Mitigating Measurement Errors in Quantum Computers by Exploiting State-Dependent Bias

Swamit S. Tannu
swamit@gatech.edu
Georgia Institute Technology

Moinuddin K. Qureshi
moin@gatech.edu
Georgia Institute Technology

ABSTRACT

Quantum computers are susceptible to errors. While quantum computers can be guarded against errors using error correction codes, near-term quantum computers will not have sufficient number of qubits to implement error correction and must perform their computation in the presence of errors. Qubit measurement is typically the most error-prone operation on a quantum computer, with measurement errors ranging from 8% to 30% reported on current machines. This goal of this paper is to mitigate measurement errors by exploiting the state-dependent bias of measurement errors.

Experiments on the IBM-Q5 and IBM-Q14 machines show variation in measurement errors depending on the state being measured. For example, measuring an all-zero state on IBM-Q5 has a fidelity of 84%; however, the fidelity drops to 62% while measuring the all-one state. To improve measurement fidelity, we propose *Invert-and-Measure*, which transforms the system from a vulnerable state to a stronger state and then performs the measurement in the stronger state. We propose two designs for Invert-and-Measure. First, *Static Invert-and-Measure (SIM)*, which executes two instances of the program, one with standard measurements and the other with inverted measurements and combines the results. Second, *Adaptive Invert and Measure (AIM)*, which learns the relative bias of different states using runtime profiling and produces specialized inversions to increase the likelihood of obtaining the correct answer. Our evaluations, using IBM-Q5 and IBM-Q14, show that SIM improves the application reliability by up to 2X, and AIM by up to 3X.

CCS CONCEPTS

• **Hardware: Quantum technologies**;

KEYWORDS

Quantum Compilers, Correlated Errors, NISQ

ACM Reference Format:
Swamit S. Tannu and Moinuddin K. Qureshi. 2019. "Mitigating Measurement Errors in Quantum Computers by Exploiting State-Dependent Bias" . In *The 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-52), October 12–16, 2019, Columbus, OH, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3352460.3358265

1 INTRODUCTION

Quantum computers promise to solve hard problems, which are beyond the capabilities of conventional computers. While quantum computers with several dozens of qubits have already been demonstrated, machines with millions of qubits (required to obtain exponential speedup on applications such as Shor's algorithm) are still several decades away. One of the biggest challenges in building a large scale quantum computer is vulnerability to errors. Quantum computers use qubits to encode and process information. Qubits are extremely susceptible to noise and cannot hold the state for more than several tens of microseconds. One of the ways we can tolerate noise is by using quantum error correction codes. Unfortunately, quantum error correction requires about 20 to 50 qubits to build one fault tolerant logical qubit. Therefore in the near future, quantum computers are likely to be operated in what is called as the *Noisy Intermediate Scale Quantum (NISQ)* model of computing [21].

In a NISQ computing model, the application performs computation, then the output is measured, and this output gets logged. This process is repeated for a large number of trials. If the error-free output appears with a reasonable probability, then the output-log can be used to infer the correct answer. Therefore, one of the key performance metrics for a NISQ machine is the likelihood that the output produced is error-free. This metric is called *Probability of Successful Trial (PST)*. Techniques that can help in reducing the impact of errors on a NISQ machine leads to improvement in PST.

A program running on a NISQ machine can encounter an error due to decoherence, qubit operation, or due to measurement at the end of the computation. Measurement is typically the most error-prone operation on the current quantum computers. For example, on the IBM machine, the average error rate for measurement is 6%-8% with the worst-case measurement error rate of 30%. Measuring a qubit is fundamentally challenging as qubits are extremely low energy devices and during a process of measurement, qubit devices are exposed to noisy measurement circuitry. Thus, even if a quantum machine performs all the computation without encountering an error, in the end, the measurement can still result in an erroneous output. The goal of this paper is to improve the reliability of NISQ computers by mitigating measurement errors.

Measurement collapse the qubit in a state of superposition into a classical binary state, 0 or 1. Thus, a measurement error manifests itself as either a "1" being read as a "0" or vice versa. In this paper, we observe that measurement errors do not affect all states equally. For example, on IBM machines, measurement errors have state-dependent bias such that the state "1" is erroneously read "0" ($1 \longrightarrow 0$) more frequently as compared to state "0" measured as the state "1" ($0 \longrightarrow 1$). While measuring a collective state of N qubits, we would expect to encounter more errors for states that have a large

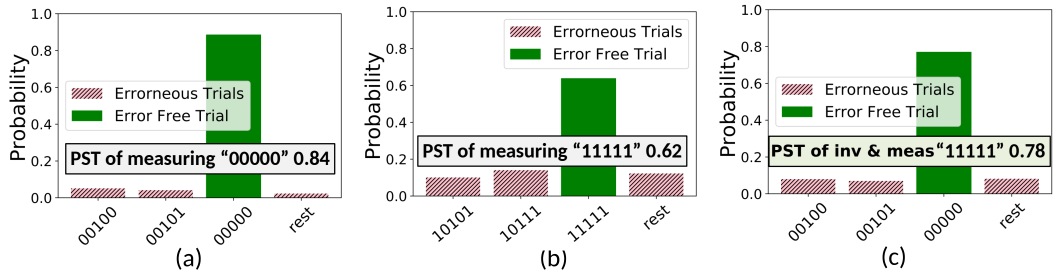

Figure 1: On five qubit ibmqx4, probability of successfully measuring a state (a) All-zero state "00000" (b) All-ones state "11111" (c) Measuring the All-ones state by first inverting the state and then performing the measurement (expected output "00000").

number of ones. We can exploit this bias to mitigate the impact of measurement errors and improve the application reliability.

To show the state-dependent bias in measurement errors we conduct a simple experiment. We initialize the IBM-Q5 (five qubit) machine into an all-zero state (00000) and measured the state. This experiment was repeated for one thousand trials. The experiment is deemed to give the right output if we obtain the 00000 state and incorrect if it gave any of the other 31 possible values. Figure 1(a) shows the probability of obtaining the correct answer and the probability of obtaining a few of the dominant incorrect states. We note that the probability of successful measurement for the all-zero state is 84%. Conversely, if we initialized the machine in an all-ones (11111) state, then the probability of successful measurement drops to 62%, as shown in Figure 1(b). Thus, reading a state of "1" is usually more error-prone than reading a "0" state.

We also conducted an exhaustive experiment with all 32 states ("00000" to "11111") and observed that the probability of successful measurement shows a strong inverse correlation with the Hamming Weight (number of ones) of the state being measured. So, states with higher number of ones are more susceptible to measurement errors than the states with fewer ones. Furthermore, this state-dependent bias is observed even for qubits in the state of superposition. For example, a GHZ state is an equal superposition of all-one and all zero state, when measured it is expected to produce the all-zero state and the all-one state with 50% probability each. We observe that on IBM-Q5, the all-zero state was four times as likely as the all-ones state. We note that our experiments with all three publicly available IBM machines (two 5-qubit machines and one 14-qubit machine) showed such state-dependent bias in measurement errors.

The key insight in this paper is to exploit the state dependent bias to reduce the impact of measurement errors. For example, if we are likely to read a vulnerable state (high Hamming Weight), then we can transform it into a stronger state (low Hamming Weight) by inverting qubits before the measurement, performing the measurement, and inverting the measured result. We refer to such a method of conditionally converting the state to obtain lower measurement errors as *Invert-And-Measure*. For example, reconsider the example of Figure 1(b) where we read the all-one state. Instead, if we inverted the state (by using an additional "X" gate at each of the qubits) before the measurement and then we perform the measurement, then the probability of successful measurement increases from 62% to 78%, as shown in Figure 1(c). Note that the measurement produces a complementary state (all-zeros on correct output) and we must invert this output to get the desired results.

Unfortunately, prior to measurement, we do not know the Hamming Weight of the output (and hence we do not know if the system is in a weak or strong state). Always using inversion before measurement can degrade reliability if the system was already in the strong state. We design two practical policies for Invert-and-Measure.

Our first policy is *Static Invert-and-Measure (SIM)*, which splits the trials into two (or more) groups, each using a different measurement mode: standard and inverted. In the standard-mode, input program is executed and qubits are read without any change. Whereas in the inverted mode, the input program is executed, the qubits are inverted and measurement is performed (the measured output is inverted to maintain correctness). We merge outputs performed in standard-mode and the inverted-mode to obtain the aggregated output. Therefore, SIM ensures that only a subset of the trials are affected by measurement in the vulnerable state. Our measurements on IBM-Q14 (ibmq-melbourne) and IBM-Q5 (ibmqx2 and ibmqx4) machines show that SIM can increase by PST by up to 2X.

While we observe a strong correlation of Hamming Weight of state and the vulnerability of the state to the measurement errors, this correlation is not always perfect, especially for the ibmqx4 machine. If we knew the strongest state for the given machine, then we can steer the output of the machine to map to that state instead of steering it towards an all-zero state. Our second policy, *Adaptive Invert-and-Measure (AIM)* is based on this insight. AIM learns the relative strength of basis state using runtime profiling and performs targeted inversions such that probable solutions get mapped to strong states when inversion is applied. Our evaluations on IBMQ machines show that AIM provides significantly higher improvement compared to SIM. AIM improves PST by up to 3X.

Overall, this paper makes the following contributions:

(1) We show that measurement errors have significant bias depending on the state being measured, with some states significantly more error-prone than other states.
(2) We propose to exploit the bias in measurement errors by performing an inversion before measurement if the state being measured is likely an error-prone state.
(3) We propose *Static Invert-and-Measure (SIM)* that obviates the need for a-priori knowledge of the state being measured by splitting the trials into standard-mode and inverted-mode measurement and merging the results.
(4) We propose *Adaptive Invert-and-Measure (AIM)* that tailors the inversion profile to suit the machine characteristics and maps the likely solutions to be read in the strong state that is specific to the machine.

2 BACKGROUND

2.1 Primer on Qubits and Quantum Gates

Quantum computers owe their powers to two properties: superposition and entanglement. A state of a qubit ($|\psi\rangle$) is a vector in a Hilbert space that is represented as a linear superposition of two basis states $|0\rangle$ and $|1\rangle$ as shown in Figure 2(a). Whereas, combined state of two qubits can be represented as $a_0 * |00\rangle + a_1 * |01\rangle + a_2 * |10\rangle + a_3 * |11\rangle$ such that $a_0^2 + a_1^2 + a_2^2 + a_3^2 = 1$. Qubit Measurement is used to measure/read the state of the qubit. When qubit is measured, it produces binary output (1 or 0) with probability depending on the state of the qubit. As shown in Figure 2(b), when a qubit with a state: $|\psi\rangle$ is measured, it produces "$|0\rangle$" with probability of $|\alpha|^2$ and "$|1\rangle$" with probability of $|\beta|^2$. Whereas, the measurement of n-qubit state outputs 2^n basis states ($000..0_n$ to $111..1_n$) with probabilities corresponding to the superposition of the n-qubit state.

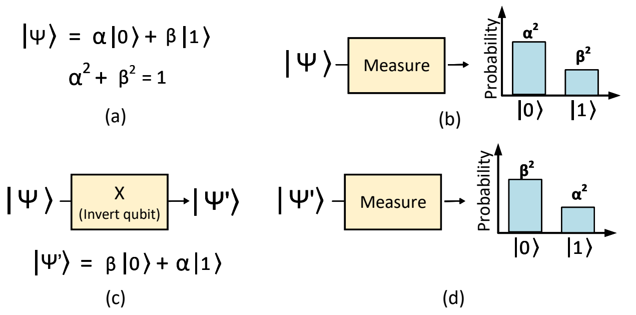

Figure 2: (a) Quantum state is a superposition of $|0\rangle$ and $|1\rangle$ (b) Measurement of a qubit is probabilistic (c) X-gate inverts the qubit state (d) Measurement of inverted qubit

A state of the qubit can be manipulated using a quantum gate. For example, the "X" gate change the superposition by altering the probability amplitudes α and β. In fact, as shown in Figure 2 (c), by applying X-gate on a qubit with state $|\psi\rangle$, we can invert the state of qubit to produce a flipped state $|\psi'\rangle$. When $|\psi'\rangle$ is measured it produces "$|0\rangle$" with probability of $|\beta|^2$ and "$|1\rangle$" with probability of $|\alpha|^2$ as shown in the Figure 2 (d).

2.2 Error Modes on Quantum Computers

Qubits are unreliable as they can lose their state within a few hundreds of microseconds or encounter operational errors. The error rate for a qubit can be defined as a probability of undesired change in the qubit state. Errors on quantum computers can be classified as - Coherence-errors, Gate-errors, and Measurement-errors.

Coherence-Errors: A qubit can retain data for only a limited time and this duration is called *Coherence Time*. A qubit in a high-energy state (state $|1\rangle$) naturally decays to the low-energy state (state $|0\rangle$), and the time constant associated with this exponential decay is called as the *T1 Coherence Time*. It dictates the error rate such that probability of qubit error scales as $e^{-t/T1}$. For existing IBM quantum computers, the average T1 time is about 60μS.

Gate-Errors: Quantum operations are imperfect, and performing gate operations on qubits can also affect their state incorrectly. For IBM quantum-computers, single qubit gates have an error rate in the range of 0.1%-0.3%, whereas the two-qubit gates have an error rate in the range of 2%-5%.

Measurement-Errors: Measurement is the most error-prone operation in current quantum computers. Table 1 shows the minimum, average, and maximum error-rates for the measurement operation (readout operation in IBM terminology) for the three IBM machines that we use in our evaluations. We note that the average error rate for measurement operation in the range of 4%-8% and as high as 31%. The high rate of measurement errors can degrade the application level reliability, especially for low depth programs. Although highly probable, measurement errors are easy to correct as they manifest as bit-flip errors, and in this paper, we leverage this insight to mitigate measurement errors.

Table 1: Error Rate of Measurement Operation

Machine	Error Rate		
Name	Min	Average	Max
`ibmqx2 (IBM-Q5)`	1.20%	3.8%	12.8%
`ibmqx4 (IBM-Q5)`	3.4%	8.2%	20.7%
`ibmq-melbourne (IBM-Q14)`	2.2%	8.12%	31%

2.3 NISQ Model of Quantum Computation

Qubits can be protected against errors by using quantum error correction (QEC) protocols. QEC use extra qubits to introduce redundancy in the information encoding. However, QEC requires a large number of physical qubits (20x-50x) to enable one fault-tolerant logical qubit. Near-term quantum computers (with few tens/hundreds of qubits), may not able to leverage error correction even for an application requiring few dozens of logical qubits. However, there exists a class of applications highlighted by Preskill [21] that can still be viable with such *Noisy and Intermediate-Scale Quantum (NISQ)* machines. Figure 3(a) describes the general computing model for the NISQ machines. In this model, the given program is run, the output qubits are read/measured, and the result is logged. This process is repeated for thousands of trials. As long as the correct results appear with high probability, we can infer the correct result by analyzing the log.

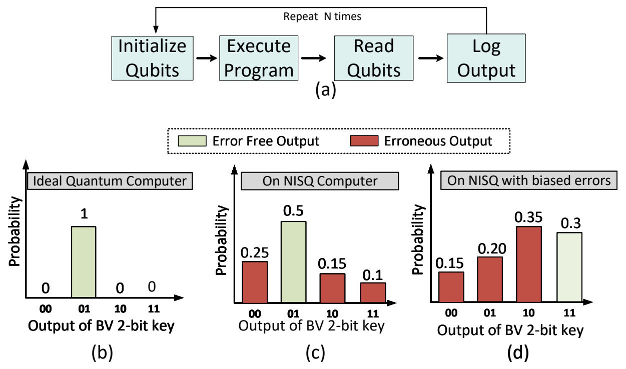

Figure 3: (a) NISQ Model of computation (b) Output of *Bernstein-Vazirani (BV) with 2-bit key* on an ideal quantum computer (c) Successful execution on NISQ machine (d) Unsuccessful Execution on NISQ machine.

2.4 Impact of Errors on NISQ Applications

Errors can cause the NISQ machine to produce an incorrect output. Figure 3(b) shows the distribution of output for a Bernstein-Vazirani (BV) kernel storing 2-bit key "01". On an ideal, error-free machine, we would get the secret key with 100% probability. Whereas on a real quantum computer, qubit errors can produce incorrect output, and we observe distribution of all possible outputs. For example, in Figure 3(c) show a case where the correct output occurs with 50% probability, and each of the incorrect output is generated with no more than 25% probability. Thus, we can correctly infer the key, even in the presence of errors. Now, suppose we stored a different key ("11"), as shown in Figure 3(d). The correct output occurs with 30% probability, but one of the incorrect outputs occurs with 35% probability. In such cases, we can not infer the correct key by picking most frequently occurring output. This can especially happen if certain states are more vulnerable to measurement errors, causing a bias in output.

2.5 Goal of Our Paper

The goal of our paper is to improve the reliability of quantum computers by mitigating errors due to measurement operations. We note that measurement errors are simply binary errors where a "1" is inferred instead of a "0" and vice versa. We observe that measurement errors do not affect all the states equally and states storing a "1" are typically more prone to measurement errors – perhaps because the measurement is a long latency operation and the coherence error can take the qubit from a high energy state to a low energy state. We use this bias in measurement errors to perform error mitigation. We provide experimental methodology next and then perform a characterization of measurement errors.

3 BIAS IN MEASUREMENT ERRORS

This section provides characterization data for measurement errors that highlight the state dependent bias in measurement errors and it's correlation with the Hamming Weight (the number of 1s).

3.1 State Dependent Bias

Qubits have a natural tendency to relax to the low energy state (0) from the high energy state (1). This creates a data dependent bias in measurements as qubit which already in State 0 is less likely to change its state due to natural relaxation process as compared to the qubit in State 1. For example, our evaluations on IBM quantum computers show that a qubit in an excited state is more likely to encounter an error as compared to the qubit in a low energy state. To understand the state-dependent bias in measurement errors, we generate all the 32 possible basis states (00000 to 11111) and measure each state 16 thousand times. We compute the *Basis Measurement Strength (BMS)* as the ratio of number of correct measurement to the total number of trials for all 32 basis states. Figure 4 show the relative BMS for all 32 states on IBM's five qubit`ibmqx2` machine. Note that the x-axis is in the ascending order of Hamming Weight. We calculate relative BMS by dividing the BMS of each state with the highest BMS. For `ibmqx2`, state "00000" is the strongest basis state, whereas state "11111" is the weakest state with relative BMS of 0.38 as the BMS reduce with increasing Hamming Weight.

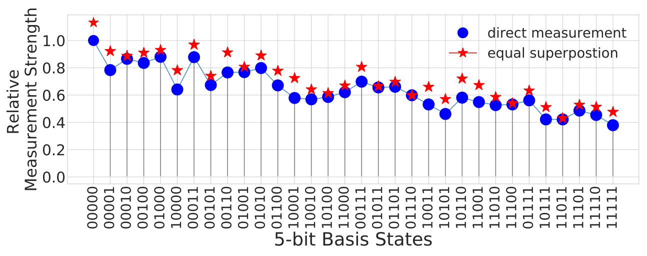

Figure 4: Probability of Successful Measurement for `ibmqx2` basis states using equal superposition and direct basis measurement. (X-axis shows five bit basis states in ascending order of hamming weights)

With unbiased measurements, BMS for all the states should be similar. However, our evaluations suggest that the probability of successful measurement is inversely proportional to the Hamming weight of basis states (Correlation coefficient = -0.93). Thus, the larger the Hamming weight, the higher the probability of measurement error. To understand the impact of the size of the quantum computer on measurement bias, we run similar experiments with 10-bit basis states on `ibmq-melbourne` for total 150 thousand trials. Figure 5 show the relative BMS for all 1024 basis categorized as per the Hamming Weight of the basis state. The data again shows a strong inverse correlation between the measurement strength and the Hamming Weight.

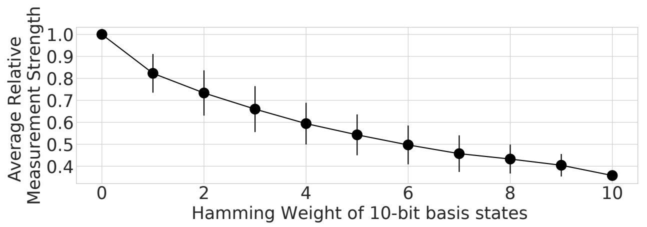

Figure 5: Relative Basis Measurement Strength for the `ibmq-melbourne` machine. The data is averaged over all the basis states with the identical Hamming Weight.

3.2 Impact of Bias on Superposition of States

Thus far, we have measured the vulnerability of measurement errors to states that do not have any superposition (classical states). However, bias in measurement error can impact qubit states with superposition. To understand how measurement bias affects the superposition of states, we create the *Greenberger-Horne-Zeilinger (GHZ)* state, which is an equal superposition of basis-states "00000" and "11111" (GHZ-5 $= \frac{1}{\sqrt{2}}(|00000\rangle + |11111\rangle)$).

If the GHZ-5 state is prepared and measured on a quantum computer with no errors, the output will be either "00000" or "11111" with 0.5 probability respectively. However, when prepared and measured on the IBM machine, the probability of measuring state "00000" and state "11111" are unequal. As shown in the Figure 6,probability of measuring state "00000" reduces from from 0.5 to 0.4 and from 0.5 to 0.1 for sate "11111". Our experiments suggest that measurement bias extends to the superposition of basis states. Note that GHZ states are considered to be the maximally entangled state; thus measurement bias also affects qubits that are entangled.

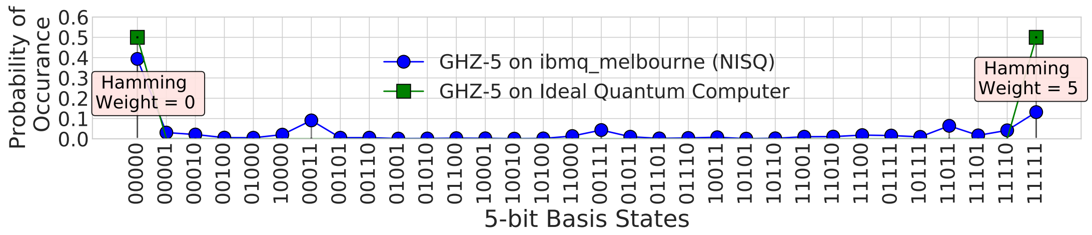

Figure 6: Output probability distribution for GHZ5 (X-axis is in ascending order of the Hamming Weight). On ideal machine both "00000" and "11111" occur with 0.5 probability – errors affect state 11111 four times as state 00000.

Similar to GHZ, bias affects other superposition states as well. For example, if we prepare superposition of all states for five qubits using Hadamard (H) gates, and measure the output probability distribution, it strongly correlate with the with the relative measurement strength as shown in the Figure 4.

3.3 Impact of Bias on NISQ Applications

State-dependent bias in the measurement errors produces non-uniformity in the measurement strength of basis states. The skew in measurement strength degrades the reliability of NISQ machines. For example, if the desired or optimal answer is a weak basis state, then the probability of measuring the answer drops significantly. Furthermore, weak states are often incorrectly measured as strong states. To understand this masking effect, we execute an instance of QAOA on the `ibmq-melbourne` machine. We use QAOA to solve the max-cut problem for five input graphs (Graph-A to Graph-E) such that each graph consists of six nodes and the desired output (the partition that maximizes the cost function) is in the increasing order of Hamming Weight, as shown in the Table 2. We execute each graph for 32 thousand trials. All five graphs use an identical number of gates and the optimal mapping (aware of variation in error rates of different qubits).

Table 2: Impact of measurement bias on QAOA

Input	Optimal	Hamming	Metric		
Graph	Output	Weight	PST	IST	ROCA
`Graph-A`	010000	1	6.5%	1.3	1
`Graph-B`	010100	2	5.5%	1.01	1
`Graph-C`	101001	3	5.0%	0.70	7
`Graph-D`	101011	4	1.9%	0.59	14
`Graph-E`	110110	4	1.5%	0.23	24

Table 2 shows the Probability of Successful Trials (PST), Inference Strength (IST) that ratio of frequency of correct answer to the frequency of most dominant incorrect answer and the rank of the correct answer for the five graphs. We observe that PST is inversely correlated to the Hamming Weight. As for input graphs A and B, the PST is 2x higher as compared input graph D and E. Thus, state-dependent measurement bias can significantly deteriorate the reliability of the application.

The bias in measurement has a significant impact on the IST and Rank of correct answer as well. IST drops significantly for the input graphs E and F. When the expected output is a weak state (Hamming Weight= 3 or 4) this indicate that incorrect answers have a higher frequency of occurrence compared to the correct answer.

Along with IST, we use the *Rank of Correct Answer (ROCA)* to understand how measurement bias impacts the ability to infer the error-free answer. For A and B the correct answer appears with the highest frequency, whereas, for D and E (high Hamming Weight) the incorrect answers are more dominant than the correct answer. Our goal is to reduce the impact of measurement errors, and make the system fidelity be less dependent on which state is being measured. We describe our methodology before discussing our solution.

4 METHODOLOGY

4.1 NISQ Benchmarks

Developing applications for near-term quantum computers is an open problem [16, 21]. *Quantum Approximate Optimization Algorithm (QAOA)* [6] has emerged as an appealing use for NISQ machines as it can be used to solve challenging optimization problems. We use QAOA as one of the kernels for our evaluation. The other kernel we use is Bernstein-Vazirani (BV)[1] which deduces a secret key hidden inside a quantum oracle. Note that both `bv` and `qaoa` produce a binary string as the solution. For example, the output of QAOA when solving a max-cut problem represents a partition that maximizes the cost function. Whereas, BV produces a secret key hidden in the quantum oracle and outputs a binary string corresponding to the secret key. On an ideal quantum computer (with no errors), these applications will produce correct output with certainty. For BV, the correct string is generated with probability of one, whereas for QAOA, the correct output string has the highest frequency of occurrence. Note that QAOA solves an optimization problem – solutions produced with QAOA can be used to calculate the cost function, and the answer corresponding to optimal value among all the tested solution can be used as the good enough solution. Table 3 shows the configurations of BV and QAOA used in our study. For both QAOA and BV the number of gate operations and measurement operations scale linearly with problem size.

Table 3: Benchmark Characteristics

Benchmark	Problem/Algorithm	Output
`bv-4A`	4 bit Bernstein-Vazirani	Secret: 0111
`bv-4B`	4 bit Bernstein-Vazirani	Secret: 1111
`bv-6`	6-bit Bernstein-Vazirani	Secret: 011111
`bv-7`	7-bit Bernstein-Vazirani	Secret: 0111111
`qaoa-4A`	max-cut for 4 node graph	Output cut: 0101
`qaoa-4B`	max-cut for 4 node graph (p=2)	Output cut: 0111
`qaoa-6`	max-cut for 6 node graph (p=2)	Output cut: 101011
`qaoa-7`	max-cut for 7 node graph (p=2)	Output cut: 1010110

4.2 Reliability Metrics

Our goal is to improve the reliability of NISQ machines. While PST has been commonly used as the metric to denote the system level reliability, we discuss two additional metrics that can provide further insight into assessing the reliability of NISQ machines, depending on the application.

4.2.1 ***Probability of Successful trial (PST):***. PST has been used to evaluate the reliability of NISQ applications [14]. To calculate PST, a NISQ program is run multiple times, and the output of each trial is logged. PST is evaluated by computing the ratio of a number of error-free trials to the total number of trials. PST can be evaluated for a class of problems that have known correct solution. However, for benchmarks such as QAOA, we can have two possible correct answers: a binary string that denotes the most optimal partition and it's inversion. In such case, for all our evaluations, we use cumulative frequency of occurrence for both of the strings. Note that by only knowing the PST, we do not know if the execution will lead to a successful outcome or not.

$$\boxed{PST = \frac{\text{Number of Trials with Correct Solution}}{\text{Total Number of Trials}}}$$

4.2.2 ***Inference Strength (IST):***. When running a NISQ application, we need to consider the frequency of error-free outcomes along with the erroneous outcome, as incorrect outputs can mask the error-free outputs. Thus, suppressing erroneous trials is essential to determine the error-free answer from the erroneous ones. To quantify this, we propose *Inference Strength (IST)*, which is the ratio of the frequency of error-free output to the number of most frequently occurring erroneous output. IST can easily capture the cases where the incorrect answer can dominate the correct answer. For example, the correct answer appears with the highest frequency in the output log only if IST exceeds 1.

$$\boxed{IST = \frac{\text{Probability of Correct Solution}}{\text{Probability of Strongest Incorrect Solution}}}$$

4.2.3 ***Rank of Correct Answer (ROCA):***. For discrete optimization problems, such as QAOA, we get a solution from the NISQ machine, and we would compute the cost associated with the solution. However, instead of only testing the most frequently occurring solution on the NISQ machine, we could also test "top K" answers for possible solution. To compute the rank of the correct solution, we sort the output log in descending order using frequency of occurrence and use the Rank of Correct Answer (ROCA) as a metric to capture the effectiveness of the NISQ machine.

4.3 Machine Configuration and Parameters

For our evaluations, we use the publicly available quantum cloud service from IBM [3]. We conduct our experiments on three quantum machines, as shown in the Table 4. We use multiple machines to understand the machine specific and general measurement bias. We evaluate all the benchmarks using the most optimal qubit allocation for both the baseline experiments and for proposed bias mitigation techniques. We use allocations that are cognizant of underlying noise and variation in the error rate such that benchmarks are mapped on strongest qubits and links with minimum number of SWAPs. When running the benchmarks, we ensure that the identical program (number of gates, and position of qubits) are performed for the baseline and proposed policy. Moreover, we run each benchmark for more than 32,000 trials and ensure that the baseline and the proposed policy are evaluated in the same calibration window such that the evaluations for the baseline and proposed policy are executed in intertwined batches.

Table 4: Quantum Machines

Platform	`ibmqx2`	`ibmqx4`	`ibmq-melbourne`
Number of Qubits	5	5	14

5 EXPLOITING BIAS USING INVERSION

Our characterization data shows that there is a significant bias in the measurement errors, and these errors tend to show a strong correlation with the Hamming Weight of the output being measured. We could exploit this bias to reduce the impact of measurement errors on the NISQ machines.

5.1 Invert-and-Measure: The Basic Concept

If we could guess the state being measured, and if it was a state that is highly vulnerable to measurement errors, then we could instead perform the measurement in an inverted mode. In the inverted mode, the qubits would first be inverted (using the X gate), and then the measurement is performed. The measurement would give an output complementary to what is expected; however, we can perform inversion on the measured output to get the desired state. We exploit this insight in our proposal, *Invert-and-Measure*. For example, if we were reading an all-ones state (highly error-prone) then inverting the state will allow us to perform our measurement in the all-zeros state (less error-prone).

Performing all the measurement in an inverted form is not always beneficial. In fact, it can lead to more errors. For example, if we were reading a strong state and we inverted and measured, then we would increase the error rate of the system. Therefore, inverted measurement makes sense if we know the state we are reading. Applying an inverted measurement thus faces a practical challenge as we do not know the state we are measuring before performing the measurement. We propose *Static Invert-and-Measure* that avoids the reliance on knowing which state is being measured.

5.2 Static Invert-and-Measure

Rather than performing all of the trials in the same measurement mode, our Static Invert-and-Measure (SIM) policy divides the trials into multiple groups and performs a different measurement mode on each group. The simplest form of SIM is to have two measurement modes: standard and inverted, and use each measurement mode for half the trials, as shown in Figure 7(a). Note that for measurements performed in the inverted mode, we flip the measured results to get the expected output. The distribution obtained from both modes of measurement is then combined to obtain an aggregate distribution over all the trials. As SIM divides

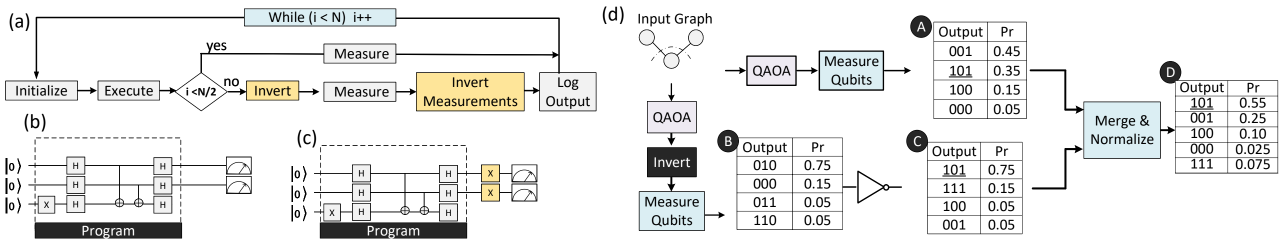

Figure 7: Static Invert-and-Measure (SIM): Split trials into standard-mode and inverted-mode, and merge results

the measurements into groups, the system may perform only half of the measurements in the vulnerable state and the other half in the stronger state so that the errors can get averaged out.

We explain the operation and effectiveness of SIM with an example, as shown in Figure 7(b). Consider a QAOA application that produces a 3-bit output. The expected output of QAOA on an ideal quantum computer is "101". If we run the application on a NISQ machine, we can get different output in different trials. The PST is 0.35, and the incorrect answer "001" (with lower Hamming Weight) is measured more frequently as compared to the correct answer 🅐. Whereas, for the inverted measurements, our expected measurement would be "010" (with high probability) 🅑. We flip the measurements obtained in an inverted mode to obtain the desired probability distribution for the inverted mode 🅒. Combining the distributions from the standard mode and inverted mode produces the correct answer with the highest probability (PST=0.55), as shown in 🅓. Thus, SIM limits the vulnerability of measurement errors to bias towards only one group and averaging the results from different measurement mode improve the overall reliability of the system.

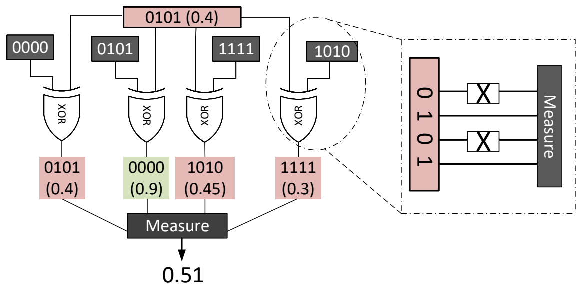

Figure 8: SIM with four inversion-Strings: all-zeros, all-ones, even-bits-1, and odd-bits-1. Averaging over four modes increases the likelihood of getting a stronger state

5.3 Generalizing SIM to Multiple Modes

The basic insight in SIM is to transform a state being measured into another state which might be less vulnerable to measurement errors. This is achieved by inverting the measurement for some of the trials. We can generalize this concept of measuring in a different basis as having an *Inversion-String* that is applied to the set of qubits being measured before doing the measurement. In standard mode, the Inversion-String is all zeros, so no inversion is applied, and the state is read as is. In the inverted mode, the Inversion-String is all ones, so all the qubits are inverted before performing the measurement. For an N-qubit machine, there are 2^N possible Inversion-Strings. In fact, if we divide all the trials into 2^N groups, and applied a unique Inversion-String to each of the group, then we would get an average value of measurement error, regardless of the state being measured.

Applying all possible Inversion-Strings may not be a viable option, especially for a machine with dozens of qubits, given that the number of Inversion-Strings grows exponentially with the number of qubits. However, even if we choose a handful of Inversion-String, we can still average out the measurement errors on those measurement modes. For example, the SIM policy described earlier had two Inversion-Strings (all-zeros and all-ones) and is optimized for cases where the state being measured is either very low Hamming Weight or very high Hamming Weight. We could add additional measurement modes which are designed for states that may have moderate Hamming Weight. For example, we could add an Inversion-String that has an alternating string of 1s and 0s. So, our policy may have four Inversion-Strings altogether: all-ones, all-zeros, even-bits-one, and odd-bits-one. Such a design with four Inversion-Strings is shown in Figure 8. In this example, we are measuring state "0101" which has BMS of 0.4, whereas the inverted state "1010" can be measured with 0.45 probability. Thus, in this case, SIM with only two modes (standard and inverted) may not be effective in mitigating measurement bias. Whereas, if we use four inversion strings ("0000", "1111", "0101", "1010") as shown in the Figure 8, then the resultant measurement has higher reliability.

We find that such an implementation of SIM with four Inversion-Strings is effective at providing measurement errors close to average, without requiring us to know the state that is being measured and without knowing the machine characteristics (which states are vulnerable and which are not vulnerable). For our experiments with SIM, we split the trials into four equal groups. We use the four Inversion-Strings: no inversion ($00000..0_n$), full inversion ($11111..1_n$), even qubit inversion ($10101..1_n$), and odd qubit inversion ($01010..0_n$). Using these inversion strings, we generate four copies of a program and run each copy for an equal number of trials. For partial and fully inverted copies we perform post-measurement flips and combine all the outputs. We use inversion strings that split the Hamming space into four equal parts. By adding more inversion strings and execution modes, we can achieve incremental benefits in IST at the cost of running extra trials.

5.4 Impact of SIM on Reliability of QAOA

To understand the effectiveness of SIM, we use QAOA to find the max-cut for the input graph-D (output string: 101011). We run the application on IBM-Q14 for 16 thousand trials in the baseline configuration. The output of QAOA with the baseline policy is shown

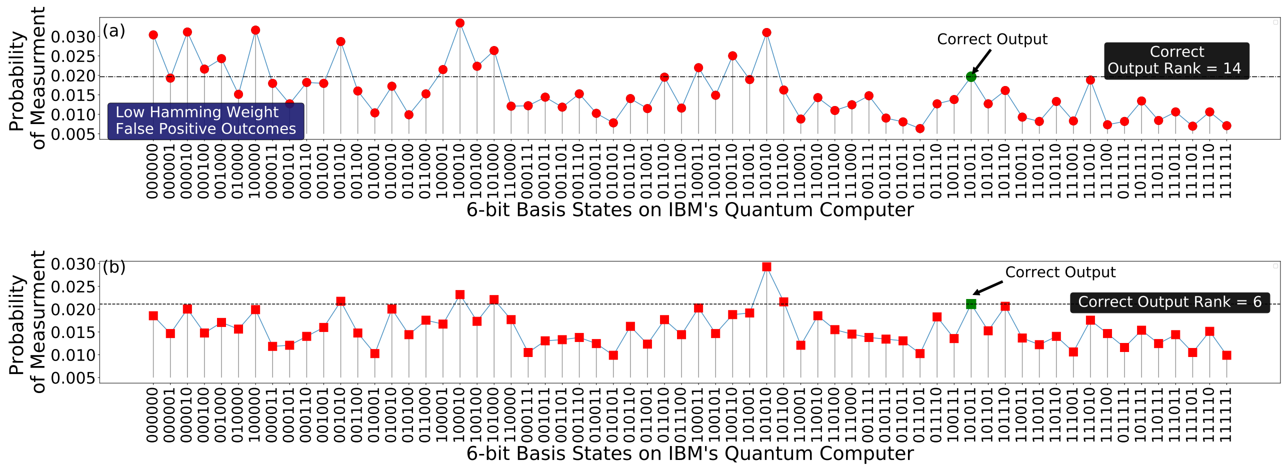

Figure 9: Output of QAOA on the IBM-Q14 machine using the (a) Baseline policy (b) SIM

in the Figure 9(a). The baseline PST is only 1.9% and the Relative Strength is 0.59. With the baseline policy, a significant number of incorrect answers are generated, especially incorrect answers that tend to have low Hamming Weight. The baseline produces 13 incorrect outcomes with a higher frequency of occurrence than the correct answer, with ROCA of 14.

To improve reliability, we run QAOA with SIM. We prepare, four copies of the executable such that first copy uses non-inverted measurements, second and third copy uses alternating partial inversions, and the fourth copy uses fully inverted measurements. We run each copy for 4096 trials and combine all four distributions after post-correcting the outputs. Figure 9(b) shows the distribution of outputs produced by SIM. SIM improves PST by 10% and the IST by 23%. The Rank of the correct answer improves from 14 to 6. Thus, SIM can attenuate several of the incorrect outputs by averaging out the measurements over a larger number of measurement modes.

5.5 Impact of SIM on PST

We conduct our experiments on three IBM machines: `ibmqx2`, `ibmqx4`, `ibmq-melbourne`. Figure 10 shows the PST of the system with SIM, normalized to the PST of the baseline machine. We observe that across all three machines, SIM improves PST. SIM can improve PST by as much as 2X for `ibmqx4`. SIM provides an improvement in other reliability metrics (such as Relative Strength and Rank) as well; however, we report results on those metrics in Section 7.

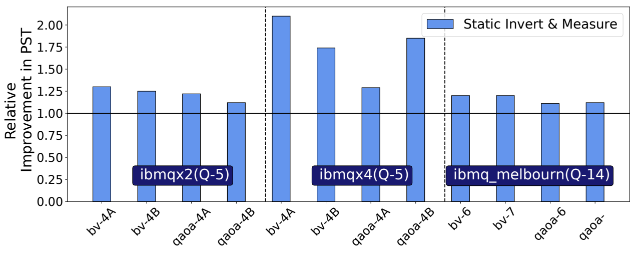

Figure 10: Impact of SIM on PST of the three machines.

6 ADAPTIVE INVERT-AND-MEASURE

With SIM, we perform measurement in four different modes with the expectation that these inversion strings will average out the errors, and overall reliability will be dictated by the average error-rate for measurement rather than the worst-case. We do this because we do not know the state being measured. However, if we could predict (using runtime profiling) the state being measured, then we would use an Inversion-String that maps the given state to the strongest state. For example, if the strongest state is all-zeros, then the Inversion-String will be the same as the state being measured. This would ensure that the system always performs measurement in state with minimum measurement error rate.

While running a few "canary" trials to estimate the likely output looks like a promising option, the success of getting the right answer with a non-negligible probability again depends on the error rate for the state being measured during the canary trials. If the state that we are interested in measuring is a highly error-prone state, then even in the canary trials we would encounter an error and get the wrong output. Therefore, we need both the profile information of the application as well as the profile information of the machine to make use of the canary trials. Unfortunately, the measurement error does not have a perfect correlation with the Hamming Weight, and for some machines (e.g.`ibmqx4`) this correlation is weak.

6.1 Arbitrary Measurement Bias and Impact

On `ibmqx2` and `ibmq-melbourne` measurement strength of the basis state is inversely proportional to its Hamming Weight. However, not all quantum computers have measurement strengths that scale predictably with Hamming Weight. For example, on `ibmqx4`, IBM's five-qubit quantum machine, we observe an arbitrary measurement bias such that measurement strength is not strongly correlated with the Hamming weight of the basis state. Figure 11 show the relative measurement strength for all 32 basis states on IBM's five-qubit machine. The data shows that the strength of the measurement is not monotonically decreasing with the Hamming Weight of the basis state. To test if the bias is repeatable, we evaluated the measurement strength of different five-qubit basis states for 35 days over 100 calibration cycles. We observe that the bias is repeatable.

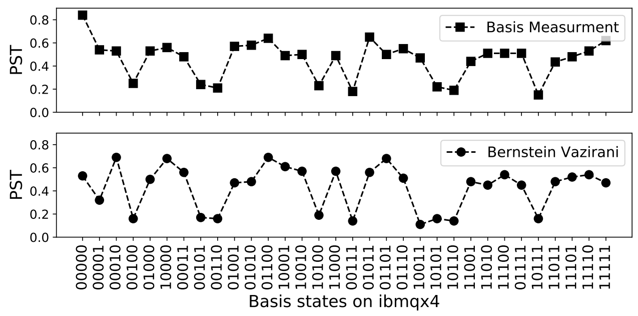

Figure 11: (a) Probability of Successful Trial (PST) for `ibmqx4` states (b) Probability of Successful Trial (PST) for BV for different desired output states.

The state-dependent bias, variability, and machine-specific errors can collectively produce an arbitrary bias. Also, measurement errors has a significant impact on the reliability of the NISQ machine. For example, we execute 32 instances of BV-4 (each for 24 thousand trials) on `ibmqx4` such that each instance outputs 5-bit basis state (4-bit secret key and 1-bit ancillary qubit). The PST for the experiments with 32 different keys is shown in Figure 11(b). The x-axis is the secret input key, and the states are arranged in the increasing order of the Hamming Weight. We can observe a positive correlation between the PST and the measurement strength as weak basis states have significantly lower PST compared to the stronger states.

6.2 Design of Adaptive Invert-and-Measure

To adapt to any arbitrary bias in measurement, we propose Adaptive Invert and Measure (AIM). AIM uses run-time profiling to build the measurement strength curve for a given quantum computer and uses targeted inversions. Figure 12 shows an overview of AIM.

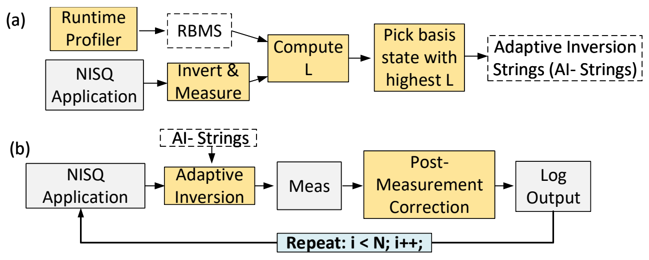

Figure 12: Design of Adaptive Invert-and-Measure

AIM contains three parts: (1) generating machine profile for measurement strength, (2) generating likely outputs for the application using canary trials, and (3) running the application with tailored Inversion-Strings that map the likely output states to the strongest state of the machine. We describe these steps below.

6.2.1 Generating Measurement Strength Function: For a small machine (such as IBM-Q5) we can build an *Relative Basis Measurement Strength (RBMS)* by measuring the probability of successful measurement for each of the possible states. However, for a larger machine (such as IBM-Q14), evaluating all possible measurement states is not a viable option due to the exponential growth in the number of states. For IBM-Q5, we use a brute-force approach (similar to Figure 11(a)). Whereas, for IBM-Q14, we use a divide-and-conquer approach, where we learn the RBMS characterizing one window of 4-qubits at a time (sliding window). For details please refer to the Appendix-A: Characterizing RBMS.

6.2.2 Generating Candidates for Likely Output: AIM performs canary trials using the four Inversion-Strings used in SIM to produce an output distribution that removes global bias. However, note that a state that is low strength may appear with low probability in this distribution simply due to measurement errors. Therefore, we scale the output distribution with an inverse value of measurement strength (for example, if the measurement strength of state X is 0.1 and state Y is 0.2, but both occur with same frequency, then we scale the likelihood of X by a factor of two compared to Y).

We denote L_i as the likelihood that the measured basis state i is correct. L_i is defined by Equation 1.

$$L_i = \frac{\text{Probability of occurrence of state i in output}}{\text{Measurement strength of the state i}} \tag{1}$$

We sort and select top "k" strings with the highest L value. In practice, these "k" strings (or the strings within one or two hamming distance) are the most likely to be the correct output.

6.2.3 Generating Inversion-Strings for Execution: When the likely outputs are available, we use the Inversion-String that can map them to the strongest state. For simplicity, if the strongest state is an all-zero state, then the Inversion-String is the same as the predicted output. We run the execution for a given number of trials with this tailored Inversion-String. We do this for all of the "k" predicted outputs (we use K=4 in our study). For our evaluations, if we have N number of trials in the baseline, we use 25% of the trials as canary trials to generate the possible outputs for the application. For the remaining 75% of the trials, we use the tailored Inversion-String to perform the experiments. The total number of trials of the application remains the same for the baseline and AIM.

6.3 Impact of AIM on Reliability of BV

Both SIM and AIM try to transform the state being measured into another state using Inversion-Strings. SIM does so without any knowledge of the application and the system characteristics using statically selected strings. AIM performs system and application profile to generate specialized Inversion-Strings to get the strongest state for the measurement. Figure 13 shows the PST of BV for the baseline, SIM, and AIM on the `ibmqx4` machine. We experiment with all possible basis states. We note that the PST with the baseline and SIM are quite variable and the fidelity depends on the states, with some states having quite low fidelity. With AIM, the PST remains rather stable across all the possible states. Compared to the baseline and SIM, AIM continues to have a consistently high PST, with the exception of state all-zeros (the all-zero state is the strongest, so the baseline has the highest PST). Thus, AIM not only improves the PST but also makes the system have less dependence on the values used by the applications.

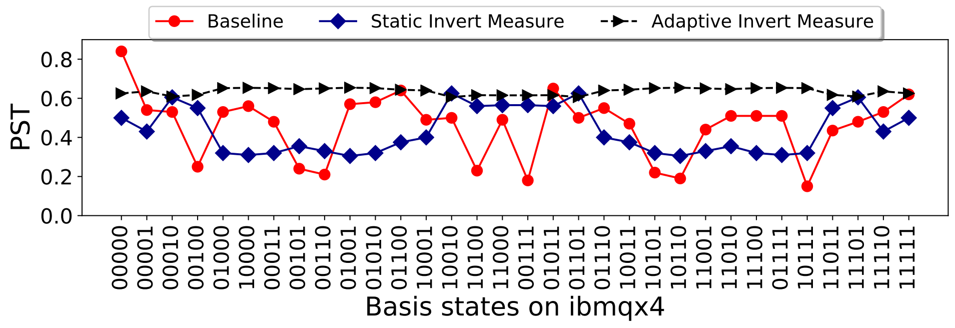

Figure 13: Bernstein-Vazirani executed on ibmqx4 for all possible keys using: Baseline, SIM, and AIM. Note that for the baseline machine the application fidelity depends on the value of the stored key. Whereas with with AIM the PST remains high for all possible states (and is close to the maximum, except for the trivial case of all-zero key).

7 EVALUATIONS

We use Probability of Successful Trial (PST), and Inference Strength (IST) as the figure-of-merit to evaluate effectiveness of SIM, and AIM on `ibmqx2`, `ibmqx4`, and `ibmqx-melbourne`.

7.1 Impact on Inference Strength

Table 5 shows the IST (ratio of the frequency of the correct output to the frequency of the strongest incorrect output) for the baseline, SIM, and AIM. If the IST is less than one, then the correct output is not the most frequently appearing output string, and errors may have masked the correct output. For example,on `ibmqx2`, with baseline policy, BV-4B, QAOA-4A, and QAOA-4B produce incorrect answer more frequently than the correct answer. Whereas, using SIM and AIM, the benchmarks produce correct output with highest frequency such that SIM improves IST by 1.2x and AIM improves it by 1.56x. Note that `ibmqx2`, was consistently the most reliable NISQ machine during our evaluations and showed relatively low variability in errors. On the other hand, `ibmqx4` had significantly higher gate and measurement error rates. SIM improves the IST by 3.4x and AIM improves it by 7.2x. In fact, on `ibmqx4`, SIM improves the IST from 0.46 to 2.85 (6.2x) and AIM improves the inference strength to 10.38 (22.5x improvement) for BV-4A. Whereas, for benchmarks executed on `ibmq-melbourne`,

Table 5: Inference Strength (IST) for Baseline, SIM, and AIM

Benchmark	Platform	Baseline	SIM	AIM
BV-4A	ibmqx2 (5 Qubits)	1.22 ✓	**1.12** ✓	**1.32** ✓
BV-4B	ibmqx2 (5 Qubits)	0.9	**1.25** ✓	**1.83**✓
QAOA-4A	ibmqx2 (5 Qubits)	0.73	0.86	**1.27** ✓
QAOA-4B	ibmqx2 (5 Qubits)	0.72	0.96	**1.12** ✓
BV-4A	ibmqx4 (5 Qubits)	0.46	**2.85** ✓	**10.38** ✓
BV-4B	ibmqx4 (5 Qubits)	**4.8** ✓	**6.4** ✓	**5.7** ✓
QAOA-4A	ibmqx4 (5 Qubits)	0.82	**1.94** ✓	**2.03** ✓
QAOA-4B	ibmqx4 (5 Qubits)	0.72	**2.67** ✓	**1.98** ✓
BV-6	ibmq-melbourne	0.70	0.93	**1.02** ✓
BV-7	ibmq-melbourne	0.62	0.84	**1.09** ✓
QAOA-6	ibmq-melbourne	0.23	0.72	0.86
QAOA-7	ibmq-melbourne	0.18	0.36	0.78

SIM improves the relative strength by 1.9x and AIM improves it by 2.8x. Note that although `ibmq-melbourne` had a high measurement error rate we did not observe as high an improvement in IST as for scaled benchmarks (BV-6, BV-7, QAOA-6, and QAOA-7) because measurement errors are not the only errors that produce incorrect answers. For example, gate errors can degrade the IST as well, and AIM and SIM can not protect against operational errors.

7.2 Impact of SIM and AIM on PST

By mitigating bias in the qubit measurement, we can improve the measurement fidelity and overall reliability on NISQ machines. Figure 14 shows the Probability of Successful Trial (PST) of SIM and AIM, normalized to the baseline that uses variability-aware qubit allocation. For `ibmqx2`, SIM improves the PST by 22% (up-to 30%) and AIM improves it by 40% (up-to 56%). Due to highly biased measurement errors on `ibmqx4`, SIM improves the PST by 74% (up-to 85%) and AIM improves it by 290% (up-to 329%). For `ibmq-melbourne`, SIM improves the PST by 16% (up-to 20%) and AIM improves it by 27% (up-to 36%). Both SIM and AIM are simple techniques that improve the reliability of the NISQ machines by mitigating measurement errors.

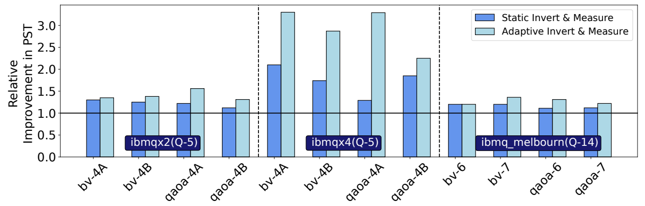

Figure 14: Probability of Successful Trial (PST) for SIM and AIM, normalized to the baseline. SIM improves PST by up-to 2X, whereas AIM improves PST by up-to 3X.

Operating NISQ machines with consistently high reliability and low variability is extremely challenging as qubit devices are sensitive to operating conditions. As the complexity of NISQ machines will increase ensuring optimal operating condition for each and every qubit device may not be possible. Therefore, it is essential to develop techniques that can alleviate reliability issues at a software level. The proposed techniques: SIM and AIM, achieve this goal by using inverted measurements to mitigate state dependent bias in measurement errors.

8 RELATED WORK

Improving the reliability of NISQ is an interdisciplinary area of research. Following are the current broad research directions.
Compiler Techniques: Several prior works have focused on the compiler techniques to eliminate redundant gates and minimize the number of SWAPs [2, 8, 13, 15, 19, 20, 22, 23, 31, 33]. While early papers on qubit allocation were focused on the SWAP minimization, recent work [26] exploit the hardware characteristic when allocating qubits. Moreover, [17, 18, 28], show significant reliability gain using variability-aware compilation for NISQ.
Application and Device Specific Techniques: To tolerate noise, researchers are developing and benchmarking algorithms that are inherently resilient to noise, and require less number of resources [4, 7, 32]. Moreover, to mitigate the errors, researchers have proposed error mitigation techniques [5, 9–12, 25, 29, 30].
Concurrent Work: In our concurrent work [27] at *MICRO'19*, we demonstrate how correlated errors can degrade our ability to infer correct answer on NISQ machines. To improve inference strength, we propose *EDM: Ensembles of Diverse Mapping* that mitigate correlated errors by using different qubit mappings for different trials. This paper and EDM use similar philosophy that on a NISQ machine, while executing a program, if we repeat identical program for all trials then we can introduce correlation and bias in errors. This can limit our ability to infer correct answers on a NISQ machine.

Our solution of flipping a vulnerable state bears resemblance to the Data Bus Inversion [24]. Thus, there may be synergy in efficient reliability solutions that we apply in classical machines to detect and mitigate errors in near-term NISQ machines.

9 CONCLUSION

We focus on mitigating measurement errors, which tend to have the highest error-rate on current machines. We observe that there is state-dependent bias in measurement errors, with some states experiencing significantly higher error rates compared to the other states. Furthermore, the disparity between the measurement strength of basis states can significantly affect the reliability of NISQ applications, especially while measuring states with a high Hamming Weight. We propose to exploit this bias in measurement errors to improve the overall system reliability. For example, while measuring a state which is highly susceptible to measurement errors, we invert the state of the qubits and perform measurement in the inverted mode. To avoid the reliance on a-priori knowing the state being measured, we propose *Static Invert-and-Measure (SIM)*, which splits the trials into multiple groups and applied a different inversion string to each group. SIM obtains measurement errors close to the average and improves the application reliability by up to 2X.

If we could predict the state that is being measured, and the error rate profile of the machine, then we can proactively map the predicted state to the strongest state using a specifically designed inversion string. We use this insight to propose *Adaptive Invert-and-Measure (AIM)*. AIM estimates the measurement strength of the machine for each state. AIM also conducts a few "canary" trials to learn the likely outcomes for the given application and uses the inversion string that maps the predicted output to the strongest state before performing the measurement. Our evaluations, using three IBM machines (ibmqx2, ibmqx4, ibmqx14), shows that AIM improves the reliability of the system by up to 3X.

ACKNOWLEDGMENTS

We thank anonymous reviewers and Amruta Vidwans for editorial and technical feedback. This work was supported, in part, by a gift from Microsoft Research.

APPENDIX-A: CHARACTERIZING RBMS

Relative Basis Measurement Strength (RBMS) captures the measurement strength of each basis state on a quantum computer. It is crucial to learn RBMS, for detecting the measurement bias and and correct it using *Adaptive Invert and Measure (AIM)*. To evaluate the RBMS, we can prepare each basis state and measure the state multiple times to estimate the measurement strength function. However, the evaluation of absolute basis measurement strength is expensive because the number of basis states scales exponentially with the number of qubits. To address this, we develop two techniques:
Measuring Equal Superposition of N qubits: To avoid the cost of preparing every basis state, we prepare a quantum state with equal superposition of N-qubit state and measure this state repeatedly. By measuring the equal superposition of basis states, we can estimate the relative measurement strength of the basis states. By using this technique we achieve highly accurate RBMS within 5% mean squared error rate as show in Figure 4 and Figure 15. However, even with superposition, we do not completely address the exponential scaling problem. For example, for 30 qubits, there are billion states and we may still need the number of trials to scale exponentially with the number of qubits.

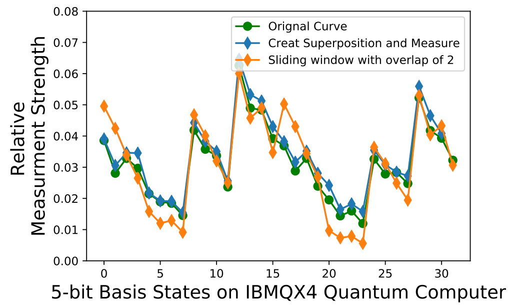

Figure 15: Validation of ESCT and AWCT on ibmqx4.

Sliding Window Characterization: To enable evaluation of RBMS with linearly scaling number of trials, we propose *Approximate Windowed Characterization Technique (AWCT)*. AWCT employs a *divide-and-conquer* approach by partitioning the N qubit in smaller groups with m qubits and perform characterization of "m" qubits at a time using uniform superposition. It uses sliding window with overlap of two qubits. When characterizing N qubit machine, AWCT characterizes fraction of m qubits at a time and moves to next set of m qubits such that there are two common qubits between consecutive superposition windows. With AWCT, the number of trials required to estimate the relative strengths of the basis states scale with $O(2^m)$ rather than $O(2^N)$. By picking small m we can estimate RBMS . Figure 15 show RBMS for ibmqx4 using direct measurement that measures all 32 basis states individually, a total superposition technique, and AWCT with sliding window of 2. AWCT shows a good match with exhaustive technique, providing a validation. For IBM-Q14, we use the AWCT with m=4 qubits (6 windows with each having 16 states, so only 96 states instead of 16 thousand states).

REFERENCES

[1] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. 1997. Strengths and weaknesses of quantum computing. *SIAM journal on Computing* 26, 5 (1997), 1510–1523.

[2] Kyle EC Booth, Minh Do, J Christopher Beck, Eleanor Rieffel, Davide Venturelli, and Jeremy Frank. 2018. Comparing and Integrating Constraint Programming and Temporal Planning for Quantum Circuit Compilation. *arXiv preprint arXiv:1803.06775* (2018).

[3] International Business Machines Corporation. 2017. Universal Quantum Computer Development at IBM:. http://research.ibm.com/ibm-q/research/. [Online; accessed 3-April-2017].

[4] Gavin E Crooks. 2018. Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv preprint arXiv:1811.08419* (2018).

[5] Suguru Endo, Simon C Benjamin, and Ying Li. 2018. Practical quantum error mitigation for near-future applications. *Physical Review X* 8, 3 (2018), 031027.

[6] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028* (2014).

[7] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Hartmut Neven. [n. d.]. Quantum Algorithms for Fixed Qubit Architectures. 2017. *arXiv preprint arXiv:1703.06199* ([n. d.]).

[8] Gian Giacomo Guerreschi and Jongsoo Park. 2018. Two-step approach to scheduling quantum circuits. *Quantum Science and Technology* (2018).

[9] Robin Harper and Steven Flammia. 2018. Fault tolerance in the IBM Q Experience. *arXiv preprint arXiv:1806.02359* (2018).

[10] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. 2019. Supervised learning with quantum-enhanced feature spaces. *Nature* 567, 7747 (2019), 209.

[11] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. 2017. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* 549, 7671 (2017), 242.

[12] 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.

[13] Gushu Li, Yufei Ding, and Yuan Xie. 2018. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. *arXiv preprint arXiv:1809.02573* (2018).

[14] Norbert M Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A Landsman, Kenneth Wright, and Christopher Monroe. 2017. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3305–3310.

[15] Dmitri Maslov, Sean M Falconer, and Michele Mosca. 2007. Quantum circuit placement: optimizing qubit-to-qubit interactions through mapping quantum circuits into a physical experiment. In *Proceedings of the 44th annual Design Automation Conference*. ACM, 962–965.

[16] Andrea Morello and David Reilly. 2018. What would you do with 1000 qubits? *Quantum Science and Technology* 3, 3 (2018), 030201.

[17] Prakash Murali, Jonathan M Baker, Ali Javadi Abhari, Frederic T Chong, and Margaret Martonosi. 2019. Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers. *arXiv preprint arXiv:1901.11054* (2019).

[18] Shin Nishio, Yulu Pan, Takahiko Satoh, Hideharu Amano, and Rodney Van Meter. 2019. Extracting Success from IBM's 20-Qubit Machines Using Error-Aware Compilation. *arXiv preprint arXiv:1903.10963* (2019).

[19] Alexandru Paler. 2019. On the Influence of Initial Qubit Placement During NISQ Circuit Compilation. In *International Workshop on Quantum Technology and Optimization Problems*. Springer, 207–217.

[20] Alexandru Paler, Alwin Zulehner, and Robert Wille. 2018. NISQ circuit compilers: search space structure and heuristics. *arXiv preprint arXiv:1806.07241* (2018).

[21] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *arXiv preprint arXiv:1801.00862* (2018).

[22] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. 2013. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In *Proceedings of the 50th Annual Design Automation Conference*. ACM, 41.

[23] Marcos Siraichi, Vinicius Fernandes Dos Santos, Sylvain Collange, and Fernando Magno Quintão Pereira. 2018. Qubit Allocation. In *CGO 2018-IEEE/ACM International Symposium on Code Generation and Optimization*. 1–12.

[24] Mircea R Stan and Wayne P Burleson. 1994. Limited-weight codes for low-power I/O. In *International Workshop on low power design*, Vol. 6. Citeseer, 6–8.

[25] Mingyu Sun and Michael R. Geller. 2019. Efficient characterization of correlated SPAM errors. arXiv:arXiv:1810.10523

[26] Swamit S Tannu and Moinuddin K Qureshi. 2018. A Case for Variability-Aware Policies for NISQ-Era Quantum Computers. *arXiv preprint arXiv:1805.10224* (2018).

[27] Swamit S Tannu and Moinuddin K Qureshi. 2019. Ensemble of Diverse Mappings: Improving Reliability of Quantum Computers by Orchestrating Dissimilar Mistakes. *The 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Oct 2019). https://doi.org/10.1145/3352460.3358257

[28] 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*. ACM, 987–999.

[29] Kristan Temme, Sergey Bravyi, and Jay M Gambetta. 2017. Error mitigation for short-depth quantum circuits. *Physical review letters* 119, 18 (2017), 180509.

[30] Takahiro Tsunoda, Andrew Patterson, Xiao Yuan, Suguru Endo, Joseph Rahamim, Peter Spring, Martina Esposito, Salha Jebari, Kitti Ratter, Sophia Sosnina, et al. 2019. Implementing the Variational Quantum Eigensolver with native 2-qubit interaction and error mitigation. *Bulletin of the American Physical Society* (2019).

[31] Davide Venturelli, Minh Do, Eleanor Rieffel, and Jeremy Frank. 2018. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology* 3, 2 (2018), 025004.

[32] Jonathan Ward, Johannes Otterbach, Gavin Crooks, Nicholas Rubin, and Marcus da Silva. 2018. QAOA Performance Benchmarks using Max-Cut. In *APS Meeting Abstracts*.

[33] Alwin Zulehner, Alexandru Paler, and Robert Wille. 2017. Efficient Mapping of Quantum Circuits to the IBM QX Architectures. *arXiv preprint arXiv:1712.04722* (2017).