HAMMER: Boosting Fidelity of Noisy Quantum Circuits by Exploiting Hamming Behavior of Erroneous Outcomes

Swamit Tannu*
University of Wisconsin
Madison, USA

Poulami Das
Georgia Tech
Atlanta, USA

Ramin Ayanzadeh
Georgia Tech
Atlanta, USA

Moinuddin Qureshi
Georgia Tech
Atlanta, USA

ABSTRACT

Quantum computers with hundreds of qubits will be available soon. Unfortunately, high device error-rates pose a significant challenge in using these near-term quantum systems to power real-world applications. Executing a program on existing quantum systems generates both correct and incorrect outcomes, but often, the output distribution is too noisy to distinguish between them. In this paper, we show that erroneous outcomes are not arbitrary but exhibit a well-defined structure when represented in the Hamming space. Our experiments on IBM and Google quantum computers show that the most frequent erroneous outcomes are more likely to be close in the Hamming space to the correct outcome. We exploit this behavior to improve the ability to infer the correct outcome.

We propose *Hamming Reconstruction (HAMMER)*, a post-processing technique that leverages the observation of Hamming behavior to reconstruct the noisy output distribution, such that the resulting distribution has higher fidelity. We evaluate HAMMER using experimental data from Google and IBM quantum computers with more than 500 unique quantum circuits and obtain an average improvement of 1.37x in the quality of solution. On Google's publicly available QAOA datasets, we show that HAMMER sharpens the gradients on the cost function landscape.

CCS CONCEPTS

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

KEYWORDS

Quantum Computing, NISQ, Quantum Compilers

ACM Reference Format:
Swamit Tannu, Poulami Das, Ramin Ayanzadeh, and Moinuddin Qureshi. 2022. HAMMER: Boosting Fidelity of Noisy Quantum Circuits by Exploiting Hamming Behavior of Erroneous Outcomes. In *27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '22), February 28-March 4, 2022, Lausanne, Switzerland*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3503222.3507703

*The corresponding author can be be reached at stannu@wisc.edu.

ASPLOS '22, February 28-March 4, 2022, Lausanne, Switzerland

ACM ISBN 978-1-4503-9205-1/22/02…$15.00
https://doi.org/10.1145/3503222.3507703

1 INTRODUCTION

Quantum computers with few tens of quantum bits (qubits) are currently available to users, and machines with several hundred qubits are expected within the next few years [22]. These machines can accelerate certain optimization problems [16], molecular simulations [25], and machine learning [6] applications. While their prospective looks promising, near-term quantum computers are noisy and prone to device errors, which severely limits the fidelity of applications. Although the number of qubits on quantum computers has notably increased in recent years, the average error rate of quantum operations has not reduced at the same pace. For example, the average two-qubit operational error rate on existing quantum computers from IBM and Google continues to be in the range of 1% to 2% [1, 11]. Such large error rates limit the number of operations that can be performed reliably. As quantum machines are unlikely to have enough resources to support error correction in the near term, such machines are typically operated in the *Noisy Intermediate-Scale Quantum (NISQ)* [36] mode, whereby the program is executed several thousand times, and each trial can produce a correct or an erroneous outcome. The application fidelity on a NISQ machine is determined by the ability to identify the correct answer from the outcomes produced during all the trials.

To improve the fidelity of applications on NISQ hardware, recent works propose compiler techniques that reduce the number of quantum operations (gates) [3, 24, 29, 38, 39, 45], perform error-aware computations [15, 26, 44], focus on reducing specific sources of errors, such as measurement errors [14, 17, 21, 23, 43], idling errors [13, 40], and crosstalk [27]. Recent works [34, 42] have also looked at the problem of mitigating correlated errors, where a particular incorrect outcome can occur with a high probability. The implicit assumption in all these prior approaches is that the erroneous outcomes do not provide any meaningful information.

Even with intelligent compilation techniques, NISQ machines produce incorrect or sub-optimal outcomes for a significant fraction of the trials. In this paper, we propose an orthogonal approach to improve the fidelity of NISQ programs. Rather than simply treating the erroneous outcomes as wasteful, we propose to leverage correlation in such erroneous outcomes. In particular, we try to address the following two questions in this paper:

(1) When a trial produces a wrong outcome, is the produced bitstring arbitrary, or does it have a specific structure?
(2) If erroneous outcomes have some structure, can we exploit that structure to improve the quality of the solution?

To understand the behavior of the incorrect outcomes produced during program execution, we analyze experimental data from three different IBM quantum computers and publicly available datasets from Google with more than 500 representative circuits containing

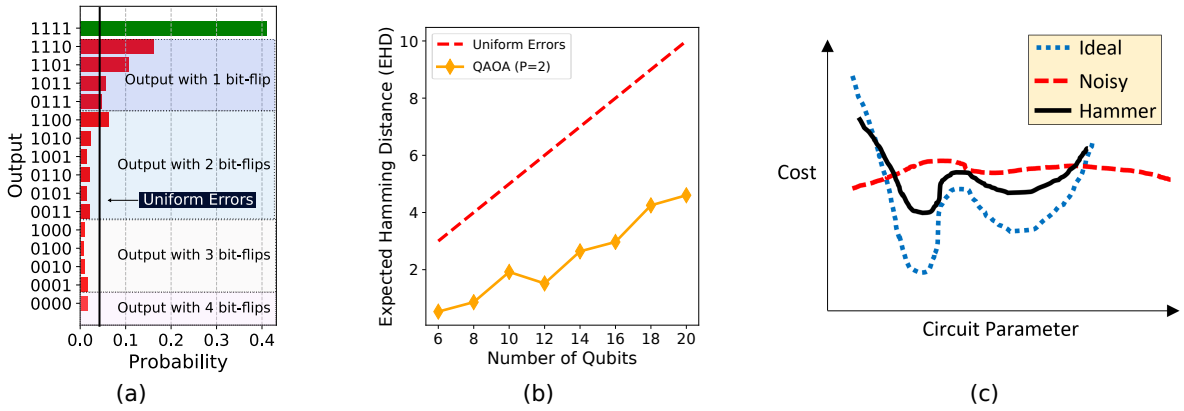


Figure 1: (a) Histogram of the output distribution for a 4-qubit Bernstein-Vazirani circuit. (b) Trend in the Expected Hamming Distance in the output distribution for QAOA circuits, run on IBM-Paris. (c) Cost landscape of a Variational Quantum Circuit.

up to 20 qubits. If the incorrect outcomes did not have any structure, we would expect all possible incorrect outcomes to occur with close to uniform probability. However, the incorrect outcomes have a well-defined structure in the Hamming space, as the incorrect outcomes tend to be within a short Hamming distance from the correct or optimal answers. For example, Figure 1(a) shows the output histogram of a four-qubit Bernstein-Vazirani (BV) circuit in which the error-free output "1111" appears with only 40% probability. We observe that the most frequent incorrect outcomes are close to the correct output in the Hamming space.

We observe a similar structure in the output of variational quantum programs such as *Quantum Approximate Optimization Algorithm (*QAOA*)*. To understand the structure of errors in Hamming space, we compute the *Expected Hamming Distance (EHD)*, which is a weighted average of the Hamming distances between correct and incorrect outcomes. EHD captures the density of outcomes in the Hamming space. If the erroneous outcomes produced are arbitrary for an n-qubit program, then we would expect that the incorrect bitstring will be $\frac{n}{2}$ bits away from the correct outcomes on an average. Alternately, when outcomes with high probabilities are clustered around the correct answers, the EHD would approach to zero. Figure 1(b) shows the EHD for QAOA circuits with two layers ($p = 2$) [16] as the size of the circuit is increased from 5 qubits to 20 qubits. We observe that although the EHD increases with the number of qubits, it increases at a much slower pace compared to a uniform-error model. Thus, incorrect outcomes are not arbitrary but exhibit a well-defined structure in the Hamming space.

As erroneous outcomes are not arbitrary, it is possible to leverage the structure they exhibit in the Hamming space. To this end, we propose *Hamming Reconstruction (*HAMMER*)*, a post-processing technique for boosting the fidelity of NISQ applications. Instead of relying solely on the probabilities associated with each outcome, HAMMER analyzes their neighborhoods in the Hamming space. More specifically, by utilizing the probabilities of individual outcomes and their structure in Hamming space, HAMMER provides more accurate estimates of the *likelihood* of each outcome. By using this likelihood function, HAMMER boosts the outcomes that are likely to be correct, while "hammering" down those that are potentially incorrect.

We observe that despite the probability of obtaining the correct bitstring being low for large quantum circuits, the correct answer has a rich neighborhood, HAMMER leverage this insight to compute a Neighborhood Score for every outcome string. The neighborhood Score is computed by obtaining a weighted sum of all the strings that are k Hamming distance away. Therefore, HAMMER uses a consensus from the individual Hamming scores of all the outcomes and generates a *weight* for evaluating the neighborhood at each Hamming distance. By aggregating these weights, HAMMER determines the final Neighbourhood Score for each outcome that is used in conjunction with its probability to estimate the likelihood of the outcome. As HAMMER uses the knowledge of the neighborhood, it boosts the probabilities of the correct outcomes while diminishing the probabilities of the incorrect ones.

Not only can HAMMER improve the fidelity of applications that produce a single correct output, but also of near-term variational quantum algorithms. For example, Figure 1(c) shows the landscape of the cost function of a QAOA circuit, as the circuit parameters are tuned. Unfortunately, due to high error rates, a significant fraction of the outcomes in the output distribution are sub-optimal. Therefore, the expected cost becomes insensitive to changes in the circuit parameters, which impedes the search for the optimal solution. As HAMMER builds the consensus using Hamming Spectrum, it amplifies the probabilities of the outcomes with Hamming structure and attenuates the spurious solutions to improve the average quality of solutions, thus helping the search for high-quality solutions.

Overall, our paper makes the following contributions:

(1) To the best of our knowledge, this is the first paper to demonstrate that incorrect outcomes produced on NISQ machines are not arbitrary but manifest a well-defined structure in the Hamming space. In particular, incorrect outcomes tend to be at a short Hamming distance from the correct outcome.
(2) We propose HAMMER, a post-processing technique that exploits the Hamming structure of incorrect outcomes to boost the likelihood of the correct outcomes.
(3) We evaluate HAMMER on Google and IBM machines with more than 500 circuits and show that HAMMER provides a consistent improvement in the average quality of solutions.

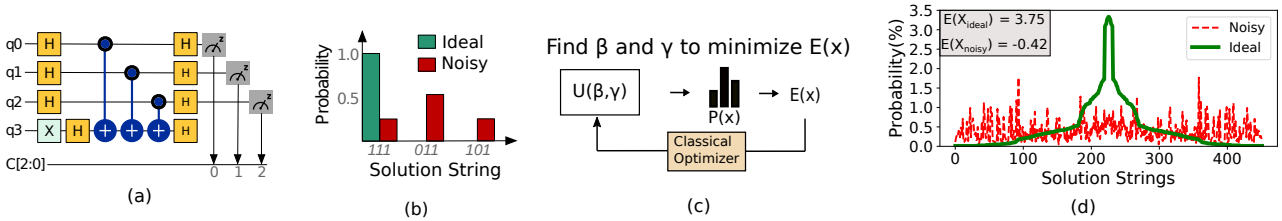


Figure 2: (a) Example of a 3-qubit Bernstein-Vazirani circuit. (b) Output on ideal (noise-free) and noisy NISQ hardware. (c) Variational Quantum Approximate Optimization Algorithm (QAOA). (d) Ideal and real output of a QAOA-9 circuit on IBM-Paris.

2 BACKGROUND

2.1 NISQ Paradigm

Existing qubit devices are vulnerable to noise. The average gate error rates range from 0.1% to 2% on current IBM and Google quantum computers. Such a high error rate limits the size of the largest quantum circuit that can be executed as the probability of encountering errors during computations increases with the number of operations. To run practical quantum applications, we need to protect against hardware errors. Unfortunately, quantum error correction codes incur large overheads requiring thousands of physical qubits. Thus, error correction on near-term systems with a few hundreds of qubits is not plausible. In the near-term, we have to operate quantum computers without error correction. Although error-prone, these *Noisy Intermediate Scale Quantum (NISQ)* [36] computers are still considered promising for very specific applications [16, 25]. Consequently, there is an active effort in devising algorithms, hardware (building better devices and gates), and software (compiler and post-processing) solutions to reduce the impact of errors.

2.2 Impact of Noise on Quantum Circuits

NISQ machines are vulnerable to errors and often produce incorrect outcomes. For example, Figure 2(a) shows a 3-qubit Bernstein Vazirani (BV) circuit that encodes the secret key "*111*". Ideally, this circuit should produce the output in a single query on a quantum computer. However, due to hardware errors, quantum computers produce incorrect outcomes "*011*" and "*101*" in addition to the correct outcome "*111*". Executing a quantum program on noisy hardware produces a large number of incorrect outcomes, along with the correct ones. Such incorrect bitstrings may occur with a very high probability, and the output distribution can become too noisy that the correct outcome is indistinguishable from the incorrect ones.

2.3 Variational Quantum Algorithms

Despite the vulnerability of NISQ machines to hardware errors, we can still use these machines for solving a class of optimization problems using variational quantum algorithms (VQA), which are robust to a certain types of errors. VQAs use a parametric circuit and search iteratively for the circuit parameters that produce high-quality solutions. For example, when solving an optimization problem using QAOA, as illustrated in Figure 2(c), we search for circuit parameters β and γ using a two-step process. First, we initialize β and γ with the best-known value and execute a quantum circuit several thousands of times. This yields a distribution of solution strings, where each solution has a fixed cost. Our objective is to find the solution string with the lowest cost. Next, we perform a second step, in which we compute the average (or expected) cost corresponding to the output distribution and search for optimal β and γ using expected cost as the objective function.

Unfortunately, high error rates of NISQ hardware disrupt the variational loop as the output distributions can be extremely noisy. Figure 2(d) shows an ideal distribution and the output on an IBMQ machine for a QAOA-9 benchmark. Due to the high error rate, a significant fraction of outcomes are solutions with suboptimal costs. These suboptimal outcomes result in inaccurate estimation of the expected cost on NISQ devices. For example, ideally the expected cost should be E(x) = 3.75 but in reality, it is E(x)=-0.42, as shown in Figure 2(d). Moreover, due to increasing noise, the expected cost becomes insensitive to changes in β and γ and the cost function landscape plateaus. This makes optimization problems at practical scales beyond the reach of QAOA on existing NISQ hardware.

2.4 Improving Quality of Solution on NISQ

Existing error-mitigation techniques focus on providing better than worst-case reliability on NISQ machines. The implicit assumption in these techniques is that erroneous outcomes may have no meaningful information to determine the correct answer. This would be true if the values produced by the erroneous trials were arbitrary – with a uniform probability over all possible incorrect answers. However, if the incorrect values have some correlation with the correct answer, then we can analyze the incorrect values to determine the correct answer.

3 HAMMING BEHAVIOR OF ERRORS

3.1 Is there a Structure in Errors?

To understand the structure in errors, we run a GHZ-10 circuit. On an ideal (error-free) quantum computer, the output state is an equal superposition of the all-zero and all-one states. However, on the IBM quantum computer, hardware errors produce incorrect outcomes along with the two desired states. In the case of the GHZ-10 circuit, we observe that correct outcomes occur with a cumulative probability of 45%, while 55% is the collective probability of all the incorrect outcomes. The incorrect outcomes are not random as the dominant incorrect outcomes that appear with high frequency are close to the correct answers in Hamming space. We also observe that majority of the dominant incorrect outcomes are within a Hamming distance of two from either correct answer.

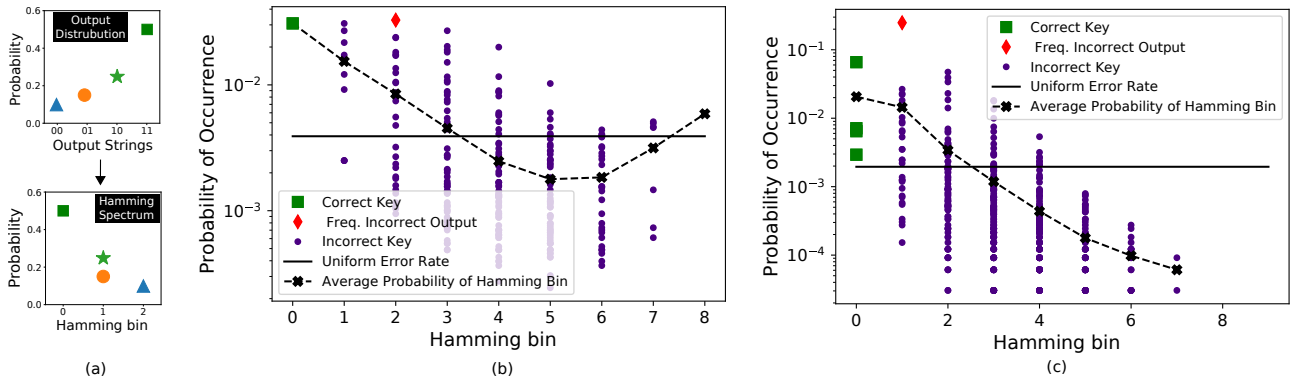


Figure 3: (a) Representing output probability distribution as Hamming spectrum (illustrative example). (b) Hamming Spectrum of a BV-8 circuit (executed on IBM-Manhattan). (c) Hamming Spectrum of a QAOA-8 circuit (executed on IBM-Manhattan).

The Hamming behavior of erroneous outcomes is not unique to GHZ. We observe a similar pattern in errors by analyzing data from more than 1500 quantum circuits executed on IBM and Google quantum hardware. These circuits capture diverse trends in the number of gates, circuit depth, degree of entanglement, and measurement basis. A detailed analysis on how these factors impact the Hamming structure is presented in the Section 7.

3.2 Hamming Spectrum

To visualize the structure in errors, we illustrate the output distribution in the form of a *Hamming spectrum*. Hamming spectrum creates a compact representation of the output probability distribution by bucketing each outcome into Hamming bins, as shown in Figure 3(a). Every string outcome in the output distribution is added to the K^{th} bin, where K is the Hamming distance between the correct answer(s) and the string. For a quantum circuit with an N-bit output, K ranges between 0 to N. Figure 3(b) shows the Hamming spectrum of BV-8 output, where correct output is the all-one state ("11111111"), and rest are erroneous outcomes. In the Hamming spectrum, we highlight - (1) correct output (2) an erroneous outcome that occurs more frequently than the correct output (3) rest of the erroneous outcomes (4) average of the Hamming bin.

In the Hamming spectrum shown in Figure 3(b), we observe that many incorrect outcomes that appear with high probability are close to the correct answer in Hamming space. Furthermore, the probability of an output string in a given Hamming bin reduces with increasing Hamming distance. For example, incorrect outputs that are four Hamming distances away from the correct answer have a lower than a random chance of occurrence. Figure 3(b) also shows the uniform probability distribution where all 2^N outcomes are equally likely with $\frac{1}{2^N}$ probability. So far, we have used BV to illustrate the structure in the output errors. However, unlike BV, most practical NISQ circuits produce output distributions with multiple correct outcomes. To understand if the structure in errors persists for such circuits, we analyze representative QAOA circuits that produce multiple correct outcomes. Figure 3(c) shows the Hamming spectrum for a QAOA-8 circuit that uses eight qubits and produces three correct outcomes with 82%,10.5% and 7.0% probability, respectively, in the absence of noise. On IBM Manhattan, the three dominant solutions appear with 7%, 0.7%, and 0.2% probability respectively. Most incorrect answers produced while running QAOA are within three Hamming distance from the correct answers. Note that for multiple correct answers, we consider the shortest Hamming distance.

3.3 How to Quantify Hamming Behavior?

To quantify the degree of Hamming structure, we use *Expected Hamming Distance (EHD)*. EHD computes the weighted sum of Hamming distances between correct outcome(s) and incorrect observations, where the weights are the probabilities of the incorrect observations. For output distributions without errors, EHD is zero. Whereas, in case of a uniform output probability distribution, EHD approaches $\frac{n}{2}$, where n is the number of qubits. Note that EHD $\in$ [0,n] as the output strings can be up to "n" hamming distance away from the correct outcomes. A quantum circuit that uses n qubits and produces correct answers all the time will have an EHD of zero, whereas, for uniform output probability distribution where all outcomes are equally likely, EHD would be close to $\frac{n}{2}$.

3.4 Hamming Clustering and QAOA

We observe Hamming structure in the output of QAOA circuits, wherein most incorrect outcomes are close to the desired outputs in the Hamming space. However, despite this closeness in Hamming space, incorrect outcomes can significantly change the expected value compared to the noise-free expected value. To illustrate how just a few bit-flips can significantly degrade the average solution quality, we show a partial cost landscape of max-cut problem used for QAOA-10 from Google dataset in Figure 5 and overlay two desired cuts with the lowest cost. Note that the maxcut problem is formulated such that cost of the desire cut is negative [20].

The staircase plot in Figure 5(a) shows the cost of all the solution strings that are one Hamming distance away from desired cuts, whereas Figure 5(b) represents the cost of all the solution strings that are two Hamming distance away from the desired cuts. Figure 5(a) show that the solutions that are just one Hamming distance away have 2x higher cost, and strings that are two Hamming distance away can have up to 10x higher cost compared to the desired solution as shown in the Figure 5(b).

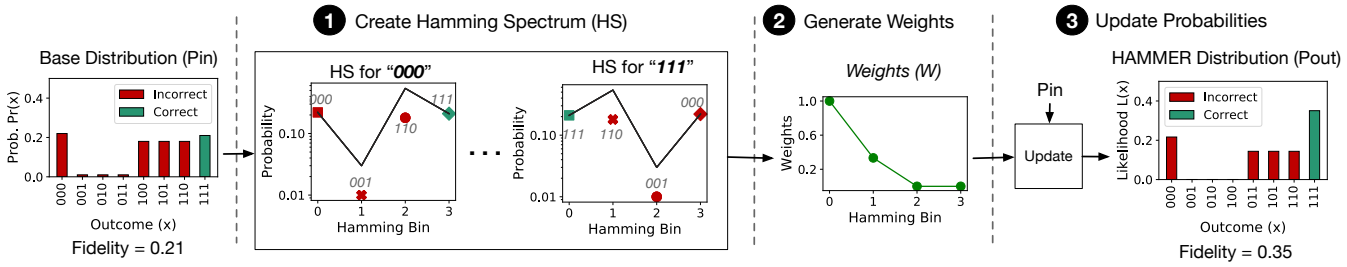


Figure 4: Overview of HAMMER to translate the distribution generated by the NISQ machine to corrected distribution.

4 HAMMING RECONSTRUCTION

HAMMER is a post-processing technique that leverages the observation about the erroneous outcomes being close in the Hamming space to the correct outcome to produce a modified distribution. In this section, we first provide an overview of HAMMER and then the details of each step.

4.1 Overview of HAMMER

The output of a NISQ program can be represented as a probability distribution, where the outcome x_i occurs with probability $Pr(x_i)$ measured across all the trials. Unfortunately, correct outcomes are often indistinguishable from incorrect ones due to errors, and therefore, the probabilities associated with them are generally inaccurate and insufficient to infer the solution for large programs. For example, Figure 4(a) shows the output distribution ($\mathbf{P}_{in}$) of a 3-qubit circuit whose correct output is "*111*". However, "*111*" does not appear with the highest frequency. If we rely on the probabilities and pick the most frequent outcome as the program solution, we will incorrectly pick "*000*". The goal of HAMMER is to *accurately* estimate the likelihood $\mathbf{L}(x_i)$ of every outcome x_i in $\mathbf{P}_{in}$.

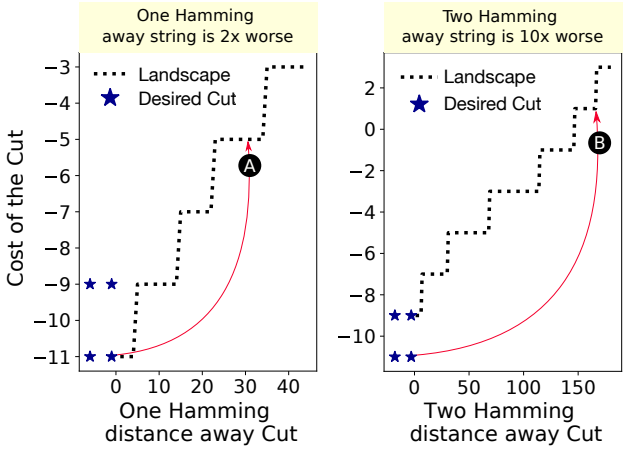


Figure 5: Landscape of a `QAOA-10` benchmark from Google data-set with cost of all solutions that are at (a) Hamming distance of one, or (b) Hamming distance of two from the desired cuts.

Figure 4 provides an overview of HAMMER. HAMMER consists of three steps. The first step identifies the Hamming distance neighborhood for each unique outcome in the histogram. This is used to compute a "Neighbourhood Score". The second step analyzes the neighborhood scores of all unique outcomes to develop a "weight" that must be assigned to each neighborhood that is at a distance K. The third step is to use the weight and the probability distribution of the neighborhood to compute the effective value for each outcome in the probability distribution (and normalize).

Thus, HAMMER determines the *likelihood* $\mathbf{L}(x_i)$ of every outcome x_i being an error-free answer, by combining (a) its probability of occurrence and (b) a "Neighbourhood Score", $\mathbf{S}(x_i)$, as described in Equation (1). We design a robust Likelihood function that obtains the neighborhood score by exploiting the structure in the Hamming space. Using this score, the singleton outcomes that appear without structure in Hamming space are penalized, whereas outcomes with Hamming structure get boosted. We discuss each step of HAMMER next.

$$\mathbf{L}(x_i) = Pr(x_i) \times \mathbf{S}(x_i) \tag{1}$$

4.2 Step-1: Create Hamming Spectrum

To capture the Hamming structure, we introduced the notion of *Hamming Spectrum* in Section 3. We can also represent this structure by using an equivalent Hamming graph where individual outcomes are the nodes of the graph, and the weight of the edges connecting the nodes is the Hamming distance between them. For example, the output distribution in Figure 6(a) can be represented as a six node Hamming graph, as shown in Figure 6(b-c).

Outcome (x)	Prob. Pr(x)
111	0.30
101	0.40
110	0.05
011	0.10
010	0.10
001	0.05

(a)

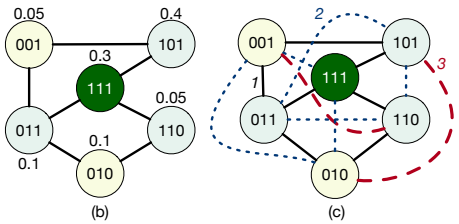


Figure 6: (a) Output Probability Distribution. (b-c) Hamming Graph Representation of the Output Distribution.

For simplicity of illustration, Figure 6(b) only shows those edges that are one Hamming distance away from each other, whereas, in reality, the graph has additional edges corresponding to other Hamming distances, as shown in Figure 6(c). In this example, although the correct outcome "111" occurs with a lower probability, by looking at the Hamming graph, we observe that "111" has more neighbors than the most frequent outcome "101".

To leverage this structure in deriving the likelihood function at scale, there are two key challenges:

(1) What is the right size of the neighborhood in Hamming space that will enable a robust likelihood function?
(2) How to compute the neighborhood score to boost program fidelity?

Size of Neighborhood: Our characterization data show that although the most frequent incorrect outcomes are close to the correct answers in the Hamming space, they disperse in multiple Hamming bins with increasing circuit depth. We observe that certain multi-bit flips become dominant in the output distribution, which increases the Expected Hamming Distance (EHD) and reduces the density of clusters. For example, the most frequent incorrect outcome "*110011111*" of a 10-qubit BV circuit is two Hamming distance away from the correct answer "*11111111*", as shown in Figure 7(a).

Thus, to compute a robust neighborhood score, we cannot rely only on neighbors at Hamming distance of one but also need to consider the neighbors at slightly larger distances.

Neighbourhood Score: The influence of a neighbor is computed from its Hamming Spectrum (*HS*). Increasing the neighborhood size causes the number of neighbors, and therefore its influence, to grow rapidly, as shown in Figure 6(c). For an n qubit program, there are $\binom{n}{k}$ possible neighbors that are within "k" Hamming distance away. In the limiting case, when the entire neighborhood is considered, the neighborhood score will be influenced by all the other outcomes, eventually yielding a uniform score across all outcomes. Thus, there exists a trade-off between the neighborhood size and its influence. While looking at very small neighborhoods may not be enough, particularly for large circuits as in the case of the BV-10 circuit example above, bigger neighborhoods can dilute the influence of the individual neighbors.

4.3 Step-2: Compute Per-Distance Weights

Ideally, we want to look at larger neighborhoods and simultaneously ensure that the neighborhood score derived from them has a positive influence on the likelihood function. To solve this problem, we limit the influence of each neighbor by assigning a weight (W) to each neighbor based on its Hamming distance (d) while computing the neighborhood score. Quite often, an erroneous outcome that occurs with a very low probability may exist in a very influential neighborhood, such as outcome "001" in Figure 6. To avoid such low probability outcomes deriving from a rich neighborhood, we introduce a filter function $\pi(x)$ that differentiates between dominant and average incorrect outcomes, as shown in Equation 2

$$\mathrm{S}(x) = \sum_{i=0}^{d} \pi(x) \times W_i \times CHS_i(x) \tag{2}$$

Next, we discuss how to design the weights and the filter function to exploit the structure of errors in Hamming space.

The outcomes generated on a NISQ machine may be grouped into (1) the correct outcome(s), (2) a few dominant incorrect outcome(s), and (3) a large number of average incorrect outcomes that appear with low frequency. To distinguish between the three, we utilize the observation that dominant incorrect outcomes lie in close proximity to the correct answers in the Hamming space. To quantify this observation, we define the Cumulative Hamming Strength (CHS), a vector that holds the total probability of all the outcomes that are "d" Hamming distance away from a given outcome. For an n-qubit program with n-bit output, the possible Hamming distances range from 0 to n. For a given string, we compute the CHS by adding the probabilities of the outcomes in each individual Hamming bin of the HS. For example, Figure 7(b) shows the CHS of a BV-10 program for the correct output, the most frequent incorrect outcome, and the average of all the outcomes. We observe that the CHS of the correct and dominant incorrect outcomes peaks at a low Hamming bin, as shown in the 7(b). On the contrary, the average (incorrect) outcome peaks at $d = \frac{n}{2}$ in the Hamming spectrum. This observation is consistent in all our experiments and closely matches the theoretical estimate from a uniform error model as the total number of entries in the Hamming bin approximately scale as $\binom{n}{d}$ whose maxima is at d= $\frac{n}{2}$. This suggests - (1) infrequent (average) incorrect outcomes exhibit weak structure in Hamming Space, (2) structure exists only for the dominant incorrect outcomes, and they appear in the close neighborhood of the correct answer. HAMMER uses this insight to differentiate between incorrect and correct outcomes.

Since infrequent average outcomes constitute the majority of the output distribution, the CHS of the average case represents the global neighborhood information. As most outcomes are erroneous, we use the average CHS to compute the weights for estimating the neighborhood score. We compute the weights (W) by inverting the average CHS as shown in Figure 7(c). Furthermore, to prevent the infrequent outcomes benefiting from a rich neighborhood, we limit the neighborhood sizes up to $\frac{n}{2}$ by assigning zero weight for Hamming bins greater than $\frac{n}{2}$. Each outcome in the output distribution is assigned its individual Neighborhood Score $S(x)$ by multiplying its CHS and the Weights. Figure 7(d) shows the neighborhood score for the correct outcome, the dominant incorrect outcome, and the average outcome.

4.4 Step-3: Update the Probability Distribution

HAMMER assigns the Neighborhood score by computing a dot product between the CHS vector and the weight vectors. However, in this process, a low probability string that is part of the rich neighborhoods gets assigned a high score. This equalization step can result in neighborhood scores that are very similar across all outcomes reducing the effectiveness of the HAMMER. For circuits with 10+ qubits, we observe that many low probability outputs are part of rich neighborhoods. For example, in the case of a QAOA-16 benchmark, there are more than 1000 outcomes within the radius of three Hamming distance. With increasing circuit size, although the expected Hamming distance grows slowly, we see an explosion in the total number of possible outcomes. For an effective score update, it is essential to account for this effect, and we introduce a filter function to update the neighborhood score.

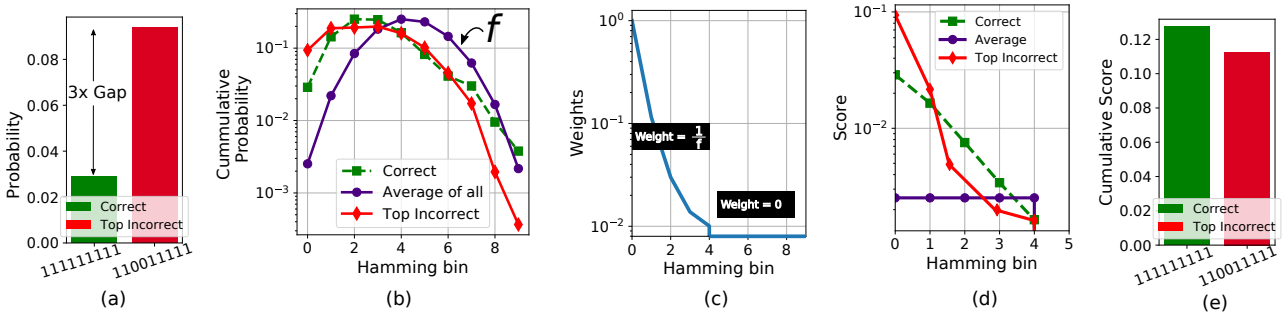


Figure 7: The probability distribution (P) produced by a BV-10 circuit. (a) Probability of correct and top incorrect outcomes in P. (b) Cumulative Hamming Spectrum (CHS). (c) Weights computed for P. (d) Neighbourhood Score for the correct, top incorrect, and average of all strings. (e) Cumulative Score.

The filter function determines if the string with low probability gets any credit from nodes that are high probability. For a given string (s), while computing neighborhood score, we only consider neighboring strings (s_i), which have a lower probability than the given string. This modulates the benefits given to the strings that are in the rich neighborhood but sampled with a high probability in the original distribution.

4.5 Putting it all Together

We use an example to show how HAMMER improves fidelity using neighborhood scores. Figure 7(a) shows the probability of two strings in the BV-10 output distribution. The output string "111111111" is an error-free output, whereas "110011111" is an incorrect string that appears with the highest probability. HAMMER's goal is to boost program fidelity by reducing the gap between correct and most frequently occurring incorrect outcomes.

In the baseline, there is a 3x gap between the two. Although the correct string has a low probability, it is part of a stronger neighborhood, as shown in 7(b). In HAMMER, we compute the average CHS and invert it to generate the weights, as shown in Figure 7(c). We compute the neighborhood score for the correct output and the incorrect output by multiplying the CHS with the weights. Figure 7(d) show neighbourhood score for each bin. We sum the neighborhood scores to compute the total score for both the output strings, as shown in Figure 7(e). Using these post-processing steps, we generate the output probability distribution. With HAMMER, the correct string now appears with a higher probability compared to the strongest incorrect string. Algorithm 1 in the Appendix describes the overall steps for HAMMER.

5 EVALUATION METHODOLOGY

We evaluate the effectiveness of HAMMER by running various benchmarks on three IBM quantum computers. Furthermore, we test HAMMER on publicly available quantum datasets from Google [37].

5.1 Quantum Hardware and Benchmarks

Google Dataset: We test HAMMER on the Google dataset (see Table 1). The output distributions in this dataset are collected by running *Quantum Approximate Optimization Algorithm (*QAOA*)* instances on the 53-qubit Sycamore processor [1]. These QAOA circuits are used to find the max-cut on Grid, Sherrington-Kirkpatrick and 3-regular input graphs [37].

Table 1: Benchmarks from Google Dataset [37]

Name	Algorithm details	#Qubits (n)	P Layers	Total Circuits	Figure of Merit
QAOA	Maxcut on Grid	6–20	1 to 5	120	CR
QAOA	Maxcut on 3-Reg Graphs	4–16	1 to 3	200	CR

IBM Dataset: We use quantum benchmarks of different sizes and structures to evaluate our proposed design. Table 2 summarizes the benchmarks used in this paper. For the QAOA benchmarks, we use it in the context of Max-Cut problems on a wide range of random and regular graphs. The graphs are generated using the Erdos-Renyi method [7]. To obtain a wide range of graphs, we vary the degree of connectivity between 0.2 (sparse) to 0.8 (highly connected), depending on the size of the problem. We adopt this approach from prior works focused on QAOA circuits [2, 18].

Table 2: Details of NISQ benchmarks on IBM Machines

Name	Algorithm details	#Qubits (n)	P Layers	Total Circuits	Figure of Merit
BV	Bernstein-Vazirani	5–15	-	88	IST, PST
QAOA	Maxcut on 3-Reg Graphs	5–20	2 and 4	70	CR, PF
QAOA	Maxcut Rand Graphs	5–20	2 and 4	70	CR, PF

Compiler and Hardware: We use the Qiskit compiler tool-chain from IBM [10, 24, 27]. Additionally, we perform the compilation step recursively to ensure minimum number of CNOTs. For all evaluations, we use three real quantum hardware from IBM. Note that although all of these machines have a Quantum Volume [12] of 32 they have very different error characteristics. Generally, NISQ programs are executed multiple times (called trials) and by default, IBMQ systems use 8K trials. For our evaluations, we execute between 8K-32K trials. This serves as our ***baseline*** for evaluation. We also evaluate HAMMER across multiple calibration cycles and observe similar results.

6 EVALUATIONS

We now discuss the effectiveness of HAMMER in improving the quality of solution for key NISQ benchmark circuits.

6.1 Figure of Merit for BV

We use two different metrics derived from prior works to evaluate our design, and these metrics are discussed below:

(1) Probability of Successful Trial (PST): measures the probability of the correct answer and is defined as the ratio of the number of trials that produce the correct outcome(s) to the total number of trials, as described in Equation (3). We use PST for BV circuits as it has one correct solution.

$$PST = \frac{\text{Number of Correct Trials}}{\text{Total Number of Trials}} \tag{3}$$

(2) Inference Strength (IST) is used to account for the magnitude of both the correct and the incorrect answers. IST is the ratio of the frequency of the correct output to the frequency of the most commonly occurring erroneous output, as described in Equation (4). If IST exceeds 1, the system will be able to correctly infer the output, whereas if IST is significantly lower than 1, then the wrong answer(s) would mask out the correct answer.

$$IST = \frac{Pr(S_{correct})}{Pr(S_{incorrect})} \tag{4}$$

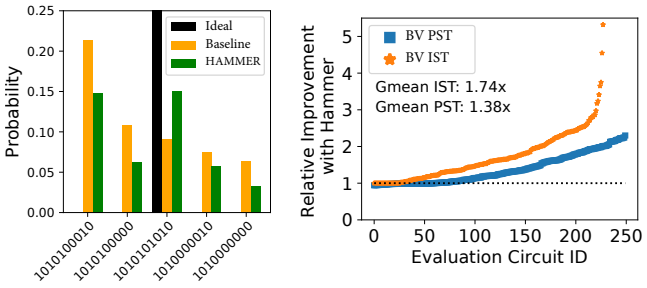


Figure 8: (a) Output of BV-10 circuit with *1010101010* key (b) Improvement in PST and IST with HAMMER for 250 BV circuits with 5-16 qubits over baseline on three IBM Machines.

6.2 Impact on Bernstein Vazirani Circuits

Figure 8(a) shows an output of a `BV-10` circuit executed on a noiseless simulator, on IBM-Paris (baseline), and a post-processed output from HAMMER. For noiseless simulation, the probability of solution string "1010101010" is 100%, whereas for the probability drops to 8% in the baseline. Furthermore, the incorrect outcome "1010100010" is the most frequent string with 20% probability in the baseline. The resultant distribution obtained by applying HAMMER on the output distribution from the baseline is shown in Figure 8(a). HAMMER improves the PST from 8% to 14% by boosting the correct solution string as it is more likely to be a correct answer based on the Hamming structure. Moreover, HAMMER attenuates the dominant incorrect outcome and increases the IST from 0.4 to 1.01, such that the solution key has the highest frequency of occurrence.

We observe a similar boost in program fidelity with HAMMER for 250 BV circuits. Figure 8(b) shows the relative improvement in PST and IST. We observe on average, HAMMER improves fidelity by 1.38x and up to 2x. At the same time, it boosts IST by up to 5x and, on average, by 1.74x.

6.3 Figure of Merit for QAOA Circuits

Cost Ratio (CR): Hybrid quantum-classical algorithms such as QAOA use parametric circuits with a classical optimization loop, in which the circuit parameters are tuned to minimize the cost function. To calculate the cost, the circuit is first executed on the NISQ hardware for thousands of trials, which generates an output distribution that consists of possible sample solution strings and their probabilities. We can evaluate the expectation value of the cost function by computing the weighted average of cost corresponding to each solution string in the output distribution [16, 20].

$$CR = \frac{C_{exp}}{C_{min}} \tag{5}$$

We analyze the impact of errors on the effectiveness of QAOA while evaluating the expectation value and discuss the efficiency of HAMMER in tackling these errors. We compute the ***Cost Ratio (CR)*** which is the ratio of the average quality (C_{exp}) and the lowest possible cost of the solution (C_{min}), as described in Equation (5). A higher CR indicates the better average quality of the solution. In the classical component of these algorithms, an optimizer uses the expectation values to tune the circuit parameters and eventually converges on a distribution that maximizes the expectation value and allows us to estimate the high-quality solution of the problem. Unfortunately, hardware errors that result in noisy distributions and low-quality expected values can disrupt the training process [4].

6.4 Impact on Quality of QAOA Solutions

Results on Google Dataset: We run HAMMER on 320 QAOA circuits used to solve Max-Cut problems with 200 *Grid* and 120 *3-Regular* input graphs to quantify improvements in CR from Google dataset [37]. The baseline data uses a post-measurement correction scheme to reduce the readout bias [20]. Figure 9(a) shows an S-curve for the Cost Ratio (CR) of the baseline and HAMMER data for 3-Regular graphs from 6 to 16 nodes and for p = 1 to 3 layers. Ideally, without noise, the CR ranges from 0.7 to 0.9, only depending on the number of layers. However, on Google Sycamore, qubit errors reduce the average quality of solution C_{exp} and CR drops to 0.08 to 0.4, as shown in Figure 9(a). HAMMER boosts the CR consistently for all input circuits showing improvements up to 2.4x in the CR. Figure 9(b) show an output distribution of a QAOA-10 circuit with p=2 layers. We plot the cumulative probability of all solutions (y-axis) corresponding to a Ratio of C_{sol}/C_{min} value on x-axis. The quality of the solution is highest when $C_{sol} = C_{min}$, and it degrades with decreasing value of the ratio. Note that $\frac{C_{sol}}{C_{min}}$ can be negative, which represents a sub-optimal cut. We want to maximize the probability of all solution strings with higher $\frac{C_{sol}}{C_{min}}$ value to boost the average cost C_{exp}. In Figure 9(b), HAMMER achieves this goal by increasing the cumulative probability from 12% to 19.5% for optimal cuts and reducing the probability of sub-optimal cuts. We observe a similar trend for grid input graphs in Figure 9(c) and (d).

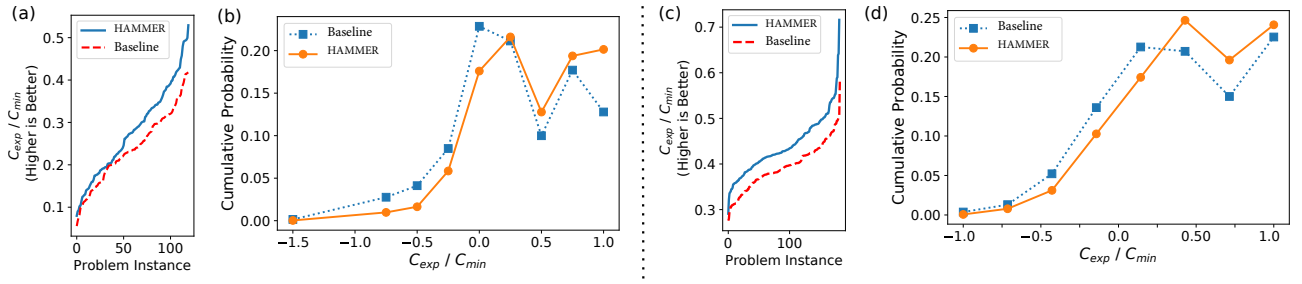


Figure 9: (a) Cost Ratio S-Curve for Baseline and HAMMER for 3-Reg input graphs. (b) Output for `QAOA-10` example on 3-Reg input. (c) Cost Ratio S-Curve for Baseline and HAMMER for grid input graphs. (d) Output for `QAOA-12` example on grid input.

For Grid graph inputs, both baseline and HAMMER have higher CR due to reduced circuit depth and total gate counts for grid circuits that do not require SWAPs.

Results on IBM Dataset: We use HAMMER to post-process 140 QAOA circuits executed on three IBMQ systems. For these benchmarks, we observe total variational distance (TVD) decreases by 1.23x and CR increases by 1.39x on average. We evaluate TVD by comparing output obtained on the hardware and ideal simulation of the circuit.

6.5 Reclaiming Algorithmic Benefits of QAOA

In theory, the solution quality of QAOA improves rapidly with the increasing number of "p" layers. However, by increasing the number of layers, QAOA circuits executed on NISQ hardware become more error-prone and produce noisy sub-optimal solutions with a higher frequency. This degrades the average quality of the solution. Furthermore, training a QAOA circuit with a higher "p" becomes challenging due to increasing noise. Figure 10 shows CR for Google baseline data with an increasing value of "p" for finding max-cut on the grid of 10 to 20 nodes. In the ideal noiseless case, the quality of solution, i.e. CR, improves monotonically with increasing "p" number of layers. Whereas, on Google's Sycamore device, the quality of solution peaks at p=2 and reduces. When we post-process Google data with HAMMER, we observe a peak in quality at p=3. As HAMMER can suppress hardware noise, it can reclaim algorithm benefits of QAOA at higher "p" values.

Moreover, to understand if HAMMER can improve the classical optimization steps in variational mode, we run HAMMER on a complete landscape of `QAOA-14`, solving maxcut for 3-regular graph. We observe that HAMMER consistently enhances the quality of solution for each data point on the grid and sharpens the gradient.

6.6 Complexity Analysis of HAMMER

For a given quantum program with n qubits, let N be the number of non-zero entries in the (noisy) distribution obtained on a NISQ computer. From the memory complexity viewpoint, HAMMER requires to store two vectors of size $n/2$ for storing Hamming score (HS) and weight vectors—denoted by HS and W in Algorithm 1, respectively. Hence, the memory required by HAMMER grows linearly with the number of qubits—i.e., $O(n)$. Our analysis shows that the memory required is less than 1 MB even for problems using up to 500 qubits. For computational complexity, HAMMER performs $N^2 + N$ steps to compute the Hamming weight vector, N^2 steps for computing the likelihood of observations being a correct outcome, and N steps for normalizing likelihoods. Thus, the execution time of HAMMER grows quadratic with the number of unique outcomes - $O(N^2)$. For HAMMER, the execution time gets determined by the number of unique outcomes, which is limited by the number of trials for large programs.

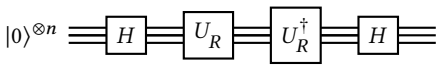
For example, Google uses a total of 25,000 trials irrespective of the size of the `QAOA` problem [20]. The largest `QAOA` instance that we evaluate with HAMMER had 24 qubits. For this instance, there were about 20K unique outcomes, and HAMMER required 56 seconds for a Python-based single-threaded code. On NISQ machines, we expect to observe a limited number of unique outcomes. Even if we use a few thousand trials and each trial produces a unique outcome, HAMMER can process this within a few minutes. Table 3 shows the complexity of HAMMER for machines with up-to 500 qubits when we run 32K or 256K trials. The time complexity of HAMMER is small, even with 256K unique outcomes.

Table 3: Number of Operations Required

Trials (T)	Unique Outcomes	Operations in Billion	
		Qubits (n) = 100	Qubits (n) = 500
32K	10%	0.001	0.001
	100%	1	1
256K	10%	0.6	0.6
	100%	64	64

7 IMPACT OF ENTANGLEMENT AND CIRCUIT SIZE ON HAMMING BEHAVIOUR

Quantum algorithms leverage entangled or correlated states to enable speedup over classical methods. Unfortunately, entangled states are prone to errors. Moreover, certain errors can rapidly spread among entangled qubits reducing the Hamming structure. To understand if the Hamming behavior persists with increasing entanglement, we run over a thousand benchmark circuits with varying degrees of entanglement on IBM quantum hardware. We use circuits with the following structure:



Each circuit starts and ends with a layer of Hadamard gates between which a random unitary, U_R, and its reverse, $U_R^\dagger$, are used. The sub-circuit U_R comprises of randomly selected single-qubit (`Rz`, `Rx`, `Ry`) and two-qubit (`CX`, `CZ`) gates. The $U_R^\dagger$ is the reverse circuit

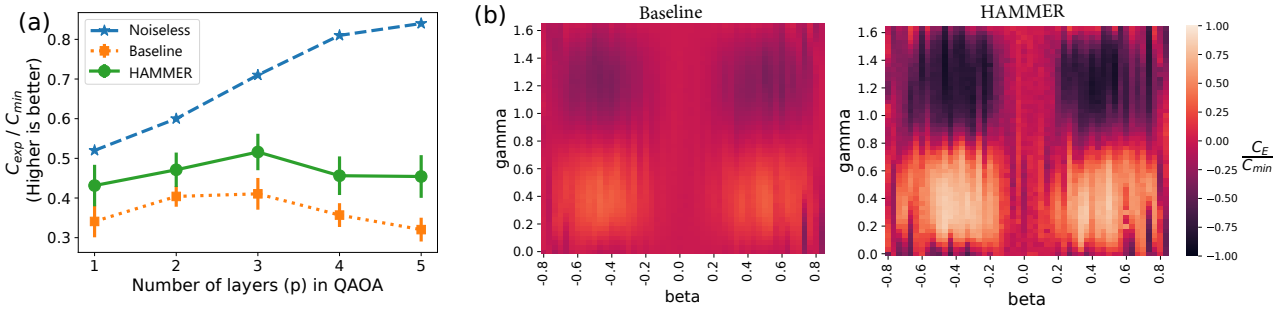


Figure 10: (a) Quality of solution with increasing layers in QAOA circuits solving Max-Cut for grid graphs with 6-20 nodes, with total 200 circuits. (b) Optimization landscape for 3-Regular graph for Google baseline and HAMMER.

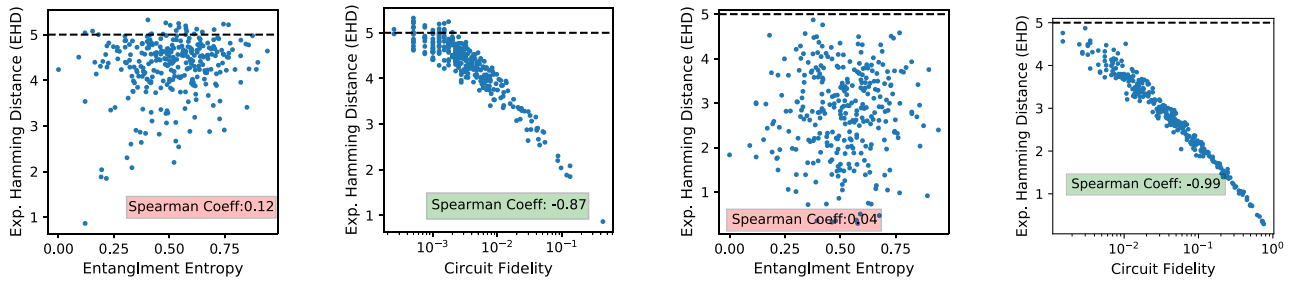


Figure 11: EHD of high depth benchmark circuits with varying (a) Entanglement Entropy (b) Fidelity. EHD of low benchmark circuits with varying (c) Entanglement Entropy (d) Fidelity. Spearman Correlation Coefficient used to quantify the correlation.

such that $U_R * U_R^{\dagger} = I$. These circuits create an entangled state and gradually untangle it, producing an all-zero state ($|0000..0_n\rangle$). We use random unitaries to generate circuits with varying entanglement. Furthermore, the all-zero output state enables high fidelity measurements. We evaluate the degree of entanglement for a benchmark circuit by computing the entanglement entropy of the state produced by the sub-circuit : $H.U_R$ using ideal simulations. We run two sets of benchmark circuits - (1) high depth circuits with depth up to 25 (2) low depth circuits with depth up to 15.

Figure 11(a) shows how EHD changes with varying entanglement entropy for 300 ten-qubit circuits running on IBM hardware. We observe a weak correlation between entanglement entropy and EHD (Spearman coefficient: 0.2). Furthermore, EHD is lower than the uniform error model (dotted line), showing a strong Hamming structure. We observe a similar trend across different benchmark circuits. In fact, for shallower circuits shown in Figure 11(c), the correlation between entanglement and structure in errors weakens.

While Hamming structure persists despite the increasing entanglement entropy or degree of entanglement, it reduces with increasing noise. As shown in Figure 11(b) and Figure 11(d) with decreasing fidelity, EHD increases for both low and high depth benchmark circuits. Increasing the size of quantum circuits increases the total number of operations and the duration for which qubits are active. This exacerbates the error rate, and with increasing errors, fidelity drops, and EHD increases. Similarly, Figure 12(a) shows EHD for BV and QAOA circuits with 6 to 20 qubits. With the increasing size of circuits, we see an increase in EHD.

With more errors, more incorrect answers are produced and scattered across the Hamming space. This results in a higher average Hamming distance between any two outcomes. However, compared to the uniform distribution where EHD is $\frac{n}{2}$, circuits discussed in Figure 12 have significantly lower EHD. Furthermore, we observe different rates of increase in EHD for different circuits. BV circuits, for example, lose the structure much faster as compared to the QAOA circuits. This is because the depth of BV circuits increases super linearly compared to the linear increase in QAOA circuits.

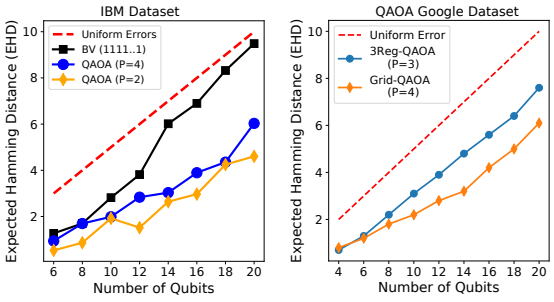


Figure 12: EHD of output (a) IBM-Paris (b) Google-Sycamore

Figure 12 highlights that circuit depth is a dominant factor that influences the EHD and the Hamming structure. Moreover, we observe this trend for over 400 QAOA circuits that are used to solve a max-cut problem on 3-regular, 2-regular, and Erdos-Renyi graphs with 6 to 20 nodes, as shown in Figure 12(a), and a similar trend

manifests for QAOA on Google-Sycamore, as shown in Figure 12(b). We observe that majority of the QAOA circuits have lower EHD than the uniform distribution, and for QAOA circuits with reasonable program fidelity, we observe a low EHD and dense clusters around the correct outcome. Our observations are consistent across three IBM machines and over twenty days of experiments. We also verify if the trend in error structure changes significantly due to qubit mapping. Our results suggest that although average fidelity and dominant incorrect answers change, the clustering effects remain similar across different mappings.

While we cannot establish the exact reasons for the Hamming behavior of incorrect outcomes, we can surmise a few reasons for why this behavior occurs in practice. Our evaluations show for shallow circuits, errors can have a localized impact preserving Hamming structure. Thus, shallower circuits can be expected to provide more Hamming structure than deeper circuits. In fact, given the constraints of NISQ systems, we are likely to run programs with shallow depth (for example, on IBM hardware, the fidelity of QAOA circuits drops below 1% beyond circuit depth of 40). Our experiments show that for wide and low depth circuits, the errors are clustered in Hamming space.

8 RELATED WORK

NISQ computers promise computational advantages for practical applications [16, 25, 31]. In the absence of quantum error correction on these systems, software policies will play a vital role in closing the gap between the devices and the algorithms [9, 30]. Therefore, mitigation of hardware errors through software techniques is an active area of research. We can broadly classify them as compiler and post-processing techniques.

Compiler-Based Error-Mitigation: These methods focus on (1) generating highly optimized program schedules by accounting for the application-specific characteristics to reduce the circuit depth and number of operations [3, 18, 19, 24, 39, 45] and (2) hardware error characteristics to perform noise-aware computations [26, 28, 32, 35, 42–44]. There are other approaches that decompose a circuit into smaller circuits and obtain the output distribution using a tensor product [41]. Generally speaking, all of these policies focus on reducing the likelihood of a program encountering errors. Some of them particularly focus on a very specific source of error such as crosstalk between ongoing CNOT operations [28], measurement errors [5, 8, 21, 23, 33, 43], correlated errors [42], or idling errors [40]. HAMMER uses a completely different approach and is compatible with all of these policies as it can be applied to a program compiled with these optimizations.

Post-processing for Error-Mitigation: Recent work [34] has looked at the problem of correlated errors and proposed diverse mappings on different machines to reduce the magnitude of correlated errors. These schemes post-process the noisy outputs obtained from the individual mappings to offer a more accurate output distribution. Post-processing schemes have also been effective in mitigating measurement errors. These schemes use characterization data to compute a noise matrix which is then used to post-process the noisy output distribution obtained on a NISQ device [8, 21].

9 CONCLUSION

We propose *Hamming Reconstruction (HAMMER)*, a post processing technique to boost the fidelity of a quantum program. It uses the insight that correct outcomes are clustered in the Hamming space. HAMMER estimates the likelihood of each outcome based on a neighborhood of answers within a small Hamming distance of given answer. We evaluate the effectiveness of HAMMER using more than 500 key quantum benchmarks on IBM and Google Sycamore Datasets and show that HAMMER improves the quality of solution by 1.37x on average. We also evaluate the scalability of HAMMER and show that it easily scales to machines with thousands of qubits.

ACKNOWLEDGMENTS

We thank Matthew Harrigan from the Google Quantum AI Group for providing us the QAOA datasets used for some of the evaluations in this paper. Poulami Das was funded by the Microsoft Research PhD Fellowship. Ramin Ayanzadeh was supported by the NSF Computing Innovation Fellows (CI-Fellows) program. 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.

APPENDIX A: HAMMING RECONSTRUCTION

The Hamming Reconstruction algorithm is described below.

Algorithm 1: Hamming Reconstruction

Input: P_{in} (input distribution), n (number of qubits)
Output: P_{out} (output distribution from HAMMER)

Function HAMMER(P_{in}):
  // Step-1: Create Hamming Spectrum
  CHS = zeros(n/2)
  for x **in** P_{in} **do**
    for y **in** P_{in} **do**
      d = Hamming Distance (x, y)
      if $d < \frac{n}{2}$ **then**
        $CHS[d] += P_{\text{in}}[y]$
  // Step-2: Compute Per-Distance Weights
  W = zeros(n/2)
  for $i = 0$ **to** $n/2$ **do**
    if $CHS[d] > 0$ **then**
      $W[d] = 1/CHS[d]$
  // Step-3: Update the Probability Distribution
  $P_{\text{out}} = \{\}$
  for x **in** P_{in} **do**
    $score = P_{\text{in}}[x]$
    for y **in** P_{in} **do**
      d = Hamming Distance (x, y)
      if $d < \frac{n}{2}$ **and** $P_{in}[x] > P_{in}[y]$ **then**
        $score += W[d] \times P_{\text{in}}[y]$
    $P_{\text{out}}[x] = score \times P_{\text{in}}[x]$
  P_{out} = normalize (P_{out})
  return P_{out}

REFERENCES

[1] Google Quantum AI. Accessed: June 19, 2021. Quantum Computer Datasheet. https://quantumai.google/hardware/datasheet/weber.pdf.

[2] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. 2020. Accelerating quantum approximate optimization algorithm using machine learning. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 686–689.

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

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B Buckley, David A Buell, et al. 2020. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *arXiv preprint arXiv:2004.04197* (2020).

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

[6] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. *Nature* 549, 7671 (2017), 195–202.

[7] Béla Bollobás and Bollobás Béla. 2001. *Random graphs*. Number 73. Cambridge university press.

[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] Frederic Chong. 2018. Closing the gap between quantum algorithms and hardware through software-enabled vertical integration and co-design. In *APS March Meeting Abstracts*, Vol. 2018. C39–004.

[10] International Business Machines Corporation. 2017. Quantum Software Development Kit for writing quantum computing experiments, programs, and applications. https://github.com/QISKit/qiskit-sdk-py. [Online; accessed 28-AUGUST-2020].

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

[12] 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.

[13] Poulami Das, Swamit Tannu, Siddharth Dangwal, and Moinuddin Qureshi. 2021. ADAPT: Mitigating Idling Errors in Qubits via Adaptive Dynamical Decoupling. In *MICRO-54*.

[14] Poulami Das, Swamit Tannu, and Moinuddin Qureshi. 2021. JigSaw: Boosting Fidelity of NISQ Programs via Measurement Subsetting. In *MICRO-54*.

[15] Poulami Das, Swamit S Tannu, Prashant J Nair, and Moinuddin Qureshi. 2019. A Case for Multi-Programming Quantum Computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 291–303.

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

[17] Lena Funcke, Tobias Hartung, Karl Jansen, Stefan Kühn, Paolo Stornati, and Xiaoyang Wang. 2020. Measurement Error Mitigation in Quantum Computers Through Classical Bit-Flip Correction. *arXiv preprint arXiv:2007.03663* (2020).

[18] 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.

[19] 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).

[20] Matthew P Harrigan, Kevin J Sung, Matthew Neeley, Kevin J Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C Bardin, Rami Barends, Sergio Boixo, et al. 2021. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nature Physics* 17, 3 (2021), 332–336.

[21] IBM. 2019. Measurement Error Mitigation. https://qiskit.org/textbook/ch-quantum-hardware/measurement-error-mitigation.html. [Online; accessed 26-July-2020].

[22] IBM. 2021. IBM's roadmap for scaling quantum technology. https://research.ibm.com/blog/ibm-quantum-roadmap.

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

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

[25] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. 2016. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics* 18, 2 (2016), 023023.

[26] 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).

[27] 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).

[28] 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).

[29] 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).

[30] 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

[31] Roman Orus, Samuel Mugel, and Enrique Lizaso. 2019. Quantum computing for finance: overview and prospects. *Reviews in Physics* (2019).

[32] 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 *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*. 705–711.

[33] Tirthak Patel and Devesh Tiwari. 2020. DisQ: a novel quantum output state classification method on IBM quantum computers using openpulse. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–9.

[34] Tirthak Patel and Devesh Tiwari. 2020. Veritas: accurately estimating the correct output on noisy intermediate-scale quantum computers. In *2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE Computer Society, 188–203.

[35] Tirthak Patel and Devesh Tiwari. 2021. Qraft: reverse your Quantum circuit and know the correct program output. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 443–455.

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

[37] Google AI Quantum and Collaborators. 2020. Sycamore QAOA experimental data. "https://figshare.com/articles/dataset/Sycamore_QAOA_experimental_data/12597590". [Online; accessed 26-July-2021].

[38] 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).

[39] 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.

[40] Kaitlin N Smith, Gokul Subramanian Ravi, Prakash Murali, Jonathan M Baker, Nathan Earnest, Ali Javadi-Abhari, and Frederic T Chong. 2021. Error Mitigation in Quantum Computers through Instruction Scheduling. *arXiv preprint arXiv:2105.01760* (2021).

[41] Wei Tang, Teague Tomesh, Martin Suchara, Jeffrey Larson, and Margaret Martonosi. 2021. CutQC: using small quantum computers for large quantum circuit evaluations. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 473–486.

[42] 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.

[43] Swamit S Tannu and Moinuddin K Qureshi. 2019. Mitigating Measurement Errors in Quantum Computers by Exploiting State-Dependent Bias. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 279–290.

[44] 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*.

[45] Alwin Zulehner, Alexandru Paler, and Robert Wille. 2018. Efficient mapping of quantum circuits to the IBM QX architectures. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1135–1138.