand our current VSCode extension boiler plate to provide users with an effective interface to our system. Examine usage of simulation to provide probability distribution data for smaller circuits.