Enabling High Performance Debugging for Variational Quantum Algorithms using Compressed Sensing

Kun Liu*
Carnegie Mellon University
Pittsburgh, PA, USA
liukun@cmu.edu

Tianyi Hao*
University of Wisconsin-Madison
Madison, WI, USA
tianyi.hao@wisc.edu

Swamit Tannu
University of Wisconsin-Madison
Madison, WI, USA
swamit@cs.wisc.edu

ABSTRACT

Variational quantum algorithms (VQAs) can potentially solve practical problems using contemporary Noisy Intermediate Scale Quantum (NISQ) computers. VQAs find near-optimal solutions in the presence of qubit errors by classically optimizing a loss function computed by parameterized quantum circuits. However, developing and testing VQAs is challenging due to the limited availability of quantum hardware, their high error rates, and the significant overhead of classical simulations. Furthermore, VQA researchers must pick the right initialization for circuit parameters, utilize suitable classical optimizer configurations, and deploy appropriate error mitigation methods. Unfortunately, these tasks are done in an ad-hoc manner today, as there are no software tools to configure and tune the VQA hyperparameters.

In this paper, we present OSCAR (cOmpressed Sensing based Cost lAndscape Reconstruction) to help configure: 1) correct initialization, 2) noise mitigation techniques, and 3) classical optimizers to maximize the quality of the solution on NISQ hardware. OSCAR enables efficient debugging and performance tuning by providing users with the loss function landscape without running thousands of quantum circuits as required by the grid search. Using OSCAR, we can accurately reconstruct the complete cost landscape with up to 100X speedup. Furthermore, OSCAR can compute an optimizer function query in an instant by interpolating a computed landscape, thus enabling the trial run of a VQA configuration with considerably reduced overhead.

CCS CONCEPTS

• **Computer systems organization → Quantum computing**.

KEYWORDS

Quantum Computing, Variational Quantum Algorithms, Debugging

*Both authors contributed equally to this research.

1 INTRODUCTION

Rapid progress in quantum computing has brought together a broad community of researchers who want to leverage quantum computers to solve problems in areas ranging from high-energy physics to finance. Using a quantum computer to solve scientific and commercial problems will require full-stack solutions and system-level abstractions that can address a diverse group of end users and allow them to access quantum computers seamlessly. This paper focuses on building methods to help debug, tune, and benchmark near-term quantum algorithms.

Variational Quantum Algorithm (VQA) [38] is one of the most promising categories of quantum algorithms that can utilize existing quantum computers with hundreds of noisy qubits to solve hard problems in optimization, chemistry, and machine learning. The paradigm of VQA is similar to that of Machine Learning (ML). In ML, we train a model by searching for the optimal parameters of the model on the loss landscape. In VQA, the difference is that the model is a variational quantum circuit (VQC), a quantum circuit with parameters. The VQC is executed on the quantum computer to produce the probability distribution of solutions, which is then used to compute the solutions' average cost. Just like the ML training, the parameters of VQC are tuned using an optimizer running on the conventional computer to maximize or minimize the average cost. The circuit parameters form a continuous parameter space, and as we vary the parameters of VQC, we obtain a cost hypersurface in the parameter space. Like in ML, we also denote it as 'landscape'.

Copying data, one of the most common computing primitives, is not available on quantum computers, as the laws of quantum mechanics prohibit the creation of exact copies of quantum information. Moreover, upon reading the qubit, we can only extract partial information about its state. This makes debugging and tuning quantum programs challenging as we can not "single step" through the program to probe the intermediate state due to probabilistic qubit readout, nor can we create copies of the data to analyze the intermediate state. In addition to the fundamental constraints, there are engineering challenges. For example, all quantum hardware platforms are extremely noisy, and running programs on error-prone quantum hardware often produces incorrect output. As a result, when we observe an undesired output on a quantum computer, it is hard to differentiate a software bug from a hardware error.

VQAs add to this challenge as they rely on optimizers to search and coverage on the optimal circuit parameters that maximize the quality of the solution. If optimizers are not configured correctly or if we pick an initial point that is close to a local minimum, the

final VQA output can be suboptimal, producing low-quality solutions. Moreover, to improve the efficacy of VQAs, researchers have developed noise mitigation techniques that execute additional runs to offset or filter the effect of noise. Although well-intentioned and theoretically proven, noise mitigation does work under a narrow set of conditions and can require precise configuration.

When running VQA on a near-term quantum computer, all components of the system must work in tandem to get high-quality solutions. For example, the programmer describes the parametric circuit (or ansatz), picks the initial point, configures the optimizer, and configures the noise mitigation technique. To enable the optimal solution on VQA, all previously described steps need to be performed accurately. Any misconfiguration can result in an undesired system behavior or a bug. Unfortunately, discovering these scenarios can be highly challenging due to: (1) The inability to trace intermediate states of quantum programs [19, 23, 26]. (2) Noisy hardware can further muddle the output [33, 53]. (3) Several tightly integrated non-deterministic components, such as optimizer and noise mitigation schemes.

To debug the VQAs, we must untangle and check components in isolation. We observe that using an optimizer adds non-determinism. Furthermore, the path traversed by the optimizer is highly sensitive to the landscape. One of the most straightforward ways to reduce non-determinism is by disengaging the optimizer from the overall VQA workflow. Instead of traversing the landscape guided by the optimizer, we propose to leverage an entire landscape. With a full landscape, we could calculate the variance of gradient and probe directly into barren plateaus, check the quality of initial points and convergence of optimization, measure the performance of different quantum error mitigation methods, visualize the landscape, and reveal deeper insights.

However, generating a full landscape by using a grid search, i.e., uniformly sampling on each dimension of the landscape, is extremely challenging due to the high cost of running VQA circuits. In real experiments, thousands of shots are needed for the near-term quantum device to compute a single point on the landscape. Generally, for each point on the landscape, we derive it by running the quantum circuit number-of-shots (usually over 1,024) many times and measuring these quantum states.

We use this insight of sparsity and propose *cOmpressed Sensing based Cost lAndscape Reconstruction (OSCAR)*, which leverages the idea of compressed sensing (CS) [5, 12] to efficiently reconstruct the full landscape by using only a small fraction of the experiments compared to performing a full grid search. CS can enable full reconstruction by leveraging the sparsity of signals in the frequency domain. We discover that optimization landscapes for the majority of VQAs are sparse in the frequency domain, and these landscapes can be faithfully reconstructed by using standard mathematical tools developed by the compressed sensing community. Based on the analysis of the landscape given by CS, we could choose initial points, optimizers, and other settings to achieve better performance. With few points on the landscape, we could reconstruct the whole landscape with reasonable loss of accuracy. We evaluate our technique on simulated landscapes and landscapes generated on Google and IBM superconducting qubit quantum processor.

To reconstruct a full landscape, we sample parameter values randomly and uniformly from the entire parameter space and then run VQA circuits to generate the expected value of the cost functions corresponding to the randomly selected parameters. Using these expected values, we execute a compressed sensing routine that reconstructs a landscape. Our experiments show that in less than a few thousand experiments, we can reconstruct an entire landscape with minuscule error. Using a simulator, we test our design in an ideal setting and with depolarizing noise. Furthermore, we use landscape data collected on Google's quantum computers to show that even on real hardware with complex noise sources, we can execute OSCAR to generate an accurate cost landscape significantly faster.

The idea of using compressed sensing to reconstruct a full landscape from a few random samples is extremely powerful because it detaches the classical optimization routine from Variation Quantum Algorithms. During the debugging process, it reduces the complexity and non-determinism added by the optimization routine, but even more importantly, it enables parallelism as the calculation of expected values is no longer serialized due to the optimizer. In a typical VQA workflow, we run a quantum circuit, compute the expected cost value and use it to determine the next set of circuit parameters using an optimizer. This model enforces serial execution. With OSCAR, we can pick random circuit parameters and run circuits in parallel on multiple quantum computers to further accelerate reconstruction to enable fast debugging. However, using OSCAR as it is in parallel mode is challenging due to variability in error rates. Different quantum hardware platforms have different noise profiles. Even on the same quantum computer, a subset of qubits exhibits large variations in error rates [53]. If we collect samples on devices with different noise-level, the reconstructed landscape would be an "artificial" mixture of original landscapes, masking hardware-specific effects. We propose a noise-compensation algorithm to minimize this mixing to retain the device-specific effects. We transform the expected values obtained on a quantum computer with one noise configuration to another machine with a different noise configuration using linear regression to preserve the impact of noise on the reconstructed landscape. Furthermore, we propose eager reconstruction, wherein OSCAR can fully leverage the speedup enabled by parallel execution despite high tail latency and queuing delays on the real systems.

In this paper, we showcase three use cases of OSCAR. First, we show that by using OSCAR, we can efficiently navigate a complex process of configuring noise mitigation techniques. Noise mitigation is highly effective on near-term quantum computers. Unfortunately, many noise mitigation techniques require a significant number of additional circuit runs, making them expensive. Furthermore, these mitigation techniques need to be carefully configured as they can affect the problem landscape. For example, we observe that noise mitigation techniques make problem landscapes sharp, amplifying the gradients but also they add jaggedness. This can negatively affect the optimization process. We show that by using OSCAR. We can use OSCAR to reconstruct a landscape that preserves certain local traits using only a 10% fraction of samples and give users the ability to benchmark and test the noise mitigation method without running a substantial number of experiments. OSCAR's second and third use cases focus on selecting and configuring the optimizer. A reconstructed landscape gives us a large number of grid points in the parameter space. We can make the grid dense by using interpolation and running an optimizer on the
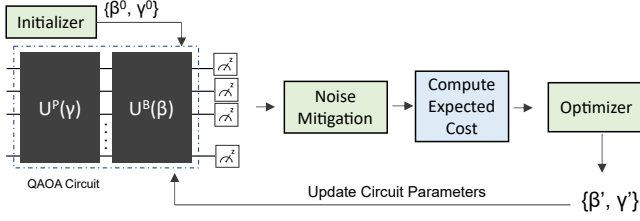
Figure 1: Typical workflow used for executing QAOA.

reconstructed landscape. In this case, we simply query the reconstructed landscape instead of running an actual circuit for specific parameters. We show that by optimizer running with default VQA workflow and on a landscape reconstructed by OSCAR, converge on points very close to each other. This gives us the ability to test optimizers with different configurations without actually querying quantum computers to faithfully capture the interaction between the optimizers and the problem landscape. Furthermore, in the third use case, we show that the interpolated landscape can assist users in picking a good initial point and the right optimizers.

2 BACKGROUND

2.1 Quantum Programming Model

Quantum computers are domain-specific accelerators that leverage properties of quantum bits (qubits) to solve computationally challenging problems. Quantum programs use sequences of quantum gates that manipulate the collective state of qubits. Today, most quantum computers are programmed using a co-processor model, in which the programmer uses a domain-specific language to describe the quantum program. Then, the program is compiled to an executable, which the host machine offloads to *Quantum Processing Unit (QPU)*. Finally, the program is executed, and output is returned to the host machine. Unfortunately, quantum hardware is prone to errors, and the execution of quantum programs can produce incorrect results. Researchers are actively developing reliable hardware and noise-resilient algorithms that we can use in the near term.

2.2 Variational Quantum Algorithms

Variational Quantum Algorithms (VQAs) are a family of quantum algorithms that promise to solve challenging optimization and learning problems using near-term quantum computers. VQAs use a parametric circuit and search iteratively for the circuit parameters that produce high-quality solutions. The Quantum Approximate Optimization Algorithm (QAOA) [14] is one of the VQAs, which is widely used to solve combinatorial optimization problems. When solving an optimization problem using QAOA, as illustrated in Figure 1, we search for circuit parameters β and γ using a two-step process. First, we initialize β and γ with the best-known value and execute a quantum circuit several thousands of times. This yields a distribution of solution strings, where each solution has a fixed cost. Our objective is to find the solution string with the lowest cost. Next, we compute the average (or expected) cost corresponding to the output distribution and search for optimal β and γ using expected cost as the objective function.

2.3 Noise mitigation with VQA Workflow

VQA circuits are expected to run on noisy hardware, where the produced output can be incorrect due to qubit errors corrupting the computation. However, most VQAs estimate the expected (or average) value of a cost function to tune the circuit parameters. Fortunately, the average estimate is resilient to small changes in the output distributions due to errors. However, with current error rates, it is challenging to get accurate results on problems at scale that are useful for practical application. With an average two-qubit gate error rate of 0.5% on leading commercial hardware platforms, we can run applications with, at most, hundreds of two-qubit gates. To expand the capabilities of noisy quantum computers, we can use noise mitigation methods such as Dynamical Decoupling (DD) [8, 55], Qubit Readout Mitigation (QRM) [2], Zero Noise Extrapolation (ZNE) [24], Probabilistic Error Cancellation (PEC) [54], and Clifford Data Regression(CDR) [7],categorized as:

Shot Frugal Mitigation - leverages methods that enhance gate or operational fidelity by augmenting the input circuit without increasing the number of shots. Dynamical decoupling, for example, is implemented by simply inserting a sequence of single-qubit gates to suppress the ZZ-crosstalk [3, 11, 31], while readout mitigation is performed by constructing the inversion matrix and filtering measurement errors using a post-processing step. The shot frugal mitigation methods do not require additional circuit executions.

Mitigation with Supplementary Shots - In this category, the circuit and circuit configurations are altered at runtime. For example, Zero Noise Extrapolation (ZNE) evaluates the expected value of a cost function with varying noise levels and extrapolates it back to estimate the expected value with zero noise. Several demonstrations at small and large scales have shown encouraging results. Unfortunately, these methods require a significantly higher (10x to 100x) number of circuit runs and non-trivial configurations.

To improve the estimates of expected values on current hardware, the user needs to set up noise mitigation strategies by using trial and error methods. The methods are theoretically sound, but there are noise and circuit-specific factors that require manual intervention, and if the user is not careful, the noise mitigation techniques can do more harm than good.

3 DEBUGGING AND TUNING VQAS

3.1 Motivation

Quantum debugging and analysis tools are essential for developing, deploying, and benchmarking quantum programs. The goal of a software debugger is to help programmers track down incorrect or inconsistent implementation, whereas the analysis tools can help improve the overall reliability and quality of the solution by tuning the quantum programs.

Building debugging and analysis tools for quantum programs are fundamentally challenging due to (1) destructive and statistical reads, (2) no-cloning theorem, and (3) uncertainty introduced by noisy operations. Reading a qubit gives only partial information as qubit readout collapses the qubit to '0' or '1' states with a certain probability, and precisely estimating the quantum state can require a significant number of repeated measurements. Furthermore, creating copies of quantum variables is prohibited due to the no-cloning theorem. Both constraints make probing the intermediate state of
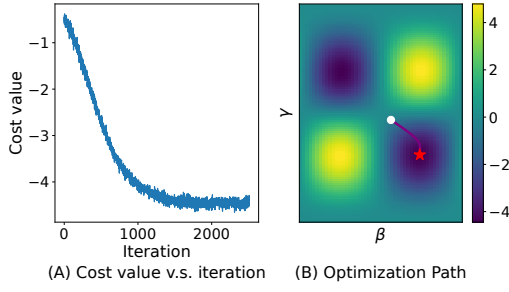
Figure 2: (A) Default view provided by the VQA workflow (B) Bird's-eye view with a complete optimization landscape.

quantum programs extremely challenging. Huang et al. proposed statistical assertions to navigate this challenge by using carefully designed statistical tests to check if the intermediate state of the quantum program satisfies specific properties [19].

Using projective measurements and runtime assertions proposed by Li et al. [23] and Liu et al. [26], we can reduce the overhead of probing the internal state of a quantum program significantly by leveraging additional ancillary qubits. However, for all circuit and statistical debugging techniques, the programmer needs to know the correct state or some properties of the correct state, and by using quantum and statistical gadgets, one could track inconsistencies in the circuit implementations. These works lay the groundwork for debugging quantum circuits and programs. However, prior work does not address debugging and tuning near-term quantum algorithms, such as VQAs running on noisy quantum computers.

3.2 Challenges in Debugging and Tuning VQAs

Debugging quantum circuits is significantly challenging in the presence of noise, as noisy operations produce incorrect outcomes, and determining if the undesired output is produced due to a hardware error because of noisy qubits or by incorrect implementation can be very difficult. Furthermore, VQAs utilize optimizer-driven circuit parameter updates, resulting in complex internal states of the program that can not be easily used to drive the debugging process. Unlike quantum kernels such as the Quantum Fourier Transform, which is at the heart of Shor's factoring algorithm [47], or Amplitude Amplification, used in Grover's Algorithm [17], VQAs do not exhibit structure that programmers can easily use to verify correctness. Similar to classical deep neural networks, there is no one correct internal state that programmers can use as a reference to debug the program [25]. Additionally, optimization inserts a significant amount of non-determinism in the program behavior. All VQAs fundamentally rely on a tightly integrated workflow between the quantum computer and optimizer, where misconfiguration on one component can affect the overall correctness.

The complexity of VQA workflows is expected to increase with the introduction of error mitigation methods that run circuits with different configurations. Many noise mitigation techniques require a significantly large number of circuit runs, and they are challenging to configure. We envision that users developing VQAs will need tools to reason with the effectiveness of noise mitigation techniques to tune current and future quantum algorithms. Especially with the current cloud computing model, with limited system-level access.

3.3 How complete landscape can help?

When debugging VQAs, we want to understand which workflow component is causing the undesired output. Unfortunately, the current software workflows provide a narrow view of the landscape as the optimizer picks the path (a tiny fraction from a cost function landscape) traversed on the landscape. The objective function of the optimizer is designed to maximize the quality of the solution. However, VQA landscapes are riddled with local maxima/minima and barren plateaus [6, 28, 30], which are further exacerbated due to noisy operations, making it challenging to establish the impact of specific configuration and their effect on the output quality when running an optimizer. We need context to understand how optimizer configuration affects the traced path. For example, assessing the efficacy of design decisions would become significantly easy if we could access the entire landscape from a bird's eye view. Figure 2, show two points of view, panel (A) shows optimizer centric view, which users see when they use standard workflow, whereas panel (B) shows the path traced by the optimizer superimposed on the landscape, providing context for the user to understand the dynamic between optimizer, landscape, and noise mitigation.

For prototyping and as a pedagogical tool, complete landscapes are often generated by doing a grid search, where a large number of β and γ values are evaluated on a quantum computer or the simulator. However, grid search is extremely slow and expensive, especially for debugging, a highly uncertain process, where the user does not know a priori how many executions they will need to trace the bug.

4 LANDSCAPE RECONSTRUCTION BY OSCAR

We propose *cOmpressed Sensing based Cost lAndscape Reconstruction (OSCAR)*. An open-source implementation in Python is available at https://github.com/haoty/OSCAR. This section details our design and demonstrates its effectiveness in practice.

4.1 Overview of OSCAR Workflow

OSCAR leverages compressive sensing to generate the entire cost landscape using a small number of samples. Our design is comprised of three phases: parameter sampling, circuit execution, and landscape reconstruction, as shown in Figure 3.

Parameter Sampling. We randomly select a small number of circuit parameters that are uniformly distributed over the parameter space (shown as β and γ points in Figure 3). The dimension of the optimization landscape scales with the number of independent parameters, which is $2p$ for a QAOA circuit with p layers. When reconstructing high-dimensional landscapes, we perform concatenations to reduce the dimension.

Circuit Execution. We execute the VQA circuit with the sampled circuit parameters for a set number of shots to calculate the expected values of the cost function.

Landscape Reconstruction. We use the cost function values from the previous phase to reconstruct the landscape using compressed sensing. Compressed sensing (CS) [5, 12] reconstructs a signal with samples significantly fewer than what Nyquist–Shannon sampling theorem requires. Nyquist-Shannon sampling theorem states that if the sampling rate is two times higher than the highest frequency of a signal, then this signal could be reconstructed perfectly. However,
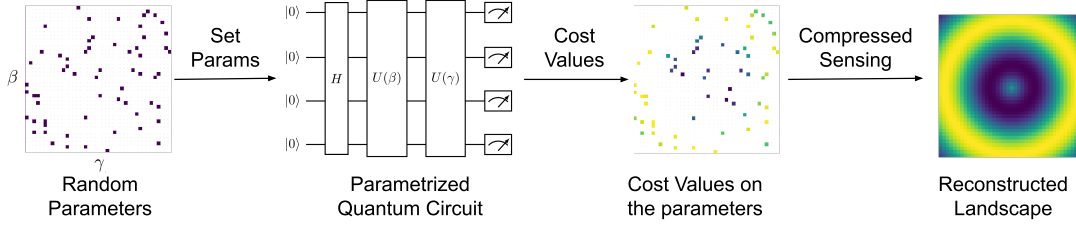
Figure 3: Overview of OSCAR illustrating three steps: random parameter sampling, circuit execution, and reconstruction.

the constraint on the sampling rate is not absolute. We can enable high-quality reconstruction with significantly fewer samples if the signal has a small number of frequency components, that is if the signal is sparse in the frequency domain. We observe that *VQA landscapes are sparse in the frequency domain and can be accurately reconstructed using compressed sensing.*

We leverage this insight to generate the entire landscape by running a significantly lower number of circuits compared to the standard grid search. For more mathematical details of CS, please refer to Appendix A.

Depth	β range, # samples	γ range, # samples	Total # samples
p=1	$[-\pi/4, \pi/4]$, 50	$[-\pi/2, \pi/2]$, 100	$50 \times 100 = 5k$
p=2	$[-\pi/8, \pi/8]$, 12	$[-\pi/4, \pi/4]$, 15	$12^2 \times 15^2 = 32k$

Table 1: Grid Definition of QAOA Ansatz.

4.2 Reconstruction Accuracy of OSCAR

We evaluate the reconstruction accuracy of OSCAR using simulators and data collected on IBM and Google hardware.

4.2.1 Method. We use a total of 200 problem instances of QAOA to evaluate the landscape reconstruction accuracy of OSCAR. To generate the ground truth, we perform a dense grid search to obtain the true landscape, by executing 5,000 to 32,000 quantum circuits corresponding to the grid points on the landscape as shown in Table 1. Each circuit is executed using state vector simulation.

4.2.2 Metrics. To evaluate the quality of reconstructed landscapes, we use the normalized root-mean-square error (NRMSE), a scale-invariant error metric to analyze a diverse set of landscapes. For x, the original landscape flattens to a 1-D array, and y the reconstructed landscape flattens to a 1-D array with T elements in total, NRMSE is calculated as

$$\text{NRMSE} = \sqrt{\frac{\sum_{t=1}^{T}(x_t - y_t)^2}{T}} \Big/ (Q_3 - Q_1), \qquad (1)$$

where Q_1 and Q_3 are the first and third quartiles of true landscape x.

4.2.3 Evaluating OSCAR with diverse Anzatzes and problems. While our primary focus is low-depth QAOA and the well-studied 3-regular graph MaxCut problem, our method is applicable to landscapes of a variety of ansatzes and problems. In addition to large-scale QAOA-MaxCut experiments shown in the rest sections, we perform small-scale experiments computing the landscapes of the SK model problem, the hydrogen molecule, and the lithium hydride molecule. We test the QAOA ansatz, the hardware-efficient Two-local ansatz, and the chemistry-inspired UCCSD ansatz, as shown in Table 2 and Table 3. We configure the depth of the ansatz so that QAOA and Two-local have 8 and 6 parameters for $n = 4$ and $n = 6$, respectively. Hydrogen and lithium hydride with UCCSD have 3 and 8 parameters respectively. For 8-parameter instances, we sample 7 points along each dimension of the grid; for 3-and-6-parameter instances, we sample 14 points. Due to the exponentially large overhead, we evaluate the reconstruction accuracy by randomly selecting two varying parameters, fixing the rest to random values, and repeating 100 times.

To demonstrate these landscapes are periodic enough to be accurately and efficiently reconstructed, we employ Discrete Cosine Transformation (DCT) to show they are sparse in the frequency domain. Table 4 shows the fractions of DCT coefficients needed to preserve 99% of the signal energy. We observe that only a few DCT terms contribute to most of the energy costs, showing a highly sparse structure in the frequency domain. Our insight is broadly applicable to VQA landscapes.

Problem	#Qubits	#Parameters	#Samples	QAOA	Two-local
3-reg MaxCut	4	8	7	0.847	0.645
3-reg MaxCut	6	6	14	0.372	0.0000001
SK Problem	4	8	7	0.847	0.765
SK Problem	6	6	14	0.372	0.057

Table 2: Reconstruction errors for QAOA and Two-local ansatzes on 4-qubit and 6-qubit problems. We configure the depth of the ansatzes so that QAOA and Two-local have 8 and 6 parameters for $n = 4$ and $n = 6$.

Molecule	Ansatz	#Qubits	#Parameters	#Samples	NRMSE
H_2	Two-local	2	4	14	0.171
LiH	Two-local	4	8	7	0.678
H_2	UCCSD	2	3	14	0.345
H_2	UCCSD	2	3	50	0.005
LiH	UCCSD	4	8	7	0.856

Table 3: Reconstruction errors of hydrogen and lithium hydride molecules. "#Samples" indicates the number of points equidistantly sampled within the range of every parameter.
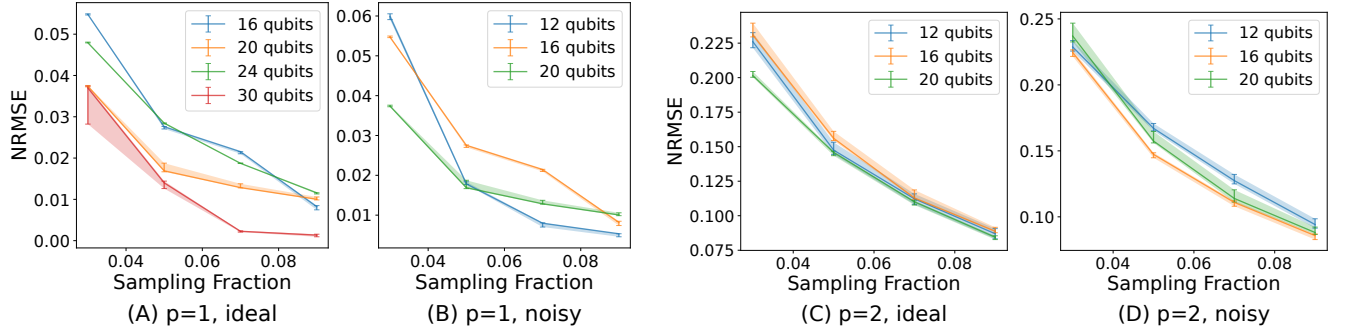
Figure 4: Median reconstruction errors of 16 MaxCut problem landscapes for (A) QAOA **with** $p = 1$ **and ideal circuit execution** (B) QAOA **with** $p = 1$ **and depolarizing noise,** 1Q **gate error of 0.003 and** 2Q **gate error of 0.007.** (C) QAOA **with** $p = 2$ **and ideal circuit execution** (D) QAOA **with** $p = 2$ **and depolarizing noise,** 1Q **gate error of 0.003 and** 2Q **gate error of 0.007.**
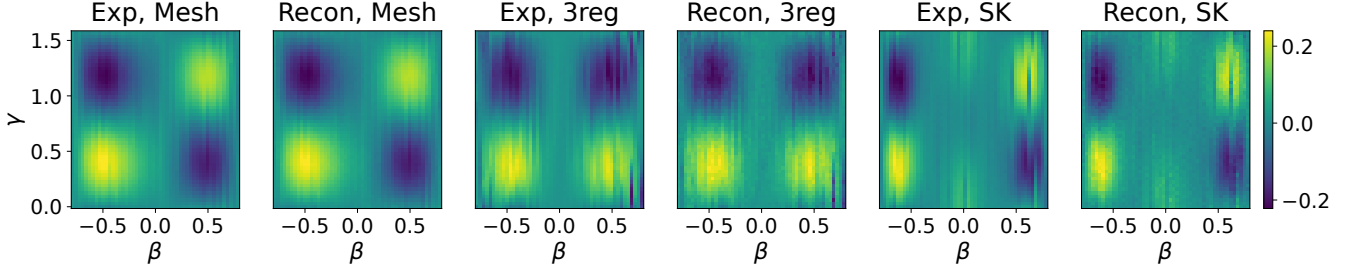

Figure 5: Original experimental [18] and reconstructed QAOA landscapes for MaxCut problem with mesh graph and 3-regular graph, and SK Model problem. The reconstructed landscapes are generated with a sampling fraction of 41%.

4.2.4 Evaluations on QAOA. Figure 4 shows the accuracy of landscape reconstruction versus sampling fractions for different QAOA problem. For ideal and noisy cases, we reconstruct landscapes of 16 different MaxCut problems and draw 25%, 50%, and 75% quartiles for given numbers of qubits under varying sampling fractions. As the sampling fraction increases, the reconstruction error of both ideal and noisy decreases steadily and keeps having small variances across instances. Figure 4 (C) and (D) show the error of reconstructing $p = 2$ ideal and noisy landscapes. We evaluate the reconstruction accuracy of the $p = 2$ case by reshaping 4-D landscapes into 2-D and then applying OSCAR. For example, we reshape the true landscape with dimensions (12,12,15,15) into a (12×12, 15×15) landscape. We observe that despite the noisy operations or higher dimensionality,

OSCAR can perform accurate landscape reconstruction, capturing the impact of noise on the landscape. However, with increased dimensionality, the reconstruction accuracy drops because of reshaping, which introduces artificial repeating patterns. Moreover, simulating $p = 2$ noisy landscapes is slow[1].

4.3 Evaluating OSCAR with Google's Dataset

To evaluate the reconstruction accuracy of OSCAR on real hardware with complex noise sources, we utilize the QAOA dataset from Google [18]. This includes the optimization landscapes generated on Google's 53-qubit Sycamore chip for the MaxCut on the 3-regular graph, MaxCut on the mesh graph, and the Sherington Kirkpatric (SK) Model problem [46]. All landscapes in the Google dataset have 50×50 data points. Figure 5 summarizes reconstruction errors with increasing sampling fraction. Note that Google's original landscape has 2500 points, which is relatively sparse compared to dense grids with 5000 to 32,000 points that we have used previously. We believe due to the sparser original landscape, the reconstruction accuracy is slightly lower. However, the error measured by the NRMSE is conservative as shown in Figure 6. For the NRMSE of about 0.2, the reconstructed landscapes are perceptually identical to the original landscapes. Figure 6 highlights successful reconstruction for the SK Model landscape, despite the highly noisy original landscape.

Problem	QAOA Ansatz	Two-local Ansatz	UCCSD Ansatz
3-reg Maxcut (n=4)	0.0420%	0.0000867%	–
3-reg Maxcut (n=6)	0.00768%	0.0000133%	–
SK Problem (n=4)	0.0420%	0.000416%	–
SK Problem (n=6)	0.00912%	0.0000398%	–
H_2 (n=2)	–	0.00260%	0.0729%
LiH (n=4)	–	0.000104%	0.0000173%

Table 4: The fractions of DCT coefficients needed to preserve 99% of the signal energy, verifying that VQA landscapes are generally sparse in the frequency domain and thus can be accurately and efficiently reconstructed.

---

[1]To generate the entire optimization landscape, we need to simulate 32k 20-qubit depth-2 MaxCut QAOA circuits. On the Qiskit state vector simulator using a single Nvidia A100 GPU, this requires 45 to 55 hours of runtime.
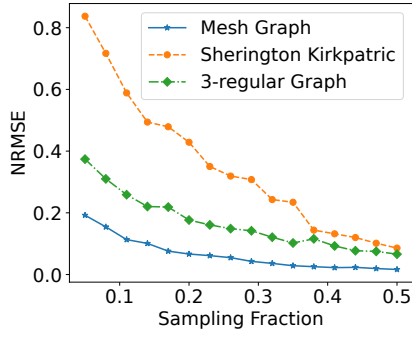
Figure 6: Reconstruction error on Google Sycamore

> **Using a small number of samples, OSCAR can accurately reconstruct the landscape by providing 2x to 20x speedups for complete landscape generation.**

5 PARALLEL LANDSCAPE RECONSTRUCTION

Generating a full landscape requires thousands of shots for a large number of points in the circuit parameter space. With OSCAR, we reduce the number of experiments significantly by using compressive sensing. Furthermore, we can accelerate the landscape generation even further by exploiting the parallelism offered by OSCAR. When debugging, OSCAR decouples the optimizer to eliminate the sequential execution model. In this section, we will discuss parallel landscape reconstruction, a design that takes advantage of the parallel circuit execution as the samples required can be generated independently.

5.1 Parallel Reconstruction with OSCAR

Figure 7 shows OSCAR can generate cost function samples in parallel by executing quantum circuits on k QPUs as samples picked on the cost function landscape are entirely independent. This constant factor speedup of k can significantly help the debugging process. Due to scarce quantum hardware and costly quantum simulations, users may have to wait for long durations to obtain the results when debugging quantum programs. For example, on publicly available QPUs, the queuing delay for running hundreds of circuits can span from a few hours to days [9, 39]. Even when using quantum simulations, runtimes can be extremely long. Using OSCAR, users can leverage multiple quantum computers (or simulators) at once. However, when we use samples generated on multiple QPUs, the reconstructed landscape can be a 'mixture' of all the landscapes that we are sampling from, which might be acceptable for specific debugging use cases. But for a use case where the user wants to study the impact of the hardware error on the VQA workflow, preserving the effect of noise on the reconstructed landscape is necessary. Note that prior works have used circuit ensembles to boost fidelity and performance [40, 48, 51, 52]. However, using them as is may not be apt for debugging VQAs.

To enable noise-preserving reconstruction in a multi-QPU setting, we propose a *Noise Compensation Model (NCM)* based on linear regression. First, we train NCM on a small fraction of circuit parameter samples executed on QPU-1 and QPU-2 as shown

in Figure 8(B). Then, using NCM, we can transform the expected values on QPU-2 to match the noise configuration of reference machine QPU-1 as shown in Figure 8(C). To illustrate the effectiveness of our method, we run three QAOA problems on two QPUs with different noise configurations. The 1Q and 2Q gate error rates on QPU-1 are 0.1% and 0.5%, respectively, whereas, for QPU-2, the error rates are 0.3% and 0.7%. Our goal is to match the landscape obtained on QPU-1 by using a mixture of QPU-1 and QPU-2 samples. Figure 8 shows the NRMSE between the reconstructed landscape that uses samples from QPU-1 and QPU-2, while the true landscape obtained on QPU-1 is used as the reference. When OSCAR is executed in uncompensated mode, the reconstruction error is high, indicating dissimilarity between the reconstructed landscape and the target landscape. However, by using NCM, OSCAR can significantly reduce the reconstruction error and closely match the target landscape as shown in Figure 8(B). For training NCM, we use 1% training samples from the landscape, while for reconstruction, a total of 10% samples with a varying fraction from QPU-1 and QPU-2 are used.

Moreover, we test the performance of NCM on real quantum hardware by introducing samples generated on IBM Lagos and IBM Perth. Table 5 shows the reconstruction errors of different combinations of IBM devices and simulations as sources, with and without NCM. We observe that NCM makes the reconstruction landscape closer to the target in all cases. Furthermore, there is little difference between using ideal and noisy simulation as QPU2. This result indicates that with a small fraction of samples from real quantum devices and the rest from ideal simulation, we can accurately reconstruct landscapes generated on real quantum devices.

QPU1	QPU2	20%-80%		50%-50%		80%-20%		100%-0%
		OSCAR	+NCM	OSCAR	+NCM	OSCAR	+NCM	OSCAR
Noisy sim-I	Noisy sim-II	0.076	0.003	0.061	0.002	0.039	0.002	0.001
Noisy sim-II	Noisy sim-I	0.075	0.002	0.059	0.002	0.037	0.002	0.001
IBM Perth	Ideal sim	1.362	0.299	0.97	0.265	0.597	0.223	0.184
IBM Perth	Noisy sim	0.767	0.272	0.564	0.247	0.379	0.213	0.184
IBM Perth	IBM Lagos	0.5	0.424	0.419	0.36	0.284	0.262	0.184
IBM Lagos	IBM Perth	0.403	0.337	0.341	0.286	0.266	0.247	0.222
Ideal sim	IBM Perth	0.478	0.226	0.363	0.18	0.215	0.109	0.042

Table 5: Errors between reconstructed and QPU1 landscapes using different combinations of IBM devices and simulations as sources, with and without the Noise Compensation Model (NCM). "+NCM" indicates the samples from QPU2 are transformed to match the noise model of QPU1 (target landscape). "(20%-80%)" means 20% of the samples come from QPU1 and 80% come from QPU2.

5.2 Relaxing Amdahl's Law with Eager Reconstruction

Generating an entire landscape is embarrassingly parallel. However, as per Amdahl's law, the serial portion fundamentally limits the parallel speedup. This applies to circuits running on parallel QPUs. Recent data suggest that a significant fraction of the wait times on quantum computers are due to queuing delays [39]. Furthermore, during our evaluations, we see 10x to 30x higher tail latency than
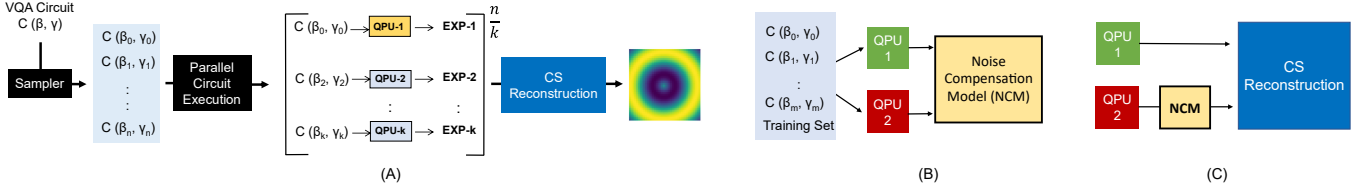
Figure 7: (A) OSCAR in Parallel Mode (B) Training Noise Compensation Model (NCM) to map expected cost values obtained on QPU-2 to QPU-1 (reference machine) (C) Leveraging NCM to transform QPU-2 values to enable noise-preserving reconstruction.

median circuit evaluation latencies. With the increasing complexity, we expect higher queuing delays and large tail latencies, making debugging challenging on future quantum computers. However, we can augment OSCAR to sidestep Amdahl's law. We propose *eager reconstruction* for OSCAR, where we leverage the tradeoff between reconstruction accuracy and sampling fraction. Based on our evaluations, we observe that a small reduction in sampling fraction does not reduce reconstruction accuracy significantly. As a result, during the circuit sampling process, we can set a soft timeout and begin the eager reconstruction for the available samples that fall under this timeout. In short, we can omit the samples that incur tail latency to significantly reduce the time to reconstruct without sacrificing the reconstruction accuracy.

6 USE CASE: BENCHMARKING AND TUNING NOISE MITIGATION

Noise mitigation methods are shown to be effective in reducing the errors introduced by the current noisy quantum hardware [21, 41]. However, deciding which mitigation method to use is not straightforward, and comparing the performance of different mitigation methods and settings can be highly resource-consuming. We show that reconstructed landscapes with OSCAR preserve properties of the original landscapes while being easily computable, thus helping us benchmark and configure noise mitigation techniques in an intuitive manner at a substantially reduced cost.

Inspired by the recent demonstration of Zero Noise Extrapolation (ZNE) [24] on multiple quantum hardware platforms [41], we use ZNE as an example to demonstrate how OSCAR can help visualize, benchmark, and configure noise mitigation methods. Given a circuit, ZNE first generates a few equivalent circuits by replacing

Figure 8: Errors between reconstructed and target (QPU-1's) landscapes using samples from two QPUs, without (A) and with NCM (B).

a gate U with equivalent substitutions, such as $UU^{-1}U$, based on the predefined noise scaling factors. The new circuits are thus functionally identical to the original but with additional noise. Then, by running these circuits with varying degrees of noise, we calculate the expected cost function values at different noise levels and then estimate expected noise with zero noise by using user-configured extrapolation models. ZNE is costly to apply since it requires running a significant number of additional circuits. Furthermore, it is even more expensive to configure, as there are various extrapolation models and many scaling factor choices.

Noise mitigation configuration can strongly influence the overall efficacy of mitigation techniques. For example, we use the Richardson extrapolation [24, 54] with {1,2,3} scaling and the linear extrapolation with {1,3} scaling for solving MaxCut problem using QAOA, and Figure 9 shows a landscape comparison of Richardson and linear extrapolation models. We can observe that Richardson extrapolation has a salt-like noise, whereas linear extrapolation leads to a smoother landscape closer to the ideal landscape almost without any noise. Richardson extrapolation adds a jaggedness to the landscape, which makes gradient-based optimization difficult.

As shown in Figure 9, the reconstructed landscapes preserve this difference. Thus OSCAR can enable users to study the impact of noise mitigation on the optimization landscape. Furthermore, we can quantitavely measure and compare the performance of different configurations. For example, we want to evaluate how mitigated features such as roughness and flatness change with different noise mitigation configurations using the following metrics:
(a) second-order derivative

$$D^2(x) = \sum_i \left[x_i - 2x_{i-1} + x_{i-2}\right]^2 / 4, \qquad (2)$$

(b) variance of gradients on the landscape

$$\text{VoG}(x) = \text{Var}\left[x_i - x_{i-1}\right] \qquad (3)$$

and (c) variance of the landscape

$$\sigma^2(x) = \text{Var}\left[x_i\right]. \qquad (4)$$

A second-order derivative on a landscape is commonly used to evaluate the roughness [16]. The variance of gradients is typically used to measure barren plateaus or flatness of the landscape [30]. Eq. 3 expresses gradients with the difference of elements. Note that all the equations above are for one-dimensional x. We compute average metrics on all dimensions.

We compare the three metrics of the original and the reconstructed landscape calculated with unmitigated results and Richardson-and-linear-extrapolated ZNE results in Figure 10. The reconstructed landscapes preserve the features of the original landscapes, enabling
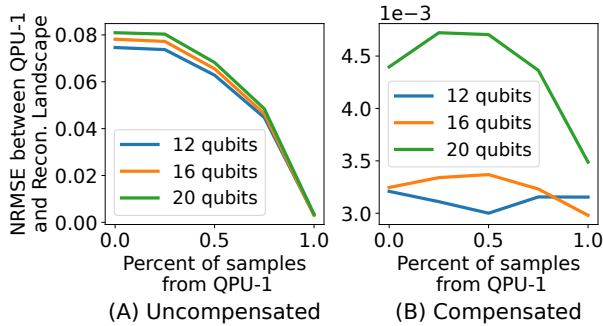
(A) Richardson   (B) Linear

(C) Reconstructed   (D) Reconstructed
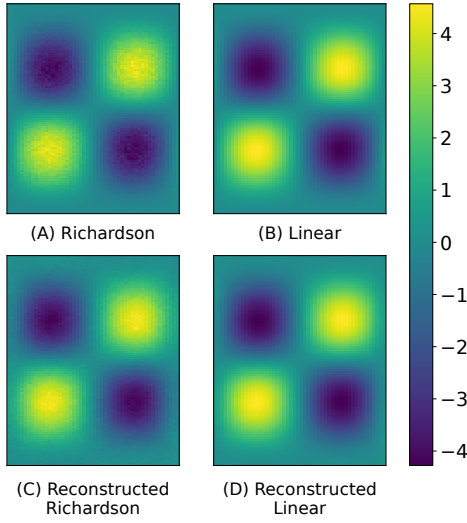   Richardson   Linear

Figure 9: Richardson extrapolation and linear extrapolation mitigation methods on a depth-1, 16-qubit landscape with depolarizing noise (1Q error of 0.001 and 2Q error of 0.02.

users to infer the impact of noise mitigation on the overall landscape. With OSCAR, we can evaluate landscape properties with substantially less costly reconstructed landscapes, which are derived with only a small fraction of points from the original landscapes.

As shown in Figure 10, ZNE with linear or Richardson extrapolation has comparable average flatness metrics as VoG and Variance values on the original landscape are comparable. A similar conclusion can be drawn by using OSCAR, where reconstructed landscapes show a similar trend. Unlike flatness, the roughness of Richardson's is significantly different as the second derivative is extremely high on the reconstructed landscape, which agrees with the ground truth of the original landscape, shown in Figure 9.

7 USE CASE: CONFIGURING AND DEBUGGING OPTIMIZER

VQAs, being heuristic algorithms and containing several configurable components, require extensive tuning to work as expected on a given problem. The classical optimizer in the VQA workflow is arguably the most tricky component to set up. Assume we are trying to configure the optimizer, and test the QAOA workflow for a large problem that is expensive to run on the QPU and difficult to simulate using a classical computer. However, when we begin the QAOA training process, the optimizer is not able to make significant progress even after running the training loop a significant number of times.

In the current quantum software development paradigm, even to test the configurations of an optimizer it is imperative to run the circuit a large number of times with different configurations. The trial and error approach of debugging may not always work as the desired outcome of the VQC training process may be hindered by multiple reasons such as bad optimizer configurations, hardware errors, software bugs, etc. Moreover, when you run the optimizer it only gives you the local information which is the path traversed by the optimizer in a small parameter range. Ideally, we would

like to have a bird's-eye view and see the full landscape and how optimizers make progress to get more insights into the problem. This is essential for near-term variational quantum algorithms as writing and debugging quantum programs are not very intuitive. Unlike classical programs which have meaningful intermediate states that programmers can use to debug their implementation of the algorithm, debugging VQAs by running quantum circuits with small sets of arbitrary parameters are hard to reason with. Furthermore, VQCs are known to have barren plateaus on the landscape, where cost becomes insensitive to the changes in the circuit parameters. Similar to the vanishing gradient problem [37] in machine learning, searching for the optimal parameters using the gradients becomes challenging. Barren plateaus can cause the optimizer to get stuck in a local region of the parameter space. In addition to barren plateaus, it may be possible to encounter saddle points and local minima, making optimization non-trivial.

Running the optimizer on the reconstructed landscape can serve as a good way to pre-check the optimizer performance before actually executing circuits on a real quantum device. To allow continuous-space optimization on the discrete grid, we use rectangular bivariate spline interpolation [35] to fill in the gaps. Figure 11 shows an example comparison between optimizing on the interpolated reconstructed landscape and with circuit executions, where the optimizer paths are identical. To evaluate the efficacy of our method, we calculate the Euclidean distance between the ending points of the two optimization paths. We use the gradient-based optimizer ADAM and the gradient-free optimizer COBYLA with default settings from Qiskit and random initial points. We run 8 instances each for ideal and noisy simulations of 16-qubit and 20-qubit problems. Figure 12 shows that interpolated reconstructed landscapes and circuit executions result in very close endpoints.

After obtaining reconstructed landscapes, our method is significantly faster and more resource-efficient compared to the normal optimization, since interpolation returns an optimizer query almost instantly. For some optimizers, the number of QPU queries required by reconstruction can even be fewer than the number needed for actual optimizations. In addition, our method allows reruns of optimizations with little overhead after reconstructing the landscape. Figure 13 shows an example of choosing and configuring optimizers based on their performance on the reconstructed landscape.

8 USE CASE: INITIALIZING OPTIMIZER USING OSCAR

Selecting accurate initialization for VQA circuit can have a considerable impact on the optimization result. For many problems, users have to resort to randomly choosing an initial point for the optimizer, leading to large number of cost function queries and worse results than using a properly chosen initial point. Past studies have suggested warm-starting with classical algorithms as well as using parameters obtained from running simpler instances [13].

We propose an alternative approach that takes advantage of the reconstructed landscape. Since parameters obtained by optimizing on the interpolated reconstructed landscape are very close to the optimal ones, as shown by Figure 12, they are suitable choices as initial points for the regular optimization workflow. Using OSCAR, we generate a reconstructed landscape, interpolate it, and run an
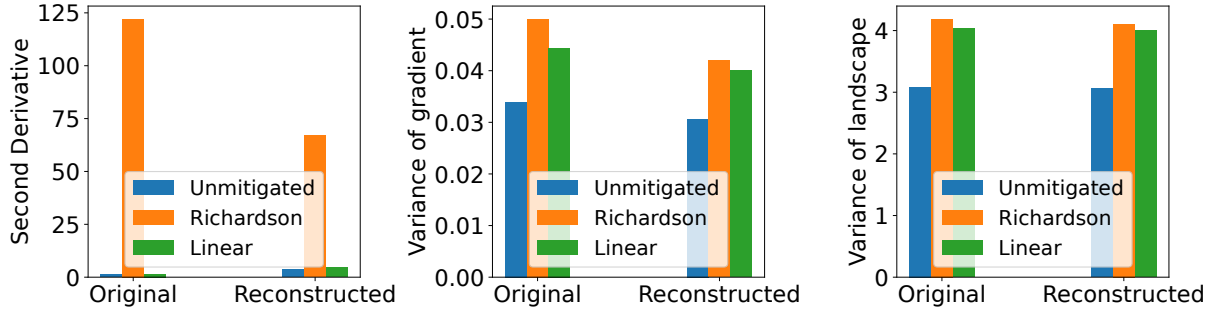
Figure 10: Reconstructed landscapes using OSCAR preserve key features of the original landscapes.

optimizer to obtain the minima on the reconstructed landscape. We use the minima that OSCAR converges to as the initial point to run optimization using regular workflow.

We run depth-1 QAOA on 14 different instances of the 16-qubit MaxCut problem with ideal and noisy simulators, starting from randomized initial points and points given by optimizing the reconstructed landscape, respectively. The resulting cost function values are within the termination tolerance of the optimizers, but OSCAR-generated initial points require much fewer queries during the optimization process. Even with the additional queries needed for reconstructing the landscapes, OSCAR can still be significantly faster for optimizers such as ADAM, as shown in the first two rows of Table 6. For optimizers that require fewer queries in nature, such as COBYLA, OSCAR is slower in paying the overhead of landscape reconstruction queries, as shown in the last two rows of Table 6. However, we stress that these queries can be computed completely in parallel. In that case, OSCAR reduces the sequential QPU computation time but adds an overhead for performing compressed sensing. The aforementioned concurrent study [15] proposes a similar approach and tests with gradient descent in the ideal setting, where they select the minima of the reconstructed landscape as initial points. While we agree with the claim that such reconstructed landscape-based initialization can greatly enhance gradient-based optimizers, we also note that it may not always be cost-effective to use our method for selecting initial points, especially for gradient-free optimizers such as COBYLA.
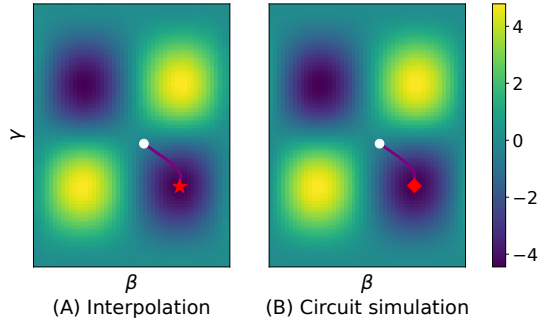

Figure 11: (A) Optimization on the reconstructed landscape by ADAM. (B) Optimization by circuit simulation. The landscape is from a 16-qubit MaxCut instance.
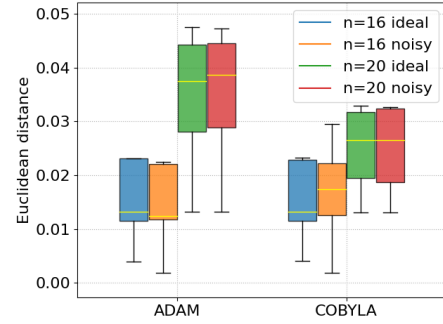

Figure 12: Comparison of the Euclidean distances between the ending points of optimizing on the reconstructed landscape and with circuit executions.
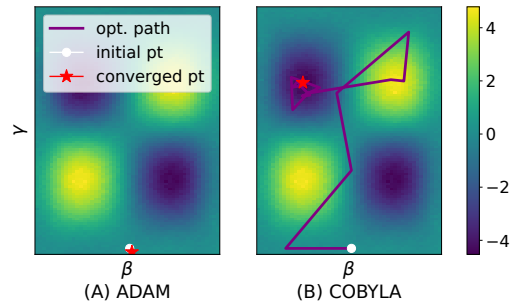

Figure 13: An example of choosing the optimizer based on the interpolated reconstructed optimization. On this Richardson extrapolated landscape, a gradient-free optimizer (COBYLA) performs better than a gradient-based one (ADAM).

9 RELATED WORK

Debugging Quantum Programs: Debugging quantum software is an open problem, and researchers have just started to study bugs in quantum software [4, 36, 56]. Prior works have focused on assertions to debug quantum programs, which use statistical tests based on classical measurements. To determine the correctness, prior work uses the chi-square test and contingency table analysis [19]. However, estimating quantum states requires repeat execution of circuits for every assertion point. To reduce the assertion cost, researchers proposed runtime assertion by performing testing and

	random, opt.	OSCAR, opt.	OSCAR, opt.+recon.
ADAM, ideal	3127	370	620
ADAM, noisy	3123	661	911
COBYLA, ideal	38	32	282
COBYLA, noisy	40	32	282

Table 6: Number of QPU queries to reach convergence by ADAM and COBYLA running depth-1 QAOA on 14 instances of the 16-variable MaxCut problem, using OSCAR and random initialization, respectively. "opt." is the number of queries during optimization, and "recon" is the number of queries during landscape reconstruction.

debugging on ancilla qubits [26], introducing dynamic and indirect verifications, and Proq, a projection-based runtime assertion [23], leveraging those projective measurements that will not change the tested state if the state meets the assertion. Besides, some quantum programming languages, such as Q# [50] and Cirq [10], provide unit tests and assertions APIs.

Periodic Landscapes: A growing body of research leverages symmetry in QAOA, VQE, and QML landscapes to enable efficient workflows. For QAOA, [45] found a connection between classical symmetries of the objective function and the symmetries of the terms of the cost Hamiltonian with respect to the QAOA energy. Furthermore, it shows that such symmetry is general and broadly applicable independent of input type. For VQE, symmetry in the landscape naturally arises due to the structure of molecule Hamiltonians. Multiple recent works leverage symmetry of the many-body systems to enable efficient VQE circuits [27, 42, 43]. Whereas for recent work focusing on QML, [22, 32, 34] explored symmetry groups of the learning problem to construct models with outcomes invariant under the symmetry of the learning task. For effective error mitigation, multiple prior works [1, 20, 29, 44, 49] leverage symmetry to enable effective noise mitigation.

Concurrent work on VQA landscape reconstruction: The concurrent work [15] provides theoretical evidence that low-depth QAOA cost landscapes are sparse in the Fourier basis and, thus, are recoverable by CS. Empirically, they conduct limited simulated experiments in the ideal setting. In contrast, we focus on a broader VQA debugging tuning problem. We provide a more thorough and statistically rigorous empirical validation and propose parallel sampling and noise compensation as extensions of this method. We show multiple concrete use cases where OSCAR can be leveraged to accelerate VQA debugging and tuning.

10 CONCLUSION

Debugging VQAs on near-term quantum platforms is extremely challenging due to limited access to the internal state of the quantum program, non-determinism introduced by the optimizer, and noisy operations on near-term quantum computers. In this paper, we propose OSCAR to significantly reduce the overhead of VQA debugging and tuning. OSCAR leverages compressed sensing to generate a complete landscape that provides the user with a bird's-eye view to assess the correctness of all the components involved in executing VQAs. Using simulators and real quantum hardware, we show that OSCAR can generate a complete optimization landscape, executing a small fraction of randomly selected circuits with

high accuracy. In addition, by using OSCAR, we break the serial execution model and enable a parallel execution model that can utilize multiple devices for accurate landscape generation and further improve landscape reconstruction. OSCAR can evaluate the efficacy of noise mitigation techniques in an optimizer-agnostic manner and help users to effectively configure noise mitigation on near-term quantum computers. Furthermore, we demonstrate a use case where OSCAR can help troubleshoot wrong optimization configurations. OSCAR can be used to untangle key components of the VQA workflow including the initialization, optimizer, circuit execution, and noise mitigation, to isolate the bugs and help tune the individual components significantly faster and at a lower cost.

ACKNOWLEDGMENTS

A PRINCIPLE OF COMPRESSED SENSING

Here is a detailed mathematical explanation of CS.

A.1 General Procedure of Compressed Sensing

Denote $x \in \mathbb{R}^n$ as a compressible vector, and it could be represented by a sparse vector $s \in \mathbb{R}^n$ in a new basis $\Psi \in \mathbb{R}^{n \times n}$:

$$x = \Psi s. \tag{5}$$

The sparsity of s is defined as the number of non-zero terms in s. The basis here could be Fourier basis, wavelet basis or other tailored basis. Every column of Ψ is a base orthogonal to each others. Eq. 5 is assuming that the original signal x could be represented by very few coefficients in another basis.

Let's say, $y \in \mathbb{R}^m$ is the measurement vector derived by measurement matrix $C \in \mathbb{R}^{m \times n}$:

$$y = Cx = C\Psi s, \tag{6}$$

where $m < n$. C defines where we measure the original signal x, while y is the measurement result. Eq. 6 is saying that with m points in original signal x, i.e. y, we could still infer s.

Then, we find a s consistent with the y by solving

$$s = \text{argmin}_{s'} \|s'\|_1, \text{ s.t. } y = C\Psi s. \tag{7}$$

Here l_1-minimization will yield a sparse s with high probability if $C\Psi$ satisfies restricted isometry property (RIP).

Once we solve Eq. 7, we reconstruct the original signal by $x = \Psi s$.

A.2 2-D Compressed Sensing

Above is an example of 1-dimensional (1-D) CS. For 2-D CS for image, general idea is to transform to 1-D CS.

Denote n_0 and n_1 as length of rows and columns separately, and $\Psi_{n_0} \in \mathbb{R}^{n_0 \times n_0}$ as DCT basis. Define $\text{vec}(X)$ as stacking columns of matrix X vertically. If $X \in \mathbb{R}^{n_0 \times n_1}$, then $\text{vec}(X) \in \mathbb{R}^{n_0 n_1}$.

Next, we define $\Psi := \Psi_{n_1} \otimes \Psi_{n_0}$ and $n := n_0 n_1$, then we solve Eq. 7 as what we do in 1-D CS. After we obtain x, we reshape $x \in \mathbb{R}^{n_0 n_1}$ into $X \in \mathbb{R}^{n_0 \times n_1}$.

ACKNOWLEDGEMENTS

REFERENCES

[1] X. Bonet-Monroig, R. Sagastizabal, M. Singh, and T. E. O'Brien. [n. d.]. Low-cost error mitigation by symmetry verification. 98, 6 ([n. d.]), 062339. https://doi.org/10.1103/PhysRevA.98.062339 arXiv:1807.10050 [quant-ph]

[2] Sergey Bravyi, Sarah Sheldon, Abhinav Kandala, David C. Mckay, and Jay M. Gambetta. 2021. Mitigating measurement errors in multiqubit experiments. *Physical Review A* 103, 4 (April 2021). https://doi.org/10.1103/physreva.103.042605

[3] T.-Q. Cai, X.-Y. Han, Y.-K. Wu, Y.-L. Ma, J.-H. Wang, Z.-L. Wang, H.-Y Zhang, H.-Y Wang, Y.-P. Song, and L.-M. Duan. [n. d.]. Impact of Spectators on a Two-Qubit Gate in a Tunable Coupling Superconducting Circuit. 127, 6 ([n. d.]), 060505. https://doi.org/10.1103/PhysRevLett.127.060505 Publisher: American Physical Society.

[4] José Campos and André Souto. 2021. Qbugs: A collection of reproducible bugs in quantum algorithms and a supporting infrastructure to enable controlled quantum software testing and debugging experiments. In *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*. IEEE, 28–32.

[5] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. [n. d.]. Stable signal recovery from incomplete and inaccurate measurements. 59, 8 ([n. d.]), 1207–1223. https://doi.org/10.1002/cpa.20124 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpa.20124.

[6] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. [n. d.]. Cost function dependent barren plateaus in shallow parametrized quantum circuits. 12, 1 ([n. d.]), 1791. https://doi.org/10.1038/s41467-021-21728-w

[7] Piotr Czarnik, Andrew Arrasmith, Patrick J. Coles, and Lukasz Cincio. [n. d.]. Error mitigation with Clifford quantum-circuit data. 5 ([n. d.]), 592. https://doi.org/10.22331/q-2021-11-26-592 arXiv:2005.10189

[8] Poulami Das, Swamit Tannu, Siddharth Dangwal, and Moinuddin Qureshi. 2021. Adapt: Mitigating idling errors in qubits via adaptive dynamical decoupling. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 950–962.

[9] Poulami Das, Swamit S Tannu, Prashant J Nair, and Moinuddin Qureshi. 2019. A case for multi-programming quantum computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 291–303.

[10] Cirq Developers. [n. d.]. *Cirq*. https://doi.org/10.5281/zenodo.6599601

[11] L. DiCarlo, J. M. Chow, J. M. Gambetta, Lev S. Bishop, B. R. Johnson, D. I. Schuster, J. Majer, A. Blais, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf. [n. d.]. Demonstration of two-qubit algorithms with a superconducting quantum processor. 460, 7252 ([n. d.]), 240–244. https://doi.org/10.1038/nature08121 Number: 7252 Publisher: Nature Publishing Group.

[12] D.L. Donoho. [n. d.]. Compressed sensing. 52, 4 ([n. d.]), 1289–1306. https://doi.org/10.1109/TIT.2006.871582

[13] Daniel J Egger, Jakub Mareček, and Stefan Woerner. 2021. Warm-starting quantum optimization. *Quantum* 5 (2021), 479.

[14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. [n. d.]. A Quantum Approximate Optimization Algorithm. https://doi.org/10.48550/arXiv.1411.4028 arXiv:1411.4028 [quant-ph]

[15] Enrico Fontana, Ivan Rungger, Ross Duncan, and Cristina Cîrstoiu. [n. d.]. Efficient recovery of variational quantum algorithms landscapes using classical signal processing. arXiv:2208.05958 [quant-ph] http://arxiv.org/abs/2208.05958

[16] P. J. Green and Bernard W. Silverman. [n. d.]. *Nonparametric Regression and Generalized Linear Models: A roughness penalty approach*. Chapman and Hall/CRC. https://doi.org/10.1201/b15710

[17] Lov K. Grover. [n. d.]. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing* (New York, NY, USA, 1996-07-01) *(STOC '96)*. Association for Computing Machinery, 212–219. https://doi.org/10.1145/237814.237866

[18] Matthew P. Harrigan, Kevin J. Sung, Matthew Neeley, Kevin J. Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C. Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B. Buckley, David A. Buell, Brian Burkett, Nicholas Bushnell, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Sean Demura, Andrew Dunsworth, Daniel Eppens, Austin Fowler, Brooks Foxen, Craig Gidney, Marissa Giustina, Rob Graff, Steve Habegger, Alan Ho, Sabrina Hong, Trent Huang, L. B. Ioffe, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Cody Jones, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Seon Kim, Paul V. Klimov, Alexander N. Korotkov, Fedor Kostritsa, David Landhuis, Pavel Laptev, Mike Lindmark, Martin Leib, Orion Martin, John M. Martinis, Jarrod R. McClean, Matt McEwen, Anthony Megrant, Xiao Mi, Masoud Mohseni, Wojciech Mruczkiewicz, Josh Mutus, Ofer Naaman, Charles Neill, Florian Neukart, Murphy Yuezhen Niu, Thomas E. O'Brien, Bryan O'Gorman, Eric Ostby, Andre Petukhov, Harald Putterman, Chris Quintana, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Andrea Skolik, Vadim Smelyanskiy, Doug Strain, Michael Streif, Marco Szalay, Amit Vainsencher, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Leo Zhou, Hartmut Neven, Dave Bacon, Erik Lucero, Edward Farhi, and Ryan Babbush. [n. d.]. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. 17, 3 ([n. d.]), 332–336. https://doi.org/10.1038/s41567-020-01105-y

[19] Yipeng Huang and Margaret Martonosi. [n. d.]. Statistical Assertions for Validating Patterns and Finding Bugs in Quantum Programs. In *Proceedings of the 46th International Symposium on Computer Architecture* (2019-06-22). 541–553. https://doi.org/10.1145/3307650.3322213 arXiv:1905.09721 [quant-ph]

[20] Ashish Kakkar, Jeffrey Larson, Alexey Galda, and Ruslan Shaydulin. [n. d.]. Characterizing Error Mitigation by Symmetry Verification in QAOA. ([n. d.]). arXiv:2204.05852 http://arxiv.org/abs/2204.05852

[21] Abhinav Kandala, Kristan Temme, Antonio D Córcoles, Antonio Mezzacapo, Jerry M Chow, and Jay M Gambetta. 2019. Error mitigation extends the computational reach of a noisy quantum processor. *Nature* 567, 7749 (2019), 491–495.

[22] Martin Larocca, Frederic Sauvage, Faris M. Sbahi, Guillaume Verdon, Patrick J. Coles, and M. Cerezo. [n. d.]. Group-Invariant Quantum Machine Learning. ([n. d.]). https://doi.org/10.48550/arXiv.2205.02261

[23] Gushu Li, Li Zhou, Nengkun Yu, Yufei Ding, Mingsheng Ying, and Yuan Xie. [n. d.]. Projection-based runtime assertions for testing and debugging Quantum programs. 4 ([n. d.]), 1–29. Issue OOPSLA. https://doi.org/10.1145/3428218

[24] Ying Li and Simon C. Benjamin. [n. d.]. Efficient Variational Quantum Simulator Incorporating Active Error Minimization. 7, 2 ([n. d.]), 021050. https://doi.org/10.1103/PhysRevX.7.021050 Publisher: American Physical Society.

[25] Zachary C Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue* 16, 3 (2018), 31–57.

[26] Ji Liu, Gregory T. Byrd, and Huiyang Zhou. [n. d.]. Quantum Circuits for Dynamic Runtime Assertions in Quantum Computation. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne Switzerland, 2020-03-09). ACM, 1017–1030. https://doi.org/10.1145/3373376.3378488

[27] Chufan Lyu, Xusheng Xu, Man-Hong Yung, and Abolfazl Bayat. [n. d.]. Symmetry enhanced variational quantum eigensolver. arXiv:2203.02444 [quant-ph] http://arxiv.org/abs/2203.02444

[28] Carlos Ortiz Marrero, Mária Kieferová, and Nathan Wiebe. [n. d.]. Entanglement Induced Barren Plateaus. https://doi.org/10.48550/arXiv.2010.15968 arXiv:2010.15968 [quant-ph]

[29] Sam McArdle, Xiao Yuan, and Simon Benjamin. [n. d.]. Error-mitigated digital quantum simulation. 122, 18 ([n. d.]), 180501. https://doi.org/10.1103/PhysRevLett.122.180501 arXiv:1807.02467 [quant-ph]

[30] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. [n. d.]. Barren plateaus in quantum neural network training landscapes. 9, 1 ([n. d.]), 4812. https://doi.org/10.1038/s41467-018-07090-4 Number: 1 Publisher: Nature Publishing Group.

[31] David C. McKay, Sarah Sheldon, John A. Smolin, Jerry M. Chow, and Jay M. Gambetta. [n. d.]. Three-Qubit Randomized Benchmarking. 122, 20 ([n. d.]), 200502. https://doi.org/10.1103/PhysRevLett.122.200502 Publisher: American Physical Society.

[32] Johannes Jakob Meyer, Marian Mularski, Elies Gil-Fuster, Antonio Anna Mele, Francesco Arzani, Alissa Wilms, and Jens Eisert. [n. d.]. Exploiting symmetry in variational quantum machine learning. ([n. d.]). arXiv:2205.06217 http://arxiv.org/abs/2205.06217

[33] Prakash Murali, Jonathan M. Baker, Ali Javadi Abhari, Frederic T. Chong, and Margaret Martonosi. [n. d.]. Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers. ([n. d.]). https://doi.org/10.48550/arXiv.1901.11054

[34] Quynh T. Nguyen, Louis Schatzki, Paolo Braccia, Michael Ragone, Patrick J. Coles, Frederic Sauvage, Martin Larocca, and M. Cerezo. [n. d.]. Theory for Equivariant Quantum Neural Networks. https://doi.org/10.48550/arXiv.2210.08566 arXiv:2210.08566 [quant-ph, stat]

[35] G. Nürnberger and Th. Riessinger. [n. d.]. Bivariate spline interpolation at grid points. 71, 1 ([n. d.]), 91–119. https://doi.org/10.1007/s002110050137

[36] Matteo Paltenghi and Michael Pradel. 2022. Bugs in Quantum computing platforms: an empirical study. *Proceedings of the ACM on Programming Languages* 6, OOPSLA1 (2022), 1–27.

[37] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. [n. d.]. On the difficulty of training Recurrent Neural Networks. ([n. d.]). https://doi.org/10.48550/arXiv.1211.5063

[38] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. [n. d.]. A variational eigenvalue solver on a photonic quantum processor. 5, 1 ([n. d.]), 4213. https://doi.org/10.1038/ncomms5213 Number: 1 Publisher: Nature Publishing Group.

[39] Gokul Subramanian Ravi, Kaitlin N Smith, Pranav Gokhale, and Frederic T Chong. 2021. Quantum Computing in the Cloud: Analyzing job and machine characteristics. In *2021 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 39–50.

[40] Salonik Resch, Anthony Gutierrez, Joon Suk Huh, Srikant Bharadwaj, Yasuko Eckert, Gabriel Loh, Mark Oskin, and Swamit Tannu. 2021. Accelerating variational quantum algorithms using circuit concurrency. *arXiv preprint arXiv:2109.01714* (2021).

[41] Vincent Russo, Andrea Mari, Nathan Shammah, Ryan LaRose, and William J Zeng. 2022. Testing platform-independent quantum error mitigation on noisy quantum computers. *arXiv preprint arXiv:2210.07194* (2022).

[42] Ilya G. Ryabinkin and Scott N. Genin. [n. d.]. Symmetry adaptation in quantum chemistry calculations on a quantum computer. arXiv:1812.09812 [quant-ph] http://arxiv.org/abs/1812.09812

[43] Kazuhiro Seki, Tomonori Shirakawa, and Seiji Yunoki. [n. d.]. Symmetry-adapted variational quantum eigensolver. 101, 5 ([n. d.]), 052340. https://doi.org/10.1103/PhysRevA.101.052340 arXiv:1912.13146 [cond-mat, physics:quant-ph]

[44] Ruslan Shaydulin and Alexey Galda. [n. d.]. Error Mitigation for Deep Quantum Optimization Circuits by Leveraging Problem Symmetries. ([n. d.]). https://doi.org/10.48550/arXiv.2106.04410

[45] Ruslan Shaydulin and Stefan M. Wild. [n. d.]. Exploiting Symmetry Reduces the Cost of Training QAOA. 2 ([n. d.]), 1–9. https://doi.org/10.1109/TQE.2021.3066275 arXiv:2101.10296

[46] David Sherrington and Scott Kirkpatrick. [n. d.]. Solvable Model of a Spin-Glass. 35, 26 ([n. d.]), 1792–1796. https://doi.org/10.1103/PhysRevLett.35.1792 Publisher: American Physical Society.

[47] P.W. Shor. [n. d.]. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994-11). 124–134. https://doi.org/10.1109/SFCS.1994.365700

[48] Samuel Stein, Nathan Wiebe, Yufei Ding, Peng Bo, Karol Kowalski, Nathan Baker, James Ang, and Ang Li. 2022. EQC: ensembled quantum computing for variational quantum algorithms. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*. 59–71.

[49] Michael Streif, Martin Leib, Filip Wudarski, Eleanor Rieffel, and Zhihui Wang. [n. d.]. Quantum algorithms with local particle number conservation: noise effects and error correction. 103, 4 ([n. d.]), 042412. https://doi.org/10.1103/PhysRevA.103.042412 arXiv:2011.06873 [quant-ph]

[50] Krysta M. Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. [n. d.]. Q#: Enabling scalable quantum computing and development with a high-level domain-specific language. In *Proceedings of the Real World Domain Specific Languages Workshop 2018 on - RWDSL2018* (2018). 1–10. https://doi.org/10.1145/3183895.3183901 arXiv:1803.00652 [quant-ph]

[51] Swamit S. Tannu and Moinuddin Qureshi. [n. d.]. 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* (Columbus OH USA, 2019-10-12). ACM, 253–265. https://doi.org/10.1145/3352460.3358257

[52] Swamit S Tannu and Moinuddin K Qureshi. 2019. Mitigating measurement errors in quantum computers by exploiting state-dependent bias. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 279–290.

[53] Swamit S Tannu and Moinuddin K Qureshi. 2019. Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 987–999.

[54] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. [n. d.]. Error Mitigation for Short-Depth Quantum Circuits. 119, 18 ([n. d.]), 180509. https://doi.org/10.1103/PhysRevLett.119.180509 Publisher: American Physical Society.

[55] Lorenza Viola and Seth Lloyd. [n. d.]. Dynamical suppression of decoherence in two-state quantum systems. ([n. d.]). https://doi.org/10.1103/PhysRevA.58.2733

[56] Pengzhan Zhao, Jianjun Zhao, Zhongtao Miao, and Shuhan Lan. 2021. Bugs4Q: A benchmark of real bugs for quantum programs. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 1373–1376.