On the predictability of execution performance of quantum circuits across inter-device and intra-device layouts

Ishant Kohar[1, *], **Anupama Ray**[2], **Ritajit Majumdar**[2]

[1]Netaji Subhash University of Technology
[2]IBM Quantum, IBM Research India
[*]ishant.ug21@nsut.ac.in

Abstract

Quantum computing presents exciting possibilities but faces challenges due to hardware noise and variability. Moreover, the users are often waiting for long in queue for their job to execute. It will beneficial for the users if they can execute on a hardware having lesser queue, thus saving time, and then predict the computation outcome is some other device which may have longer queue. On the other hand, a user may also save time by simultaneously executing multiple circuits on different layouts of the same hardware. This also has the trade-off since the layouts of a hardware vary in noise profile, and not all circuits can be simultaneously executed on the best layout of that hardware. In this paper, we investigate the possibility of predicting the computation outcome of a quantum circuit on a layout l_2, when it was originally executed on a layout l_1, which may or may not belong to the same hardware. Our results show that such a predictability is difficult for inter-device scenario, and holds only for few cases. On the other hand, our initial result is promising when both the layouts belong to the same hardware.

Introduction

Quantum computing stands at the forefront of technological innovation, offering solutions to problems that are beyond the reach of classical computing methods. By harnessing the principles of quantum mechanics, quantum computers can potentially perform complex calculations at unprecedented speeds. Despite these advancements, the current generation of quantum devices faces significant challenges, primarily due to inherent noise and variability in hardware characteristics.

One major issue in quantum computing today is the selection of optimal hardware amidst varying noise profiles. Quantum devices differ widely in their noise characteristics, introducing errors into computations and impacting their reliability. Users often confront a critical decision: whether to wait for access to hardware with lower noise levels or to utilize sub-optimal devices that offer shorter wait times. This decision is complicated by the need to balance result accuracy with hardware availability.

Our first hypothesis addresses the challenge of predicting the deviation of results on one quantum hardware based on results obtained from another. Specifically, we aim to develop a service that, by knowing the results on a sub-optimal hardware, can predict the deviation of results on optimal hardware. This approach will provide a trade-off between time and result accuracy, enabling users to make informed decisions.

To illustrate, consider a user selecting a quantum backend: the one with the lowest noise profile might have a long queue of hours, causing delays. On the other hand, a noisier backend may offer quicker result due to lower queue, but with reduced accuracy. Our service will allow the user to choose whether to wait for a better device or to get results on a sub-optimal one. This service aims to enhance user experience and efficiency in utilizing quantum computing resources by providing insights into the potential deviations between different hardware platforms. We will utilize the Quantum Approximate Optimization Algorithm (QAOA) Path Circuit as a test case for our example. This circuit is hardware-native and requires no extra swap gates for mapping onto the hardware.

Our second hypothesis focuses on scheduling multiple quantum circuits on a hardware device to enhance the efficiency of quantum computers and improve user experience. Figure 1 illustrates this problem with a user running multiple quantum circuits in parallel on non-overlapping layouts to optimize hardware use.

Previous work has explored packing multiple circuits on quantum hardware, understanding that not all circuits can be given the mapped to the best layout. We pursue this problem by considering a scenario where a user is strict about the fidelity of their results, allowing up to some fixed degradation of ϵ. In this case, we choose the top k% of layouts to run our circuits on, ensuring that the fidelity deviation remains under ϵ. For example, consider two layouts, l_1 and l_2. We measure the percentage degradation from l_1 to l_2 when both circuits are run individually on a backend, and then measure the percentage degradation when both circuits are packed together on l1 and l2, respectively.

We aim to establish a relationship between the percentage degradation when circuits are run individually and when they are packed together. By knowing the degradation individually, we can estimate the deviation if multiple circuits are scheduled on the same hardware, thereby saving time while maintaining acceptable fidelity levels.

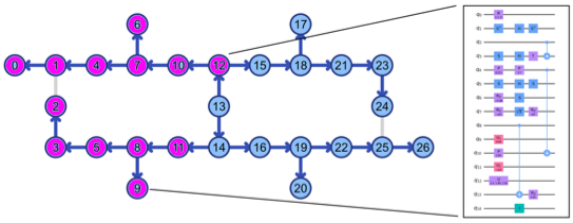

Figure 1: An example of a 15-qubit circuit assigned to a 27-qubit hardware. The used qubits are shown in purple while the unused qubits are shown in blue. The hardware still has room to accommodate one or more quantum circuit(s) using the free qubits. This figure is taken from (Bhoumik, Majumdar, and Sur-Kolay 2024)

This paper systematically evaluates the trade-offs between increased execution speed and quality degradation, providing a comprehensive analysis to optimize quantum computing resource utilization.

Background Study

QAOA

The Quantum Approximate Optimization Algorithm (QAOA) (Farhi, Goldstone, and Gutmann 2014) is a hybrid quantum-classical algorithm used for solving combinatorial optimization problems. It operates by alternating between two Hamiltonians for multiple steps. The number of steps is usually denoted as p. The depth of the circuit, is proportional to p. In the absence of noise, as p increases, the algorithm can explore a larger solution space, leading to non-decreasing improvement. However, in noisy scenario, increasing p can even lead to worse results due to the depth of the circuit.

In this study, we use QAOA for the n-qubit path graph, and call it the QAOA path circuit henceforth. It is hardware-native, meaning it does not require swap gates to map it onto quantum hardware. This design reduces error rates and circuit depth, leading to more efficient and accurate quantum computations.

Transpilation

Transpilation ensures quantum circuits are optimized for the target hardware's capabilities and constraints. This involves several key steps: circuit optimization, which focuses on reducing gate count and depth; mapping to hardware, which involves aligning with qubit connectivity; layout selection, which assigns logical to physical qubits to minimize extra operations; and error mitigation, which applies error correction for increased reliability. Overall, transpilation ensures that quantum circuits are effectively tailored for the capabilities and limitations of the target quantum hardware.

Mapomatic

Mapomatic (Nation and Treinish 2023) addresses the challenge of selecting the optimal qubit mapping to minimize noise and improve performance on IBM Quantum hardware. Traditionally, choosing the best qubit layout involves an initial selection based on noise characteristics followed by SWAP mapping to adapt the circuit to the hardware's connectivity constraints.

The process involves several steps. First, a post-compilation routine reconfigures the circuit to match the two-qubit gate structure with a specific sub-graph of the target quantum system. Next, the VF2 mapper (via Qiskit's retworkx) performs a sub-graph search to identify and rank sub-graphs based on error rates from system calibration data. Finally, optimal qubit selection determines the best qubit layout for a single system or among multiple systems to achieve superior performance.

Layouts: In quantum computing, layouts determine the arrangement of qubits on a device. Optimal layouts align circuit operations with hardware connectivity, reducing the need for additional SWAP gates and improving execution efficiency.

Mapomatic Score(*layout score*): Mapomatic automates the selection of the best qubit layout by evaluating and ranking layouts based on error rates from device calibration. Lower scores indicate better performance, helping to identify the most efficient arrangement for executing quantum circuits.

Fake Backends

Fake backends in Qiskit are virtual simulators that emulate real quantum hardware. They mimic qubit connectivity, error rates, and decoherence times, allowing developers to test and debug circuits without actual quantum devices. Examples include FakeWashington, FakeCairo, and FakeManhattan. These simulators enable validation and optimization of quantum circuits in a controlled environment, ensuring thorough testing before deployment on real quantum hardware.

CatBoost

CatBoost (Dorogush, Ershov, and Gulin 2018) is a gradient boosting library by Yandex that efficiently handles categorical features. It converts categorical values to numerical while preserving structure and reducing overfitting.

XGBoost

XGBoost (Chen and Guestrin 2016) is a gradient boosting library known for its speed and efficiency. It effectively manages large and high-dimensional datasets, utilizing L1 and L2 regularization to improve model generalization and reduce overfitting.

RandomForest

The RandomForest Regressor aggregates predictions from multiple decision trees to enhance accuracy and stability. By averaging results, it reduces overfitting and improves performance, making it less sensitive to noise in the data.

Related Work

First Hypothesis

Our first hypothesis suggests that the fidelity of quantum computations decreases with the Mapomatic score and that this trend is consistent across different hardware platforms.

While specific studies on this hypothesis are lacking, related research in quantum error correction and optimization has explored how hardware characteristics impact computation fidelity. These studies provide insights into hardware noise effects, but a comprehensive analysis of the Mapomatic score's correlation with fidelity across various hardware remains unexplored. Our research aims to address this gap.

Second Hypothesis

Recent advancements in quantum computing emphasize the importance of optimizing qubit arrangements and scheduling to improve performance and minimize noise in Noisy Intermediate-Scale Quantum (NISQ) devices. Significant contributions include:

Resource-aware Scheduling: Research by (Bhoumik, Majumdar, and Sur-Kolay 2024) highlights the benefits of nearest-neighbor configurations to reduce cross-talk and qubit wastage, demonstrating improved hardware efficiency.

Community Detection Assisted Partition (CDAP): The CDAP algorithm (Liu and Dou 2020) improves throughput and resource utilization by partitioning qubits based on topology and error rates, reducing SWAP overheads. We align with this approach by optimizing circuit layouts.

Quantum Multi-programming Compiler (QuMC): The QuMC framework (Niu and Todri-Sanial 2023) addresses quantum hardware under-utilization through parallelism management and qubit partitioning algorithms. Our research complements this by focusing onA circuit layouts and integrating multi-programming insights to enhance throughput.

Our research integrates insights from these studies to optimize QAOA circuits and address hardware constraints, aiming to enhance quantum computing performance and hardware utilization.

First Hypothesis(*Inter-backend*)

Methodology

In this study, we employ two types of quantum circuits, the Quantum Approximate Optimization Algorithm (QAOA) Path circuit and the QAOA ComputeUncompute circuit, to optimize our quantum computations and evaluate the performance of quantum hardware.

QAOA Path Circuit This circuit operates directly on the path graph structure, taking advantage of the hardware's inherent connectivity. The performance of the QAOA Path circuit is governed by two sampling parameters: γ (gamma) and β (beta).

QAOA ComputeUncompute Circuit In contrast, the QAOA ComputeUncompute circuit is used for scenarios where we can sample parameters randomly. This circuit is advantageous because its ideal expectation value is zero; otherwise, the parameters would directly influence the outcome if the uncompute stage were not included.

The QAOA ComputeUncompute circuit comprises two main stages:

- **Compute Stage:** This stage involves applying problem-specific unitary operations to prepare the quantum state in accordance with the objective function. The operations applied during this stage are designed to encode the problem constraints into the quantum state.
- **Uncompute Stage:** In this stage, the unitary operations applied during the Compute stage are undone. This step simplifies the quantum state by removing the intermediate computational details, retaining only the information necessary for measurement. To ensure accurate results and prevent Qiskit from returning zero, a barrier is introduced between the Compute and Uncompute stages. This barrier ensures that no unintended operations occur during the transition between the two stages.

For the ComputeUncompute circuit, the ideal outcome is $00\ldots0$. When measuring Z-type observables, where Z is the Pauli-Z operator, this ideal state should yield an expectation value of +1. This is because all qubits are in the $|0\rangle$ state, and the Z operator has eigenvalues of +1 and -1, corresponding to the $|0\rangle$ and $|1\rangle$ states, respectively. Thus, the maximum possible expectation value of +1 is achieved for Z-type observables in the ideal state.

Dataset Generation

QAOA Path Circuit To generate the dataset for the QAOA Path circuit, we perform the following steps:

1. **Setup and Parameters:** We vary the number of qubits (n_qubits) and the parameter p (number of layers) across different fake backends. For each configuration, we calculate both the Mapomatic score and the expectation value. We use Qiskit estimator to calculate expectaion value
2. **Dataset Features:** The dataset comprises the following columns:
   - n_qubits: Number of qubits used in the circuit.
   - p: Number of layers in the QAOA Path circuit.
   - `best_layout`: Optimal qubit layout for the given configuration.
   - `Mapomatic score`: The Mapomatic score, which assesses the quality of the qubit layout.
   - `Expectation value`: The expectation value obtained from the circuit execution.
   - `Fake_Backend`: The simulated backend used for the circuit execution.
   - `Observable`: The observable used to measure the circuit output.
3. **Observables Used:** We employ the following observables to evaluate the performance of the QAOA Path circuit:
   - `ZZ`: Measures the correlation between the Z components of two qubits.
   - `XZ`: Measures the correlation between one qubit's X component and another qubit's Z component.
   - `YZ`: Measures the correlation between one qubit's Y component and another qubit's Z component.
   - `XX`: Measures the correlation between the X components of two qubits.

The observables are always between neighbouring qubits. Example for ZZ Observable: For a system with 6 qubits, the observables for the ZZ operator include:

ZZIIII, IZZIII, IIZZII, IIIZZI, IIIIZZ

These represent weight-2 observables, as they involve the interaction between two qubits.

For QAOA Path ComputeUncompute Circuit We generate a dataset using the QAOA Path ComputeUncompute Circuit, varying the number of qubits and p on different fake backends, and calculate the Mapomatic score and expectation value.

Dataset Features:

- n_qubits
- p
- *best_layout*
- *Mapomatic score*
- *Expectation value*
- *Fake_Backend*

Experiments

QAOA Path Circuit We conducted several experiments using machine learning models to predict the expectation value based on different input features.

multirow

Results and Discussion

Experiment Analysis **Experiment 1:** The initial results from Experiment 1 were not satisfactory. To better understand the underlying issues, we conducted a detailed analysis using a correlation matrix. The correlation matrix, shown in **Figure 2**, provided insights into the relationships between different features and their impacts on the model's performance. The matrix helped identify potential areas for improvement and guided further experimentation.

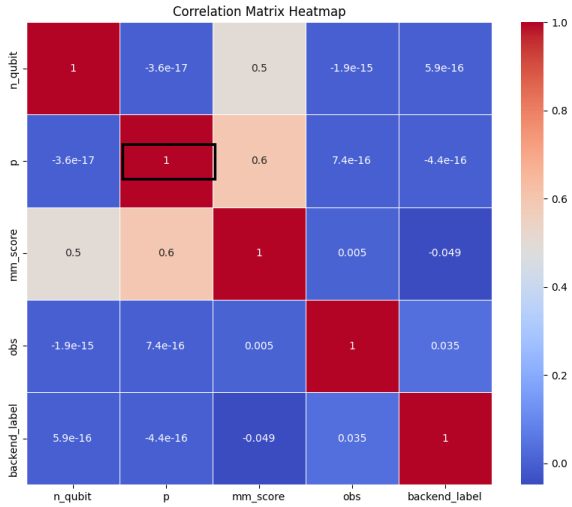

Figure 2: Correlation Matrix

Experiment 2: In this experiment, we applied one-hot encoding to backend and observable labels as input features. This approach significantly improved the results, highlighting the importance of encoding categorical variables appropriately. However, it was observed that observable labels could not be used as generalized features across all experiments, suggesting that their utility is context-dependent.

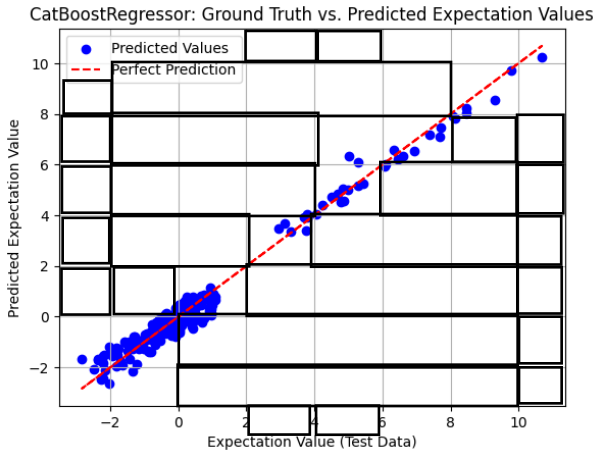

Figure 3: Experiment 2

Experiments 3 and 4: The findings from Experiments 3 and 4 indicated that the impact of the backend on model performance was less significant compared to the observable label. This was evidenced by the R-squared scores:

- Training with observable: R^2 score = 0.9486
- Training with backend: R^2 score = 0.2487

These results demonstrate that observable labels provide more meaningful contributions to the model's explanatory power than the backend information alone.

Experiment 5: Despite achieving a high R-squared score of 0.8617 in Experiment 5 in **Figure 4**, the presence of outliers and the proximity of expectation values to zero resulted in a mean square error of less than one. This indicates that while the model performed well in terms of variance explained, the low mean square error suggests potential issues with scalability and reliability when applied to broader datasets.

In summary, the experiments reveal the varying impacts of different features on model performance and highlight the importance of feature selection and data preprocessing in achieving robust results. The use of correlation matrices, appropriate encoding techniques, and careful consideration of feature contributions are crucial for optimizing quantum circuit analysis and performance prediction.

Mapomatic Score vs. Expectation Value Analysis QAOA Path Circuit

Figure 5 illustrates the relationship between Mapomatic score and expectation value for fake_washington with varying p values and different observables.

To understand the uneven trends, we compared ideal expectation values (calculated using the StatevectorSampler

	Input Features	Best Model	MSE	R2 Score	MAE
Trained on Combined Data	Number of Qubits p Mapomatic Score	CatBoost	1.9322	0.2545	0.8002
	Number of Qubits p Mapomatic Score Observable Backend	**CatBoost**	**0.0674**	**0.9740**	**0.1700**
	Number of Qubits p Mapomatic Score Observable	XGBoost	0.1332	0.9486	0.1982
Trained on (6,12) Tested on (13,15)	Number of Qubits p Mapomatic Score Backend	CatBoost	1.9470	0.2487	0.8049
	Number of Qubits p Mapomatic Score Observable Backend	RandomForest	0.5500	0.8617	0.3651

Table 1: Experiments on QAOA Path Circuit data

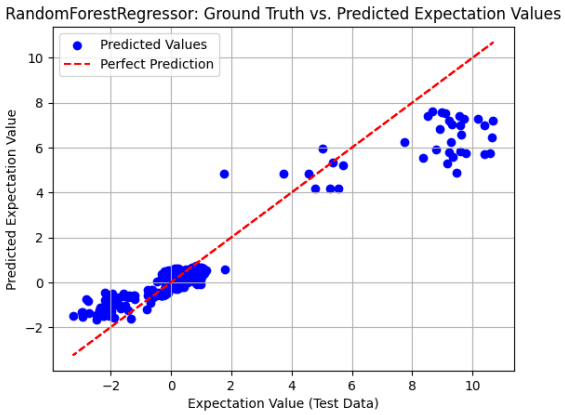

Figure 4: Experiment 5

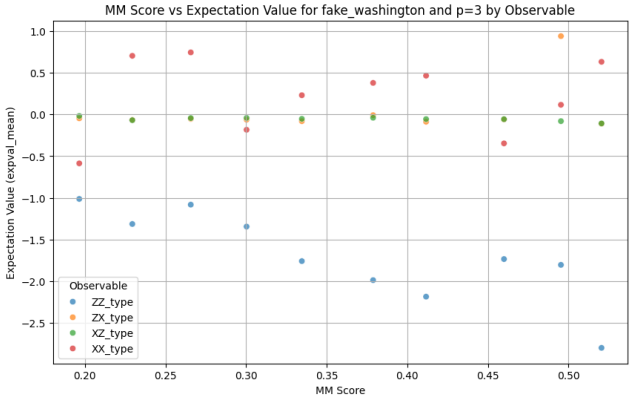

Figure 5: Mapomatic score vs Expectaion value(p=3)

estimator) and noisy expectation values (obtained from simulations on the fake backend).

We also analyzed the expectation value trends for different backends and p values: In figure 6, we have ideal vs noisy expectation value data

QAOA ComputeUncompute Circuit

In this section, we explore the relationship between the Mapomatic score and the expectation value across various fake backends for the *QAOA* at $p = 1$.

Figure 7 illustrates the Mapomatic score plotted against the expectation value for all fake backends when $p = 1$. This plot provides a comprehensive view of how the qubit layout impacts computational accuracy across different hardware simulations. In **Figure 8**, a notable pattern emerges in the form of a polynomial fit across different values of p ranging from 1 to 5. This observation suggests a potential correlation between the Mapomatic score and the expectation value, which warrants further investigation.

Curve Fitting Analysis on Fake Brooklyn In **Figure 8**, we focus on the specific case of the *fake_brooklyn* backend. A linear fit was applied to the data for $p = 1$ in **Figure 9**, revealing a close alignment between the observed values and the linear model. The mean square error (MSE) for this fit was calculated to be 0.01185, indicating a strong correlation and suggesting that a linear relationship adequately describes the dependency between the Mapomatic score and the expectation value for this backend.

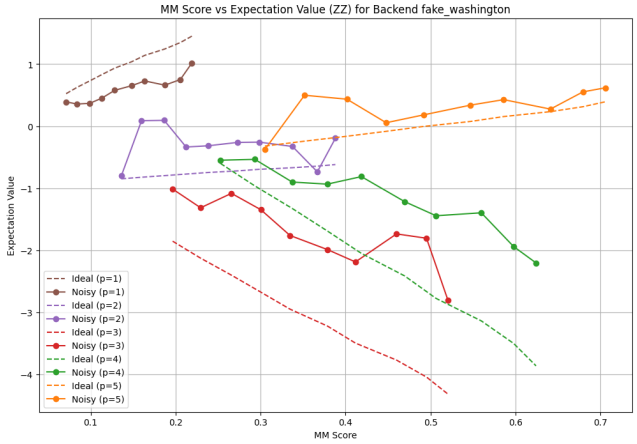

Figure 6: Plot for fake_washington with p = 1, 2, 5 show normal trends, whereas p = 3, 4 have low ideal expectation values.

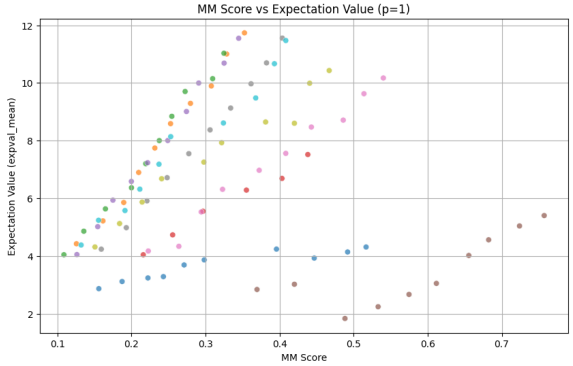

Figure 7

Fitting curve on Fake Mumbai Extending the analysis, we applied the *fake_brooklyn* polynomial curve fit to another backend, *fake_mumbai*. Interestingly, the fit yielded a mean square error (MSE) of 2.76193. This higher MSE value compared to *fake_brooklyn* indicates that the relationship between the Mapomatic score and the expectation value might be more complex on *fake_mumbai*, possibly due to different hardware characteristics simulated in this backend. Despite the larger error, the fit remains reasonable, suggesting that our hypothesis of consistent behavior across hardware platforms holds to some extent.

Interpretation of Results The varying fit quality between *fake_brooklyn* and *fake_mumbai* highlights the nuanced impact of hardware topology and noise characteristics on QAOA performance. While the linear fit's success on *fake_brooklyn* suggests that simple models may suffice to predict the behavior of quantum circuits for certain hardware configurations, this is not universally applicable.

These findings suggest that our first hypothesis—developing a service that predicts the deviation of results on optimal hardware based on sub-optimal hardware results—is only applicable to a very limited extent. This is

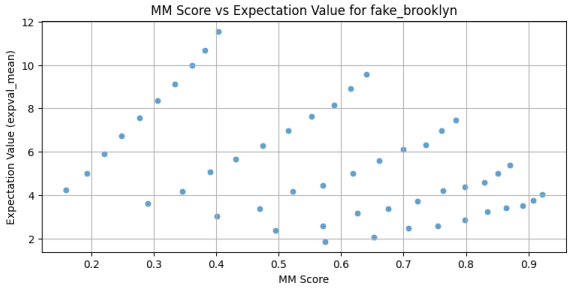

Figure 8

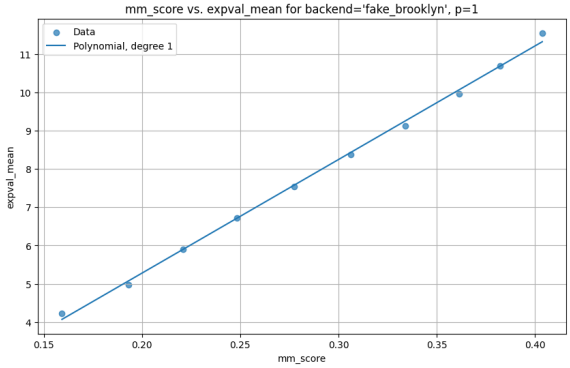

Figure 9: *Linear fit on fake_brooklyn*

particularly evident in the analysis of the $p = 1$ data. As shown in **Figure 8**, the generalized relationship may not hold universally, indicating that while the hypothesis might work under specific conditions, such as when p is constant, it does not extend to broader scenarios.

Second Hypothesis(*Intra-backend*)

Methodology

Isomorphic Layouts and Qubit Mapping Isomorphic layouts in quantum circuit mapping involve identifying equivalent sub-graphs on different quantum hardware systems that match the compiled circuit's gate structure. Mapomatic leverages this concept to optimize qubit mapping by selecting the qubits with the lowest noise and minimizing the need for SWAP gates. This approach ultimately enhances the execution quality of quantum circuits on IBM Quantum hardware.

Circuit Packing In our experiment, we first select a QAOA ComputeUncompute circuit and map it onto the layout with the lowest Mapomatic score, denoted as `qc1`, representing the best layout. We then map an identical circuit onto the next best non-overlapping layout, denoted as `qc2`. These circuits are measured on classical registers `cr1` and `cr2`, respectively. Next, we compose `qc1` and `qc2` into a single circuit, denoted as `qc`, and transpile `qc` on `fake_washington` using the initial layouts as the best layout and next best layout. After transpilation, we run a sampler to obtain counts for `cr1` and `cr2`. To account for

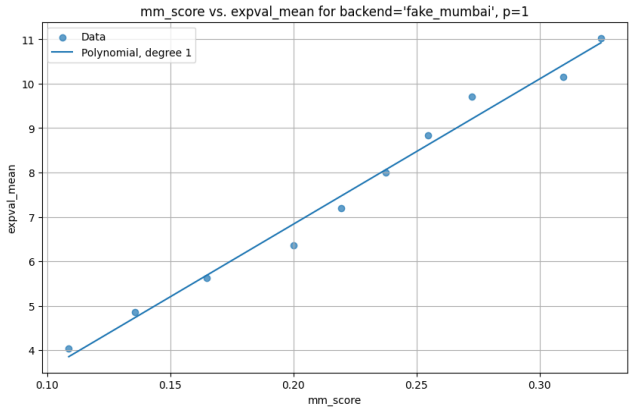

Figure 10: *Linear fit on fake_mumbai*

noise on the fake backend, we resample the counts using bootstrapping and calculate the mean of 10 samples. Finally, we compute the expectation value using these samples and observables.

After transpilation, we run a sampler to obtain counts for `cr1` and `cr2`. To account for noise on the fake backend, we resample the counts using bootstrapping and calculate the mean of 10 samples. Finally, we compute the expectation value using these samples and observables.

Dataset Generation

Dataset 1: QAOA ComputeUncompute Circuit We generate Dataset 1 by using the QAOA ComputeUncompute circuit on a 127-qubit fake backend, `fake_washington`. The dataset is created by varying the number of qubits and the parameter p, while considering all isomorphic layouts. For each configuration, we compute the Mapomatic score and gather counts from the sampler. The expectation value is derived from these counts using ZZ-type observables. This dataset contains 721 data points corresponding to the 721 isomorphic layouts.

Features:

- n_qubits - Number of qubits in the circuit.
- p - Parameter defining the circuit depth.
- **Isomorphic Layout** - The layout to which circuit is mapped.
- **Mapomatic Score**
- **Expectation Value**

Dataset 2: Circuit Packing Results Dataset 2 focuses on the results of circuit packing. We fix `qc1` on the best layout, determined by the lowest Mapomatic score, and iterate through all next best non-overlapping layouts for `qc2`.

Features:

- **Next Best Layout** - The layout for `qc2` selected as the next best non-overlapping option.
- **Individual Expectation Value at L1** - Expectation value of `qc1` on layout 1 when run individually
- **Individual Expectation Value at L2** - Expectation value of `qc2` on layout 2 when run individually
- **Packing Expectation Value at L1** - Expectation value of `qc1` on layout 1 when packed with `qc2`
- **Packing Expectation Value at L2** - Expectation value of `qc2` on layout 2 when packed with `qc1`
- **Percentage Degradation Individual**: The percentage degradation in expectation value from the best layout to the next best layout (i.e., from L1 to L2) when the circuits are run individually.
- **Percentage Degradation Packing**: The percentage degradation in expectation value from the best layout to the next best layout (i.e., from L1 to L2) when both circuits are packed together.

Experiments

We trained a Polynomial Regression model to predict the percentage degradation in packing from the individual degradation. The model's performance metrics are:

- **MSE**: 22.5938
- **R-squared (R^2)**: 0.8786
- **MAE**: 3.8036

Figure 11 displays the regression results.

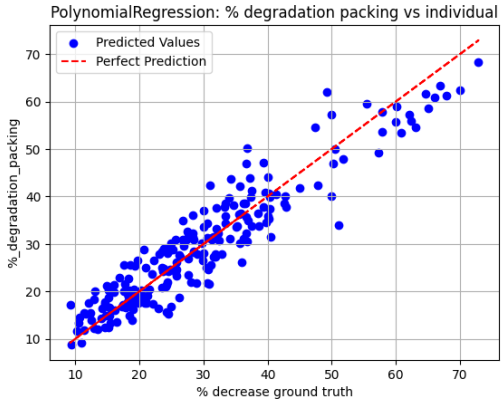

Figure 11: Polynomial Regression Results: Packing vs. Individual Degradation

Results and Discussion

Mapomatic Score vs. Expectation Value We plotted the Mapomatic score against the expectation value for various layouts in **Figure 12**. Curve fitting on this plot in **Figure 13** yielded an average Mean Squared Error (MSE) of 0.0618. This indicates a relatively accurate fit of the curve to the data points, reflecting how the Mapomatic score relates to the expectation value across different layouts.

Curve Fitting Analysis When examining the Mapomatic Score vs. Expectation Value for $p = 1$, we observed deviations between the fitted curves for $p = 1$, $p = 2$, and $p = 3$ in **Figure 14** This suggests that the relationship between the Mapomatic score and expectation value is influenced by the parameter p, with varying degrees of fit quality.

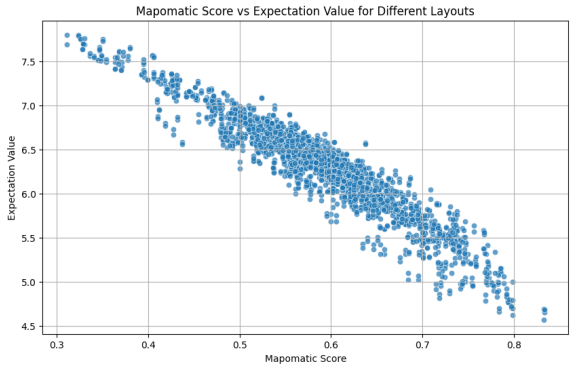

Figure 12: Mapomatic Score vs Expectation Value for Different Layouts(p=1)

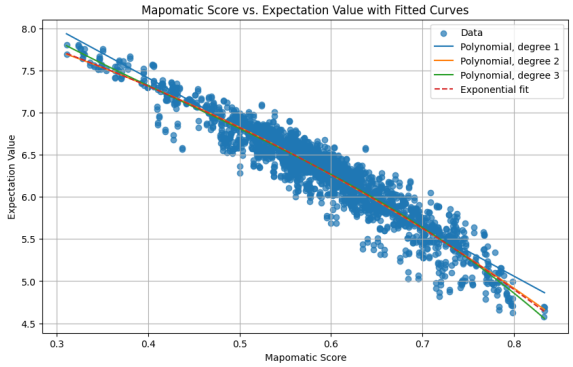

Figure 13: Curve Fitting on figure 12

Percentage Degradation Analysis We plotted the percentage degradation in packing against the percentage degradation observed when circuits are run individually in **Figure 15**.

Curve fitting on this plot in **Figure 16** revealed a generally linear relationship, with an average MSE of 21.8891. However, note that the goal here is not to predict the exact expectation value across layouts, but to predict the *degradation*. Our initial study shows that the trend of degradation remains similar when executing circuits individually, or packing them together – although the exact curve may not. This requires further exploration to confirm whether it is possible to predict the trend, but initial results are optimistic.

In summary, the findings highlight the importance of layout optimization and the impact of the parameter p on circuit performance. The relatively low average MSE in the Mapomatic Score vs. Expectation Value plot demonstrates good model fitting, while the linear trend in percentage degradation suggests a predictable relationship between individual and packing performance. These results inform strategies for improving quantum circuit execution by optimizing layouts and considering the effects of different parameters.

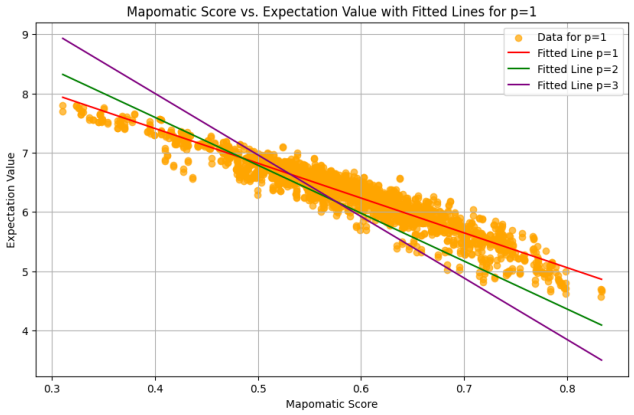

Figure 14: Fitted curves for p=1,2,3 on scatter plot of p=1

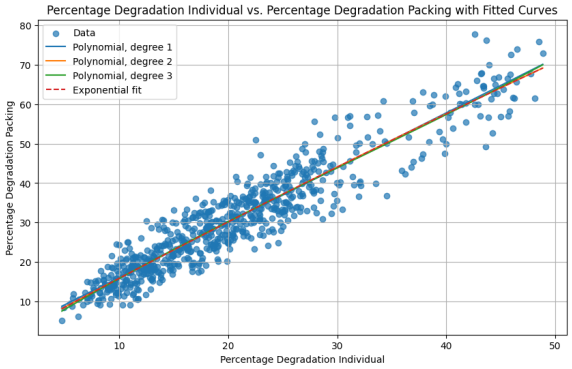

Figure 15

Code Avaliability

The code for all experiments and analyses conducted in this study is publicly available on IBM GitHub. The repository includes detailed instructions for reproducing the results and exploring the datasets used in our research.

You can access the repository at the following URL: https://github.ibm.com/ishantkohar/hardware-selection

Conclusion and Future Scope

Quantum computing presents numerous opportunities, but significant challenges still remain due to hardware variability and noise. In this paper, we examined two key hypotheses aimed at enhancing quantum computation efficiency and reliability across different backends.

Our first hypothesis explored the possibility of predicting result deviations on optimal hardware by leveraging known outcomes from sub-optimal hardware. Through our experiments using the QAOA Path Circuit, we found that this approach is not universally applicable across different quantum devices. While some limited scenarios showed promise, a generalized model for cross-device prediction remains elusive. This indicates the inherent complexities and differences between quantum devices, which require deeper exploration for accurate prediction and optimization.

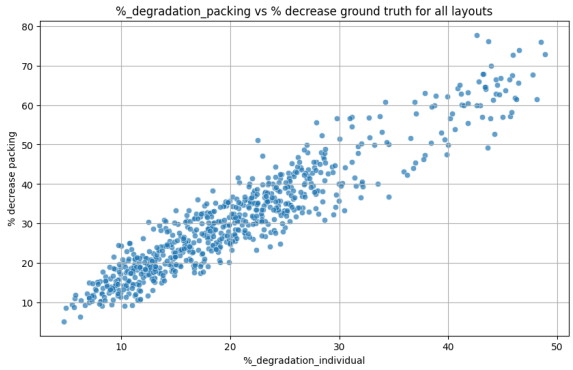

Figure 16: Percentage Degradation packing vs Percentage Degradation individual

Overall, the second hypothesis confirms that layout-dependent optimization can significantly enhance the performance of quantum computations. However, it also highlights the complexity of achieving consistent improvements across diverse quantum platforms. This reinforces the need for continued research and development of advanced scheduling algorithms and optimization techniques that can be tailored to the specific requirements of various quantum backends

Future research should focus on evaluating these methods across diverse quantum hardware platforms to validate and generalize the findings. Improving the accuracy of predictive models for deviations between sub-optimal and optimal hardware is also crucial. Additionally, exploring advanced scheduling algorithms and optimizing the packing of multiple circuits on quantum hardware could enhance performance and resource utilization. Further development of adaptable frameworks for various quantum architectures and tasks, along with advancements in calibration and error correction, will be essential for broader applicability and improved computational reliability.

References

Bhoumik, D.; Majumdar, R.; and Sur-Kolay, S. 2024. Resource-aware scheduling of multiple quantum circuits on a hardware device. arXiv:2407.08930.

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System.

Dorogush, A. V.; Ershov, V.; and Gulin, A. 2018. CatBoost: gradient boosting with categorical features support.

Farhi, E.; Goldstone, J.; and Gutmann, S. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028.

Liu, L.; and Dou, X. 2020. A New Qubits Mapping Mechanism for Multi-programming Quantum Computing. arXiv:2004.12854.

Nation, P. D.; and Treinish, M. 2023. Suppressing Quantum Circuit Errors Due to System Variability. *PRX Quantum*, 4(1).

Niu, S.; and Todri-Sanial, A. 2023. Enabling Multi-programming Mechanism for Quantum Computing in the NISQ Era. *Quantum*, 7: 925.