JigSaw: Boosting Fidelity of NISQ Programs via Measurement Subsetting

Poulami Das
poulami@gatech.edu
Georgia Institute of Technology
Atlanta, USA

Swamit Tannu
stannu@wisc.edu
University of Wisconsin
Madison, USA

Moinuddin Qureshi
moin@gatech.edu
Georgia Institute of Technology
Atlanta, USA

ABSTRACT

Near-term quantum computers contain noisy devices, which makes it difficult to infer the correct answer even if a program is run for thousands of trials. On current machines, qubit measurements tend to be the most error-prone operations (with an average error-rate of 4%) and often limit the size of quantum programs that can be run reliably on these systems. As quantum programs create and manipulate correlated states, all the program qubits are measured in each trial and thus, the severity of measurement errors increases with the program size. The fidelity of quantum programs can be improved by reducing the number of measurement operations.

We present *JigSaw*, a framework that reduces the impact of measurement errors by running a program in two modes. First, running the entire program and measuring all the qubits for half of the trials to produce a global (albeit noisy) histogram. Second, running additional copies of the program and measuring only a subset of qubits in each copy, for the remaining trials, to produce localized (higher fidelity) histograms over the measured qubits. JigSaw then employs a Bayesian post-processing step, whereby the histograms produced by the subset measurements are used to update the global histogram. Our evaluations using three different IBM quantum computers with 27 and 65 qubits show that JigSaw improves the success rate on average by 3.6x and up-to 8.4x. Our analysis shows that the storage and time complexity of JigSaw scales linearly with the number of qubits and trials, making JigSaw applicable to programs with hundreds of qubits.

CCS CONCEPTS

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

KEYWORDS

Quantum Computing, Error Mitigation, NISQ Computing

ACM Reference Format:
Poulami Das, Swamit Tannu, and Moinuddin Qureshi. 2021. JigSaw: Boosting Fidelity of NISQ Programs via Measurement Subsetting. In *MICRO'21: 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '21), October 18–22, 2021, Virtual Event, Greece.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3466752.3480044

1 INTRODUCTION

Quantum computers can solve very hard problems by using properties of quantum bits (qubits) [29, 46]. Recently demonstrated quantum hardware with fifty-plus qubits are getting to the regime where they can outperform the most advanced supercomputer for some computations [52]. Unfortunately, these machines are not sufficiently large to implement quantum error correction and are operated in the *Noisy Intermediate Scale Quantum (NISQ)* [41] model, whereby the computation is run a large number of times (called trials) and the answer is inferred from the output log. The ability to obtain the correct answer on a NISQ machine depends on the error-rates and the size of the program. Recently, various software techniques have been investigated to improve application fidelity that either perform noise-aware computations to enable better than worst-case error rate [30, 33, 48, 50, 51] or reduce the program length and number of computations [27, 45, 57].

This paper focuses on measurement operations, which are often the most dominant source of error on current superconducting quantum computers, with average error-rates ranging between 2-7% [20, 52]. Measurement errors constrain the size of the largest program that can be run on NISQ machines with high fidelity [13]. These errors arise from the imperfections in the qubit readout protocol [7, 53] and the long latency of these operations (about 4-5 microseconds on IBM quantum systems). Furthermore, measurement operations suffer from *measurement crosstalk* [4, 23], which means performing several measurement operations concurrently increases the error rate of each measurement operation. Our experiments on IBMQ machines show that the average measurement error rate increases by up-to 2% when five qubits (and by up-to 4% when ten qubits) are measured simultaneously compared to measuring a single qubit in isolation. This indicates larger programs become even more susceptible to measurement crosstalk due to higher number of measurement operations. Similar observations are reported for the Google Sycamore, where simultaneous measurement operations have 1.26x higher error-rates compared to isolated measurements [52]. Consequently, fast and accurate qubit measurement at scale remain an open problem [24].

Spatial variation of measurement errors imposes further challenges in attaining high fidelity for large programs. For example, on IBM's 27-qubit Toronto device, the median measurement error rate is about 2.7%, whereas the worst-case error rate is 22.2%. Existing state-of-the-art noise-aware compilers avoid mapping programs onto unreliable physical qubits [30, 51], to alleviate the impact of worst-case errors. However, it is not always feasible for large programs, particularly in the context of measurement errors because the qubits with the lowest measurement error rates are typically not co-located in space. For example, it is challenging to map any

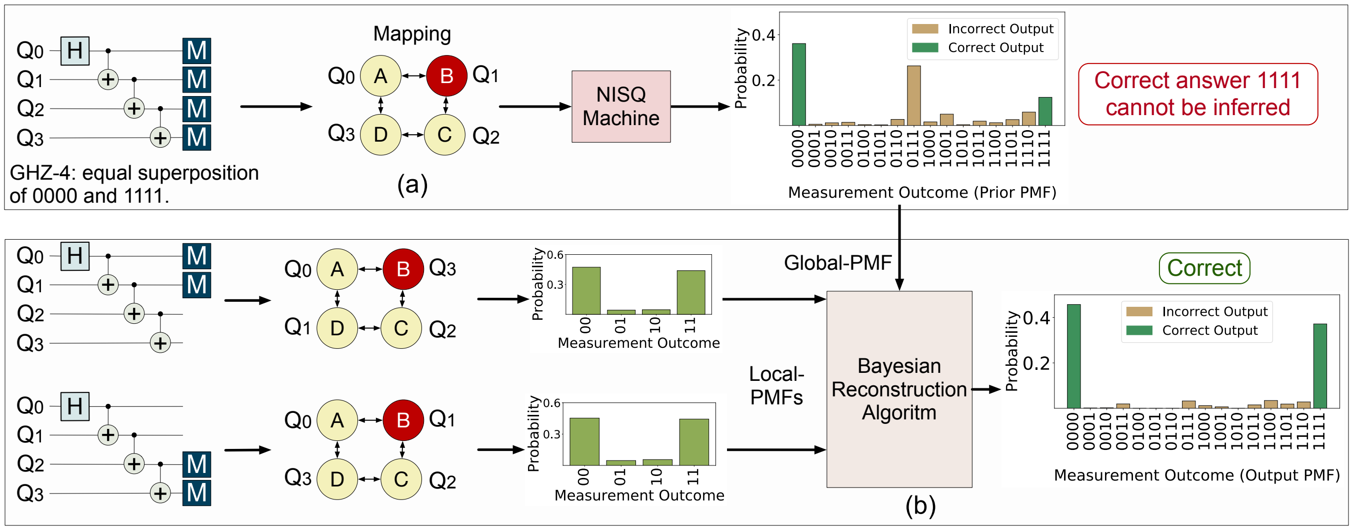
Q0
Q1
Q2
Q3
H
M
GHZ-4: equal superposition of 0000 and 1111.
Mapping
A
B
C
D
(a)
NISQ Machine
Probability
Incorrect Output
Correct Output
Measurement Outcome (Prior PMF)
Correct answer 1111 cannot be inferred
Measurement Outcome
Global-PMF
Local-PMFs
Bayesian Reconstruction Algoritm
Correct
Measurement Outcome (Output PMF)
(b)

Figure 1: (a) Executing GHZ-4 kernel on NISQ machine (b) JigSaw obtains a global-PMF for the program by measuring all the qubits, a high-fidelity local-PMF for each circuit with partial measurement (CPM), and uses the local-PMFs from CPMs to update the global-PMF via a Bayesian Reconstruction algorithm.

program with more than six qubits on IBMQ-Toronto without using a physical qubit with more than 2.7% measurement error-rate (the median). In the limiting case, the compiler is forced to use qubits with more than 20% measurement error-rate for any program that uses sixteen qubits or more.

NISQ programs measure all the program qubits in each trial, and thus, suffer from the accumulated measurement errors across all the program qubits. The severity of measurement errors is compounded in large programs by the fact that a correct output is obtained only if all these measurements are error-free. The key insight in our paper is to measure only a subset of program qubits for some of the trials, so that these trials encounter fewer errors due to reduced measurement crosstalk, and by avoiding the physical qubits most susceptible to measurement errors. However, subset measurements lack information about the global correlation across all the qubits. Thus, we need a way to combine the benefits of both global runs (full correlation but low fidelity) and subset runs (limited correlation over fewer qubits but higher fidelity). To that end, we propose JigSaw,[1] a framework that reduces measurement errors by running a program in two modes: global and subset. The PMF (probability mass function) produced for each mode are then combined using a post-processing algorithm to generate a higher fidelity PMF.

To illustrate our design, we use the GHZ-4 program shown in Figure 1(a) as an example. Ideally, GHZ-4 produces 0000 or 1111 with 50% probability each. However, errors can produce incorrect outputs, some with probability higher than the two correct answers. To improve the application fidelity using JigSaw, first, we run the program in *global-mode*, wherein we measure all the qubits to produce the global-PMF. In the second step, we run the circuit in *subset-mode*, where multiple copies of the original program are run, but where each copy only measures a subset of qubits and produces a PMF only over the measured qubits. For example, in Figure 1(b), we run two such *Circuits with Partial Measurements (CPM)* that perform all the computations but only measure two qubits in each copy (Q_0 and Q_1 for the first copy, and Q_2 and Q_3 for the second copy). Each CPM is recompiled to ensure that the two measurement operations are performed on the physical qubits with the least amount of measurement error. Therefore, we would expect the measurement fidelity which is closer to the best-case qubits rather than average-case or worst-case qubits. Measurement subsetting and recompilation results in the higher reliability of CPM marginals compared to the scenario of deriving the marginals from the global-PMF.

Although fewer measurements can offer marginal information with higher fidelity, the CPM may still not be able to produce the output PMF as the correlation between different CPM are unknown. To reconstruct the global-PMF using the local-PMFs, we employ a *Bayesian Reconstruction* algorithm, analogous to Bayesian updating in statistics, where a prior probability estimate is updated using newer information. For JigSaw, the global-PMF is the prior estimate, which is updated using more accurate marginal information from the CPM. By performing Bayesian updates, the algorithm accentuates the probabilities of the correct outcome(s) while reducing the probabilities of the incorrect outcomes, thus improving the fidelity. Our evaluations with three different quantum hardware from IBM, with 27 and 65 qubits, show that JigSaw without recompilation (and measurement subsetting only) improves the success rate of typical quantum benchmarks on average by 1.85x and by up-to 3.26x. Note that by default JigSaw uses CPM of size 2. Alternately, JigSaw with both measurement subsetting and CPM recompilation improves the fidelity of applications on average by 2.9x and by up-to 7.9x.

For an N-qubit program, it is possible to design a polynomial number of CPM of subset size 2 ($^{N}C_2$). However, our experiments show that the effectiveness of JigSaw saturates even when more CPM of the same subset size are used because after a certain limit these additional CPM do not offer incremental and unique information. To overcome this limitation, we propose *Multi-Layer JigSaw*

[1]The name is inspired by the popular Jigsaw puzzle – the global-PMF represents a noisy skeleton, and the subset measurements are small tiles. When both are used together, the full picture is revealed.

(JigSaw-M), which creates CPM of different sizes by exploiting the trade-off between fidelity and the amount of correlation for a CPM. While small CPM provide higher fidelity; they do not capture sufficient correlation. Alternatively, larger CPM provide greater correlation but also encounters more errors due to larger number of measurements. Overall, our default JigSaw-M uses CPM of subset sizes 2 to 5 qubits and improves application success rate on average by 3.65x and by up-to 8.42x.

Overall, our paper makes the following contributions:

(1) We propose *JigSaw*, a framework that does not subject all trials to measurement errors across all qubits. It performs half of the trials with global measurements (for correlation) and the other half with subset measurements (for higher fidelity).
(2) We propose a *Bayesian Reconstruction* algorithm that uses the PMFs from the subsets to improve the global-PMF.
(3) We propose *JigSaw-M*, which generates more unique non-uniform sized CPM to further enhance the global-PMF.

Our scalability analysis of JigSaw show that the storage and time complexity is linear with the number of trials and qubits, making JigSaw applicable to programs with hundreds of qubits. The algorithm for JigSaw and datasets used for evaluations in this paper is available at this link.

2 BACKGROUND

2.1 Basics of Quantum Computing

A qubit is the fundamental unit of information on a quantum computer and may be represented using a vector $|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, a superposition of the basis states $|0\rangle$ and $|1\rangle$ such that $|\ \alpha\ |^2 + |\ \beta\ |^2 = 1$. When measured, a qubit collapses to a binary 0 or 1 with probabilities $|\ \alpha\ |^2$ and $|\ \beta\ |^2$ respectively. Similarly, an n-qubit system exists in a superposition of 2^n basis states and can produce any of the 2^n bitstrings depending on the probabilities associated with them upon measurement.

2.2 Errors and NISQ Model of Computing

Qubits are extremely sensitive devices and error prone. These errors can corrupt their states, producing incorrect outcomes during program execution. The state of a qubit naturally decays due to its interactions with the environment, a phenomenon called decoherence, whereas imperfections in quantum operations can lead to gate errors. *Measurement* errors manifest due to errors in measurement operations. Noisy Intermediate Scale Quantum (NISQ) computers will be operated in the presence of noise as they may not be large enough to achieve fault-tolerance [41]. By repeatedly executing a program several times on the NISQ hardware (called *trials*), the program solution can be inferred from the output distribution.

2.3 Measurement Errors

Qubit measurement error rates can constrain the size of the largest program (in terms of number of qubits) that can be run reliably on a NISQ machine [13]. As a result, software policies particularly aimed at reducing the impact of measurement errors are currently being developed [5, 8, 19, 26, 50].

How are measurements performed? Superconducting qubits, similar to the devices from IBM and Google, are measured using a *dispersive qubit readout protocol.* In this protocol, a qubit is coupled to a measurement resonator whose resonance frequency experiences a shift depending on the state of the qubit and by measuring this shift, the state of the qubit is determined [7, 24, 53]. For superconducting devices, a signal corresponding to the state of a qubit is obtained when a readout pulse is applied to the qubit [52]. This signal is translated into a single-valued complex number to classify the state of the qubit as "0" or "1" using a measurement *discriminator.*

Why are measurement operations error prone? Measurement errors can be attributed to a variety of factors.

(1) The shift in resonance frequency during readout is very sensitive to noise, is device-specific, and drifts in time.
(2) The measurement set-up involves various complex instruments operating across multiple thermal domains that introduces errors due to *crosstalk* and unwanted couplings. The impact of crosstalk is generally not fully understood and considered to be hard to minimize at device level [23, 52].
(3) Existing discriminators are inefficient and perform poorly for several quantum states [37].
(4) Measurement operations are slow (typically takes about 4-5 microseconds on recent IBMQ hardware and about 800 nanoseconds on Google devices [3]) and often cause qubits to decay to the ground state during the readout process.

These factors limit the ability of existing quantum systems to perform fast and accurate qubit measurements at scale. On existing IBMQ and Google hardware, measurement operations tend to be the dominant sources of errors, with median error rates between 2.76% to 7.1%, and worst-case error rates as high as 11.7% to 22.2%.

3 PROBLEM AND MOTIVATION

Measurement operations limit the fidelity of large programs mainly due to two reasons:

(1) The impact of *measurement crosstalk* increases with the number of measurement operations.
(2) *Spatial variation in measurement errors* limit the ability of compilers to avoid the most error-prone physical qubits.

3.1 Impact of Measurement Crosstalk

Measurement operations are more vulnerable to errors when a larger number of qubits are simultaneously measured due to measurement crosstalk. For example, the average error rate of simultaneous measurements is 1.26x higher than isolated measurements on Google Sycamore, as shown in Table 1.

Table 1: Measurement Errors on Google Sycamore [4]

Measurement Mode	Measurement Error Rates (%)			
	Min	Average	Median	Max
Isolated	2.60	6.14	5.70	11.7
Simultaneous	3.30	7.73	7.10	20.9

Characterization of Measurement Crosstalk:
We perform several characterization experiments and observe similar behavior on IBMQ hardware. To study measurement crosstalk, we perform several characterization experiments on IBMQ hardware. Figure 2(a) shows an N-qubit circuit that creates arbitrary quantum states using single-qubit U3 gates [2]. During the experiment, the Probe-Qubit (Q_1) is always mapped to the physical qubit on which the impact of measurement crosstalk is being determined, whereas the other N-1 qubits are randomly mapped to the remaining physical qubits of the machine in each sample and generate multiple samples for each N. Note that N denotes the number of measurements and N=1 corresponds to the case when the Probe-Qubit is measured in isolation. For the results shown in Figure 2(b), we vary N from 1 to 10 and take 10 samples for each N.

We compute the mean fidelity of the Probe-Qubit for each N by measuring the *Total Variation Distance (TVD)* between the experimental output and the output from a noise-free quantum computer. Figure 2(b) shows the impact of increasing the number of measurements from 1 to 10 for four different quantum states, prepared by specifying the Euler angles for the U3 gates, when Qubit-6 is probed on 27-qubit IBMQ-Paris. We observe that simultaneously measuring a larger number of qubits can reduce the fidelity of these operations significantly. We make similar observations on other qubits and hardware devices. Further experiments also show that the impact of such crosstalk depends on the quantum state and physical qubit and therefore, is hard to characterize. Prior studies state that it is hard to fully understand and minimize measurement crosstalk at device level [4, 23].

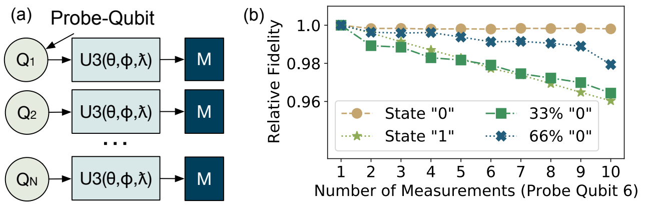

Figure 2: (a) Measurement crosstalk characterization circuits (b) Impact of increasing number of measurements on the Fidelity of the Probe-Qubit.

3.2 Impact of Spatial Variation

To execute a program on a NISQ device, a compiler maps the program qubits onto the physical qubits of the device and translates the high-level program into low-level machine specific instructions. Compilers also need to insert SWAP instructions to overcome the limited connectivity of NISQ devices. Noise-aware compilers [30, 51] account for the error characteristics of the underlying quantum hardware and avoid mapping program qubits on to hardware qubits with worst-case errors. However, while this works very efficiently for small programs, spatial variation in measurement error rates often force compilers to map program qubits on unreliable physical qubits, as programs grow in size, because the qubits with the lowest error rates are not spatial neighbors, as shown in Figure 3. For example, it is not possible to map any program with more than six qubits on the 27-qubit IBMQ-Toronto without using a physical qubit with more than 2.7% measurement error-rate (the median error-rate). Further, the compiler is forced to use physical qubits A and B with more than 20% measurement error rate for programs with sixteen and twenty-one qubits or more respectively.

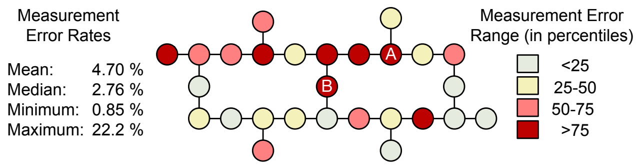

Figure 3: Spatial variation in measurement error rates of qubits on IBMQ-Toronto

3.3 The Insight: Measurement Subsetting

NISQ applications measure all the program qubits in each trial, subjecting all the trials to the accumulated measurement errors from all the qubits. We can reduce the impact of measurement errors on the fidelity of NISQ programs by:

(1) *Reducing the number of measurement operations by performing some trials with measurements only on a subset of qubits and effectively lowering the impact of crosstalk.*
(2) *Remapping to ensure that the subset measurements are performed on the qubits with the lowest error rates. This allows the us to get an effective measurement error rate that is closer to the minimum rather than the average.*

Correlation versus Fidelity Trade-off in Subsetting:
Quantum computers obtain their exponential power by creating and manipulating highly correlated states. To measure this correlated state, a NISQ program measures all the qubits in each trial. If there was no correlation, and each qubit had an independent probability of being in the 0 or 1 state, then one could simply split the trials into N groups (one group for each qubit), obtain the independent probability of being in 0 or 1 state for each qubit, and then obtain the probability distribution over all the qubits through multiplication. While this approach has high fidelity for each trial (only one measurement), it captures zero correlation between the qubits. Measuring a subset of qubits (larger than a single qubit but not all qubits) captures some correlation (within the measured qubits) but the correlations between these marginal distributions remain unknown, and therefore, multiplication or a tensor product may not yield the correct output distribution.

3.4 The Goal: Reducing Measurement Errors

Thus, measuring all the qubits provides full correlation (but low fidelity) and fewer measurements provide higher fidelity (but weaker correlation). Ideally, we want both full correlation and high fidelity. The goal of this paper is to design scalable and effective policies that can improve application fidelity by retaining the global correlation of the original program, while simultaneously benefiting from the higher fidelity obtained from fewer measurements. Next, we discuss our proposed design, JigSaw.

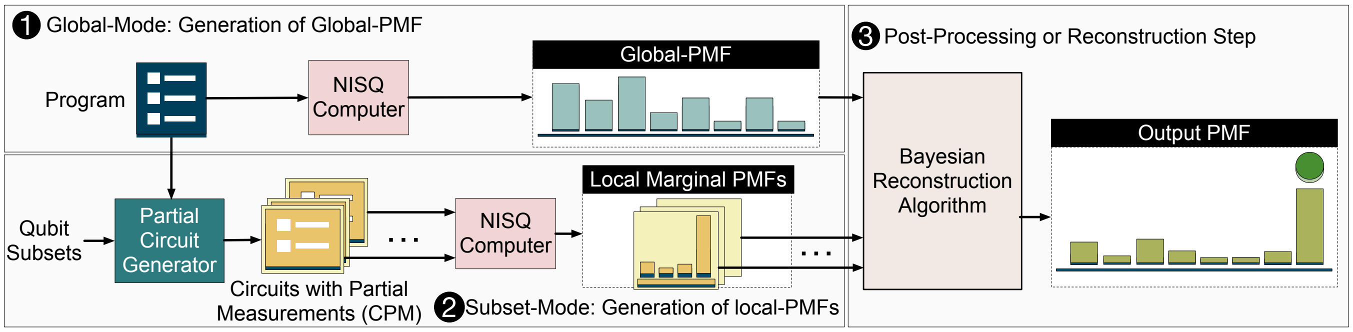
1 Global-Mode: Generation of Global-PMF
Program
NISQ Computer
Global-PMF
3 Post-Processing or Reconstruction Step
Bayesian Reconstruction Algorithm
Output PMF
Qubit Subsets
Partial Circuit Generator
Circuits with Partial Measurements (CPM)
NISQ Computer
Local Marginal PMFs
2 Subset-Mode: Generation of local-PMFs

Figure 4: Overview of JigSaw. JigSaw runs the program in two modes: global-mode (all program qubits are measured) and subset-mode (only a subset of qubits are measured), and then uses the local-PMFs generated in subset-mode to update the global-PMF obtained during the execution in global-mode.

4 JIGSAW: OVERVIEW AND DESIGN

We propose *JigSaw*, a framework that relieves the NISQ program from requiring to measure all the qubits of a program in each trial. Figure 4 shows the overview of JigSaw. JigSaw executes a program in two modes. First, JigSaw executes half of the trials in the *global-mode* in which a program is executed in its entirety and all the qubits are measured to obtain the global probability mass function (PMF).[2] Second, for the remaining trials, JigSaw runs additional copies of the program or *Circuits with Partial Measurements (CPM)* that measure fewer qubits in the *subset-mode*, to obtain more accurate marginal or local-PMFs over the *subset* of qubits being measured. However, CPM alone cannot be used to infer the output PMF of a program without information about the correlations between these local-PMFs. To address this challenge, JigSaw employs a post-processing or *reconstruction* step that updates the global-PMF using the local-PMFs. This enables JigSaw to improve the application fidelity while simultaneously retaining the global correlation without requiring any additional trials.

4.1 Global-Mode: Generation of Global-PMF

In this mode, JigSaw executes the entire program and measures all the program qubits to produce a global-PMF. This is identical to the baseline policy and is done for half of the trials. We use Noise-Aware SABRE [27] for compilation to obtain a global-PMF with high fidelity. A NISQ compiler maps the logical qubits of a program on to the physical qubits of the hardware and generates a schedule by translating the high-level instructions into low-level machine specific operations. To overcome limited connectivity on NISQ hardware, compilers also insert SWAP instructions to bring two non-adjacent qubits next to each other, so that CNOT operations can be performed between them. Noise-Aware SABRE accounts for the hardware error characteristics and generates a schedule that maximizes the *Expected Probability of Success* (EPS) [34]. EPS is the expected probability of successfully executing each gate and measurement operation in a schedule and is computed at compilation time by using the error rates obtained from the daily calibration report of the NISQ hardware.

[2]We use Probability Mass Function (PMF) for the results of program execution because these include discrete and not continuous values.

4.2 Subset-Mode: Generation of Local-PMFs

In the subset-mode, JigSaw runs several Circuits with Partial Measurements (CPM), for the remaining half of the trials which are equally distributed between the CPM. We discuss how CPM are generated and optimized for greater fidelity.

4.2.1 Circuits with Partial Measurements.

A CPM is identical to the original program, except that it measures only a subset of qubits. CPM produces high-fidelity local-PMFs over the qubits measured. For example, Figure 5(a) shows a CPM of a BV-4 program that measures 2 qubits, Q_0 and Q_1. The key parameter in JigSaw is the number of qubits measured in a CPM and is called the *subset size.* Our default design uses CPM that measure 2 qubits. This is the smallest possible subset size that captures some correlation while performing fewest possible measurements. Measuring only one qubit in a CPM captures zero correlation and therefore, not used. By default, we use a sliding window method to generate the CPM so that we get N unique CPM for an N-qubit program. For example, for a 4-qubit program with qubits q_0, q_1, q_2, q_3, we generate 4 CPM, measuring $(q_0, q_1), (q_1, q_2), (q_2, q_3)$, and (q_0, q_3). Therefore, the number of CPM is same as the number of qubits.

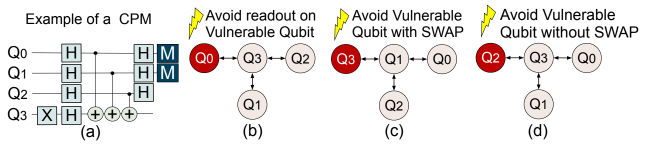
Example of a CPM
Q0
Q1
Q2
Q3
H
X
M
(a)
Avoid readout on Vulnerable Qubit
Q0
Q3
Q2
Q1
(b)
Avoid Vulnerable Qubit with SWAP
Q3
Q1
Q0
Q2
(c)
Avoid Vulnerable Qubit without SWAP
Q2
Q3
Q0
Q1
(d)

Figure 5: (a) Example of a CPM (b) Compiler avoids vulnerable qubit. Mapping that avoids vulnerable qubit (c) with extra SWAP (d) without incurring extra SWAP

4.2.2 Optimizations to Improve Fidelity of the CPM.

As JigSaw heavily relies on the accuracy of the local-PMFs, the compiler recompiles each CPM to maximize their fidelity.

Optimizing for Measurement Errors: We recompile each CPM to exploit variability in measurement errors and ensure that measurements of desired program qubits are performed on the strongest physical qubits. The compiler avoids measurements on the physical qubit(s) with the highest readout error rate, referred to as *vulnerable*

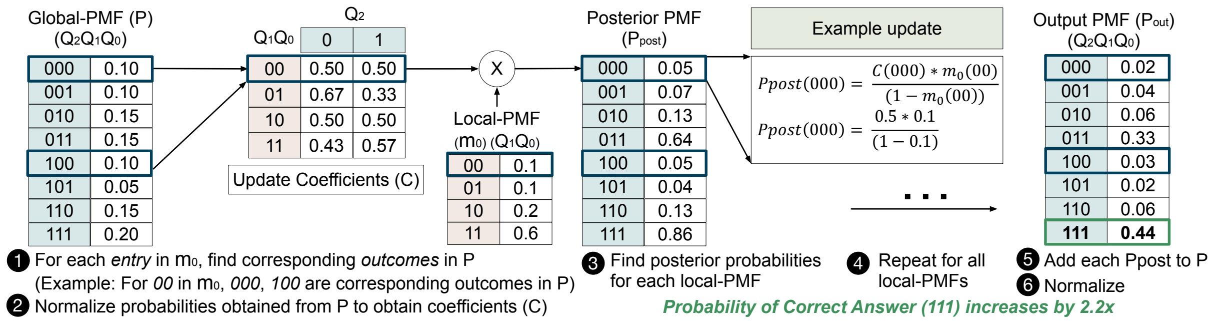
Global-PMF (P) (Q2Q1Q0)
000 0.10
001 0.10
010 0.15
011 0.15
100 0.10
101 0.05
110 0.15
111 0.20
Q2
Q1Q0 0 1
00 0.50 0.50
01 0.67 0.33
10 0.50 0.50
11 0.43 0.57
Update Coefficients (C)
Local-PMF (m0) (Q1Q0)
00 0.1
01 0.1
10 0.2
11 0.6
Posterior PMF (Ppost)
000 0.05
001 0.07
010 0.13
011 0.64
100 0.05
101 0.04
110 0.13
111 0.86
Example update
Ppost(000) = C(000) * m0(00) / (1 − m0(00))
Ppost(000) = 0.5 * 0.1 / (1 − 0.1)
Output PMF (Pout) (Q2Q1Q0)
000 0.02
001 0.04
010 0.06
011 0.33
100 0.03
101 0.02
110 0.06
111 0.44
1 For each entry in m0, find corresponding outcomes in P (Example: For 00 in m0, 000, 100 are corresponding outcomes in P)
2 Normalize probabilities obtained from P to obtain coefficients (C)
3 Find posterior probabilities for each local-PMF
4 Repeat for all local-PMFs
5 Add each Ppost to P
6 Normalize
Probability of Correct Answer (111) increases by 2.2x

Figure 6: Steps in Bayesian Reconstruction for a 3-qubit program (qubits Q_2, Q_1, Q_0) using a CPM measuring Q_1, Q_0.

qubit(s), in an allocation. For example, the compiler eliminates the allocation in Figure 5(b) for the CPM in Figure 5(a) to avoid readout of Q_0 on the vulnerable qubit and selects an alternate allocation.

Avoiding Extra SWAPs: To avoid measurement on the vulnerable qubit(s), the compiler often uses alternate qubit allocations, some of which may need insertion of extra SWAP instructions. However, we avoid such allocations that require extra SWAPs to avoid additional gate errors. For example, the compiler selects the allocation shown in Figure 5(d) over Figure 5(c) because the latter requires an extra SWAP (SWAP Q_0, Q_1). While in most cases, the compiler finds alternate mappings without incurring extra SWAPs, for cases where the compiler cannot not find a mapping without inserting an extra SWAP, it picks the mapping that maximizes the EPS.

The ability to find alternative mappings depends on device connectivity, spatial location of good qubits, and program characteristics. Our studies show most CPM can reuse qubit allocations chosen by the Ensemble of Diverse Mappings [48] policy. Also, we use SABRE, which has low latency.

4.3 Post-processing via Bayesian Method

JigSaw produces (N+1) PMFs for a program with N qubits: one global-PMF and one local-PMF for each of the N CPMs. The post-processing step aims to combine the higher fidelity local-PMFs from each CPM into the global-PMF. To this end, we propose *Bayesian Reconstruction* algorithm, which is inspired by Bayesian updating in statistics, whereby a prior probability estimate is updated using additional information [22]. For JigSaw, the global-PMF (P) offers the prior probability estimate, whereas the set of marginals (M) obtained from the CPM provides the additional information. We use the term *marginal* to denote a set comprising of a subset of qubits measured in a CPM and the local-PMF produced by the CPM.

Consider an example of a 3-qubit program. Let $P = \{000 : a, 010 : b, 010 : c, 011 : d, 100 : e, 101 : f, 110 : g, 111 : h\}$ represent a generic global-PMF of a 3-qubit program (with qubits Q_2, Q_1, Q_0), where a to h are the probabilities of observing outcomes 000 to 111 respectively. Similarly, $m_0 = [\{00 : \alpha, 01 : \beta, 10 : \gamma, 11 : \delta\}, [1, 0]]$ refers to a generic marginal for a CPM measuring qubits $[Q_1, Q_0]$.

The steps for the Bayesian Reconstruction algorithm are described in Algorithm 1 (Appendix). The algorithm uses each marginal to update the probabilities of each outcome in the global-PMF (P). The algorithm starts by searching for all outcomes in P associated with each outcome in a marginal, by evaluating the bits at the corresponding qubit positions. For example, for $(Q_1, Q_0) = 00$ in the marginal m_0, the candidates in P are $(Q_2, Q_1, Q_0) = 000$ and 100. For each marginal in M, the Bayesian Update function generates an updated PMF (P_{post}) by updating the probabilities of each outcome in P using the associated probabilities in the marginals. For example, the probabilities of 000 and 100, i.e., a and d respectively, are updated in proportion with α (corresponding to 00 in m_0). The algorithm produces a posterior output PMF (P_{out}) by adding all the intermediate PMFs (P_{post}) to the global-PMF (P). The algorithm is recursively called and terminates when the Hellinger Distance [17] between the output PMF, P_{out}, prior to and post the function call does not change, implying the two PMFs are similar.

We explain the steps involved in the Bayesian update using a quantitative example. Figure 6 shows the update sequence using experimental data for a 3-qubit program whose global-PMF is denoted by P (in order Q_2, Q_1, Q_0) using a marginal m_0 from a CPM that measures (Q_1, Q_0).

- **Step 1** : The algorithm searches for all candidate outcomes in P for each entry in marginal m_0 by evaluating the bits at the corresponding qubit positions. For example, 000 and 100 are the candidate outcomes in P for 00 in m_0, obtained by matching the values for (Q_1, Q_0) in P and m_0.
- **Step 2** : Next, the function computes the Update Coefficients C for each of the outcomes in P by normalizing their respective probabilities of occurrence in P.
- **Step 3** : Next, the algorithm computes the posterior probabilities for each observed outcome in P by scaling them using the corresponding probabilities observed in m_0. Figure 6 explicitly shows the computation for obtaining the posterior probability of outcome 000 ($P_{post}[000]$) using the Update Coefficient C[000] and marginal information $m_0[00]$.
- **Step 4** : The algorithm repeats Steps 1-3 to generate an intermediate posterior PMF (P_{post}) for each marginal.
- **Step 5** : Each P_{post} is added to the global-PMF (P).
- **Step 6** : The final output PMF (P_{out}) is obtained by normalizing the probabilities.

For readability, Figure 6 only shows the steps for marginal $[Q_1, Q_0]$, whereas the output PMF shown is obtained from recursive updates with additional marginals. Note that as the Bayesian updates for each CPM are performed independently and the intermediate PMFs are added in the final step, the order of updates does not matter.

4.4 Multi-Layer JigSaw (JigSaw-M)

Our studies show that the performance of JigSaw saturates when additional CPM of the same subset size that do not offer any incremental information are used. However, we can design more unique CPM by using different subset sizes and improve the application fidelity even further.

4.4.1 *Global and Subset-Modes for JigSaw-M.*

The global mode for JigSaw-M is identical to JigSaw that generates the global-PMF by executing the program and measuring all the qubits. The subset-mode for JigSaw-M executes CPM of non-uniform subset sizes s, such that $s_{min} \leq s \leq s_{max}$, where s_{max} and s_{min} are the maximum and minimum subset sizes respectively. We use a sliding window method to generate unique CPM for each subset size, similar to JigSaw, but other methods can be used too. If CPM of $\mathbb{S}$ different sizes are used, JigSaw-M produces ($\mathbb{S}$N+1) PMFs for an N-qubit program, one global-PMF and N local-PMFs for each subset size. By default, our design uses CPM of sizes 2 to 5.

4.4.2 *Adapting Reconstruction for JigSaw-M.*

JigSaw-M comprises of CPM of $\mathbb{S}$ different sizes. There is a choice between which CPM must be used first to update the global-PMF. However, note that there exists a *trade-off* between the fidelity of a CPM and the correlation it can capture depending on the number of qubits measured in the CPM. A smaller CPM offers higher fidelity due to fewer measurement errors but captures limited correlation. Alternately, a larger CPM offers higher correlation, but has lower fidelity since it is more prone to measurement errors. Thus, for JigSaw-M, the reconstruction algorithm first updates the global-PMF using the CPM of the highest size (s_{max}), limiting the loss of global correlation. The updated PMF (P_s) is then further enhanced using information from CPM of the next higher size. The process is repeated until the smallest CPM are used. This top-down ordering maximally preserves the global correlation, while simultaneously improving the fidelity.

We discuss the evaluation methodology before discussing the impact of JigSaw on the fidelity of NISQ applications.

5 EVALUATION METHODOLOGY

5.1 Quantum Hardware Platforms

For all our evaluations, we use three different quantum computers from IBM: 27-qubit *IBMQ-Toronto*, 27-qubit *IBMQ-Paris*, and 65-qubit *IBMQ-Manhattan*.

5.2 Baseline Compiler

For the baseline, we use Noise-Aware SABRE [27] to compile and map the program onto the physical qubits with the lowest error rates. We also evaluate JigSaw against an Ensemble of Diverse Mappings (EDM) policy that runs independent copies of the program on different groups of physical qubits [48] and improves the ability to infer the correct answer. Note that while we use Noise-Aware SABRE, other noise-adaptive compilers [30, 51] may be used too.

5.3 Benchmarks

We use the benchmarks described in Table 2. The type and size of benchmarks are derived from prior works [27, 30, 48, 51]. Note that the IBMQ machines used for evaluations have a Quantum Volume [10] of 32, which means that square circuits of only up to size 5 can be run reliably and therefore, the size of the benchmarks used is already much larger even if they do not use all the qubits that are present on the machine.

Table 2: Details of NISQ benchmarks

Name	Algorithm	#Qubits (n)	1Q Gates	2Q Gates
BV-n	Bernstein-Vazirani [6]	6	2(n+1)	n
Graycode-n	Graycode Decoder	18	n/2	(n-1)
QAOA-n (p=1)	Maxcut with p=1 [12]	8	4n	(n-1)
QAOA-n (p=2)	Maxcut with p=2	10, 14	6n	2(n-1)
QAOA-n (p=4)	Maxcut with p=4	10, 12	10n	4(n-1)
GHZ-n	Greenberger-Horne -Zeilinger [16]	14	1	(n-1)
Ising-n	Ising model [21]	10	n(4.5n-2)	n(n-1)

5.4 Experimental Setup: Number of Trials

We use between 32K to 256K trials for the baseline depending on program size. This represents the highest fidelity that can be obtained by increasing trials alone and serves as a strong baseline as more trials do not improve fidelity (mainly due to correlated errors). Figure 7 shows the Probability of Successful Trial (PST) of several GHZ and QAOA benchmarks executed on IBMQ-Paris for up to 4 million trials. We observe similar behavior for other workloads and machines.

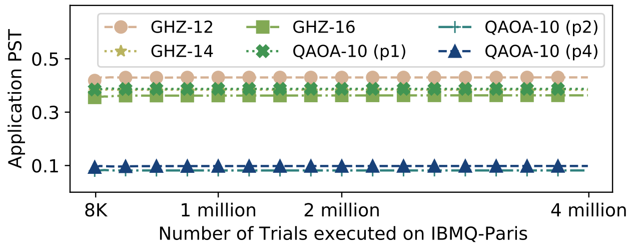

Figure 7: Impact of Number of Trials on Probability of Successful Trial (PST) of Applications.

For EDM [48], we use an ensemble of four mappings and the trials are equally divided among the mappings. For JigSaw and JigSaw-M, the trials are equally split between the global-mode and the subset-mode. In the subset-mode the trials are equally split between all the CPM for both JigSaw and JigSaw-M. Therefore, JigSaw, JigSaw-M, and EDM all use the **same number of trials** as the baseline. We run all the experiments within the same calibration cycle but observe similar results across different calibration cycles. We use equal split for simplicity because the fidelity saturates for the number of trials used in our evaluation. If the number of trials is severely limited, the split between global-mode and subset-mode can be tuned to possibly obtain even larger gains. We provide an estimate of the number of trials required for each CPM in Appendix A.2.

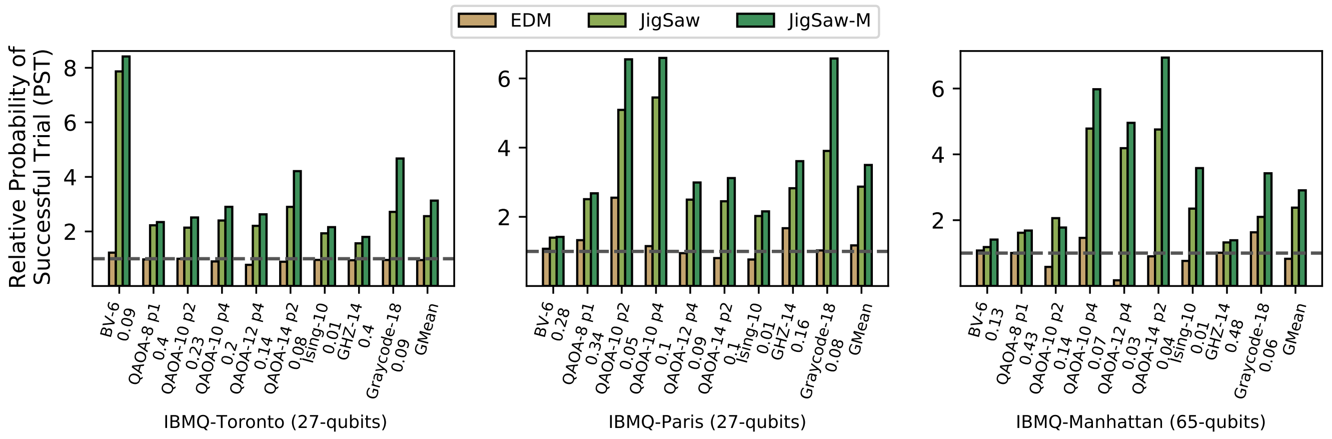

Figure 8: Probability of Successful Trial (PST) from JigSaw and JigSaw-M relative to baseline and comparison with prior work Ensemble of Diverse Mappings (EDM) [48]. Number below the label shows absolute PST for the benchmark.

5.5 Figure-of-Merit

At present, there is no standard single metric to evaluate NISQ application fidelity. Therefore, for our evaluations, we study three generic metrics and an application-specific figure-of-merit for the QAOA benchmarks. These metrics are derived from prior works and the details are discussed below:

(1) ***Probability of Successful Trial (PST)*** [11, 28, 30, 48, 49, 51] is the ratio of the number of trials with the correct output to the total number of trials, as described in Equation (1).

$$PST = \frac{\text{Number of trials with the correct output}}{\text{Total number of trials}} \tag{1}$$

(2) Inference Strength (IST) [28, 38, 48] is used to quantify the ability to infer the solution and distinguish it from incorrect outcomes. IST is defined as the ratio of the probability of occurrence of the correct outcome to the probability of occurrence of the most frequent erroneous outcome, as shown in Equation (2).

$$IST = \frac{\text{Probability of correct outcome}}{\text{Probability of most frequent incorrect outcome}} \tag{2}$$

(3) Fidelity of a program is obtained by measuring the Total Variation Distance (TVD) [54] between the output distributions on a noise-free quantum computer (P) and real hardware (Q). TVD allows us to measure the fidelity of quantum programs [42] whose output can be a probability distribution with more than one correct answer. The Fidelity ranges between 0 and 1, where 1 represents two identical distributions and 0 means completely dissimilar distributions, as shown in Equation (3). While we use TVD, Hellinger Distance [11, 17] or Kullback–Leibler divergence [25] may be used too and these metrics are closely related [54].

$$\begin{gathered} TVD(P,Q) = \sum_{i=1}^{k} || P_i - Q_i || \\ Fidelity(P,Q) = 1 - TVD(P,Q) \end{gathered} \tag{3}$$

A higher Probability of Successful Trial (PST), Fidelity, and Inference Strength (IST) is desirable.

(4) Approximation Ratio Gap (ARG) [1] is an application-specific metric for *Quantum Approximate Optimization Algorithm (QAOA) [12]* benchmarks. To solve MaxCut problems with QAOA, the classical cost function which must be maximized is translated into a cost Hamiltonian and the goal is to maximize the expectation value of the cost Hamiltonian. The expectation value of a cost function is computed by taking a mean over the samples in the output distribution of the QAOA circuit. The Approximation Ratio (AR) is defined as the ratio between the mean cost function value over these samples and the actual maximum function value of the optimal solution and is used to quantify QAOA performance [9, 56]. The Approximation Ratio Gap (ARG) denotes the percentage difference between the approximation ratio obtained on an ideal quantum computer (AR_{ideal}) and real hardware (AR_{real}), as described in Equation (4). *A lower ARG is desired as it indicates a performance closer to the noise-free scenario.*

$$ARG = \frac{100 * (AR_{ideal} - AR_{real})}{AR_{ideal}} \tag{4}$$

6 RESULTS AND SENSITIVITY STUDIES

In this Section, we discuss the impact of JigSaw on the reliability of NISQ applications.

6.1 Results for Probability of Successful Trial

Figure 8 shows the improvement in Probability of Successful Trial (PST) using JigSaw and JigSaw-M. Our evaluations using three different quantum hardware from IBM and tens of quantum benchmarks show that JigSaw the PST by 2.91x on average and by up-to 7.87x compared to the baseline. JigSaw-M improves PST by 3.65x on average and up-to 8.42x compared to the baseline. Compared to JigSaw, JigSaw-M improves the PST of applications by 1.26x on average and up-to 1.72x.

6.2 Results for Inference Strength

Inference Strength (IST) determines the capability to suppress correlated errors and infer the correct answer of a program from the the output PMF. Table 3 shows the improvement in IST for EDM, JigSaw, and JigSaw-M relative to the baseline. Note that the average here is the geometric mean. JigSaw improves the IST on average by 2.19x and up-to 21.7x compared to the baseline, whereas JigSaw-M improves the IST on average by 2.82x and up-to 27.9x. Unlike EDM which only improves the IST, JigSaw improves both PST and IST.

Table 3: Inference Strength (IST) obtained from EDM, JigSaw, and JigSaw-M relative to the Baseline

IBMQ	EDM			JigSaw			JigSaw-M		
(Hardware)	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
Toronto	0.92	2.25	1.36	1.22	21.7	2.87	1.23	27.9	3.84
Paris	0.78	6.54	1.36	1.07	9.07	2.33	1.09	28.1	3.13
Manhattan	0.75	2.74	1.27	0.81	3.12	1.35	0.83	3.40	1.46

6.3 Results for Fidelity

Table 4 compares the Fidelity for EDM, JigSaw, and JigSaw-M relative to the baseline. For instance, on IBMQ-Toronto, EDM degrades Fidelity by 0.96x on an average, whereas JigSaw and JigSaw-M improve it by 2.17x and 2.54x respectively. Overall, JigSaw improves the Fidelity on average by 2.12x, whereas JigSaw-M improves the Fidelity on average by 2.47x and by up-to 8.41x compared to the baseline. Therefore, the output distributions of programs obtained from JigSaw and JigSaw-M have higher Fidelity and are significantly more similar to the distributions obtained on a noise-free quantum computer.

Table 4: Fidelity obtained from EDM, JigSaw, and JigSaw-M relative to the baseline

IBMQ	EDM			JigSaw			JigSaw-M		
(Hardware)	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
Toronto	0.78	1.22	0.96	1.07	7.86	2.17	1.07	8.41	2.54
Paris	0.77	2.54	1.19	1.09	5.07	2.33	1.11	6.52	2.77
Manhattan	0.43	1.62	0.93	1.18	3.26	1.84	1.28	4.43	2.10

6.4 Results for Approximation Ratio Gap

We use Approximation Ratio Gap (ARG) as an application-specific metric for the QAOA benchmarks. A lower ARG is desired as it indicates a performance closer to the noise-free scenario [1]. Table 5 compares the ARG for the QAOA benchmarks evaluated in this paper. We observe that JigSaw reduces the ARG by 0.41x on average and by up-to 0.14x compared to the baseline. Alternately, JigSaw-M reduces the ARG by 0.31x on average and up-to 0.08x compared to the baseline. Overall, JigSaw and JigSaw-M consistently outperforms the baseline and EDM. Overall, JigSaw outperforms the baseline and prior work, EDM [48], across all the four metrics.

Table 5: Comparison of Approximation Ratio Gap (ARG) (values are %) between Baseline, EDM, JigSaw, and JigSaw-M

Machine	Workload	Baseline	EDM	JigSaw	JigSaw-M
Tor	QAOA-8 p1	19.6	19.4	2.83	1.59
	QAOA-10 p2	24.5	24.0	12.3	10.6
	QAOA-10 p4	23.4	24.3	10.5	8.50
	QAOA-12 p4	12.3	13.8	4.82	3.11
	QAOA-14 p2	9.86	9.74	4.06	2.48
Par	QAOA-8 p1	21.7	17.5	3.91	2.50
	QAOA-10 p2	35.0	29.2	19.0	16.3
	QAOA-10 p4	30.3	29.4	8.60	6.19
	QAOA-12 p4	10.5	11.8	5.63	4.98
	QAOA-14 p2	8.50	9.66	3.95	2.95
Man	QAOA-8 p1	18.2	18.4	8.87	8.23
	QAOA-10 p2	28.1	31.9	19.3	19.8
	QAOA-10 p4	31.1	29.9	13.7	11.1
	QAOA-12 p4	14.1	26.6	5.12	2.94
	QAOA-14 p2	11.5	14.3	5.93	4.49

6.5 Impact of Number of Circuits with Partial Measurements and Selection Method

Our default design uses a sliding window method to generate a handle of unique CPM. To understand the impact of the number of CPM and selection method, we perform an empirical study using a 12-qubit QAOA program on IBMQ-Paris.

Sensitivity to Number of CPM: The total number of possible CPM (measuring two qubits) for a Q qubit program is $^{Q}C_2$. To understand the impact of the number of CPM (N) on its effectiveness, JigSaw randomly generates N circuits with partial measurements of subset size 2 out of all the 66 possibilities ($^{12}C_2 = 66$) and uses these N local-PMFs to update the global-PMF. The process is repeated hundreds of times for each N and Figure 9(a) shows the average improvement in Application PST from JigSaw as N is increased.

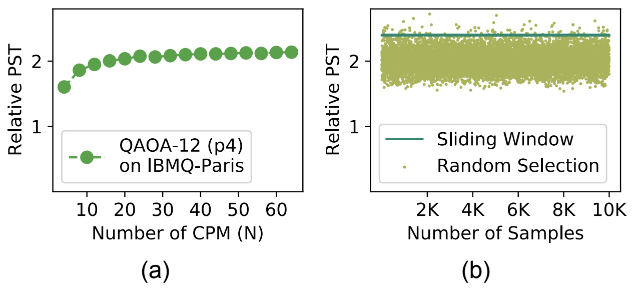

Figure 9: Impact of (a) Number of CPM and (b) CPM Selection Method on the Performance of JigSaw.

We observe that the performance of JigSaw saturates when additional CPM do not offer incremental information. Thus, only a few and unique CPM are sufficient for JigSaw to be effective. Further, to obtain a greater number of unique CPM, JigSaw-M generates CPM of non-uniform subset sizes.

Sensitivity to CPM Selection Method: To study the impact of CPM selection method, JigSaw randomly selects a group of CPM of subset size 2 from all the 66 possibilities while ensuring that each program qubit is measured in a CPM at least once. As there are 12 qubits in the program JigSaw selects 12 random CPM each time and the process is repeated 10,000 times. Figure 9(b) shows the relative improvement in PST for this study, and we observe that we get similar results irrespective of the CPM. Thus, although our default design uses a sliding window method, JigSaw is equally effective even if any other technique to generate the CPM is used.

6.6 Impact of Recompilation

JigSaw mainly benefits from measurement subsetting and recompilation. By recompiling each CPM, the effective measurement error-rates for CPM are close to the best-case qubits rather than close to the average-case qubits (which is the case for the global-mode and the baseline). For example, Figure 10 shows that the probability of correctly measuring a qubit in a CPM increases by up-to 3.25x compared to the baseline for a BV-6 benchmark on IBMQ-Toronto. Note that the probability of correctly measuring a qubit is computed from the set of outcomes where the particular qubit is correctly measured, even if the overall outcome is erroneous and does not represent the correct answer. Thus, recompiling CPM can significantly enhance the effectiveness of JigSaw.

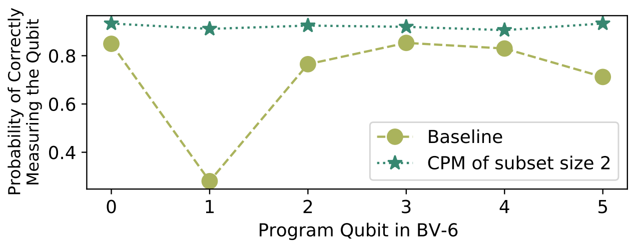

Figure 10: The probability of successfully measuring each qubit for a 6-qubit BV program on IBMQ-Toronto in the (a) baseline (b) in each CPM after recompilation.

Figure 11 shows the Mean PST from JigSaw without recompilation (subsetting only), JigSaw with recompilation, and JigSaw-M relative to the baseline. Without recompilation, JigSaw improves the PST by 1.92x on average and up-to 3.26x, whereas with recompilation JigSaw improves the PST by 2.91x on average and up-to 7.8x compared to the baseline. With recompilation, JigSaw-M improves the PST by 3.65x on average and by up-to 8.4x.

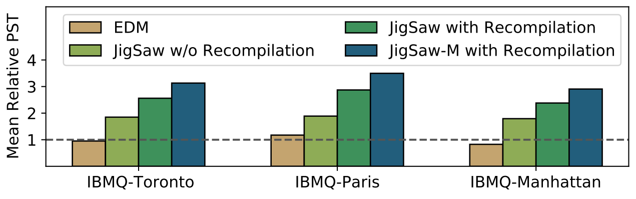

Figure 11: Comparison of Mean PST relative to the Baseline.

7 SCALABILITY OF RECONSTRUCTION

The applicability of reconstruction algorithms is often limited by their memory and time complexity. For example, tensor-product based algorithms are extremely hard to scale due to their exponential complexity. Therefore, we study the scalability of our proposed Bayesian Reconstruction using an analytical model, described next.

7.1 Insight: Bounded by Observations

JigSaw limits the complexity of the reconstruction algorithm by storing and updating only the non-zero entries generated in the global-mode. Although the number of possible non-zero entries in the global-PMF can scale exponentially with the program size, the actual number of entries observed is much lower and is *limited by the number of trials*, particularly for large programs. For example, there are 2^{100} possible outcomes for a 100-qubit program, and assuming the circuit produces a uniform distribution, it would require a minimum of 2^{100} or 10^{30} trials to observe each of these outcomes at least once, which is impractical. In practice, we may be able to execute at-most a few million trials, since the time to execute the trials still increases linearly with the number of trials. Moreover, practical quantum algorithms are designed to produce output distributions with relatively low variance and bounded possible outcomes. For example, Table 6 shows that a Graycode-18 benchmark produces only up to 18.5K unique outcomes when executed for 512K trials, even though 256K ($=2^{18}$) outcomes are possible. We bound the complexity of JigSaw reconstruction by focusing only on the outcomes observed rather than all the possible outcomes.

Table 6: Number of Outcomes in the Global-PMF for a Graycode-18 Benchmark on IBMQ Hardware

Outcomes	IBMQ-Toronto	IBMQ-Paris	IBMQ-Manhattan
Observed (Obs)	17.0 K	17.3 K	18.5 K
Maximum (Max)	256 K	256 K	256 K
Ratio (Obs/Max)	6.6 %	6.8 %	7.2 %

7.2 Memory Complexity

JigSaw only stores non-zero entries in the global, output, local PMFs, and an intermediate PMF for each CPM. We assume N is the number of CPM. For simplicity, we assume the global-mode and each CPM in subset-mode uses T trials.[3]

Global, Intermediate, and Output PMFs: Each global-PMF entry comprises of an n-bit string outcome and its probability of occurrence, as shown in Figure 12(a). We assume ϵT entries in the global-PMF, where $0 < \epsilon \leq 1$. As JigSaw only updates the probabilities of the global-PMF entries, the intermediate and output PMFs are only required to store the updated probabilities, as shown in Figure 12(b). Hence, the global-PMF requires $(n + 8)$ bytes per entry, whereas the intermediate and output PMFs each require 8 bytes per entry.

[3]For simplicity, we assume up-to 1 million trials each for the global-mode and each CPM, which is a severely pessimistic assumption. In practice, the trials are split between a large number of CPM, so the storage and timing complexity gets reduced further.

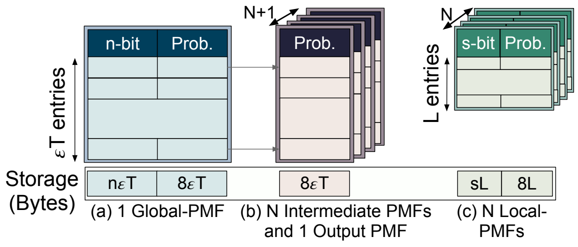

Figure 12: Memory required to store (a) the global, (b) N intermediate and 1 output PMF, and (c) N local-PMFs.

Our experiments on IBMQ systems show that $\epsilon \ll 1$ and does not change rapidly with increasing trials. For example, Figure 13 shows the number of unique outcomes and ϵ when some GHZ and QAOA benchmarks are executed for 4 million trials on IBMQ-Paris.

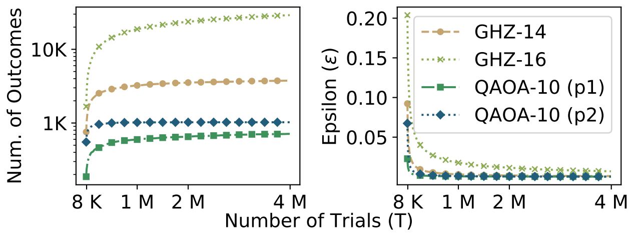

Figure 13: (a) Number of Global-PMF entries and (b) Epsilon (ϵ) with increasing trials (T) on IBMQ-Paris.

Local-PMFs: A local-PMF for a CPM of subset size s consists of $L = min(2^s, \delta T)$ entries, where $0 < \delta \leq 1$, and requires $L(s + 8)$ bytes, as shown in Figure 12(c). To minimize errors on CPM, s must be small such as 2 in the default JigSaw design. For such small s, a local-PMF consists of all possible 2^s entries. But for large s, $L \ll 2^s$ and is denoted by δT. For example, local-PMFs of size 2 and 10 for a GHZ-14 program on IBMQ-Toronto contain 4 and 297 entries respectively, even though 1024 entries are possible for s = 10.

JigSaw stores one Global-PMF, N intermediate PMFs, one output PMF and N local-PMFs. JigSaw-M stores one Global-PMF, N intermediate PMFs, one output PMF and $\mathbb{S}$N local-PMFs where $\mathbb{S}$ subset sizes are used. Although JigSaw-M uses more CPM, since it employs hierarchical reconstruction from the highest to the lowest subset size, only N intermediate PMFs are required which are reused across reconstruction rounds. Thus, the total memory capacity (in bytes) is given by Equation (5). The memory complexity for JigSaw is obtained for $\mathbb{S} = 1$ since it uses CPM of a single subset size only.

$$\text{Memory} = \{n + 8(2 + N)\}\epsilon T + L(s + 8)\mathbb{S}N \qquad (5)$$

7.3 Time Complexity

JigSaw updates each Global-PMF entry for each entry in a local-PMF. Obtaining the update coefficients require ϵT operations and the update itself requires $3\epsilon T$ operations per local-PMF. Assuming JigSaw uses N CPM, it requires $4\epsilon NT$ operations. Similarly, JigSaw-M requires $4\epsilon \mathbb{S} NT$ operations. As JigSaw only stores and updates non-zero PMF entries, which is much lower than the maximum possible, and is limited by the number of trials, the time-complexity increases linearly with the number of trials and qubits.

7.4 Results for Scalability Analysis

Table 7 shows the memory and number of operations required for programs of different input sizes n, values of ϵ, δ, and number of trials T. To obtain the typical complexity, we use T = 1 million and $\epsilon = \delta = 0.05$ (from Figure 13), whereas we use $\epsilon = \delta = 1$ to obtain the upper bound. For JigSaw, we assume CPM of subset size 5 and the number of CPM (N) to be same as the number of qubits in the program (as our default design). For JigSaw-M, we assume sizes 5,10,15, and 20. We observe that the storage and time complexity is linear with the number of trials and qubits in the program, making JigSaw applicable to programs with hundreds of qubits.

Table 7: Scalability Analysis of JigSaw and JigSaw-M: Memory (in GB) and Number of Operations (in million)

Qubits (n)	$\epsilon = \delta$	Trials (T)	JigSaw		JigSaw-M	
			Mem	OPs	Mem	OPs
100	0.05	32K	0.01	0.66	0.02	2.64
		1024K	0.05	21.0	0.42	83.9
	1.0	32K	0.03	13.1	0.20	52.4
		1024K	0.96	419	3.97	1677
500	0.05	32K	0.01	3.28	0.1	13.12
		1024K	0.24	105	2.09	419
	1.0	32K	0.15	65.5	0.99	262
		1024K	4.74	2097	19.8	8388

8 RELATED WORK

Software mitigation of NISQ hardware errors [11, 14, 15, 18, 30–32, 34–36, 39, 44, 45, 48, 49, 55, 57] is an active area of research. More recently, schemes that specifically reduce the impact of measurement errors have been proposed [5, 8, 19, 26, 37, 50]. We compare with two of these proposals, and then other works.

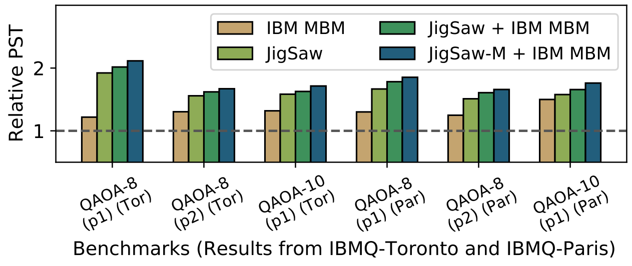

Figure 14: JigSaw vs. IBM's Error Mitigation [19]

Matrix-Based Error Mitigation: IBM's matrix-based complete measurement error mitigation (MBM) post-processes the outputs of an n-qubit program using a 2^n x 2^n inverse noise matrix prepared by calibrating 2^n basis states [19]. Jigsaw can be combined with MBM for even higher fidelity than either scheme standalone, as shown in Figure 14. Note that the complexity of MEM grows exponentially with the program size, whereas JigSaw needs no characterization and its post-processing step has linear complexity.

State-Based Error Mitigation: Few prior schemes transform a more error-prone quantum state to a less susceptible one [26, 50] using single qubit gates, but has limited applicability on recent devices as they do not exhibit considerable bias in measuring state "1" over state "0". For example, the average probabilities of incorrectly measuring states "0" and "1" on IBMQ-Manhattan are 2.3% and 3.6%, respectively. We make similar observations on other machines too.

Related Works in Circuit Decomposition: Peng et. al. [40] proposed circuit decomposition techniques and discussed the mathematical validity of executing a larger program on smaller quantum machines. A scalable approach for efficient circuit cutting was introduced in [47]. However, these approaches use tensor products and are hard to scale. On the contrary, JigSaw utilizes circuits identical to the original program except the reduced measurements and incurs linear complexity, making it scalable and applicable to all programs. Shehab et. al. [43] use partial circuits to reduce computations and improve fidelity for QAOA applications.

9 CONCLUSION

In this paper, we propose *JigSaw*- a design that mitigates the impact of measurement errors by executing programs in two modes: *global-mode* in which the program measuring all the qubits generates a global Probability Mass Function (PMF) over all the qubits, and *subset-mode* in which multiple *Circuits with Partial Measurements (CPM)* produce local-PMFs over only the measured qubits. The global-PMF offers full correlation but low fidelity, whereas the local-PMFs offer higher fidelity but poor correlation. By employing Bayesian updates and the local-PMFs, JigSaw enhances the global-PMF and improves the success rate of applications on average by 2.91x and up-to 7.87x. We also propose *Multi-Layer JigSaw (JigSaw-M)* which uses a larger number of unique CPM of non-uniform subset sizes and employs an ordered reconstruction algorithm to enhance the global-PMF. Overall, JigSaw-M improves the success rate of applications on average by 3.65x and up-to 8.42x.

ACKNOWLEDGMENTS

We thank Nicolas Delfosse for helpful discussions and feedback. We also thank Sanjay Kariyappa, Ananda Samajdar, and Keval Kamdar for editorial suggestions. Poulami Das was funded by the Microsoft Research PhD Fellowship. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

A APPENDIX

A.1 Bayesian Reconstruction

The Bayesian Reconstruction algorithm is shown in Algorithm 1.

A.2 Estimate for the Number of Trials

For simplicity, by default we execute the global-mode for half of the trials and the subset-mode for the remaining half. We also equally distribute the trials in the subset-mode between the CPM for both JigSaw and JigSaw-M. However, if the trials are severely limited, (1) the distribution of trials may be fine-tuned or (2) the subset-mode may be executed for few thousands of extra trials (global-mode corresponds to the baseline). We perform an analysis of how the trials may be allocated for each CPM.

Algorithm 1: Bayesian Reconstruction Algorithm

```
Input: (1) Global-PMF P = {$B_x : pr_x$} where $B_x$ is a n-bit
       outcome (2) Set of j Marginals M = {$m_j$} where
       $m_j$ = [{$B_y : pr_y$}, {$i_0 ... i_k$}] for k-bit outcome $B_y$
Output: PMF $P_{out}$ = {$B_x : \mathbb{P}_x$}, $\mathbb{P}_x \in [0, 1]$
Function Bayesian_Update(P,m):
    $P_o$ = P
    for each (entry $B_y : pr_y$) in m: do
        candidate = [ ]
        for each $B_x$ in P: do
            // Obtain list of outcomes in P
            outcome ← bits in $B_x$ corresponding to
            qubits {$i_0 ... i_k$}
            candidate.append((outcome,$pr_x$))
            // Obtain Update Coefficients
            normalize candidate
        for each outcome in candidate: do
            // Obtain posterior probabilities
            $P_o[outcome] = \frac{candidate[outcome] \times pr_y}{(1 - pr_y)}$
        normalize $P_o$
    return $P_o$
Function Bayesian_Reconstruction(P,M):
    $P_{out}$ = P
    for each $m_j$ in M: do
        $P_{post}$ = Bayesian_Update(P, $m_j$)
        $P_{out} = P_{post} + P_{out}$
    normalize $P_{out}$
    return $P_{out}$
```

Let there be $N(= 2^n)$ possible outcomes for a program that measures n qubits. If p is the probability of observing an outcome and each of the N outcomes is equally likely to appear at the end of a trial, then $p = 1/N$. The probability that a given outcome has appeared at least once after t trials is then given by Equation (6).

$$\mathbb{P} = [1 - (1 - p)^t] \tag{6}$$

If $t \approx \alpha N$ and N is large, $\mathbb{P}$ may be approximated as Equation (7).

$$\mathbb{P} = 1 - e^{-\alpha} \tag{7}$$

Thus, in order to obtain the given outcome at least once with probability $\mathbb{P}$, the number of trials required is given by Equation (8).

$$t = -\ln(1 - \mathbb{P})N \tag{8}$$

Hence, the total number of trials required to observe every possible outcome at least once with probability $\mathbb{P}$ is given by Equation (9).

$$\text{Total number of Trials} = -\ln(1 - \mathbb{P})N^2 \tag{9}$$

We measure only 2 qubits in each CPM in the default JigSaw design and thus, only about 150 trials are required to ensure (with 99.99% probability) that we obtain each possible answer at-least one time. Similarly, as JigSaw-M uses CPM of different sizes, the estimated number of trials would still range within a few thousands.

REFERENCES

[1] 2020. Circuit Compilation Methodologies for Quantum Approximate Optimization Algorithm, author=Alam, Mahabubul and Ash-Saki, Abdullah and Ghosh, Swaroop. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 215–228.

[2] J Abhijith, Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, Andreas Bärtschi, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, et al. 2018. Quantum algorithm implementations for beginners. *arXiv e-prints* (2018), arXiv–1804.

[3] Google Quantum AI. 2021. Exponential suppression of bit or phase errors with cyclic error correction. *Nature* 595, 7867 (2021), 383.

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 2019. Supplementary information for" Quantum supremacy using a programmable superconducting processor". *arXiv preprint arXiv:1910.11333* (2019).

[5] George S Barron and Christopher J Wood. 2020. Measurement error mitigation for variational quantum algorithms. *arXiv preprint arXiv:2010.08520* (2020).

[6] Ethan Bernstein and Umesh Vazirani. 1997. Quantum complexity theory. *SIAM Journal on computing* 26, 5 (1997), 1411–1473.

[7] Alexandre Blais, Ren-Shou Huang, Andreas Wallraff, Steven M Girvin, and R Jun Schoelkopf. 2004. Cavity quantum electrodynamics for superconducting electrical circuits: An architecture for quantum computation. *Physical Review A* 69, 6 (2004), 062320.

[8] Sergey Bravyi, Sarah Sheldon, Abhinav Kandala, David C Mckay, and Jay M Gambetta. 2020. Mitigating measurement errors in multi-qubit experiments. *arXiv preprint arXiv:2006.14044* (2020).

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

[10] Andrew W Cross, Lev S Bishop, Sarah Sheldon, Paul D Nation, and Jay M Gambetta. 2019. Validating quantum computers using randomized model circuits. *Physical Review A* 100, 3 (2019), 032328.

[11] Poulami Das, Swamit S Tannu, Prashant J Nair, and Moinuddin Qureshi. 2019. A Case for Multi-Programming Quantum Computers. In *MICRO*. ACM, 291–303.

[12] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. *arXiv preprint arXiv:1411.4028* (2014).

[13] Michael R Geller and Mingyu Sun. 2021. Toward efficient correction of multiqubit measurement errors: pair correlation method. *Quantum Science and Technology* 6, 2 (2021), 025009.

[14] Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I Schuster, Henry Hoffmann, and Frederic T Chong. 2019. Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines. In *MICRO*. ACM, 266–278.

[15] Pranav Gokhale, Ali Javadi-Abhari, Nathan Earnest, Yunong Shi, and Frederic T Chong. 2020. Optimized Quantum Compilation for Near-Term Algorithms with OpenPulse. *arXiv preprint arXiv:2004.11205* (2020).

[16] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. 1989. Going beyond Bell's theorem. In *Bell's theorem, quantum theory and conceptions of the universe*. Springer, 69–72.

[17] Ernst Hellinger. 1909. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik* 136 (1909), 210–271.

[18] Yipeng Huang and Margaret Martonosi. 2019. Statistical assertions for validating patterns and finding bugs in quantum programs. In *ISCA*. 541–553.

[19] IBM. 2010. Measurement Error Mitigation. https://qiskit.org/textbook/ch-quantum-hardware/measurement-error-mitigation.html. [Accessed July-2020].

[20] IBM. 2020. IBM Quantum Systems and Simulators. https://quantum-computing.ibm.com/. [Online; accessed 7-March-2021].

[21] Ernst Ising. 1925. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik* 31, 1 (1925), 253–258.

[22] James Joyce. 2003. Bayes' theorem. (2003).

[23] Mostafa Khezri, Justin Dressel, and Alexander N Korotkov. 2015. Qubit measurement error from coupling with a detuned neighbor in circuit QED. *Physical Review A* 92, 5 (2015), 052306.

[24] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. 2019. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews* 6, 2 (2019), 021318.

[25] Solomon Kullback. 1997. *Information theory and statistics*. Courier Corporation.

[26] Hyeokjea Kwon and Joonwoo Bae. 2020. A hybrid quantum-classical approach to mitigating measurement errors. *arXiv preprint arXiv:2003.12314* (2020).

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

[28] Ji Liu and Huiyang Zhou. [n. d.]. Reliability Modeling of NISQ-Era Quantum Computers. ([n. d.]).

[29] Seth Lloyd. 1996. Universal quantum simulators. *Science* (1996), 1073–1078.

[30] 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).

[31] Prakash Murali, Norbert Matthias Linke, Margaret Martonosi, Ali Javadi Abhari, Nhung Hong Nguyen, and Cinthia Huerta Alderete. 2019. Full-stack, real-system quantum computer studies: architectural comparisons and design insights. In *Proc. of the 46th International Symposium on Computer Architecture*. 527–540.

[32] Prakash Murali, Norbert M Linke, Margaret Martonosi, Ali Javadi Abhari, Nhung Hong Nguyen, and Cinthia Huerta Alderete. 2020. Architecting Noisy Intermediate-Scale Quantum Computers: A Real-System Study. *IEEE Micro* 40, 3 (2020), 73–80.

[33] Prakash Murali, David C McKay, Margaret Martonosi, and Ali Javadi-Abhari. 2020. Software Mitigation of Crosstalk on Noisy Intermediate-Scale Quantum Computers. *arXiv preprint arXiv:2001.02826* (2020).

[34] 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).

[35] National Academies of Sciences Engineering and Medicine. 2019. *Quantum Computing: Progress and Prospects*. The National Academies Press, Washington, DC. https://doi.org/10.17226/25196

[36] Tirthak Patel, Baolin Li, Rohan Basu Roy, and Devesh Tiwari. 2020. {UREQA}: Leveraging Operation-Aware Error Rates for Effective Quantum Circuit Mapping on NISQ-Era Quantum Computers. In *USENIX ATC*. 705–711.

[37] Tirthak Patel and Devesh Tiwari. 2020. DisQ: a novel quantum output state classification method on IBM quantum computers using OpenPulse. In *ICCAD*.

[38] Tirthak Patel and Devesh Tiwari. 2020. Veritas: accurately estimating the correct output on noisy intermediate-scale quantum computers. In *SC20*. IEEE, 1–16.

[39] Tirthak Patel and Devesh Tiwari. 2021. Qraft: reverse your Quantum circuit and know the correct program output. In *ASPLOS*. 443–455.

[40] Tianyi Peng, Aram Harrow, Maris Ozols, and Xiaodi Wu. 2019. Simulating large quantum circuits on a small quantum computer. *preprint arXiv:1904.00102* (2019).

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

[42] Yuval R Sanders, Joel J Wallman, and Barry C Sanders. 2015. Bounding quantum gate error rate based on reported average fidelity. *New Journal of Physics* 18, 1 (2015), 012002.

[43] Omar Shehab, Isaac H Kim, Nhung H Nguyen, Kevin Landsman, Cinthia H Alderete, Daiwei Zhu, C Monroe, and Norbert M Linke. 2019. Noise reduction using past causal cones in variational quantum algorithms. *arXiv preprint arXiv:1906.00476* (2019).

[44] Yunong Shi, Pranav Gokhale, Prakash Murali, Jonathan M Baker, Casey Duckering, Yongshan Ding, Natalie C Brown, Christopher Chamberland, Ali Javadi-Abhari, Andrew W Cross, et al. 2020. Resource-Efficient Quantum Computing by Breaking Abstractions. *Proc. IEEE* (2020).

[45] Yunong Shi, Nelson Leung, Pranav Gokhale, Zane Rossi, David I Schuster, Henry Hoffmann, and Frederic T Chong. 2019. Optimized compilation of aggregated instructions for realistic quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 1031–1044.

[46] Peter W Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* 41, 2 (1999), 303–332.

[47] Wei Tang, Teague Tomesh, Jeffrey Larson, Martin Suchara, and Margaret Martonosi. 2020. CutQC: Using Small Quantum Computers for Large Quantum Circuit Evaluations. *arXiv preprint arXiv:2012.02333* (2020).

[48] Swamit S Tannu and Moinuddin Qureshi. 2019. Ensemble of Diverse Mappings: Improving Reliability of Quantum Computers by Orchestrating Dissimilar Mistakes. In *MICRO*. ACM, 253–265.

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

[50] Swamit S Tannu and Moinuddin K Qureshi. 2019. Mitigating Measurement Errors in Quantum Computers by Exploiting State-Dependent Bias. In *MICRO*. 279–290.

[51] 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 *ASPLOS*. 987–999.

[52] Benjamin Villalonga, Dmitry Lyakh, Sergio Boixo, Hartmut Neven, Travis S Humble, Rupak Biswas, Eleanor G Rieffel, Alan Ho, and Salvatore Mandrà. 2019. Establishing the Quantum Supremacy Frontier with a 281 Pflop/s Simulation. *arXiv preprint arXiv:1905.00444* (2019).

[53] Andreas Wallraff, David I Schuster, Alexandre Blais, Luigi Frunzio, R-S Huang, Johannes Majer, Sameer Kumar, Steven M Girvin, and Robert J Schoelkopf. 2004. Strong coupling of a single photon to a superconducting qubit using circuit quantum electrodynamics. *Nature* 431, 7005 (2004), 162.

[54] Wikipedia. 2020. Total Variational Distance. https://en.wikipedia.org/wiki/Total_variation_distance_of_probability_measures. [Online; accessed 7-March-2021].

[55] Ellis Wilson, Sudhakar Singh, and Frank Mueller. 2020. Just-in-time Quantum Circuit Transpilation Reduces Noise. *arXiv preprint arXiv:2005.12820* (2020).

[56] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. 2020. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X* 10, 2 (2020), 021067.

[57] Alwin Zulehner, Alexandru Paler, and Robert Wille. 2018. Efficient mapping of quantum circuits to the IBM QX architectures. In *2018 DATE*. IEEE, 1135–1138.