Quarry:
Providing Fast Quantum Circuit Fidelity Estimation

Hayden Coffey Kanghee Park Saurabh Kulkarni

***Abstract*—We present Quarry, an assistant tool for providing fast quantum circuit fidelity estimations across a variety of computing platforms. Quarry processes these estimated values and provides researchers with recommendations for what platforms their particular circuit would perform best on. Our design allows Quarry to use a variety of methods for determining these recommendations, and we explore the effectiveness and performance of these implementations in this report. Our code is available at https://github.com/hcoffey1/Quarry.**

I. Introduction

Noise is a dominant aspect of modern quantum computers. Hardware noise has a significant effect on the state fidelity of quantum circuits, and a great deal of research in the field of quantum computing has been dedicated to mitigating this noise. However, most mitigation strategies are limited by current technology [1], and it is difficult to predict how helpful they will be quantitatively. Learning to live with noise and understand its impact on quantum circuit fidelity is an active challenge to researchers in the field. With limited hardware availability and long queue times, researchers could benefit from having a rough understanding of how their circuit will perform on a given platform ahead of time. However, simulating a noise model involves a significant cost. The goal of this project is to provide a tool to help researchers and quantum programmers decide which quantum computing platform will yield the best results for their circuit by providing rough estimation of circuit fidelity on a variety of available platforms.

Our tool, Quarry, aims to assist researchers designing quantum circuits by providing real time fidelity estimates for circuits across a variety of available quantum platforms. We implemented and evaluated 3 primary designs for the tool: simulation, estimated success probability (ESP) [2] [3], and predictive (machine-learning based) models.

II. Related Work

Quantum simulation is a method commonly used to check the fidelity of the quantum circuit [4] [5]. However, quantum simulation is also known to be a challenging computational problem. Since the errors from real quantum systems are so complex, many simulators greatly simplify the problem. Nevertheless, it simulation requires maintaining large quantities of complex numbers with high precision, and the time required for simulation also increases significantly as the noise model becomes more complex and the number of qubits increases.

Some simulations approach this problem in a stochastic way [6]. Stochastic quantum simulation has the advantage that various noise sources caused by a real quantum system are applied, but it often requires unrealistically large amount of data to simulate all qubits.

Quantum tomography is also closely related to the fidelity estimation and simulation [7]. Quantum tomography has two categories: state tomography and process tomography. Quantum state tomography reconstructs quantum states using measurements on an ensemble of identical quantum states. On the other hand, quantum process tomography uses known quantum states to calculate the process. However, this approach has a state explosion problem: the number of possible states increases exponentially to the number of qubits and the depth of the circuit.

Another approach is to statically estimate the upper and lower bounds of fidelity [8]. However, the bounds suggested are often very wide, or the method requires very strong assumptions to estimate fidelity. Also, most of methodology is about the estimation of the circuit itself without considering platform specific information, such as the qubit topology.

An existing work uses a machine-learning based approach to evaluate the fidelity of a quantum state [9]. Instead of full reconstruction of its density matrix, the paper uses machine-learning methods to estimate fidelity of quantum state directly, by transforming quantum state fidelity estimation problem into a classification problem. It divides the quantum state space according to their fidelity, and use neural network to predict subspace which contains the quantum state. Our models instead focus primarily on the impact of platform hardware characteristics on quantum circuit fidelity.

III. Design

Figure 1 outlines the overall system design of Quarry. At a high level, the user provides a quantum assembly (Open QASM 2.0) file alongside an API framework which is capable of retrieving characterization data for a set of quantum platforms. Quarry will then select a configurable number of platforms that are compatible with the circuit (prioritizing smaller platforms) and perform an estimation of circuit performance on each platform. The results of these queries are then used to sort the platforms in order of effectiveness. This sorted list, alongside the estimated performance metrics, are displayed to the user. Our system design allows for us to change the implementation of the query method responsible for ordering platforms fairly easily. We discuss our different query method implementations in later sections including how they define what platform is most “effective”.

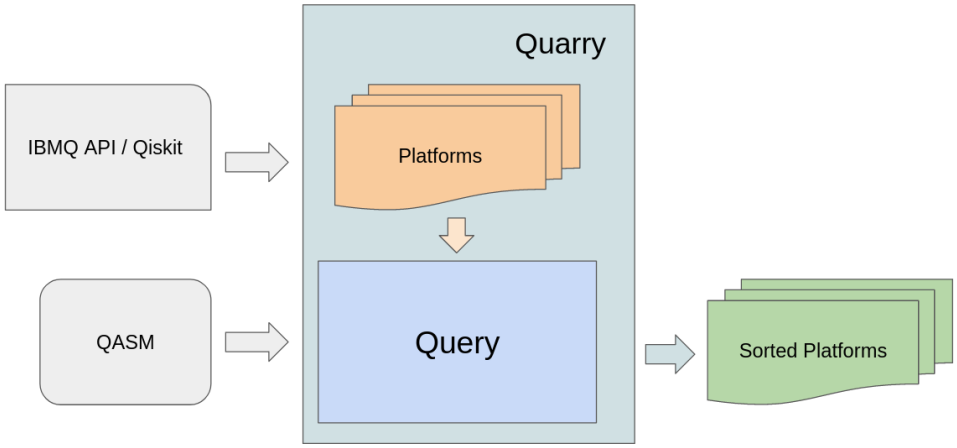

Fig. 1: Quarry system design overview.

In our work we utilize Qiskit with for its set of simulated platforms and IBMQ for access to more recent characterization data. Quarry is capable of detecting whether or not an IBMQ API token is present in the development environment and will prioritize IBMQ characterization data if the token is present. Our report focuses on the performance of Quarry with Qiskit simulation characterization data. We leave the evaluation of IBMQ's impact on system performance and accuracy as potential future work.

A. Simulation

Simulation of the quantum circuit on each platform is our first approach for estimating circuit performance. This method is the most straight forward to implement, but also the most naive, and suffers from the poor performance scaling as we increase the number of qubits and circuit depth. In exchange for high computational cost, however, we are able to collect a variety of metrics with reasonable (depending on the noise model) accuracy. With our current implementation, the metrics we collect are:

- Probability of Successful Trial (PST)
- Total Variational Distance (TVD)
- Entropy
- L2 Distance
- Hellinger Distance
- Compiler Inserted Swaps (From routing to platform topology)

We aggregate these results into a singular naive "fitness" metric (Equation 1) that can be used to sort the platforms.

$$\text{Fitness} = \frac{\text{PST}}{\text{TVD} + \text{Swaps} + \text{Hellinger} + \text{L2}} \tag{1}$$

While Equation 1 functions for our purposes of sorting the platforms, it does have several flaws which we elaborate on later in this report. Regardless, we treat the simulation results as a control set for our experiments when evaluating our other implementations.

Figure 2 shows the output displayed by Quarry when running with the simulation configuration. These results are done with a Qiskit optimization level of 0 in order to prioritize speed in obtaining results. Even with this setting, we can see that the system requires nearly a minute to obtain the 10 simulation results for the given 13 qubit circuit.

```
./qasm/QASMBench/medium/multiply_n13/multiply_n13.qasm 58.859384(s) ++++++++++++++
Backend Name        PST     TVD     Entropy   Swaps   L2      Hellinger Fitness
fake_rueschlikon    1.000   0.000   -0.000    63      0.000   0.000     0.159
fake_johannesburg   0.135   0.865   2.642     43      0.900   1.125     0.018
fake_cairo          0.163   0.837   2.501     67      0.878   1.092     0.017
fake_singapore      0.112   0.888   2.733     39      0.918   1.153     0.016
fake_boeblingen     0.110   0.890   2.734     41      0.920   1.156     0.015
fake_almaden        0.089   0.911   2.728     35      0.944   1.185     0.013
fake_guadalupe      0.111   0.889   2.667     58      0.924   1.154     0.012
fake_poughkeepsie   0.081   0.919   2.657     40      0.956   1.196     0.011
fake_tokyo          0.063   0.937   2.712     36      0.971   1.223     0.009
fake_melbourne      0.025   0.975   2.600     48      1.018   1.297     0.003
```

Fig. 2: Example simulation output.

Additionally, we observe one of the challenges presented to the system in the form of faulty characterization data as *fake_rueschlikon* displays highly improbable measurements for PST and state distances (these outlier values are not specific to the circuit tested). Identifying faulty simulations and filtering them from the output is an opportunity for future work.

B. Estimated Success Probability

Estimated success probability (ESP) [2] [3] provides an alternative method to ranking platforms without requiring simulation. ESP is calculated via Equation 2 where m^s and g^s are the success rates of measurements and gates present in the circuit.

$$ESP = \prod_{i=1}^{N_{gates}} g_i^s * \prod_{i=1}^{N_{meas}} m_i^s \tag{2}$$

By avoiding simulation, calculating ESP can be a substantially faster method for ordering platforms as compared to the approach detailed in Section III-A. However, as we can see in Figure 3, we now have access to less information describing the circuit. Additionally, this approach still requires routing of the circuit to the platform topology as it is dependent on qubit specific gate and measurement success rates.

```
./qasm/QASMBench/medium/seca_n11/seca_n11.qasm 8.389343(s) ++++++++++++++
Backend Name        ESP
fake_rueschlikon    1.000
fake_guadalupe      0.118
fake_singapore      0.010
fake_boeblingen     0.005
fake_cairo          0.002
fake_melbourne      0.000
fake_almaden        0.000
fake_poughkeepsie   0.000
fake_johannesburg   0.000
fake_tokyo          0.000
```

Fig. 3: Example ESP output.

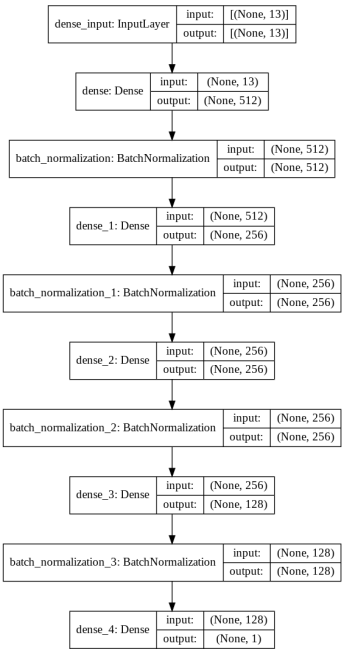

Fig. 4: V1 Predictor model structure.

C. V1 Predictor

The trade-off between computational performance and information exemplified by the simulation and ESP based approaches motivated us to explore alternatives that may offer a middle ground. Recent work [10] has used neural network models to estimate fidelity of quantum states. With machine-learning appearing to be a promising option, we implemented a neural network model, shown in Figure 4, designed to read in circuit and platform features and recreate the output

```
./qasm/QASMBench/medium/qf21_n15/qf21_n15.qasm 0.642284(s) ++++++++++++++
Backend Name        PST     TVD     Entropy   Swaps   L2      Hellinger Fitness
fake_johannesburg   0.516   0.650   0.000     0       N/A     N/A       0.195
fake_poughkeepsie   0.516   0.658   0.000     0       N/A     N/A       0.194
fake_almaden        0.516   0.658   0.000     0       N/A     N/A       0.194
fake_boeblingen     0.516   0.658   0.000     0       N/A     N/A       0.194
fake_singapore      0.516   0.663   0.000     0       N/A     N/A       0.194
fake_rueschlikon    0.516   0.667   0.000     0       N/A     N/A       0.194
fake_tokyo          0.516   0.673   0.000     0       N/A     N/A       0.193
fake_guadalupe      0.551   0.736   5.094     3       N/A     N/A       0.154
fake_cairo          0.551   0.737   5.000     3       N/A     N/A       0.153
fake_hanoi          0.551   0.730   5.094     3       N/A     N/A       0.153
```

Fig. 5: Example V1 predictor output. L2 and Hellinger distance models were not implemented at this point in the project.

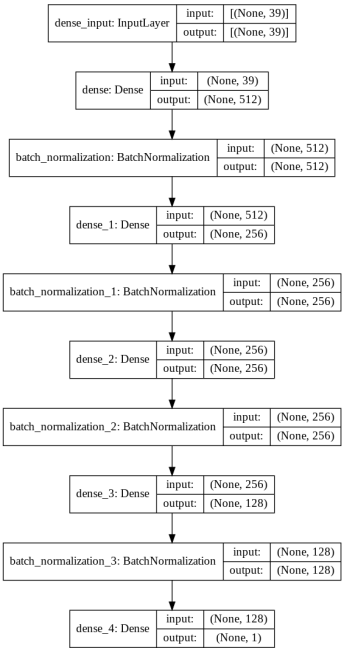

Fig. 6: V2 Predictor model structure.

features we originally collected via simulation. Training data was produced using simulation runs and a separate model was trained for each output feature. The model has 13 input features consisting of:

- IBM basis gate counts.
- Platform ID (Numerical representation)
- Platform topology average node degree.
- Platform qubit count.

Unlike ESP or simulation, this approach only requires the circuit to be unrolled to platform compatible basis gates. Figure 5 shows the output for a circuit not included in the training set. The primary performance bottleneck for the model is collecting the input features from a given quantum circuit. Due to the input features being few and easy to calculate, the model is very fast (often faster than ESP). However, the accuracy is poor on circuits not included in the training set. We believe one of the primary reasons behind the low accuracy of the model is poor feature coverage of the circuit and platform with our current model inputs. We next discuss our second iteration of the model and the improvements we made to its input features.

D. V2 Predictor

Figure 6 details the structure of the second version of the predictive model. The structure is nearly identical to the first iteration, but with the addition of 26 new input features comprised of:

- Average measurement and gate success rates.
- Topology graph metrics (density, connectivity, clustering, shortest path, etc.)
- QASMBench [11] circuit metrics (gate/measurement density, size factor, entanglement variance, etc.)

```
./qasm/QASMBench/medium/qf21_n15/qf21_n15.qasm 6.844908(s) ++++++++++++++
Backend Name        PST     TVD     Entropy   Swaps   L2      Hellinger Fitness
fake_guadalupe      0.880   0.611   0.000     0       0.145   0.000     0.337
fake_hanoi          0.880   0.611   0.000     0       0.145   0.000     0.337
fake_cairo          0.879   0.611   0.000     0       0.145   0.000     0.337
fake_rueschlikon    0.714   0.733   0.000     0       0.227   0.000     0.261
fake_poughkeepsie   0.696   0.860   0.000     0       0.107   0.000     0.243
fake_tokyo          0.665   0.860   0.000     0       0.084   0.000     0.233
fake_singapore      0.666   0.866   0.000     0       0.125   0.000     0.232
fake_johannesburg   0.666   0.866   0.000     0       0.106   0.000     0.232
fake_almaden        0.666   0.866   0.000     0       0.093   0.000     0.232
fake_boeblingen     0.666   0.870   0.000     0       0.107   0.000     0.232
```

Fig. 7: Example V2 predictor output.

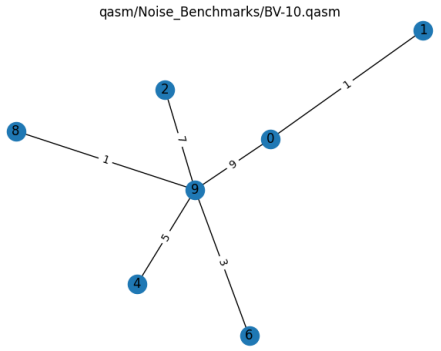

Fig. 8: CX graph of BV-10 circuit.

Figure 7 shows the output for the V2 predictor on the same circuit evaluated with V1 in Figure 5. We observe that the execution time has increased from under a second to approximately 7 seconds as our collected input features scale computationally with circuit size. However, we can now see a more distinct ordering to the platforms than what was shown with V1. We do find that this model tends to be optimistic about predictions and tends to produce faulty output on metrics such as swap counts.

E. SWAP Predictor

For our final implementation of Quarry, we focused on designing a model to sort platforms based on the number of swaps required to map the circuit to the platform topology. Our primary insight for this model is the inclusion of graph metrics (similar to those collected for platform topology) for

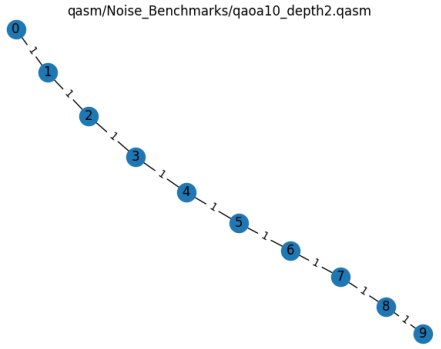

Fig. 9: CX graph of QAOA10, Depth=2 circuit.

```
./qasm/QASMBench/medium/qf21_n15/qf21_n15.qasm PredTime: 6.762290(s) ActTime: 7.773383(s)
Backend Name        Predicted Swaps     Actual Swaps
fake_tokyo          65                  54
fake_rueschlikon    84                  90
fake_singapore      91                  89
fake_boeblingen     96                  111
fake_guadalupe      98                  115
fake_almaden        103                 83
fake_hanoi          103                 119
fake_cairo          104                 133
fake_johannesburg   111                 114
fake_poughkeepsie   113                 110
```

Fig. 10: Example SWAP predictor output.

the CX graph of the circuit. We define a CX graph to be a graph where edges between nodes define the presence of a CX gate in the circuit between those two qubits. The weights of these edges are defined as the numerical distance between the qubits (i.e. an edge between qubits 9 and 5 would have a weight of $|9 - 5| = 4$). The reasoning behind this is that qubits numerically far apart tend to be physically far apart on a platform's topology. Figures 8 and 9 show example CX graphs for implementations of BV-10 and QAOA-10. We observe that the structure of these two graphs clearly differ from one another. As CX operations in a circuit often require the insertion of swaps when routing the circuit, features describing the layout of qubit interactions on the initial circuit provide meaningful input to a predictive model.

With the addition of CX graph metrics, our model now has 48 input features. As we are now only focused on predicting swaps, we no longer require a full simulation of a circuit to produce training data, allowing us to build a much larger training set. Figure 10 shows an example of the swap predictor being compared to the Qiskit compiler with an optimization level of 2. We observe in this example that the predictor is slightly faster than the compiler, but this could be further improved by narrowing the input feature selection. While these results seem to be the most promising in terms of accuracy, they are also our newest and from a circuit that was part of the training data set. We leave verifying the robustness of the model on untrained circuits to future work.

IV. Experimental Results

We performed our experiments on an AMD Ryzen 9 3950X 16-Core Processor. Table I details our initial system performance findings (runtime and memory usage) for our first 4 implementations of our query function on circuits from the SupermarQ [12] and QASMBench [11] benchmark suites. We make the following observations from this data:

1) Simulation scales poorly in both runtime and memory consumption (as expected).
2) ESP, Predictor V1, and Predictor V2 show reasonable scaling capabilities.
3) The predictive models have a relatively high fixed memory cost. This is due to loading the models into memory.

A. Wall Clock Time (Simulation, ESP, V1, V2)

With these observations in mind, we evaluate our system on a variety of circuits from the QASMBench [11] suite for small (2-5 qubit) and medium (6-15 qubit) circuits. As we

Circuit Name	Method	Wall Clock Time (s)	User Time (s)	System Time (s)	Peak RSS (KB)
qasm/SupermarQ/qaoa_fermionic_10.qasm	Simulation	18.025	113.93	5.98	927344
qasm/SupermarQ/qaoa_fermionic_10.qasm	ESP	8.3	13	2.5	717196
qasm/SupermarQ/qaoa_fermionic_10.qasm	PredV1	0.927	5.44	2.37	1623340
qasm/SupermarQ/qaoa_fermionic_10.qasm	PredV2	6.169	11.3	2.51	1722360
qasm/QASMBench/medium/qf21_n15/qf21_n15.qasm	Simulation	790.637	14758.46	9.43	1410568
qasm/QASMBench/medium/qf21_n15/qf21_n15.qasm	ESP	9.192	14.17	2.61	717368
qasm/QASMBench/medium/qf21_n15/qf21_n15.qasm	PredV1	1.080	5.49	2.45	1619380
qasm/QASMBench/medium/qf21_n15/qf21_n15.qasm	PredV2	7.046	12.39	2.57	1721648

TABLE I: Sample of system performance for query methods.

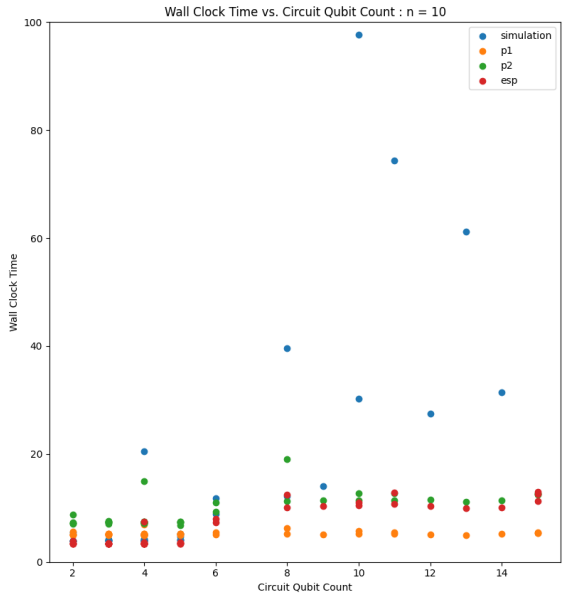

Fig. 11: System wall clock time vs circuit qubit count.

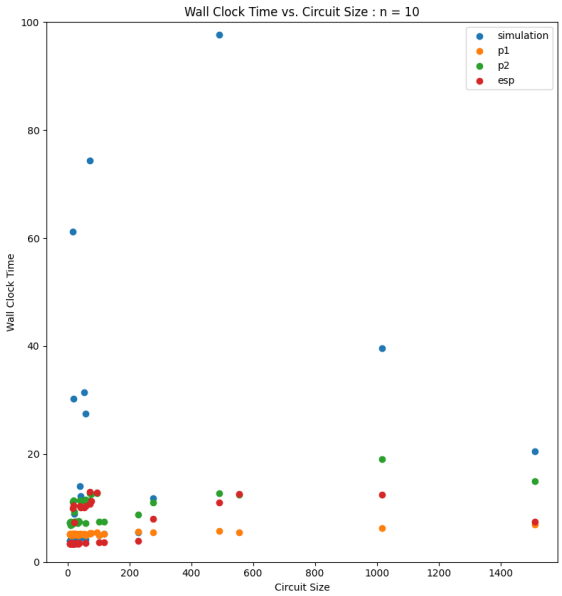

Fig. 12: System wall clock time vs number of gates in circuit.

are primarily concerned about observable latency, we plot the wall clock time against circuit qubit count for each of our 4 primary methods of estimation in Figure 11. Some simulation data points have been omitted from the graph for readability as they reach as high as 35 minutes. We observe again that simulation scales poorly in comparison to the other methods. Additionally, we observe from the data:

1) The V1 model performance is largely independent of qubit count. This is due to its small input feature selection that scales well with larger circuits.
2) ESP outperforms the V2 model on smaller circuits (less than 8 qubits), but then tends to have similar performance on larger (qubit wise) circuits.

Figure 12 shows the same data but plotted against circuit size (number of gates). We see a limitation with our evaluation as our data points are primarily from smaller circuits. However, for the data we collected, we find the typical order of the methods in regards to wall clock performance to be:

1) V1 Predictor (Best)
2) ESP
3) V2 Predictor
4) Simulation (Worst)

It should be noted that the performance overhead of simulation is far greater than what wall clock time would imply as the simulation process is highly parallelized and will utilize every core allocated to it. As our goal with the predictor model was to achieve a middle ground in terms of performance between ESP and simulation, these results are acceptable.

B. Wall Clock Time (SWAP Predictor)

Figure 13 compares the runtime performance (wall clock) of the SWAP predictor model and the Qiskit compiler (optimization level 2). While the predictor can sometimes outperform the compiler on larger circuits, it usually performs worse than actually routing the circuit onto the topology. These results prevent the model from being useful, as it is currently more expensive to obtain an estimate than calculate the actual answer to the number of swaps inserted. The model is largely hindered by unnecessary input features that are expensive to obtain. For instance, while gate error rates may have an impact

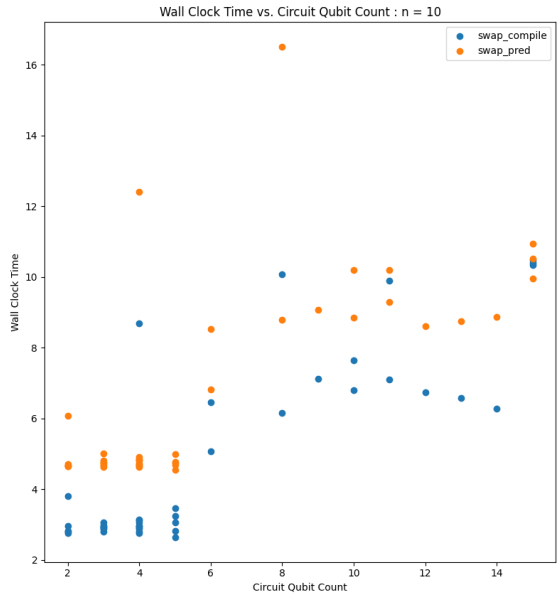

Fig. 13: System wall clock time vs circuit qubit count. (SWAP Predictor)

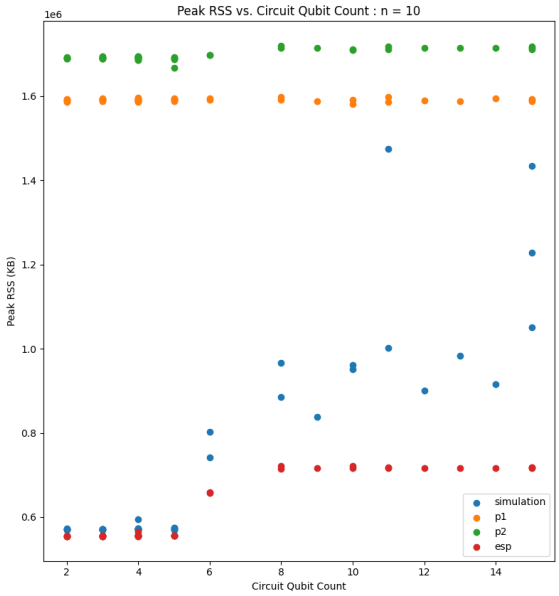

Fig. 14: System peak memory usage vs circuit qubit count.

on the routing if the compiler is noise aware, they could be ignored in order to improve runtime performance.

C. Peak Memory Usage

Figures 14 and 15 detail the peak resident set size (PRSS) of the runs performed for each circuit across the variety of estimation methods implemented. We observe simulation suffers from poor memory scaling as qubit count increases, meanwhile, the predictive models maintain largely fixed memory footprint sizes. ESP and the Qiskit routing algorithm share a similar memory usage pattern across the circuits as ESP requires routing in order to calculate qubit specific success rates for the given circuit.

Our main takeaway from these charts is that the predictive models will not provide the same savings in memory usage as they did for runtime when compared to simulation. Running tests on circuits larger than 15 qubits may result in simulations that exceed the memory cost of the predictive models, but are left for future work.

D. Accuracy

Both V1 model and V2 model showed reasonable accuracy on circuits similar to the training data. Here, similar circuits mean that same circuits transpiled to different platforms. As the training loss decreased, the validation loss also decreased reasonably. Tables II and IV show accuracy on similar circuits for the V1 and V2 models.

However, the V1 model failed to predict the metric of new circuits at all as seen in Table III. On the other hand, the V2 model did tend to overestimate the target values (Table V), but

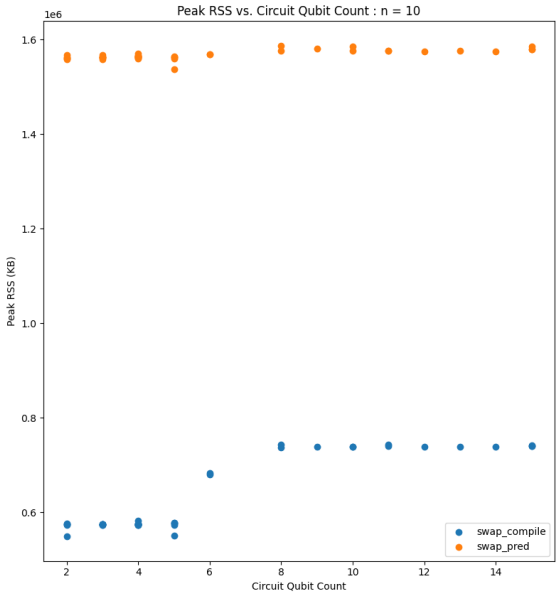

Fig. 15: System peak memory usage vs circuit qubit count. (SWAP Predictor)

Predicted TVD	Actual TVD
0.510997	0.526367
0.041219	0.078125
0.060580	0.058594
0.643715	0.949414
0.132602	0.142578

TABLE II: V1 predictor with similar circuits

Predicted TVD	Actual TVD
0.134381	0.612305
0.083689	0.344727
0.144355	0.000977
0.191454	0.465820
0.175614	0.560547

TABLE III: V1 predictor with new circuits

Predicted TVD	Actual TVD
0.741174	0.965625
0.696171	0.649414
0.862628	0.898438
0.865645	0.875000
0.663417	0.703125

TABLE IV: V2 predictor with similar circuits

Predicted TVD	Actual TVD
0.536968	0.139695
0.625335	0.306641
0.407491	0.086914
0.358605	0.133789
0.439015	0.068359

TABLE V: V2 predictor with new circuits

the output did display a correlation with the actual simulated values. Running more experiments with categorized circuits may give better results, but we were unable to do so due to time constraints.

V. Design Evaluation

We now examine the effectiveness and limitations of the current system.

A. Simulation

The simulation implementation of Quarry could be improved in a number of ways. First, Equation 1 is a naive design and could be revised. Normalizing the data passed in and implementing detection for faulty inputs from bad characterization data would lead to a more robust system. Our features used to calculate fitness are also highly correlated. Finding new features that provide a wider range of information would improve the effectiveness of the platform recommendations.

B. ESP

One of the largest questions for this work is, "Why not just use ESP for ranking?" As we have seen previously, ESP scales well in both memory usage and runtime with increasing number of circuit qubits. We also find it to be correlated with our output features as shown in Table VI. If we further optimize the code used to calculate ESP, we would have access to a metric that is ideal for our use case. However, despite these benefits, there are some reasons to consider utilizing a predictive model instead of (or in conjunction) with ESP:

Metric	Spearman Correlation	PValue
PST	0.473	0.000
TVD	-0.708	0.000
Entropy	-0.637	0.000
Swaps	-0.769	0.000
L2	-0.525	0.000
Hellinger	-0.551	0.000

TABLE VI: Spearman correlation coefficient of output features and ESP.

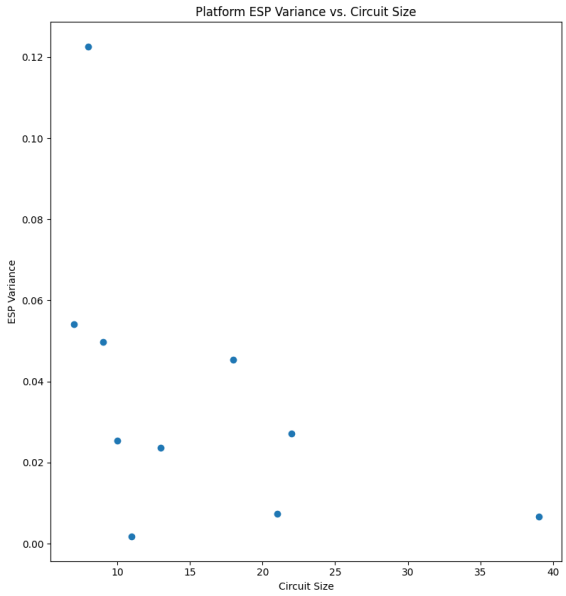

Fig. 16: ESP variance vs quantum gate count.

1) ESP provides less information than simulation or a model that can estimate simulation results. While ESP is correlated with our other metrics, a researcher may benefit from having access to more fine-grained features.
2) ESP appears to be less effective for distinguishing platforms as circuit size increases. Figure 16 details the variance of ESP measurements across the tested platforms. We have found on larger circuits most platforms yield an ESP of approximately 0.

We believe these reasons provide justification for pursuing the development of a predictive model capable of estimating quantum circuit fidelity.

C. Predictive Models

The low accuracy of the currently implemented predictive models on new circuits is one of the largest challenges for the current system. Training data is produced via simulation, limiting our work to fairly small data sets due to the high cost of simulation. The overhead from collecting the input features in test cases further limits the effectiveness of the models (especially in the case of the SWAP predictor). Redesigning

the network layout and pruning unnecessary input features could lead to better model performance.

VI. Future Work

Quarry at its current stage can function as a basic predictor, but it still needs improvement.

The algorithm for selecting platforms to evaluate a given quantum circuit on could be revised. The current naive approach is to select the smallest n platforms that are large enough for the circuit. There may be a more sophisticated way to approach this problem, such as selecting machines based on current availability and priority given to machines with a known reputation for accurate hardware.

As we have discussed previously, Equation 1 should be revised so that it better represents performance. Fitness is used as the standard for sorting machine preferences, and its current definition is a simple combination of metrics that are correlated with one another. It is difficult to say that this measure represents performance very well. Additionally, finding the best scaling coefficients for the formula that give a suitable measure of performance would allow us to fine tune the system's recommendations. As an alternative, we may give the choice to users so that they can adjust the effect of each metric to best suit their needs.

Another potential avenue for future work is exploring the impact of the circuit optimization level on wall clock time. It may be more efficient to spend time optimizing a circuit in the compiler stage in order to simulate a smaller circuit later. Additionally, accounting for compiler optimization levels when making recommendations presents a large area of potential work.

The neural network design needs improvement. The current implementation uses input features chosen manually, and these features may lose many important properties that we didn't consider. To obtain better features for circuit, we may use an encoder model to transform quantum circuit into a feature vector. We may also introduce Graph Nearning Network (GNN) instead of dense network. A typical neural network is not suitable for unstructured data, such as circuits or graphs, because they use fixed size input features. However, GNN has an appropriate structure to learn unstructured data that is more suitable for our purpose.

The way in which we query the predictive models could be further optimized. Currently the input features are recalculated for every query (i.e. querying 10 platforms involves generating 10 sets of input features). Identifying which features are consistent across platforms would allow us to calculate them only once and re-use them for subsequent queries.

We have observed that ESP appears to be an effective metric for ordering platforms. However, it lacks the full scope of information provided by simulation. Designing a system that can provide a fast initial estimate of platform ordering using ESP and then later refine the results via simulation or machine-learning would allow us to combine the strengths of the different methods we have explored.

Finally, the goal of this project was to create a Visual Studio Code plugin to function as the front end to the tool. While focusing on the predictor itself, the priority of plugin implementation has been pushed back. However, a plugin that is capable of showing the fidelity of the code being written in real time would be of great help to researchers.

VII. Conclusion

Quarry aims to assist researchers designing quantum circuits by providing real time fidelity estimates for circuits across a variety of available quantum platforms. We implemented and evaluated 3 primary designs for the tool: simulation, ESP, and predictive (machine-learning based) models.

We find simulation to be too computationally intensive for larger quantum circuits, and ESP to be too limited in scope of information provided. Additionally ESP struggles to distinguish platform performance for large circuits. We turn to predictive models as an alternative that provides a middle ground between ESP and simulation.

Our predictive models show reasonable accuracy and latency for circuits similar to previously trained circuits. However, for new circuits, the predicted values do not match well. In our initial model (V1), there was no correlation between the actual value and the predicted value, so it was not usable at all. After adding more advanced input features to the second iteration (V2), we were able to make some simple comparisons between platforms, however, the predicted value on each metric was not very accurate.

As it stands now, we believe that the current tool can be used for simple comparison if there is a sufficiently sized data set available to train the model. We were unable to generate enough data as noisy simulations took too much time. Since the model showed high accuracy for circuits similar to previous circuits, we expect that the model can be trained with a reasonable size of data set if we can divide categories of circuits.

We believe that these results show promise, and with further revision of the model and input features, we may be able to achieve a system capable of effectively estimating platform performance for a given circuit.

References

[1] D. Monroe, "Closing in on quantum error correction," *Commun. ACM*, vol. 62, p. 11–13, sep 2019.

[2] S. Nishio, Y. Pan, T. Satoh, H. Amano, and R. V. Meter, "Extracting success from ibm's 20-qubit machines using error-aware compilation," *J. Emerg. Technol. Comput. Syst.*, vol. 16, may 2020.

[3] S. S. Tannu and M. Qureshi, "Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, (New York, NY, USA), p. 253–265, Association for Computing Machinery, 2019.

[4] I. Georgescu, S. Ashhab, and F. Nori, "Quantum simulation," *Reviews of Modern Physics*, vol. 86, pp. 153–185, mar 2014.

[5] K. Georgopoulos, C. Emary, and P. Zuliani, "Modeling and simulating the noisy behavior of near-term quantum computers," *Phys. Rev. A*, vol. 104, p. 062432, Dec 2021.

[6] T. Grurl, R. Kueng, J. Fuß, and R. Wille, "Stochastic quantum circuit simulation using decision diagrams," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 194–199, 2021.

[7] G. M. D'Ariano, M. G. A. Paris, and M. F. Sacchi, "Quantum tomography," 2003.
[8] M. Cerezo, A. Poremba, L. Cincio, and P. J. Coles, "Variational quantum fidelity estimation," *Quantum*, vol. 4, p. 248, mar 2020.
[9] X. Zhang, M. Luo, Z. Wen, Q. Feng, S. Pang, W. Luo, and X. Zhou, "Direct fidelity estimation of quantum states using machine learning," *Physical Review Letters*, vol. 127, sep 2021.
[10] X. Zhang, M. Luo, Z. Wen, Q. Feng, S. Pang, W. Luo, and X. Zhou, "Direct fidelity estimation of quantum states using machine learning," *Physical Review Letters*, vol. 127, no. 13, p. 130503, 2021.
[11] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, "Qasmbench: A low-level qasm benchmark suite for nisq evaluation and simulation," *arXiv preprint arXiv:2005.13018*, 2021.
[12] T. Tomesh, P. Gokhale, V. Omole, G. S. Ravi, K. N. Smith, J. Viszlai, X.-C. Wu, N. Hardavellas, M. R. Martonosi, and F. T. Chong, "Supermarq: A scalable quantum benchmark suite," 2022.