

# Dynamic Resource Allocation with Quantum Error Detection

Quinn Langfitt<sup>1</sup>, Alvin Gonzales<sup>2</sup>, Joshua Gao<sup>3</sup>, Ji Liu<sup>2</sup>, Zain H. Saleem<sup>2</sup>, Nikos Hardavellas<sup>1</sup>, Kaitlin N. Smith<sup>1</sup>

<sup>1</sup>*Department of Computer Science, Northwestern University, Evanston, IL, USA*

<sup>2</sup>*Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA*

<sup>3</sup>*Department of Computer Science, Virginia Tech, Blacksburg, VA, USA*

**Abstract**—Quantum processing units (QPUs) are highly heterogeneous in terms of physical qubit performance. To add even more complexity, drift in quantum noise landscapes has been well-documented. This makes resource allocation a challenging problem whenever a quantum program must be mapped to hardware. As a solution, we propose a novel resource allocation framework that applies Pauli checks. Pauli checks have demonstrated their efficacy at error mitigation in prior work, and in this paper, we highlight their potential to infer the noise characteristics of a quantum system. Circuits with embedded Pauli checks can be executed on different regions of qubits, and the syndrome data created by error-detecting Pauli checks can be leveraged to guide quantum program outcomes toward regions that produce higher-fidelity final distributions. Using noisy simulation and a real QPU testbed, we show that dynamic quantum resource allocation with Pauli checks can outperform state-of-art mapping techniques, such as those that are noise-aware. Further, when applied toward the Quantum Approximate Optimization Algorithm, techniques guided by Pauli checks demonstrate the ability to increase circuit fidelity  $> 11\%$  on average, and up to 33%.

## I. INTRODUCTION

Programming a quantum computer (QC) is a challenging task. First, arbitrary quantum programs must be compiled so that they respect QPU architectural constraints such as qubit-qubit connectivity and supported basis gate set. Then, the circuit must be mapped to a robust region of hardware, which can be a moving target on today’s devices. The consequence of a poorly mapped circuit is a low fidelity result, which is sub-optimal with the rising cost to access quantum hardware through the quantum cloud [26]. Thus, we are motivated to develop techniques that enable informed quantum resource allocation that work toward maximal utilization of hardware and improved fidelity of quantum program outcomes. The problem of mapping an application to a highly diverse QPU noise profile is illustrated in Fig. 1.

Much work has been dedicated to improving both quantum program success and machine utilization during runtime in an effort to ease bottlenecks that hinder scaling. For example, noise-adaptive mapping is a state-of-art quantum circuit optimization technique used to place and route quantum programs on quantum hardware [20]. Noise-aware schemes, however, depend on accurate calibration data either from the QC provider or from high-overhead benchmarking. Even in the case that a snapshot of machine properties is available, QC noise is observed to fluctuate over time in ways that

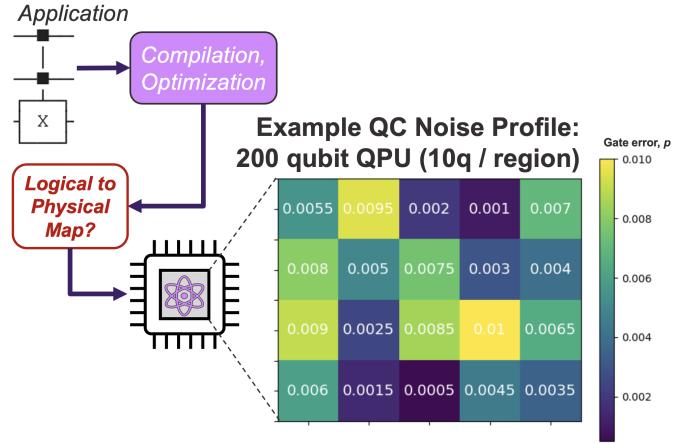


Fig. 1. A challenging problem facing quantum programmers involves mapping applications to QPUs with heterogeneous noise profiles. Pictured here is a QPU with 20 regions (10 qubits per region) of unique 1q error rates,  $p \in [0.0005, 0.01]$ , and 2q errors,  $2p$ . Our proposal leverages Pauli check based characterization to learns a QPU’s noise when confidence in device properties are low.

are challenging to capture with analytical models [3]. With this in mind, it is very unlikely to fully know a QC’s error properties with high accuracy and confidence at any moment in time. Moreover, calibration data typically only reveal average error rates via randomized benchmarking [6], [14], and the errors affecting a specific circuit can deviate significantly from the idealized noise model [11], [24]. If used for hardware allocation, sub-optimal mappings can result.

The magnitude of average error rates prevents practical implementations of quantum error correction (QEC) on today’s QPUs. Fortunately, quantum error mitigation (QEM) techniques, or techniques that require classical postprocessing, have shown to be extremely valuable for handling errors that occur on a quantum processor during runtime. Examples of QEM include zero noise extrapolation [7], Clifford data regression [1], measurement error mitigation [21], and Pauli checks [4], [8]. Pauli checks are of particular interest in this work as they produce a syndrome that when measured, provides information about whether or not an error was detected. This inspired the idea to leverage Pauli check syndrome data to learn about QPU noise, especially in the use case where calibration data is unavailable.

In our work, we seek alternative methods to determine

the noise profile of a QC when its properties are unknown. This work presents a novel framework that combines the power of circuit ensembling with Pauli checks for dynamic quantum resource allocation. In this work, we define resource allocation as the process of identifying hardware regions that are available for use, analyzing their performance, and using the performance insights to guide what resources should be prioritized for use. Syndrome data produced by the Pauli checks provides a method to roughly characterize QC error, and this data is leveraged to allow higher priority contributions to quantum program final results from the most robust regions on the QPU. To learn noise that is most important to the success of target applications, our techniques intelligently embed Pauli checks into the quantum applications of interest. These programs are then executed across a QPU for noise profile characterization. Knowledge of the QPU’s best regions creates a weighted ensemble result with non-trivial gain, found to be  $> 11\%$  on average, and up to 33%, during our experiments using the Quantum Approximate Optimization Algorithm (QAOA).

Our contributions are as follows:

- We provide a study of the benefit associated with Pauli checks for quantum error detection and mitigation, while highlighting the considerations for their practical implementation. Pauli check sandwiching (PCS) and ancilla-free Pauli checks (AFPC) are of focus. To the best of our knowledge, AFPC is evaluated here at the application level for the first time.
- We propose the use of Pauli checks for quantum device characterization. We develop a framework that combines ensembled circuits with Pauli check based characterization and error mitigation in a novel manner for improved runtime results.
- The concept of a Pauli check weighted distribution is introduced, along with its threshold and hybrid variations.
- We show that we either perform as well or outperform state-of-art noise-aware mapping techniques. Further, we are able to achieve high-fidelity program outcomes without the guidance of QPU-provided error profile.
- We demonstrate how dynamic quantum resource allocation with Pauli checks can be leveraged to significantly improve program success on a variety of quantum applications, including QAOA.

We speculate that the methods presented here will also be advantageous in the not-too-distant future – as larger quantum processors are deployed on the quantum cloud, quantum resource providers may not provide fine-grained particulars about QC operational characteristics and error rates as we are familiar with today.

## II. PRIOR WORK AND MOTIVATION

QC hardware is noisy during use and expensive to access. As a result, much work has been dedicated to improving both quantum program success and machine utilization during runtime. This Section will discuss QC noise-adaptive mapping, multi-programming, and circuit-based training, three quantum

program optimization techniques that push the frontier of existing hardware. We discuss pros and cons of these methods while presenting new questions that motivate the new contributions of this work.

This Section does not provide an exhaustive introduction to quantum computation. We recommend that the reader refers to Ref. [23] for a more complete introduction to topics in quantum information processing and Ref. [5] for an overview of the fundamentals in quantum computing systems.

### A. Noise-adaptive Mapping

There are a variety of different qubit technologies that exist today, such as superconducting circuits, trapped ions, and neutral atoms, among others. Each of these platforms have their respective benefits and costs, and they all demonstrate a wide range of performance characteristics in terms of fidelity, coherence, and gate times. Even QCs comprised of the same qubit technology tend to demonstrate performance differences. Due to a combination of imperfect fabrication procedures and intrinsic environmental conditions, each quantum system is characterized by a unique noise landscape during computation. This presents the challenge of determining how to best allocate logical qubits to physical qubits during runtime.

Noise-adaptive mapping is a state-of-art mapping technique used to place and route quantum programs on quantum hardware [13], [20], [28], [32]. Given a circuit and a target QC, tools like `mapomatic` optimize for the best low-noise device sub-graph using the QC’s error rates. Noise-aware schemes, however, often depend on the availability of calibration data from the QC service provider. Further, these schemes also assume that the calibration data provided by the quantum hardware vendor is complete and correct. If trusted machine data is unavailable, protocols such as SABRE [16] will be left without (or with misinformed) guidance for logical to physical qubit mapping, resulting in poor quantum program performance. Even in the case that machine characterization data is complete and derives from a reliable source, it is known that machine noise and performance characteristics have a tendency to fluctuate over time in ways that are challenging to capture with analytical models [3]. If stale calibration data is used in this case, sub-optimal mappings result.

Noise characterization has previously been used to improve the accuracy of an ensemble of hardware executed circuits. Quancorde [25] utilizes a form of Clifford noise estimation circuits [29] to estimate the noise present in a circuit execution. The final distribution is then calculated as a weighted sum of the individual distributions. In our work, we explore alternative methods to determine the noise profile of quantum hardware due to the challenges associated with depending on provider-supplied data. It is desirable to develop techniques with minimal overheads associated with gathering characterization data. In a work with similar motivations as our own, Ref. [25] attempts to characterize the noise that an application will experience on a target QPU through the use of easy-to-simulate Clifford-circuits that approximate the structure of the circuit of interest. However, this Clifford-based technique

has drawbacks. First, additional QC runtime is required to collect characterization data from the Clifford circuits, and these characterization results must be compared to simulated Clifford circuit outcomes. Second, the actual payload circuits must be run soon after characterization data is retrieved to avoid sub-optimal results from machine property drift. Third, depending on target circuit structure, a Clifford approximation might result in a very different unitary that changes its noise sensitivity. These drawbacks inspire us to develop techniques that allow QC noise learning to be interleaved into the target circuits of interest.

### B. Maximizing QC Utilization

Some QCs on today’s quantum cloud contain more than 100 qubits, but low qubit coherence times and high operator errors prohibit the use of all of these qubits collectively for computation. As a result, QC applications are typically much smaller than the qubit capacity of the machine, leaving many QC qubits unused during runtime. Recent work has explored how to take advantage of these unused qubit resources to improve quantum program outcomes.

Ideally, compute resources are scheduled for maximum utilization. As the work in Ref. [34] highlights, parallelized quantum circuit execution, as compared to a sequential approach, offers the benefit of reduced runtime latency because a quantum circuit executed  $n$  times simultaneously can reach a threshold number of observables in a distribution in  $1/n$  time. However, work in Ref. [2] highlights that multi-programming quantum hardware, especially machines existing today, is a non-trivial task as improper scheduling procedures can cause undesirable crosstalk between two different workloads. The observation of crosstalk is especially likely whenever two quantum programs of different duration are executed adjacently – the measurement operations of the shorter circuit could impact the longer circuit that still has quantum circuit operations in-progress. If implemented with care, however, multi-programming can increase quantum system throughput without having a detrimental impact on program outcomes.

We are interested in exploring the potential that multi-programming has for improving quantum program performance. The work in [27] demonstrates how ensembling techniques employed to create a distribution from many circuit mappings can help smooth errors, improving fidelity. Creating a cumulative distribution from many instances of the same application implemented in parallel could have a similar effect that boosts performance.

### C. Pauli Check Sandwiching

‘Pauli Check Sandwiching’ (PCS) is a technique used to detect and mitigate errors [4], [8]. Prior work in PCS has shown its viability for mitigating gate and measurement errors in quantum circuits [17] and for fault injection [18] that studies error in quantum circuits. In our work, we also find that PCS is a powerful tool for guiding QC characterization, and we make the novel contribution of combining PCS with ensembling.

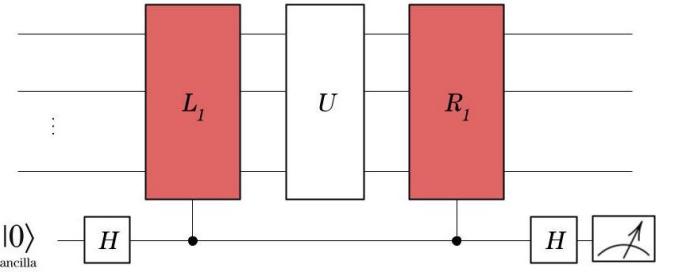


Fig. 2. General PCS circuit layout. The red unitaries represent the Pauli checks that sandwich the main payload circuit. Measurement of the ancillas provide detection of errors that occurred in the payload circuit.

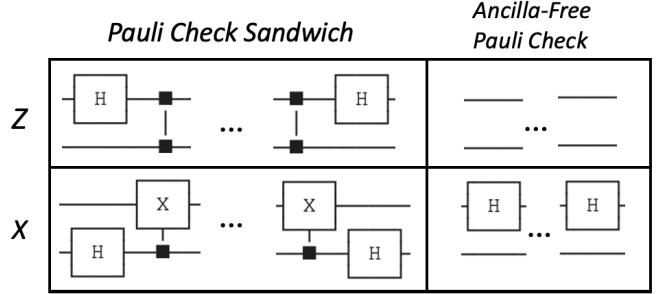


Fig. 3. Example of the conversion between PCS and AFPC for  $Z$  and  $X$  checks.

As seen in Fig. 2, PCS surrounds a payload circuit,  $U$ , with controlled Pauli operator checks. Errors on  $U$  can be detected on an ancilla through phase kickback. It is important that the relationship

$$R_1 U L_1 = U \quad (1)$$

holds so the introduced checks do not disturb the original payload circuit semantics. As a note, the operations contained in  $L_1$  and  $R_1$  do not have to comprise the same Pauli gates as long as Eqn. 1 is satisfied.

Recognizing that all errors can be decomposed into Pauli operators is a key concept in understanding PCS. By cleverly constructing Pauli sandwich checks dependent on  $U$ , anti-commutativity relations between the errors and checks can reveal errors in the ancilla qubits. Measuring 1 in any ancillas indicates that phase kickback occurred, meaning anti-commutation between the Paulis, and thus, a present error. As part of the PCS protocol, we discard this error-corrupted shot, leading to an increase in final distribution fidelity of our circuit of interest,  $U$ .

Practical application of PCS requires consideration of trade-offs. First, determining the Pauli unitary used in the check increases in complexity with the circuit. Second, Pauli check logic must be decomposed into the supported gate set of the QC, and as a result, large multi-qubit gates will end up being high-depth. Third, the overhead of adding PCS into the circuit must not outweigh the corrective benefit in terms of gate error. Fourth, while increasing the number of check pairs

improves error detection, it also introduces additional gate noise to the circuit and exponentially increases the required sampling size. We note that the exploration of an application’s ‘check limit’ is included in Ref. [15]. All of these limitations must be considered for practical implementation of PCS within quantum computing workflows.

### III. ANCILLA-FREE PAULI CHECKS

Pauli checks are a useful tool for error detection in quantum circuits. In addition to using Pauli checks for quantum error mitigation like in prior art, our proposed framework leverages Pauli checks in a novel manner to characterize quantum hardware during runtime for quantum program steering. We consider Pauli Check Sandwiching and Ancilla-Free Pauli Checks within this work.

We can estimate the postselection rate in PCS via an alternative method that does not require an ancilla and uses state preparation and measurements (SPAM) on the data qubits. We refer to this method as Ancilla-Free Pauli checks (AFPC). Note that we use AFPC only for the postselection information since in general the logical circuit is not preserved by AFPC in contrast to PCS. AFPC is a generalization of the one-sided checks introduced in Ref. [30]. State preparation and measurements are used in tomography [23] to extract channel information. AFPC can also be seen as a method that tracks a stabilizer state [10] through the circuit. However, the correspondence between the PCS postselection rate and AFPC is our novel contribution.

Given that the noise channel is Pauli and the PCS method is ideal, i.e., the noise channel is only on the data qubits, the error syndrome from PCS only reveals information on the operations between the checks. Thus, state preparation and measurement gates lying outside the checks do not affect the postselection rate.

**Theorem 1.** *For an ideal PCS scheme (noiseless checks and ancilla), the prepared input quantum state before the left checks and measurements after the right checks on the data qubits do not influence the postselection rate provided that the noise channel on the data qubits is Pauli.*

An analytical proof of Theorem 1 is provided in the Appendix. Given Theorem 1, the ancilla-free version of PCS can be derived from the stabilizer formalism [9].

First consider the controlled Pauli operation  $U_p = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes P$ . Let  $|\psi\rangle$  be stabilized by  $P$  and  $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ . Then, the operation  $U_p$  is a stabilizer of  $|+\rangle \otimes |\psi\rangle$ . Thus, in AFPC we prepare the initial state to be a  $+1$  eigenstate of  $L_i$ . Then, the PCS circuit reduces to a Hadamard test circuit as shown in Fig. 4. The Hadamard test, checks if the state is in the  $+1$  eigenspace of  $R_i$ . Since  $R_i$  is a Pauli string, we can instead perform the test without an ancilla by measuring each of the data qubits in the  $R_i$  Pauli basis and checking the parity. Thus, we have the following correspondence

**Proposition 1.** *For a Pauli noise channel on the data qubits, the post-selection rate of non-overlapping pairs (i.e., different*

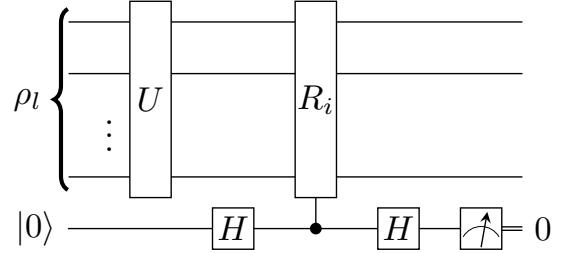


Fig. 4. When  $\rho_l$  is stabilized by  $L_i$  the PCS circuit reduces to a Hadamard test circuit.

target qubits) of ideal PCS checks can be extracted with ideal (noiseless SPAM) AFPC.

Prop. 1 follows from Theorem 1 and the construction of AFPC. An example of the correspondence is provided in Fig. 3. Biasing the noise closer to Pauli can be achieved through twirling methods such as randomized compiling [31]. Equivalence between the postselection rates of ideal PCS and ideal AFPC also happens when the prepared state in AFPC is equivalent to the input state in the PCS circuit. This happens, for example, when the left check is a tensor product of Pauli Z gates on the ground state.

### IV. METHODOLOGY

The previous sections describe how check data, or syndromes, from PCS or AFPC can inform whether or not an error occurred during the execution of a sandwiched circuit,  $U$ . Over many circuit samples, the amount of times that an ancilla is measured to be 1 can provide insight to the magnitude of quantum system noise. Consider the case where two identical circuits embedded with Pauli checks are executed in parallel on different regions of a QPU. If the check data is compared between two circuits, we can infer which execution contained more noise since a higher count of checks measuring as 1 implies that more error was present. In this Section, we first describe our techniques for embedding Pauli checks within quantum circuits. Next, we discuss how to leverage Pauli check data to guide quantum resource allocation for higher fidelity program outcomes. In the rest of this Section, we propose two key Pauli check protocols that estimate the noise profile of a target QC and guide output distributions to take most advantage of the best regions on a QPU.

#### A. Evaluating Tradeoffs for Check Placement

PCS can provide error mitigation benefits since post-selection can filter measured counts with errors. Both PCS and AFPC produce check data that can be used to learn about the noise profile of a QPU. We are interested in exploring the strengths of these approaches while being aware of the practical constraints of a realistic runtime operation environment. For this reason, we keep the PCS overhead minimal in terms of gate noise by only protecting single qubits that are on the edge of an application and thus assumed to be adjacent to

an ancilla. This avoids 1) decomposing complicated unitaries for multi-qubit checks and 2) SWAP operations needed to transfer quantum state information to a non-adjacent ancilla. Because AFPC is lower overhead in terms of required gates and associated gate error, we are able to use them to check more qubits.

We inject Pauli checks that use the same Pauli operation (i.e.  $L_1, R_1 = X$  or  $L_1, R_1 = Z$ ) to simplify the demonstration of Pauli checks injected into circuits. However, we note that this constraint can be relaxed as that Pauli checks do not need to be the same unitary for Pauli checks to be effective. For the benchmarks used in this work, it is probable that more optimal checks with higher circuit coverage exist, potentially providing more robust error mitigation and more significant fidelity improvements in certain noise models. However, we note that discovering optimal checks has poor scaling as  $L_1$  and  $R_1$  span more qubits. This serves as additional motivation to only use Pauli checks that check single qubits in both the ancilla and the ancilla-free case. For more complicated circuits (where Pauli checks cannot be found), sub-circuits instead of the full circuit can be examined instead.

### B. Protocols for PC-enhanced Distributions

Many of today's QPUs feature more qubits than are required for target applications associated with general use. In this situation, numerous mapping permutations that assign logical to physical qubits are possible, and this observation introduces the questions of 1) how to make the best use of a quantum device and 2) how to maximize the exploration of the noise landscape to discover the regions that currently compute with highest fidelity. On one hand, one could prioritize using the 'best' regions on-chip according to a mapping method guided by calibration data. This technique would serialized circuit execution to strategically target the historically low-error regions. However, this approach requires a priori knowledge of the properties of the QPU. In addition, serialized execution necessitates longer periods of dedicated hardware access. On the other hand, multi-programming to generate an ensemble distribution amplifies throughput via parallelization, potentially enabling lower-latency runtime. Additionally, it enables more of the device to be explored, potentially discovering unexpectedly high-performing regions, but it comes at the cost of also sampling less reliable qubits if the error profile varies drastically on the QPU.

In our work, we assume that the number of qubits on chip is larger than the number of logical qubits required for an application. We also assume no prior knowledge of the noise profile on-chip in our main approach. We see that leveraging Pauli checks to learn quantum noise and boost program outcomes provides the substantial advantage of limiting dependence on data from the quantum hardware provider. While exploring the noise frontier of the QPU, we attempt to maximize hardware utilization as much as possible.

*1) PCS Weighted Distribution:* We inject PCS into our target application, but only single-qubit checks on edge qubits are employed to minimize PCS gate count overhead. To

maximally utilize hardware via multi-programming, a QC with  $n$  physical qubits holds

$$\text{threads} = \lfloor n / (q_{\text{algorithm}} + q_{\text{ancilla}}) \rfloor \quad (2)$$

instances of a circuit running in parallel, assuming a graph with sufficient connectivity. In this equation,  $q$  indicates qubits, both algorithm and ancilla, that are in the logical quantum circuit. Ancillas are only required in the PCS approach. We note while Eqn. 2 indicates a maximum number of parallel maps, any smaller set of qubit layouts may also be used with the techniques described in this Section.

Assuming the same shot budget in each local thread, values of the checks (i.e. PCS ancilla measurements of 1) are used during post processing to filter error-corrupted outcomes from each local thread's measurement distribution. This process creates a vector of error-mitigated counts for each thread,  $c_i$ . Circuits that encounter more noise will discard a greater count of shots. Discarded shot count for each local thread is represented with the scalar  $r_{\text{pcs},i}$ . The percentage of discarded measurement outcomes for each thread,

$$d_{\text{pcs},i} = \frac{r_{\text{pcs},i}}{\text{shots}_i}, \quad (3)$$

gives insight into the underlying noise profile of the QC when different regions are compared. Each local  $d_i$  is then used as a weight that scales  $c_i$  to create  $s_i$ :

$$s_{\text{pcs},i} = c_i * \frac{\min(d_{\text{pcs}})}{d_{\text{pcs},i}}. \quad (4)$$

In Eqn. 4,  $\min(d_{\text{pcs}})$  represents the minimum percentage of discarded shots from all the parallel threads. A cumulative distribution,  $S_{\text{pcs}}$ , is created from the sum of the  $s_{\text{pcs},i}$  vectors from the PCS-protected local threads,

$$S_{\text{pcs}} = \sum s_{\text{pcs},i}. \quad (5)$$

*2) AFPC Weighted Distribution:* AFPC can also be used to steer quantum program outcomes so that higher-performing QPU regions are favored while creating a final, ensembled distribution. The structure of the AFPC does not include an ancilla that is measured for syndrome information - syndrome information is read out directly from the qubit targeted by Pauli checks. This causes the AFPC approach for a weighted distribution to differ from that of the PCS approach in two ways. First, the upper bound of threads that can explore a QPU in parallel increases since  $q_{\text{ancilla}} = 0$  in Eqn. 2. Second, a payload circuit embedded with AFPC only returns syndrome data in contrast with a PCS circuit that produces an error mitigated program result along with syndrome data for a local thread. Because of this, the AFPC approach also needs data from a baseline run of the target circuit (i.e. a circuit without Pauli checks) in each of the local threads. To accurately capture noise data, the baseline and AFPC circuits must utilize the same physical qubits and should only differ because of any gates added for the AFPC. The syndrome data resulting from AFPC creates a weighted distribution using the

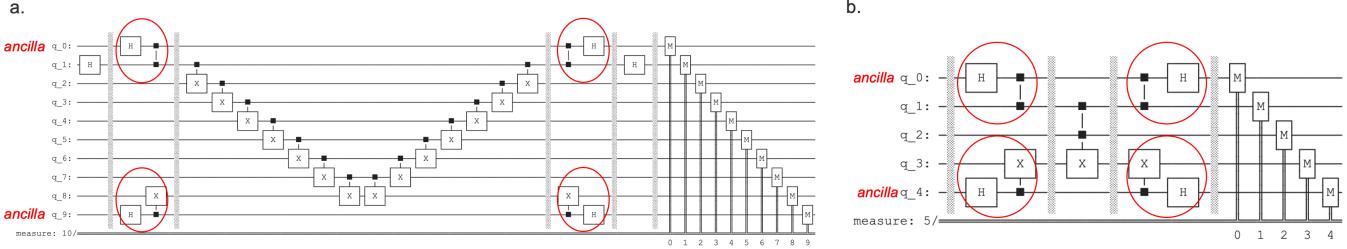


Fig. 5. Benchmarks used for analysis of methods included (a) an eight-qubit GHZ / sensing circuit and (b) the three-qubit Toffoli operation. PCS checks are circled in red. The portion of each circuit protected by the PCS checks,  $U$ , lie in between the circled checks. Two additional ancilla qubits were needed for the two single-qubit PCS checks in both (a) and (b). AFPC versions of these circuits are also possible by inserting a single-qubit gate on the protected qubit that prepares the eigenstate of the Pauli operator in the PCS controlled-Pauli. An example of this conversion is included in Fig. 3.

baseline circuit outcomes,  $b_i$ , from local threads. The value used to scale local outcomes is determined by the count of AFPC measured outcomes where any syndrome was equal to one,  $r_{af,i}$ . A percentage of syndrome-generating outcomes can be calculated:

$$d_{af,i} = \frac{r_{af,i}}{\text{shots}_i}, \quad (6)$$

which can then be used to create an AFPC weighted distribution,

$$S_{af} = \sum b_i * \frac{\min(d_{af})}{d_{af,i}}. \quad (7)$$

3) *AFPC+threshold*: The AFPC require two circuit evaluations: a baseline along with an AFPC injected equivalent. The check data can be used not only to assign weights to regions (i.e., regions with fewer detected errors receive higher weights in the combined distribution) but also to select which regions contribute to the combined distribution. This is done by ranking regions based on detected error rates and considering only the  $n$  regions with the lowest detected errors. For  $n = 1$ , the counts are taken exclusively from the single best region. For  $n > 1$ , Eq. 7 is applied over the top  $n$  regions. While the optimal number of regions to include cannot always be predicted beforehand, the experimental results in Section V-C of this paper indicate that taking the counts from only the top 1–3 regions typically yields the best performance for applications like QAOA.

4) *Hybrid Pauli Check Approach*: The primary benefit of the Pauli check techniques is that check data guides QC resource allocation rather than requiring backend-provided noise data for circuit mapping. However, the Pauli check techniques work synergistically with alternative quantum circuit optimizations that rely on noise data, if it is available. For example, `mapomatic` is a state-of-art tool that maps compiled quantum circuits to the best performing physical qubits of a QC according to device's calibration data [22]. While a powerful optimizer, the quality of layouts that `mapomatic` returns are only as good as the data used. Drift in quantum systems causes actual machine parameters to stray from their calibrated metrics [3], which will reduce the effectiveness of a `mapomatic` provided layout – what was once the best may

no longer be. However, Pauli checks can inform the quality of maps if multiple are returned from `mapomatic`. We propose in this approach, multiple `mapomatic` top picks could be generated and either the PCS or AFPC approaches could be applied to generate a PC +`mapomatic` weighted ensemble.

## V. EXPERIMENTAL RESULTS

In this Section, we apply the concepts from Sections III and IV to evaluate the effectiveness of Pauli checks for error mitigation, quantum device characterization, and steering the quantum circuit distribution. This Section includes the three different experiments designed to evaluate PCS and AFPC enabled performance benefits on quantum circuits. First, the Pauli check methods described in Section IV were evaluated in depolarizing noise channels. The second set of experiments targeted a QPU available in the IBM Quantum cloud in November 2024. The final experiment focused on the corrective benefits Pauli checks could provide to optimization problems.

The benchmarks used during evaluation represented meaningful tasks in quantum computing:

- **GHZ / Sensing** : The GHZ mirror circuit is used for evaluating quality of multi-qubit entanglement as well as in quantum sensing applications [33].
- **Toffoli**: The Toffoli gate is a fundamental operator in quantum computing applications (and reversible logic), especially in its generalized form.
- **QAOA**: The Quantum Approximate Optimization Algorithm is a near-term quantum algorithm that finds approximate solutions to optimization problems.

### A. Pauli Checks in Depolarizing Noise

We focus analysis on two circuits: GHZ and Toffoli. The GHZ circuit sizes include 2, 4, 6, and 8 qubits. The Toffoli gate has three qubits. Examples of the 8-qubit GHZ and 3-qubit Toffoli implementing PCS appear in Fig. 5. The translation from PCS to AFPC can be found in Fig. 3.

We are motivated to make best use of a QC with an unknown error profile at runtime though the application of Pauli checks to learn the noise landscape. Learned noise data then steer resource contributions to final circuit output distributions. To emulate this, we developed a noisy simulation implemented

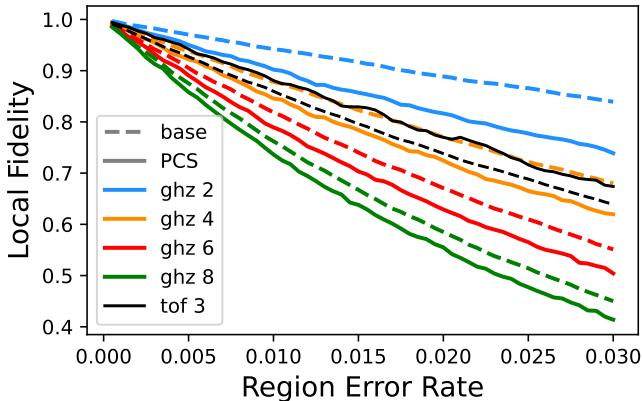


Fig. 6. Local fidelity vs. region error rate of simulated QPU. Each of the sixty regions had a unique value of  $p \in [0.0005, 0.03]$  (two-qubit gates  $2p$ ).

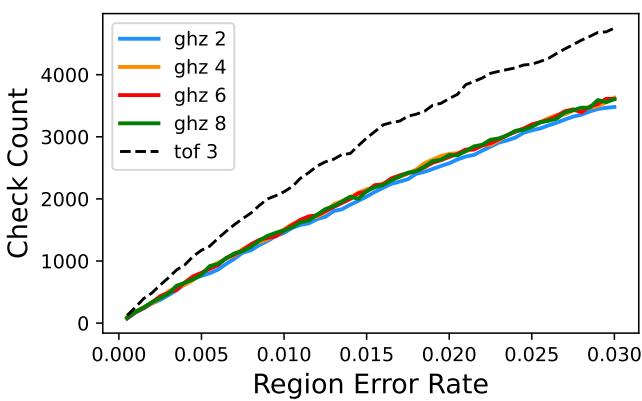


Fig. 7. PCS check counts vs. QPU region error rate. 10,000 shots total were used in each experiment.

with Qiskit Aer [12]. The test bench used during evaluation was an expansion of the model pictured in Fig. 1 to emulate a QC with 60 regions where single-qubit depolarizing noise  $p \in [0.0005, 0.03]$  and two-qubit gates  $2p$ . One instance of each application was allocated per region so that 60 threads of each benchmark are executed in parallel. The circuits pictured in Fig. 5 were transpiled to use the basis gate set that included  $CX$ ,  $X$ ,  $SX$ , and  $RZ$ .

Relationships between fidelity and PC check / syndrome data were studied first. For each benchmark, four variations were generated: baseline with no Pauli checks, PCS on edge qubits, AFPC on edge qubits, and AFPC on all qubits. This experiment assumes parallel execution of all circuits of the same type, where one circuit is allocated to one of the 60 QPU regions. 10,000 shots are used for each experiment.

First, we study error mitigated PCS vs. the baseline performance in all simulated regions. This is shown in Fig. 6 where dotted lines represent the no PCS baseline and solid lines represent PCS outcomes. We observe that while the Toffoli circuit reaps benefit from PCS within all regions, the GHZ outcomes suffer in all of the local (error mitigated) distributions. This

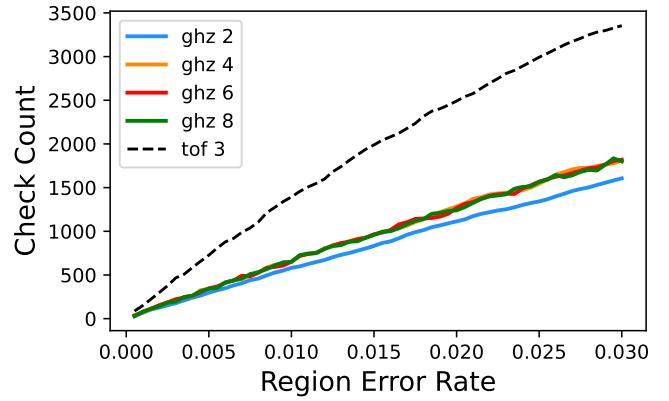


Fig. 8. AFPC (edges only) check counts vs. QPU region error rate. 10,000 shots total were used in each experiment.

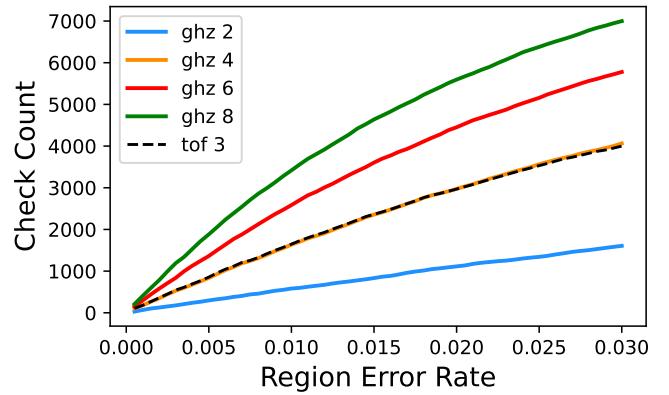


Fig. 9. AFPC (all qubits) check counts vs. QPU region error rate. 10,000 shots total were used in each experiment.

is a clear demonstration that the tradeoffs of PCS must be carefully considered – the error sensitivity of the circuit may not be able to withstand the effects of the additional 2q gate overhead. Despite PCS showing local fidelity degradation, we later apply the methods in Section IV and find that parallelized PCS circuit outcomes and check data can be combined to produce significant fidelity improvements in the intelligently ensembled final distribution on the depolarizing error model.

Next, we study the relationships between Pauli check syndrome counts and device error rate for all Toffoli and GHZ circuits. Fig. 7 reports PCS data, Fig. 8 reports AFPC placed only on the edge qubits (similar to PCS placement), and Fig. 9 reports AFPC when checks protect all qubits. These plots clearly show the relationships between syndrome data and region error. We highlight that Fig. 9 generally reports more checks counted for each circuit, which is an intuitive result as more of the circuit is protected by Pauli checks as compared to the circuits included in Figs. 7 and 8. The exceptions are the 3q Toffoli and 2q GHZ as their Pauli check protection remains minimally or completely unchanged going from edge to full AFPC.

To further instill confidence that Pauli checks can navigate

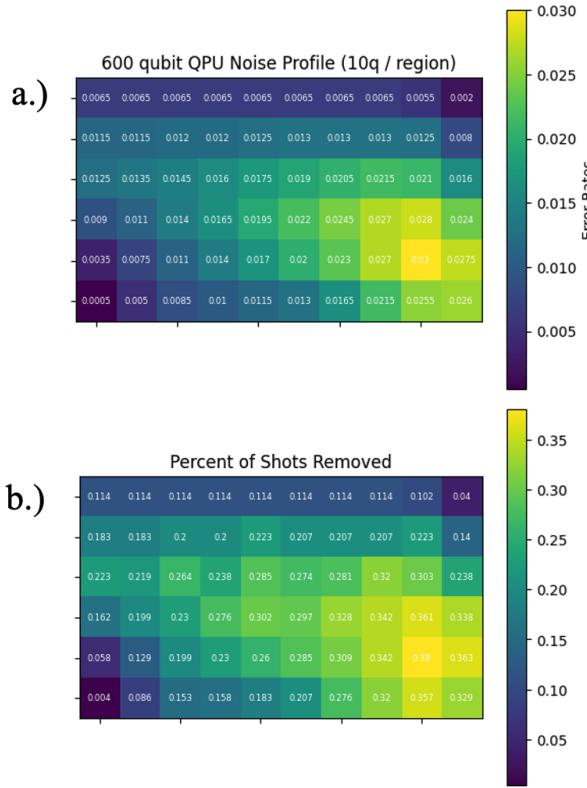


Fig. 10. (a) Heatmap of an example QC chip noise profile, where each square is a 10 qubit region of the chip characterized by depolarizing noise,  $p$ . (b) Heatmap of plotting the percentage of shots removed after running 10,000 shots on PCS-protected 8-qubit GHZ circuit (seen in Fig. 5) on each region. The underlying QC chip is the same as pictured in (a).

a potentially unknown quantum compute landscape, a QPU was initialized using random error rates. Once again, a 600 qubit processor with 60 regions was targeted. A heat map representing this device is pictured in Fig. 10(a). A 8-qubit GHZ circuit with PCS was executed in parallel for 10,000 shots, and the local percentage of shots removed in each region were used to generate a second heat map, Fig. 10(b). The resemblance between the gradients of Figs. 10(a) and 10(b) shows that the syndrome data of the PCS checks can provide an estimate of the underlying QPU error rates. This gives us further indication that Pauli checks can be used in conjunction with multi-programming to scope out the noise profile of a QC, pushing towards higher fidelity results for quantum applications while minimizing dependence on characterization data.

Fig. 11 shows the results for all circuits and Pauli check methods on the simulated QPU with a depolarizing noise model. The baseline comparison is generated from a basic ensemble of all QPU region outcomes. PCS and AFPC fidelities result from weighted distributions generated with the methods of Section IV. The weighted ensembles created from both the PCS and the AFPC edge approaches demonstrate the best average improvement,  $\sim 14\%$ .

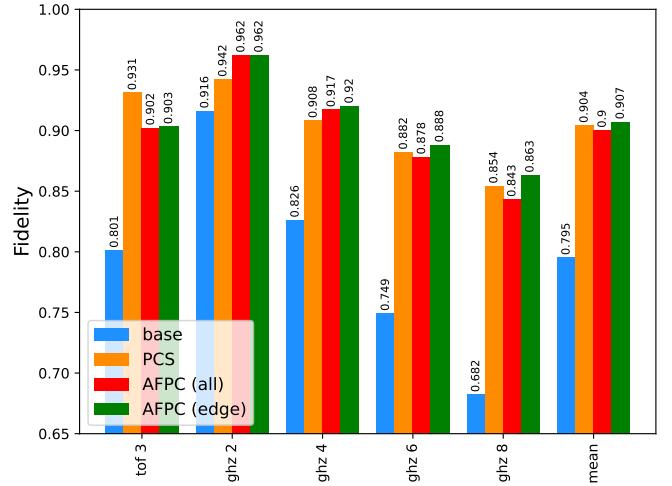


Fig. 11. Fidelity results across different Pauli check methods.

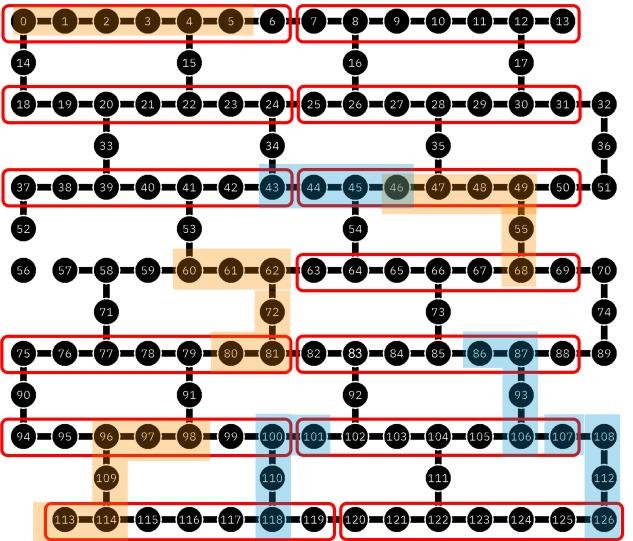


Fig. 12. Layout of IBM Sherbrooke QPU. Target QPU regions for four-qubit GHZ experiments are emphasized. Regions circled in red (13 total) were used for logical to physical qubit mapping for the noise-unaware parallelized GHZ. Regions highlighted with blue are Mapomatic-selected regions (top four) for the four-qubit GHZ. Regions highlighted with orange are Mapomatic-selected regions (top four) for the four-qubit GHZ circuits with PCS. These regions were produced by Mapomatic on 14 November 2024.

## B. IBM QPU Analysis

Pauli checks combined with a check-weighted distribution improved quantum circuit outcomes in a depolarizing noise channel. As a next form of evaluation, we tested PC and AFPC in a more irregular noise environment – Pauli checks were placed in circuits run on real quantum hardware. Our real QPU experiments targeted `ibm_sherbrooke` with the `ibm-q/open/main` instance that allows for 10 minutes of QC access per month. An image of this processor can be found in Fig. 12.

We chose the GHZ/sensing circuit as the benchmark (Fig. 5(a)) for real QC evaluation as the circuit structure scales

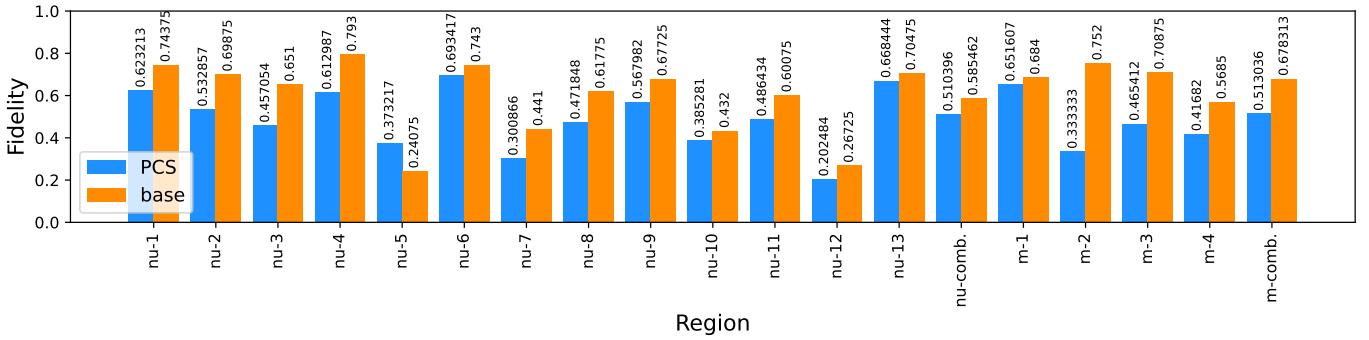


Fig. 13. Four-qubit GHZ baseline and PCS results for the regions described in Fig. 12. The 13 noise-unaware regions are indicated by ‘nu’ and the mapomatic-generated layouts are indicated by an ‘m’. The top mapomatic layout corresponds to ‘m-1’. Combined distributions are produced using the techniques outlined in Section IV-B.

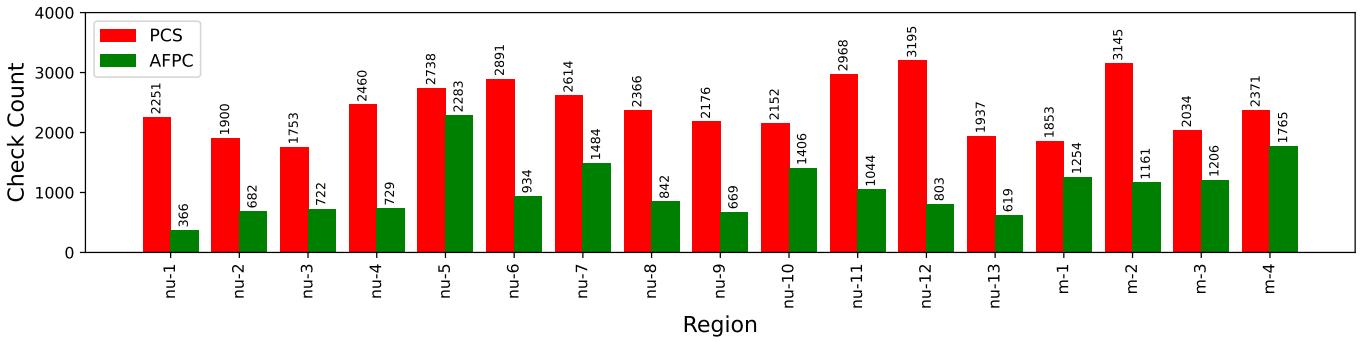


Fig. 14. Four-qubit GHZ PCS and AFPC check counts for the regions described in Fig. 12. The 13 noise-unaware regions are indicated by ‘nu’ and the mapomatic-generated layouts are indicated by an ‘m’. The top mapomatic layout corresponds to ‘m-1’.

easily. The baseline was scaled from 2 - 4 qubits. Each circuit size had three variations: baseline with no Pauli checks, PCS on edge qubits (adds two extra qubits for the required ancilla), and AFPC on all qubits. All circuits of the same type were executed using 4,000 shots.

We were interested in how well Pauli checks could navigate a QPU noise landscape whenever arbitrary mappings were selected. To test this, the QC was divided into 13 uniform regions, indicated in red in Fig. 12, so that we could evaluate the quality of the Pauli checks weighted distribution when benchmarks were scheduled cross-chip. For consistency, we used the same regions for all circuit scales in our cross-QPU tests. We note that a region connecting physical qubits 56-62 was omitted due to a break in the device graph. In this test, we evaluated a naive ensemble, PCS, AFPC, and AFPC+threshold. In the threshold case (described in Section IV-B3), the region with the lowest count of checks was sampled. We also compared our Pauli check approaches to two alternative error suppression approaches: noise aware mapping and noise aware ensemble. Noise aware mapping, referred to as ‘mapomatic top’ chose the best region of circuits for the baseline using the best scoring layout returned by mapomatic. Default options were selected during all mapomatic layout optimization procedures. Noise aware ensemble, referred to as ‘mapomatic 4 ensemble’ returned the

top four maps from mapomatic and created an ensembled distribution from those circuit outcomes. Four maps were chosen in this approach as it was the default setting in Ref. [27]. Finally, we implemented a hybrid, mapomatic-guided Pauli check approach where all the Pauli check methods previously mentioned were used only in the four regions selected by mapomatic. The Pauli check hybrid method is described in Section IV-B4.

First, we study the results from the four-qubit GHZ circuit. We examine the performance of the 13 regions of the noise unaware mapping to gain a better understanding of performance cross-chip. We also analyze the outcomes guided by mapomatic. The regions selected by mapomatic for the four qubit GHZ are found in Fig. 12 – the baseline and the PCS injected circuit are emphasized in blue and orange, respectively. The baseline and PCS fidelity results for each region are found in Fig. 13. These results show that the performance varies significantly cross-chip. When focused on the baseline, we observe that the top mapomatic pick does not have the highest fidelity – seven other mappings demonstrate better performance. Comparing the baseline to PCS, we see that the error overhead associated with the four additional two-qubit gates required for the Pauli checks outweigh the corrective potential of PCS error mitigation. This is consistent with the results we saw in the depolarizing channel for GHZ

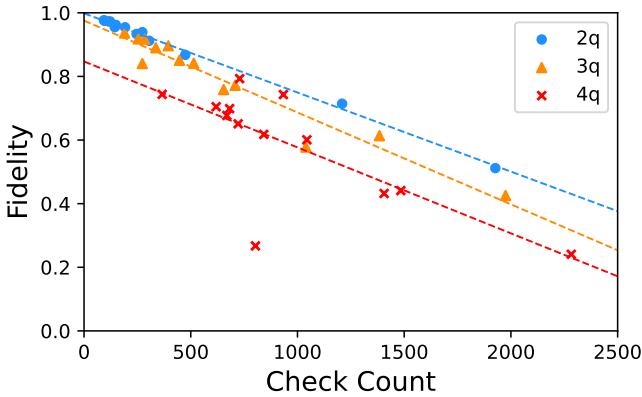


Fig. 15. QPU results for fidelity vs. AFPC check counts for 2-4 qubit GHZ circuits. Here, a clear relationship is seen between region performance and syndrome data from the AFPC circuits.

(Fig. 6). Fig. 14 shows the check counts for the PCS and AFPC circuits mapped to the 13 regions of Fig. 12 circled in red. We see in every case that check counts are lower for full AFPC as compared to PCS, which is opposite from our findings in Section V-A.

We find final fidelity results for each benchmark of each type in Table I. In this table, the highest fidelity approach for each circuit size is highlighted in orange while the second highest fidelity is highlighted in blue. Although some cases are within noise margins, technique on Pauli checks resulted in the highest fidelity in each case. We find that when the AFPC utilizes a threshold, it has a high probability of finding a quality mapping from a selection of provided maps. We note that as the circuit size gets larger, the Pauli check guided techniques are able to distance themselves from the baseline.

As a final analysis, we examine how well Pauli checks learn the error profile of a real quantum machine. We do this by comparing the AFPC check count with baseline circuit fidelity for the 13 arbitrary regions that span IBM Sherbrooke. This analysis is included in Fig. 15. Here we see a direct correlation between baseline fidelity and AFPC check count for all GHZ benchmarks. This result agrees with the relationships found in the depolarizing noise model study (Fig. 9).

#### C. Pauli Checks for Quantum Approximate Optimization

In this Section, we present results for a realistic application circuit executed on the FakeBrisbane mock backend provided by Qiskit. We focus on the hardware-efficient Quantum Approximate Optimization Algorithm (QAOA) ansatz, as described in [19], and use pre-optimized parameters that were obtained under a noiseless setting. The circuits evaluated are of size 8 and 10 qubits with depths ( $p$ ) of 1 and 2. Two checks are applied to the QAOA circuit: one targeting the controlled-X gate on the bottom-edge qubit and another targeting a controlled-X gate on one of the center qubits. The circuit is then converted to its AFPC variant, which is used to calibrate the error rates of the candidate regions. An example of the pre-optimized 8-qubit QAOA circuit with AFPC is shown

in Fig. 16. It is important to note that better performance is expected with more checks. However, these experiments demonstrate that even a small number of checks can provide significant performance boosts, even for larger systems. This is particularly valuable because identifying suitable checks becomes increasingly challenging as system size grows [8].

We compare two methods for evaluating performance improvement over the baseline: AFPC and AFPC + threshold. In the AFPC + threshold approach, we evaluate the top 1, 2, and 3 regions, where the counts for the top 2 and top 3 regions are combined using the weighted distribution in Eq. 7. The PCS ensemble approach is excluded from this comparison, as it consistently underperformed relative to the baseline under the assumed noise model. This was likely due to the noise overhead introduced by the SWAP operations when mapping the checks from PCS to the physical device qubits if algorithm qubits protected by a check could not be placed next to an ancilla qubit.

Fig. 17 presents the results for the various QAOA circuits. The initial mappings we used were selected in a similar manner as for the IBM Sherbrooke backend detailed in the previous subsection. Across all test circuits, we observe substantial improvements in fidelities when comparing the top region(s) selected by AFPC to the baseline circuit ensemble. For the 8-qubit circuits, the average fidelity improvement is approximately 27%, while for the 10-qubit circuits, fidelity improves by approximately 33%. Also, we can see that the best performance is sometimes achieved using only the top region, whereas in other cases, combining the top 2 or 3 regions results in greater improvements (see the fidelities for QAOA with  $n = 10$  in Figure 17).

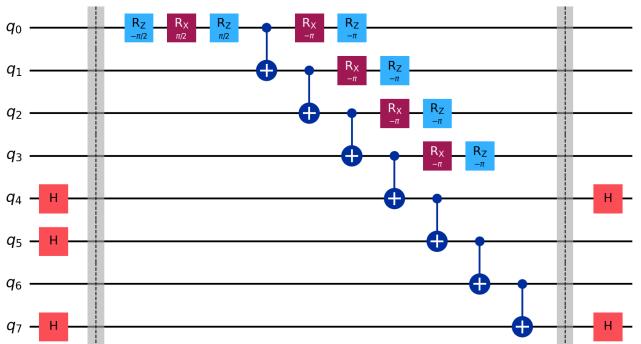


Fig. 16. Quantum circuit diagram of the hardware-efficient QAOA ansatz with AFPC. The checks are indicated by the Hadamard gates positioned outside the barriers.

## VI. DISCUSSION AND FUTURE DIRECTIONS

Our experiments found that while PCS ensembling could result in performance improvements, tradeoffs with gate error needed to be carefully considered. In noisy environments, like IBM Sherbrooke or the FakeBrisbane mock backend, PCS was not helpful. On the other hand, AFPC was shown to have more uniform performance. An important contribution

Qubits	13 Regions (noise-unaware)				Mapomatic-guided PCs				Mapo. Top	Mapo. 4 Ens.
	Base Ens.	PCS	AFPC	AFPC+Thr.	Base Ens.	PCS	AFPC	AFPC+Thr.		
2	0.8958	0.9042	0.9532	0.9765	0.9767	0.9607	0.9771	0.9783	0.9783	0.9767
3	0.7876	0.7575	0.8580	0.9365	0.9219	0.7035	0.9236	0.9218	0.9133	0.9219
4	0.5855	0.5104	0.6328	0.7438	0.6783	0.5130	0.6884	0.7520	0.6840	0.6783

TABLE I

FIDELITY RESULTS OF GHZ BENCHMARK ON IBM QC SHERBROOKE. PC TECHNIQUES (**BOLD**) ARE COMPARED TO THE BASELINES OF ENSEMBLE AND MAPOMATIC APPROACHES. AT EACH CIRCUIT SIZE, HIGHEST FIDELITY IS INDICATED IN **ORANGE** AND SECOND HIGHEST FIDELITY IS INDICATED IN **BLUE**. WE FIND THAT WHEN THE AFPC UTILIZES A THRESHOLD, IT HAS A HIGH PROBABILITY OF FINDING A QUALITY MAPPING FROM A SELECTION OF PROVIDED MAPS.

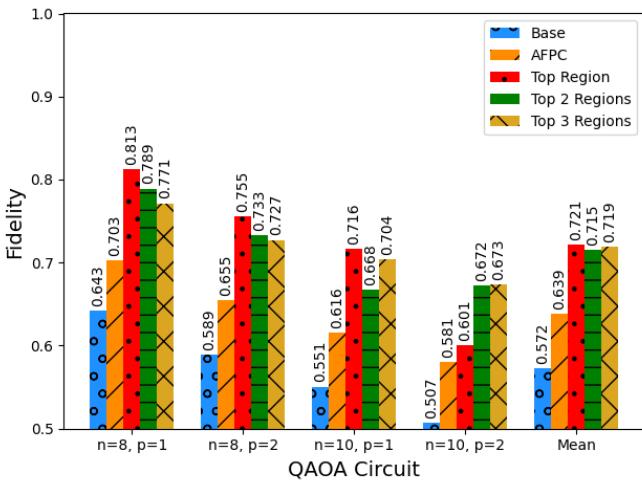


Fig. 17. Fidelities for each AFPC method across various QAOA circuits. ‘Mean’ represents the average fidelities across all four circuit instances.

of this work is highlighting the ability of AFPC to coarsely calibrate error rates across QPU regions with minimal resource overhead. While identifying suitable Pauli checks can become challenging for larger circuits, the AFPC method introduces only single-qubit gates, which add relatively little error to the circuit. For very complicated circuits (where Pauli checks cannot be found), sub-circuits instead of the full circuit can be examined to still learn noise that matters to a quantum application. Despite this low overhead, AFPC was able to accurately identify the best-performing regions, resulting in substantial fidelity improvements for circuits as large as 10 qubits, requiring only two single-qubit checks. This efficiency makes AFPC a practical and scalable solution for error mitigation and hardware optimization in near-term devices.

Another key advantage of Pauli checks is their flexibility in quantum circuit mapping. As demonstrated in Section V-B, Pauli checks, when leveraged in a hybrid approach, can be combined with prior knowledge or noise distribution estimates to enhance performance. For example, we used mapomatic to identify approximate optimal regions, which were further refined with Pauli checks to achieve significant improvements in fidelity. This integration highlights the potential for combining hardware-level noise characterization tools with PCS to optimize qubit mappings. An area for future work would be to explore other quantum circuit optimization techniques that work synergistically with Pauli checks.

A potential method for future exploration is a dynamic shot allocation approach. In this method, given a shot budget, shots are incrementally sent to each region of the chip. For each region, errors are monitored in real time (by either the PCS or AFPC circuit), and once the number of discarded shots exceeds a specified threshold, no additional shots are allocated to that region. The remaining shot budget is then gradually focused towards regions with lower error rates, allowing resources to be dynamically steered toward the best-performing regions on chip. This approach contrasts with the Pauli check methods demonstrated in this work, where a fixed number of shots are allocated per region to estimate the least noisy regions. While dynamic shot allocation offers the potential for more efficient resource usage, its effectiveness depends on accurately selecting the discard threshold, as this parameter influences the likelihood of identifying the best regions. We anticipate that the optimal threshold will vary across devices and applications, and future research should focus on identifying domain-specific thresholds to ensure consistent and reliable results.

## VII. CONCLUSION

This work highlights the versatility and potential of Pauli checks in improving quantum program performance, particularly in the resource-constrained environments of current quantum devices. By combining Pauli checks with multi-programming, we present a framework that not only mitigates errors in individual circuit distributions but also enables intelligent resource allocation and region selection across QPUs.

Beyond their role in error mitigation, Pauli checks can guide efficient qubit mapping and optimize hardware utilization, even in scenarios with little to no prior knowledge of the noise distribution. This capability for “noise-unaware” mapping is especially valuable for real-time hardware characterization or when operating under a strict shot budget. Looking forward, expanding these methods to include dynamic resource allocation strategies, such as the proposed dynamic shot allocation, could further improve resource efficiency and performance. Additionally, extending these approaches to more complex quantum algorithms beyond QAOA and scaling them to larger systems represents an exciting avenue for future research, paving the way for scalable, efficient, and easily adoptable quantum computing solutions.

## VIII. APPENDIX

Proof of Theorem 1.

*Proof.* We denote the arbitrary input density matrix as  $\rho$ . First, consider a single pair of ideal checks sandwiching a noisy circuit  $U$ . Let the noisy channel be  $\mathcal{E}(\psi) = \sum_i E_i \psi E_i^\dagger$ . The postselection probability is [8]

$$p = \frac{1}{4} \sum_i \text{tr}([RE_i R + E_i]U\rho U^\dagger[RE_i^\dagger R + E_i^\dagger]). \quad (8)$$

Let  $E'_i = \frac{RE_i R + E_i}{2}$ . Then Eq. (8) can be written as

$$p = \text{tr} \left( U\rho U^\dagger \sum_i E'^\dagger E'_i \right). \quad (9)$$

Expand  $E_i$  in the +1 Pauli basis as  $E_i = \sum_j a_{ij} P_j$ . Notice that the Paulis that anticommute with  $R$  vanish. Let the set of +1 Paulis that commute with  $R$  be indexed by the set  $S$ . Thus  $E'_i = \sum_{j \in S} a_{ij} P_j$ . From direct calculation

$$\sum_i E'^\dagger E_i = \sum_{ij \in S, k \in S} a_{ij}^* P_j a_{ik} P_k \quad (10)$$

$$= \sum_i \left( \sum_{j \neq k | j \in S, k \in S} a_{ij}^* a_{ik} P_j P_k + \sum_{j=k \in S} |a_{ij}|^2 I \right). \quad (11)$$

Finally, if  $\sum_{j \neq k | j \in S, k \in S} a_{ij}^* a_{ik} P_j P_k = 0$ , we have that  $\sum_i E'^\dagger E_i = \sum_{j \in S} |a_{ij}|^2 I$ . From substitution into Eq. (9), we have that  $p = \sum_{j \in S} |a_{ij}|^2 \forall \rho$ . Pauli channels have the property that

$\sum_{j \neq k | j \in S, k \in S} a_{ij}^* a_{ik} P_j P_k = 0$  since  $E_i = a_i P_i$ . The extension to multiple checks follows.  $\square$

## IX. ACKNOWLEDGMENTS

JL, AG, and ZHS acknowledge support by the Q-NEXT Center. NH was partially supported by NSF CCF-2119069. The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory (“Argonne”). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan. <http://energy.gov/downloads/doe-public-access-plan>.

## REFERENCES

- [1] Piotr Czarnik, Andrew Arrasmith, Patrick J Coles, and Lukasz Cincio. Error mitigation with clifford quantum-circuit data. *Quantum*, 5:592, 2021.
- [2] Poulami Das, Swamit S Tannu, Prashant J Nair, and Moinuddin Qureshi. A case for multi-programming quantum computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 291–303, 2019.
- [3] Samudra Dasgupta and Travis S Humble. Stability of noisy quantum computing devices. *arXiv preprint arXiv:2105.09472*, 2021.
- [4] Dripto M. Debroy and Kenneth R. Brown. Extended flag gadgets for low-overhead circuit verification. *Physical Review A*, 102(5), November 2020.
- [5] Yongshan Ding and Frederic T Chong. *Quantum computer systems: Research for noisy intermediate-scale quantum computers*. Springer Nature, 2022.
- [6] Joseph Emerson, Robert Alicki, and Karol Źyczkowski. Scalable noise estimation with random unitary operators. *Journal of Optics B: Quantum and Semiclassical Optics*, 7(10):S347–S352, September 2005.
- [7] Tudor Giurgica-Tiron, Yousef Hindy, Ryan LaRose, Andrea Mari, and William J Zeng. Digital zero noise extrapolation for quantum error mitigation. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 306–316. IEEE, 2020.
- [8] Alvin Gonzales, Ruslan Shaydulin, Zain H Saleem, and Martin Suchara. Quantum error mitigation by pauli check sandwiching. *Scientific Reports*, 13(1):2122, 2023.
- [9] Daniel Gottesman. Stabilizer codes and quantum error correction, 1997.
- [10] Daniel Gottesman. The heisenberg representation of quantum computers, 1998.
- [11] J. Helsen, I. Roth, E. Onorati, A.H. Werner, and J. Eisert. General framework for randomized benchmarking. *PRX Quantum*, 3(2), June 2022.
- [12] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024.
- [13] Yanjun Ji, Sebastian Brandhofer, and Ilia Polian. Calibration-aware transpilation for variational quantum optimization. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, page 204–214. IEEE, September 2022.
- [14] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland. Randomized benchmarking of quantum gates. *Physical Review A*, 77(1), January 2008.
- [15] Quinn Langfitt, Ji Liu, Benchen Huang, Alvin Gonzales, Kaitlin N Smith, Nikos Hardavellas, and Zain H Saleem. Pauli check extrapolation for quantum error mitigation. *arXiv preprint arXiv:2406.14759*, 2024.
- [16] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for nisq-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1014, 2019.
- [17] P. Li, J. Liu, A. Gonzales, Z. Saleem, H. Zhou, and P. Hovland. Outracer: Mitigating quantum gate and measurement errors by tracing subsets of qubits. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 103–117, 2024.
- [18] Shubdeep Mohapatra and Huiyang Zhou. Understanding error sensitivity of quantum circuits. In *IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2024.
- [19] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Müller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 3(3):030503, jun 2018.
- [20] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems*, pages 1015–1029, 2019.
- [21] Paul D Nation, Hwajung Kang, Neereja Sundaresan, and Jay M Gambetta. Scalable mitigation of measurement errors on quantum computers. *PRX Quantum*, 2(4):040326, 2021.
- [22] Paul D. Nation and Matthew Treinish. Suppressing quantum circuit errors due to system variability. *PRX Quantum*, 4:010327, Mar 2023.
- [23] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.
- [24] Timothy Proctor, Kenneth Rudinger, Kevin Young, Mohan Sarovar, and Robin Blume-Kohout. What randomized benchmarking actually measures. *Phys. Rev. Lett.*, 119:130502, Sep 2017.
- [25] Gokul Subramanian Ravi, Jonathan M Baker, Kaitlin N Smith, Nathan Earnest, Ali Javadi-Abhari, and Frederic T Chong. Quancorde: Boosting fidelity with quantum canary ordered diverse ensembles. In *2022 IEEE International Conference on Rebooting Computing (ICRC)*, pages 66–77. IEEE, 2022.
- [26] Gokul Subramanian Ravi, Kaitlin N Smith, Pranav Gokhale, and Frederic T Chong. Quantum computing in the cloud: Analyzing job and

- machine characteristics. In *2021 IEEE International Symposium on Workload Characterization (IISWC)*, pages 39–50. IEEE, 2021.
- [27] Swamit S Tannu and Moinuddin Qureshi. Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 253–265, 2019.
  - [28] Swamit S Tannu and Moinuddin K Qureshi. 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*, pages 987–999, 2019.
  - [29] Miroslav Urbanek, Benjamin Nachman, Vincent R. Pascuzzi, Andre He, Christian W. Bauer, and Wibe A. de Jong. Mitigating depolarizing noise on quantum computers with noise-estimation circuits. *Phys. Rev. Lett.*, 127:270502, Dec 2021.
  - [30] Ewout van den Berg, Sergey Bravyi, Jay M Gambetta, Petar Jurcevic, Dmitri Maslov, and Kristan Temme. Single-shot error mitigation by coherent pauli checks. *Physical Review Research*, 5(3):033193, 2023.
  - [31] Joel J. Wallman and Joseph Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Physical Review A*, 94(5), November 2016.
  - [32] Jean-Baptiste Waring, Christophe Pere, and Sébastien Le Beux. Noise aware utility optimization of nisq devices, 2024.
  - [33] Ken X Wei, Isaac Lauer, Srikanth Srinivasan, Neereja Sundaresan, Douglas T McClure, David Toyli, David C McKay, Jay M Gambetta, and Sarah Sheldon. Verifying multipartite entangled greenberger-horne-zeilinger states via multiple quantum coherences. *Physical Review A*, 101(3):032343, 2020.
  - [34] Wenjie Wu, Yiqian Wang, Ge Yan, Yuming Zhao, and Junchi Yan. On reducing the execution latency of superconducting quantum processors via quantum program scheduling. *arXiv preprint arXiv:2404.07882*, 2024.