Performance of surface codes in realistic quantum hardware

Antonio deMarti iOlius,[1,*] Josu Etxezarreta Martinez,[1] Patricio Fuentes,[1] Pedro M. Crespo,[1] and Javier Garcia-Frias[2]

[1]*Department of Basic Sciences, Tecnun–University of Navarra, 20018 San Sebastian, Spain*

[2]*Department of Electrical and Computer Engineering, University of Delaware, Newark, Delaware 19716, USA*

(Received 13 October 2022; accepted 2 December 2022; published 21 December 2022)

Surface codes are generally studied based on the assumption that each of the qubits that make up the surface code lattice suffers noise that is independent and identically distributed (i.i.d.). However, real benchmarks of the individual relaxation (T_1) and dephasing (T_2) times of the constituent qubits of state-of-the-art quantum processors have recently shown that the decoherence effects suffered by each particular qubit actually vary in intensity. In consequence, in this paper we introduce the independent nonidentically distributed (i.n.i.d.) noise model, a decoherence model that accounts for the nonuniform behavior of the decoherence parameters of qubits. Additionally, we use the i.n.i.d. model to study how it affects the performance of a specific family of quantum error correction codes known as planar codes. For this purpose we employ data from four state-of-the-art superconducting processors: ibmq_brooklyn, ibm_washington, Zuchongzhi, and Rigetti Aspen-M-1. Our results show that the i.i.d. noise assumption overestimates the performance of surface codes, which can suffer up to 95% performance decrements in terms of the code pseudothreshold when they are subjected to the i.n.i.d. noise model. Furthermore, we consider and describe two methods which enhance the performance of planar codes under i.n.i.d. noise. The first method involves a so-called reweighting process of the conventional minimum weight perfect matching (MWPM) decoder, while the second one exploits the relationship that exists between code performance and qubit arrangement in the surface code lattice. The optimum qubit configuration derived through the combination of the previous two methods can yield planar code pseudothreshold values that are up to 650% higher than for the traditional MWPM decoder under i.n.i.d. noise.

DOI: 10.1103/PhysRevA.106.062428

I. INTRODUCTION

Quantum computing heralds the arrival of a new era in computer science where problems that are not within reach for classical computers will become tractable. The principal tenet of quantum computing is to design ingenious algorithms that are capable of exploiting the superposition property of quantum states, which ultimately allows them to consider large portions of problem solution spaces concurrently. Generally, quantum computers are understood as ensembles of qubits, two-level coherent quantum systems that can be employed to leverage the quantum mechanical property of superposition. It must be mentioned, however, that quantum processors can also be constructed using more complex and higher discrete-level coherent quantum systems, known as qudits [1,2], or even continuous intervals [3,4]. At the time of writing, significant efforts and resources are being destined towards the construction of a large-scale universal fault-tolerant quantum computer. Nonetheless, even though substantial progress has been made in the field in recent years, machines with the capacity to fulfill the complete promise of quantum computing remain, as of yet, nonexistent.

The main cause for this is that currently existing quantum computers are too noisy to run sophisticated quantum algorithms reliably. The noise of a qubit is generally defined by its decoherence parameters (relaxation time, T_1, and dephasing time, T_2), which are measures of how long the qubit can maintain its coherence and, thus, be employed reliably to perform calculations [5–7]. Unfortunately, present-day qubits lack sufficiently long coherence times to enable reliable quantum computing. This occurs because the coherence time of modern qubits is too short relative to the amount of time that is required to interact with them. Qubits are manipulated through the action of quantum gates, the application of which consumes much of the coherence time of qubits, and makes it difficult to perform complex and reliable quantum calculations. While the coherence time of qubits varies depending on how they are built (qubits constructed with ion traps present decoherence times in the order of seconds while these times are in the order of hundreds of microseconds for superconducting qubits), so do their gate operation times (superconducting quantum gates are much faster than ion trap quantum gates). For this reason, regardless of which technology is used to implement them, currently existing qubits will suffer from similar noise processes.

Quantum states experience coherence losses as a result of the unwanted interactions that qubits have with their environment. These interactions arise through myriads of physical mechanisms, many of them unavoidable, and they are all grouped under the same term: decoherence. In fact, other sources of errors in quantum computers, such as faulty gates or inaccurate measurements, can also fall under the umbrella of decoherence. Thus, within the abstraction that decoherence provides to represent quantum noise, the technological odyssey of building a reliable quantum computer can be

[*]ademartio@tecnun.es

simply summarized as the search for strategies to effectively fight the effects of decoherence. It is in answer to this challenge that the discipline of quantum error correction (QEC) arose, to study the phenomenon of decoherence and to design strategies to protect qubits from quantum noise. Similar to what is done in the classical computing framework, QEC strategies, known as QEC codes (QECCs), employ additional qubits to protect quantum information from the impact of decoherence-related effects. In fact, thanks to certain similarities between the classical and quantum computing paradigms, QECCs can be built from existing classical codes. This is achieved by casting existing groups of classical codes into the QEC framework by means of the well-known stabilizer formalism [8]. In consequence, many classical-inspired QEC code families like quantum low density parity check codes [9–12] or quantum turbo codes [13–16], among many others, are already being studied.

However, because many of these code families require large numbers of fully connected qubits to successfully battle quantum noise, practical QEC solutions for present day quantum computers are generally based on surface codes, a different type of stabilizer code that is not based on previously existing classical codes [17,18]. Surface codes are constructed by encoding logical quantum states (those qubits that contain the information that will be processed) into two-dimensional lattices of physical qubits. Particular qubits in the lattice act as measurement qubits that can be used to extract quantum syndromes, binary vectors that enable error diagnosis of qubits while avoiding direct measurement of quantum states, effectively allowing us to perform the appropriate recovery operations without destroying the superposition state of the logical qubits.

Research within the field of QEC, including the surface code niche, typically assumes that the qubits that make up error correction codes will suffer decoherence-related errors that can all be described by the same probability distribution, i.e., that in each error correction round every qubit experiences noise defined by an independent identically distributed (i.i.d.) process [9–18]. However, recent results have shown that the relaxation and dephasing times of qubits in real quantum hardware are actually significantly different [19–23], with drastic variation in the decoherence parameter values of particular qubits. Given that the decoherence-induced noise experienced by superconducting qubits is characterized by their corresponding T_1 and T_2 times [6,7], the data from real quantum processors suggest that studying surface codes under the i.i.d. qubit noise assumption does not provide the most accurate portrayal of their performance. For this reason, in this paper we introduce the independent nonidentically distributed (i.n.i.d.) qubit-noise model as a way to capture and reflect the differences in the decoherence parameters of real qubits. Additionally, we use the aforementioned model to study how the performance of surface codes changes over the proposed i.n.i.d. model. The primary difference between the i.n.i.d. decoherence model and the conventional i.i.d. assumption is that the former model considers that each particular qubit of the surface code lattice has its own decoherence defining parameters (T_1 and T_2) and will experience different noise levels, whereas the latter model (the i.i.d. model) assumes that all qubits are defined by the same decoherence parameters and hence every physical qubit is subjected to the same noise level.

Recently, several works have discussed the experimental performance of surface codes and the way that realistic noise can be modeled. For example, in [24] the performance of belief matching [25] for an experimentally implemented rotated planar code has been studied. In that work, the authors consider a hardware specific noise model and decode the planar code by standard belief-matching and tensor network decoders. Moreover, a noise model considering nonuniform gate errors is discussed for the heavy hexagon code and the decoder is adapted for such scenario in [26]. These, however, do not explicitly consider the i.n.i.d. model proposed in this paper. Note that even if [24] considers individual T_1 and T_2 for the noise model the decoding does not directly tackle this issue, and observing the fact that their values are pretty uniform the performance will not be considerably compromised. To provide a realistic view of the impact that the more accurate i.n.i.d. model can have on the performance of surface codes, we have used the values of the relaxation and dephasing times of modern quantum processors (ibm_washington [20], ibmq_brooklyn [21], Zuchongzhi [22], and Aspen-M-1 [23]) in order to simulate noise in a planar surface code. Our results show that the i.i.d. qubit-noise assumption provides an overly optimistic portrayal of the performance of surface codes when they operate on real hardware. Fortunately, we also show that methods that remarkably enhance performance (at some points surpassing that which would be expected based on i.i.d. results) exist.

II. PRELIMINARIES

A. Planar codes

Surface codes are a family of quantum error correcting codes the constituent qubits of which are laid out on a two-dimensional lattice. The qubits within the code interact locally, i.e., only with their nearest neighbors. Depending on the geometry of the lattice, different types of surface codes can be constructed. For our paper we consider surface codes where qubits are arranged over a square lattice [18,27]. Surface codes the fundamental lattice of which is the square are named planar codes. Specifically, we will consider the square planar codes, in which qubits are arranged on the center, edges, and vertices of the square lattice and have the same amount of edges and vertices, thus, forming a square shaped qubit distribution. With a slight abuse of notation we will refer to those planar codes throughout the paper.[1]

There are two different types of qubits within the square lattice that define a planar code: the data qubits, which are located on the edges of the lattice and encode the quantum state of the code, and the measurement qubits, which are continuously initialized and measured in order to obtain information regarding errors that may have arisen. Measurement qubits interact locally with their nearest data qubit neighbors and will act differently depending on where they are located. Based

[1]Note that codes with rectangular qubit distributions are also planar codes, which are mostly used for correcting biased noise [27]. Those codes are not considered in this paper.

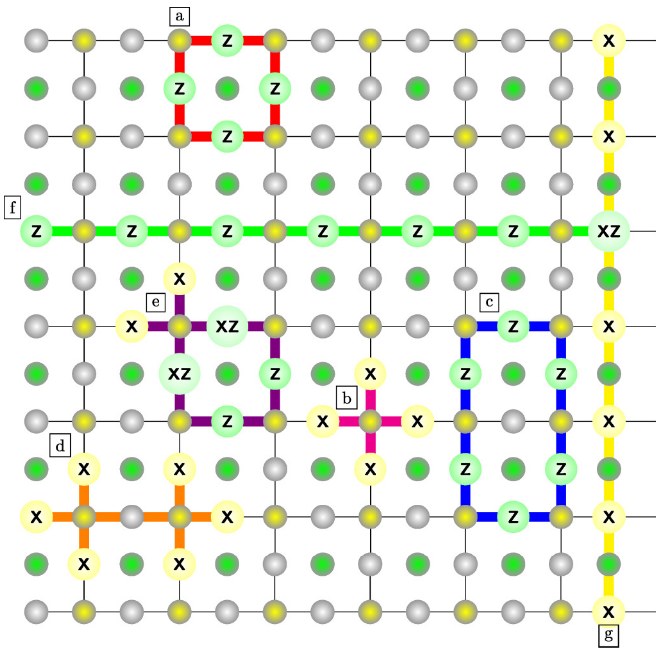

FIG. 1. Graphical representation of a 7×7 planar code. The data qubits, the measure-X qubits, and the measure-Z qubits are depicted by white, yellow, and green dots, respectively. Data qubits that suffer X, Z, and XZ operators are portrayed by light green, light yellow, and lighter green dots. The action of various stabilizer elements is highlighted using different color patterns: (a) action of a measure-Z qubit, (b) action of a measure-X qubit, (c) combination of two adjacent plaquette-plaquette stabilizers, (d) combination of two adjacent vertex-vertex stabilizers, (e) combination of two adjacent vertex-plaquette stabilizers, (f) Z_L operator, and (g) X_L operator.

on how they act on their neighboring data qubits, we can also classify measurement qubits into two separate groups. On one hand we have vertex qubits or "measure-X" qubits, which force their surrounding data qubits into an eigenstate of the operator product $X_1X_2X_3X_4$, where X_i is the Pauli X operator for a specific qubit i and 1, 2, 3, and 4 are the nearest-neighbor data qubits of the considered measurement qubit. On the other hand, we also have plaquette qubits or "measure-Z" qubits, which force the surrounding data qubits into an eigenstate of the operator product $Z_1Z_2Z_3Z_4$, where Z is the Pauli Z operator. These concepts are reflected in Fig. 1, which portrays a graphical representation of a 7×7 planar code. Notice that the boundaries are not equal, i.e., the top and bottom lattice boundaries apply vertex measurement qubits while the right and left lattice boundaries apply plaquette measurement qubits.

The code is initialized by collapsing all measurement qubits so that the data qubits are forced into an eigenstate of all their operator products. The resulting state is known as the quiescent state [27]. Once the quiescent state has been obtained, subsequent measurements of measurement qubit states will remain unchanged since the data qubits are in an eigenstate of their operator products [27]. For this reason, any change in the measurement of any measure-X or measure-Z qubit will imply that the code is no longer in the quiescent state it was initialized in. Moreover, since the qubits interact locally with their nearest neighbors, this means that one or three of its adjacent data qubits have experienced a Pauli error. Measure-X qubits will be susceptible to Z and Y errors, while measure-Z qubits will be susceptible to X and Y errors, as shown in Eq. (1):

$$\begin{aligned} X_aX_bX_cX_dZ_a\,|\psi\rangle &= -Z_aX_aX_bX_cX_d\,|\psi\rangle\,, \\ Z_aZ_bZ_cZ_dX_a\,|\psi\rangle &= -X_aZ_aZ_bZ_cZ_d\,|\psi\rangle\,, \\ X_aX_bX_cX_dY_a\,|\psi\rangle &= X_aX_bX_cX_d iX_aZ_a\,|\psi\rangle \\ &= -iX_aZ_aX_aX_bX_cX_d\,|\psi\rangle \\ &= -Y_aX_aX_bX_cX_d\,|\psi\rangle\,, \end{aligned} \tag{1}$$

where $a, b, c,$ and d refer to the four surrounding measurement qubits, and $X, Y,$ and Z are the nonidentity Pauli matrices. The fact that measure-X qubits detect Z errors and measure-Z qubits detect X errors is the reason why they are often referred to as Z checks and X checks, respectively. As a result, collapsing the measurement qubits serves to extract the syndrome associated to the error that has taken place at any given instance.

Planar codes like the one depicted in Fig. 1 encode one logical qubit. Logical operators modify the logical state of the surface code in a nontrivial manner while remaining within the codespace (the resulting state commutes with all the stabilizers). We label the Pauli logical operators of the code as X_L, Y_L, and Z_L. The logical operators X_L and Z_L can be applied by manipulating the degrees of freedom of the surface code. This is shown in Fig. 1. Consider the set of Z operators that traverse the entire planar code lattice horizontally (green line in the figure). These operators commute with all the stabilizer generators of the code, hence they end up forming a Z_L operator. Similarly, a series of adjacent X operators that cross the surface code lattice vertically end up forming a X_L operator. We refer to adjacent Z operators that traverse the edges of the lattice as chains while adjacent X operators within the center of lattice squares are known as cochains. When the combination of logical operators X_LY_L is applied, X and Z operators coincide on the same qubit. We construct Y_L operators as the product of the aforementioned X_L and Z_L logical operators: $Y_L = Z_LX_L$. Also, $X_L^2 = Y_L^2 = Z_L^2 = \mathrm{I}$, since the square of any of these logical operators can be written in terms of the stabilizer generators and so they will not modify the state of the code. Whenever the noise introduced by an n-qubit Pauli channel results in the formation of chains or cochains and the creation of a logical operator, a logical error will take place. This modifies the state of the logical qubit in a nontrivial manner but results in a trivial quantum syndrome when collapsing the measurement qubits (recall that logical operators preserve the codespace). More specifically, the combination of the operator induced by the quantum noise and the recovery operator can form logical bit-flip errors (X_L), logical phase-flip errors (Z_L), and logical bit-and-phase-flip errors (Y_L). Decoding failures in which wrong error estimates still result in a nontrivial syndrome are also considered to be logical errors. All in all, logical errors act harmfully on our encoded quantum states and make it difficult to maintain the logical qubit in the desired original quantum state. Avoiding and minimizing the likelihood of chain and cochain formation is critical for the planar code to successfully correct errors.

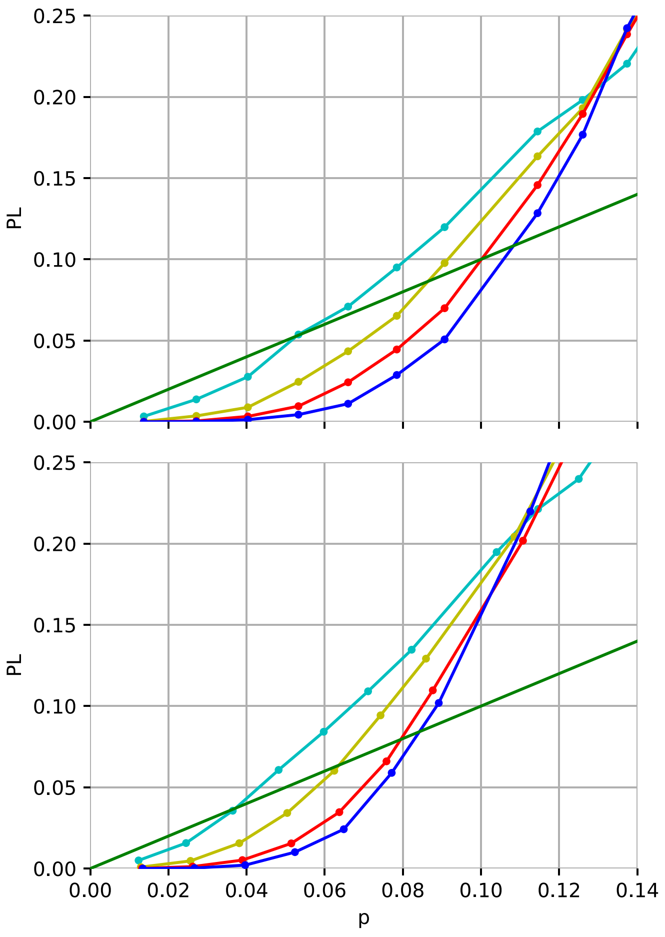

FIG. 2. Logical error rates of planar codes for the i.i.d. and i.n.i.d. models. (a) Planar codes operating over the i.i.d. noise model. (b) Planar codes operating over the i.n.i.d. noise model constructed with data from the ibm_washington hardware [20]. The green line represents the performance of the uncoded system, i.e., $P_L(p_{\mathrm{PT}}) = p_{\mathrm{PT}}$. Note that the x axis represents the probability that a Pauli error occurs $p = p_X + p_Y + p_Z$ and represents the value of p obtained for the mean values of T_1 and T_2 (for the i.n.i.d. channel, since each value of p is different for each qubit, we use the overall mean for the plot).

B. Probability pseudothreshold

The quantum code probability threshold indicates the maximum physical error probability at which increasing the distance—the distance of a planar code scales in terms of the square root of the number of qubits n, i.e., $d[O(\sqrt{n})]$—of the code lowers the logical error probability [5]. Thus, if the physical error probability is below this threshold, $p < p_{\mathrm{th}}$, adding qubits to the error correction code will result in better code performance. Determining the code threshold is one of the primary ways to benchmark the performance of surface codes in the literature when i.i.d. noise is considered [27]. However, as can be seen in Fig. 2, when the differences in the T_1 and T_2 values of physical qubits that make up quantum systems are accounted for, there is no longer a threshold physical probability value at which the performance of surface codes with different distances crosses. While in the top subfigure of Fig. 2 all the distance curves converge onto the same point, we can see in the bottom subfigure that this convergence region has spread out.[2]

To circumvent this issue and maintain our ability to benchmark the quality of planar codes over the i.n.i.d. channel, we apply the so-called probability pseudothreshold [28,29]. The code pseudothreshold is the physical error probability at which the logical error rate meets the physical error probability, $P_L(p_{\mathrm{PT}}) = p_{\mathrm{PT}}$. More specifically, it is the physical error rate at which a code of distance d performs as well (or as poorly) as an uncoded system. This physical error probability value can be thought of as the point beyond which building the error correction code will defy its own purpose, as the code will fail more frequently than the system it seeks to correct. The p_{PT} of Fig. 2 can be seen in the intersections of each distance curve with the green line.

C. Minimum weight perfect matching decoder

In surface codes, decoding a syndrome is equivalent to finding paths between the stabilizer generators the syndrome elements of which have been triggered. We employ the minimum weight perfect matching (MWPM) decoder to estimate the errors that have taken place based on the measured syndrome [27,30,31]. Multiple paths are associated to a given syndrome and the task of the decoder is to produce an estimate of the path that is associated to the error that has the highest probability of taking place. The MWPM decoder finds a solution to this problem by searching for the minimum weight path within the lattice. In graph theory, the MWPM problem is described as that of finding a matching (a set of edges without common vertices) the weight sum (the sum of edge weights) of which is minimized. The term *perfect* refers to the fact that the matching matches all vertices of the graph [30]. The lattice of a surface code that has suffered an error can be converted to a complete graph, where the generators with nontrivial syndrome components are the nodes [31]. The edges between the vertices have a weight equal to the minimum number of qubits between them. In this manner, by deriving a graph associated to the code lattice, solving the MWPM problem by finding the path of minimum weight over this graph serves to produce an estimate of the most likely error that the code has suffered [30].

The MWPM algorithm is an effective method for low physical error probabilities, but numerical methods have shown that at a given threshold of around $p* = 10.3\%$ its performance drops severely. This occurs because at such noise levels, the error operator with the highest probability (the decoder estimate) does not usually belong to the error equivalence class with the highest probability [32]. This is conventionally known as the degeneracy of quantum error correcting codes [33].

III. INDEPENDENT NONIDENTICALLY DISTRIBUTED DECOHERENCE MODEL

The decoherence induced errors in superconducting qubits arise mainly from the combination of energy relaxation and

[2]Notice that different distance curves cross at different p values.

pure dephasing processes. The so-called combined amplitude and phase damping channel, $\mathcal{N}_{\text{APD}}$, provides a fairly complete mathematical abstraction of the aforementioned processes that corrupt quantum information [6,7]. Simulating the combined amplitude and phase damping channel requires an exponential amount of resources, and so it is impractical to track the effects of this channel through classical means. However, by invoking the well-known twirling technique, we can obtain a more symmetric version of the amplitude and phase damping channel that preserves the noise dynamics of the original quantum channel and that can also be simulated on a classical machine [6]. Additionally, it has been shown that correctable codes constructed for this twirled version of the channel will also be correctable codes for the original channel (up to unitary correction) [34]. Thus, it is a common convention in the field of QEC for quantum coding theorists to work with the twirled approximated channels in order to design and simulate QECCs. In particular, in this paper we consider the Pauli twirled approximation (PTA) of the $\mathcal{N}_{\text{APD}}$ channel, denoted by $\mathcal{N}_{\text{APDPTA}}$, as our primary decoherence model. This twirled approximation is obtained by averaging the original channel over the set of unitaries defined by the Pauli group, which results in a Pauli channel, $\mathcal{N}_{\text{APDPTA}}(\rho) = (1 - p_X - p_Y - p_Z)\rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z$, with the following probabilities [6]:

$$\begin{aligned} p_{\text{I}} &= 1 - p_X - p_Y - p_Z, \\ p_X = p_Y &= \frac{1}{4}\left(1 - e^{-\frac{t}{T_1}}\right), \\ p_Z &= \frac{1}{4}\left(1 + e^{-\frac{t}{T_1}} - 2e^{-\frac{t}{T_2}}\right), \end{aligned} \tag{2}$$

where $I, X, Y,$ and Z are the identity, bit-flip, bit-and-phase-flip, and phase-flip Pauli matrices, respectively. Notice how the probabilities that the Pauli operators have of taking place are directly related to the relaxation time, T_1, and the dephasing time, T_2.

The literature on quantum error correction usually considers that each of the qubits of the system is subjected to a noise operation which is independent and identically distributed [9–18]. This implies that each particular qubit will have the same probability of suffering a particular Pauli operator within a given error correction block. We refer to this model as i.i.d. noise. Against this backdrop, assuming that we have an n-qubit system, the i.i.d. channel that arises can be described as

$$\begin{aligned} \mathcal{N}_{\text{APDPTA}}^{(n)}(\rho) &= \mathcal{N}_{\text{APDPTA}}^{\otimes n}(\rho, \mu_{T_1}, \mu_{T_2}) \\ &= \sum_{A \in \{I,X,Y,Z\}^{\otimes n}} p_A(\mu_{T_1}, \mu_{T_2}) A\rho A, \end{aligned} \tag{3}$$

where $A = A_1 \otimes \cdots \otimes A_{n-1} \otimes A_n$ with $A_j \in \{I, X, Y, Z\}$ denotes each of the possible n-fold Pauli error operators; probability distribution

$$p_A(\mu_{T_1}, \mu_{T_2}) = \prod_{j=1}^{n} p_{A_j}(\mu_{T_1}, \mu_{T_2}), \tag{4}$$

with $p_{A_j}(\mu_{T_1}, \mu_{T_2})$ described by Eq. (2); and μ_{T_1} and μ_{T_2} represent the mean values of the relaxation and dephasing times averaged across n qubits.

However, real T_1 and T_2 measurements for various modern superconducting processors disprove the assumption that all the qubits of a superconducting processor are subjected to the same level of noise [19–23]. The actual values of these parameters vary substantially from qubit to qubit (these differences can sometimes reach an order of magnitude), which, naturally, cannot be reflected by the noise model of (3) (see Appendix A for more details). For this reason, we must come up with a noise model that can account for such qubit behavioral differences. Therefore, we will consider that the errors experienced by each of the qubits of quantum systems are governed by probability distributions that are independent, and nonidentically (i.n.i.d.) distributed. This means that the values of p_X, p_Y, and p_Z for each of the qubits within the system will be different. Thus, we will refer to this model as i.n.i.d. noise. Following this rationale, these i.n.i.d. n-qubit channels will have the following structure:

$$\begin{aligned} \mathcal{N}_{\text{APDPTA}}^{(n)}(\rho) &= \bigotimes_{j=1}^{n} \mathcal{N}_{\text{APDPTA}}\left(\rho, T_1^j, T_2^j\right) \\ &= \sum_{A \in \{I,X,Y,Z\}^{\otimes n}} p_A\left(\left\{T_1^j\right\}_{j=1}^n, \left\{T_2^j\right\}_{j=1}^n\right) A\rho A, \end{aligned} \tag{5}$$

where $A = A_1 \otimes \cdots \otimes A_{n-1} \otimes A_n$ with $A_j \in \{I, X, Y, Z\}$ denotes each of the possible n-fold Pauli error operators with probability distribution

$$p_A\left(\left\{T_1^j\right\}_{j=1}^n, \left\{T_2^j\right\}_{j=1}^n\right) = \prod_{j=1}^{n} p_{A_j}\left(T_1^j, T_2^j\right), \tag{6}$$

with $p_{A_j}(T_1^j, T_2^j)$ described by Eq. (2).

Finally, it is important to state that there are other sources of errors that do not stem from environmental qubit interactions that may also be taken into account to study surface codes. These errors are caused by faulty implementations of gates (gate errors) and measurements that are inaccurate (measurement errors) [27]. Considering these additional sources of corruption is important to construct surface codes that are effective, but it is outside the scope of this paper. Herein, we limit our analysis to studying the impact that including the differences in qubit T_1 and T_2 values can have on the performance of error correction codes.

IV. REWEIGHTED MWPM

Conventional minimum weight perfect matching decoding suffers harsh performance degradation when it is applied to decode a surface code exposed to i.n.i.d. noise (this is shown further on in the Results section). Primarily, this loss stems from the fact that the qubits of the code are no longer identical, which means that some will perform better than others. The standard MWPM decoder considers that the minimum weight set of chains matching the measurement qubits with one-syndrome contribution is the most probable, where all the edges are of the same weight. Unfortunately, this no longer holds when the physical qubits of the code exhibit different error parameters. Nonetheless, it is possible to substantially minimize the degradation suffered by MWPM decoders over the i.n.i.d. channel by applying a set of modifications to the decoding algorithm.

Once a surface code experiences an error, a syndrome can be extracted by measuring the measurement qubits. Since the planar code is a CSS code, the syndrome result is mapped onto two subgraphs, a check-X subgraph, susceptible to physical X and Y errors, and a check-Z subgraph, susceptible to Z and Y errors. In both subgraphs, the respective measurement qubits act as nodes while their adjacent data qubits act as the edges that connect each measurement qubit to its four nearby measurement qubits. The MWPM decoder applied for i.i.d. channels resolves the graph problem by considering all the weights of the subsequent graph to be equal, which results in "close" measurement qubits (syndromes) being connected via lower weight paths. However, the equal weight assumption is inappropriate when facing i.n.i.d. noise. Over this more restrictive channel, because each qubit suffers different levels of noise, the weights of the edges must be reweighted according to the error parameters of the particular data qubits they represent. We refer to this modified decoding approach as reweighted MWPM decoding. The weights we use in reweighted MWPM (rMWPM) decoding are different for each subgraph, as each subgraph relates to a different error recovery:

$$\begin{aligned} w_{i,X} &= -\ln(1-e^{-t/T_{1,i}}) \propto -\ln(p_{x,i}+p_{y,i}),\\ w_{i,Z} &= -\ln(1-e^{-t/T_{2,i}}) \propto -\ln(p_{z,i}+p_{y,i}), \end{aligned} \tag{7}$$

where $T_{1,i}$ and $T_{2,i}$ are the relaxation and dephasing parameters specific to the data qubit of the edge i, t indicates the time that has passed since the code was initialized, and p_i indicates the probability of a qubit to experience an error of type i. This weight consideration significantly increases the complexity of the graph problem, since the distance between two syndromes can no longer be determined through the taxicab metric [32]. Instead, we use Dijkstra's algorithm [35] to determine the weight of the minimum weight paths between syndrome one-measurement qubits. Dijkstra's algorithm finds the shortest path between nodes in a graph and has a maximum complexity of $O[N\ln(N)+M]$, where N is the number of nodes within the graph and M represents the number of edges. Based on our weight convention, the weight of a chain or cochain between two syndromes i and j in a subgraph k will be given by

$$\sum_{k=i}^{j} -\ln(p_{k,l}) = -\ln\left(\prod_{k=i}^{j} p_{k,l}\right), \tag{8}$$

where $p_{k,l}$ is the probability of errors susceptible to the syndromes of the k subgraph for a qubit l. Higher failing probabilities p_k will imply a lower weight and, thus, longer chains and cochains with worse performing qubits will weigh less than shorter ones with better data qubits. This redistribution of weights alters the minimum weight perfect matching result and enhances the performance of the code. In Fig. 3 we can see an example of how the rMWPM decoding process unfolds in the corresponding graphs. Subfigure 1 shows the error experienced by a 5×5 planar code along with the resulting one-measurement qubits (indicated with exclamation marks). Subfigures 2 and 3 represent the check-X and check-Z measurement qubit subgraphs, respectively. Notice that the data qubits lay over the edges of the graph. Data qubits with high probability of failing in each subgraph are depicted as pink circles, while those qubits with longer relaxation times are depicted in blue. Subfigure 4 portrays the overall graph comprised by the two independent check-X and check-Z subgraphs. Using different weight conventions results in different decoding outputs. This can be seen by comparing subfigures 5 and 6, which represent the result of applying a conventional MWPM decoder and that of using the rMWPM decoder, respectively. Consequently, the recovery operators proposed by each of these decoders will also be different (pictures 7 and 8). The MWPM decoder has prioritized the lowest Hamming weight error while the rMWPM has accounted for the individual noise parameters of each data qubit in order to select the best-possible recovery operator.

Later on in the Results section we will see how the rMWPM decoding rule significantly improves the performance of generic MWPM decoding when facing i.n.i.d. errors. Nonetheless, it must be mentioned that it does so at the expense of a higher decoding complexity. Reweighted MWPM decoding has a complexity of $O[n^3\ln(n)]$ while the conventional MWPM decoder has a complexity of $O[n^2\ln(n)]$ [32], where n represents the number of data qubits within the square planar code. The source of this increase in complexity comes from the introduction of Dijkstra's algorithm, which has $O[n\ln(n)]$ complexity [conventional MWPM decoding uses the taxicab distance which has complexity $O(1)$].

V. ARCHITECTURE OPTIMIZATION METHOD

In this subsection we describe the guidelines that make up another strategy that can be employed to improve the performance of planar codes when they are subjected to i.n.i.d. noise. It involves rearranging the planar code lattice qubits according to the noise they suffer. To start, consider the fact that some qubits of the lattice are less likely to experience errors than others. Naturally, this occurs because some qubits have longer relaxation and dephasing times than others. Against this backdrop, we can place the qubits on the lattice in a way in which better qubits (less likely to fail) are positioned in the most important sites, effectively minimizing the probability of harmful events (chains and cochains).

The overall planar code lattice that encodes a logical qubit can be be split into two separate sublattices. Both of these sublattices are shown in Fig. 4. The sublattice composed by qubits placed along the horizontal edges of the overall lattice does not have measurement qubits on its edges. Consequently, a horizontal Z_i chain and a vertical X_i cochain will commute with all the stabilizer generators and, thus, will be in the code. On the other hand, the vertical edge sublattice has four adjacent measurement qubits for all data qubits; hence, any chain or cochain will be detected by the measurement qubits located at its end points.

In order to accurately reorder the qubits that make up a planar code in a way that improves performance, it is important that we come up with a way to differentiate good qubits from bad qubits. However, accurately classifying qubits according

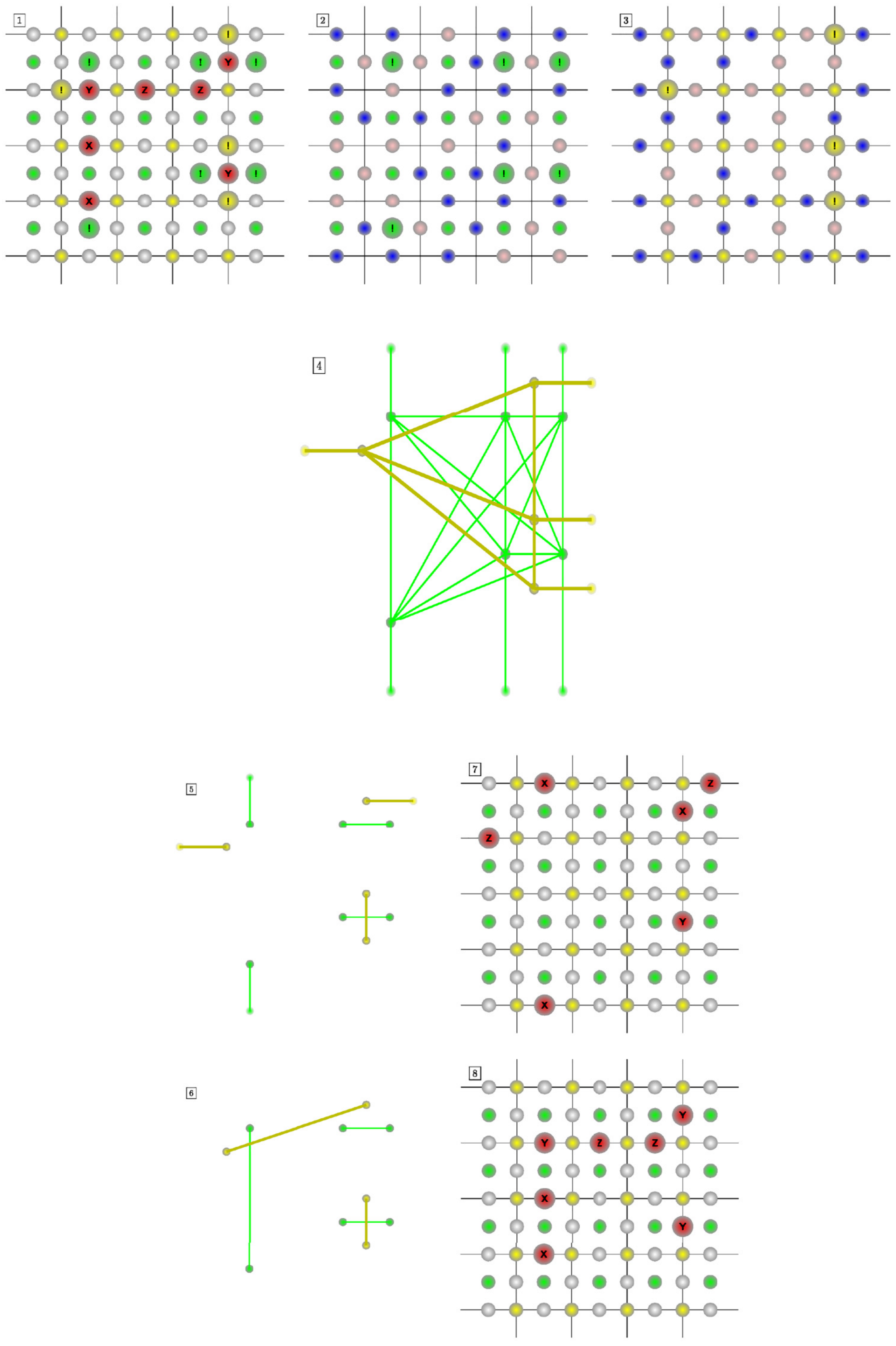

FIG. 3. Example of a MWPM and rMWPM performance towards an error in a 5 × 5 planar code. Diagram 1 shows the proposed error composed by data qubits (white circles), data qubits experiencing nontrivial error operators (red circles), measure-z qubits, and measure-x qubits (green and yellow circles). One-syndrome measurement qubits are labeled with an exclamation mark. Diagrams 2 and 3 represent the two CSS subgraphs, where pink and blue denote the qubits with lower and higher relaxation parameters, respectively. Diagram 4 shows the overall graph. Diagrams 5 and 6 show the minimum weight perfect matching of the graphs considering taxicab distance and reweighted distance, and diagrams 7 and 8 show the recovery operators proposed by the MWPM and rMWPM decoders.

to their “quality” is not a simple task. Since the physical error probability that each qubit experiences varies as a function of its relaxation and dephasing times $\{T_1^j\}_{j=1}^n$, $\{T_2^j\}_{j=1}^n$, defining a metric that determines how good a specific qubit is with regard to the rest of the ensemble is relatively nuanced. For this reason, in our algorithm we employ the lowest relaxation time as our noise “quality” indicator, i.e., a larger relaxation time implies a better performing qubit. Note that dephasing is

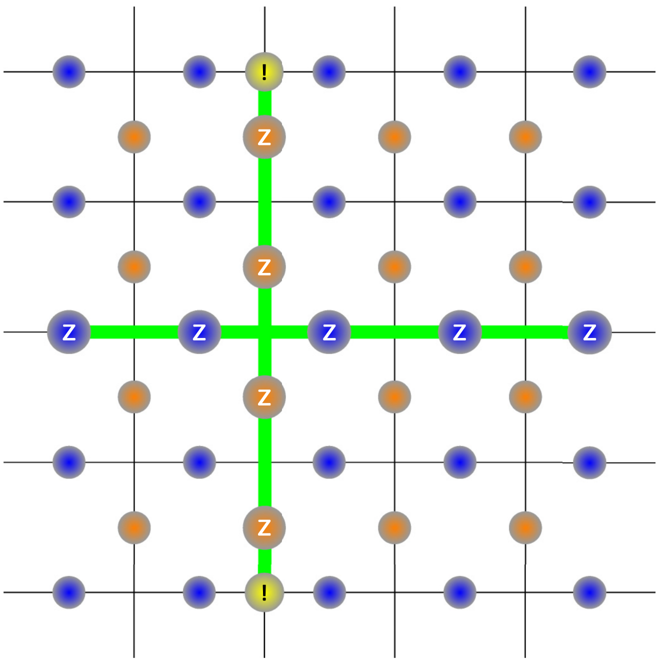

FIG. 4. The 5×5 planar code codifying the information of a logical qubit. The data qubits of both sublattices are highlighted with identifiable blue and orange qubits. A Z chain of weight 4 in the orange lattice is detected by two measuring qubits highlighted in yellow with an exclamation mark. Moreover, a Z chain in the blue sublattice commutes with all the measuring qubits.

intricately related to relaxation: $1/T_2 = 1/2T_1 + 1/T_\phi$, where T_ϕ is the pure dephasing time [7].

The minimum weight of a logical operator in an $N \times N$ planar code is N. For the example in Fig. 4, logical operators will have a minimum weight of 5. For a logical error caused by a chain to take place, a minimum of N horizontal edge qubits must experience an error. If we place the best qubits in the horizontal edge sublattice (blue in Fig. 4), we will make logical error chains and cochains more unlikely, which should ultimately lead to better code performance. Moreover, placing the worst qubits within the vertical edge sublattice (orange in Fig. 4) guarantees that, at least, half of the nearest data qubits will be good ones. Based on these ideas, the algorithm we have designed to optimize the architecture of planar codes operates based on the following two principles.

(1) Surround good qubits with bad qubits and vice versa. This is done to prevent the propagation of errors through the code and the formation of chains and cochains.

(2) Separate differently performing qubits in both sublattices. While the best qubits are in the d^2 (blue in Fig. 4) sublattice, the worst ones will be in the $(d-1)^2$ (orange in Fig. 4) lattice. We do this to make it unlikely for shortest weight (weight d) logical errors to occur. As shown in Fig. 4, a chain or cochain in the orange sublattice does not commute with the stabilizer set and, thus, placing the worst qubits within it ensures that they will not contribute towards decoding failures.

As will be shown in the next section, applying these two guidelines to redesign planar codes ends up improving their performance. In particular, we have concentrated the worst and best performing qubits in the bulk (the center) of the code, while the most average ones have been spread out along the outer walls of the lattice. Additionally, the $(d-1)^2$ worst qubits have been placed in the orange sublattice, preventing them from contributing to the formation of minimum weight logical errors. In this manner, our method minimizes the probability of chain and cochain formation and makes it likelier for the MWPM algorithm to be successful. Furthermore, we will also see in the Results section that the performance of the rMWPM decoder is also improved when optimizing the architecture of the code.

VI. RESULTS

A. Planar code numerical simulation

To estimate the performance of the various $d \times d$ planar codes with $d \in \{3, 5, 7\}$ [27] that we have considered in this paper we have carried out Monte Carlo numerical simulations. We have constructed the planar codes using a customized version of the QECSIM tool [17] that we have modified so that it can work with the i.n.i.d. decoherence model. Each round of a numerical simulation is performed by generating an N-qubit Pauli operator, calculating its associated syndrome, and finally running the one-cycle decoding algorithm using this syndrome as its input. Once the decoder produces an estimation of the channel error, the syndrome is extracted again. For the sake of simplicity and restricting our view to the effects of i.n.i.d. noise, we will not consider measurement errors. Following this second syndrome extraction, we check that the code state commutes with the X and Z logical operators. If the second syndrome is nontrivial or if the quantum state of the code no longer commutes with the logical operator, we will consider the code to have undergone a logical error. The logical error probability is obtained by computing many realizations of the aforementioned procedure. The specific properties of the constituent data qubits can be found in the Appendices.

To estimate the logical error rate, P_L, of the planar codes, we choose $N_{\text{blocks}} = 10^4$ Pauli error realizations for each considered value of the physical error probability p. In this way, we can guarantee that the estimated values of the logical error probability are accurate because we fulfill the Monte Carlo rule of thumb [36]

$$N_{\text{blocks}} = \frac{100}{P_L}. \tag{9}$$

This rule tells us that, under the assumption that the observed error events are independent, the estimated value, $\hat{P}_L$, lies in the 95% confidence interval of about $(0.8\hat{P}_L, 1.25\hat{P}_L)$.

Finally, we estimate the average performance of the planar codes for the particular relaxation and dephasing rates of each system by performing Monte Carlo simulations in the order of 10^3 randomized qubit arrangements defined over the particular planar code lattices.

B. Independent nonidentical distribution model performance

Our simulation results are depicted in Fig. 5. The consideration of i.n.i.d. noise harshly decreases the probability pseudothreshold of all the codes, a detriment that ranges

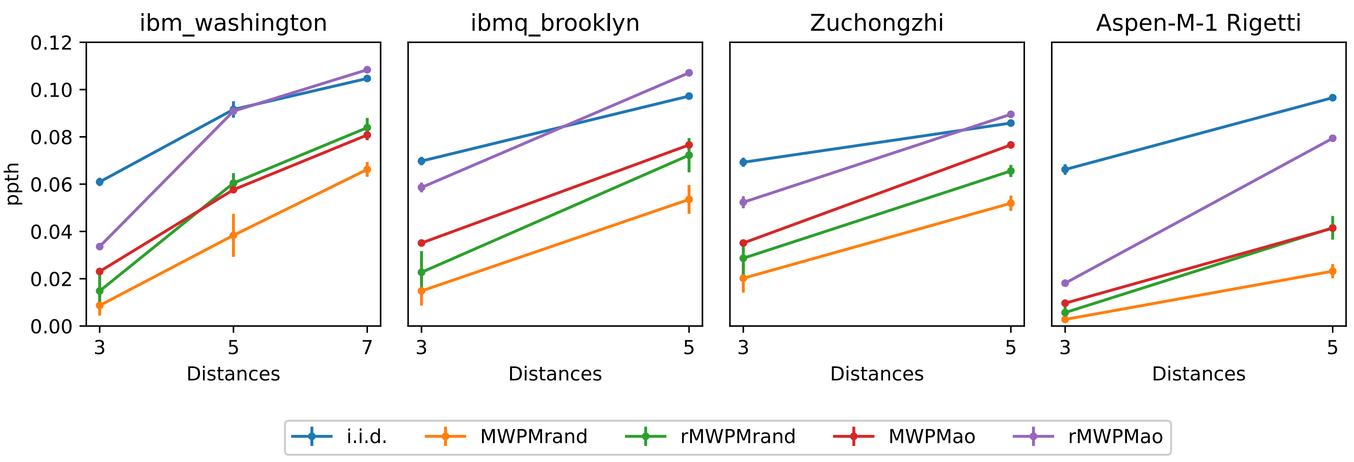

FIG. 5. Code pseudothresholds obtained for the qubit data of four quantum processors in different distance square planar codes. From left to right: ibm_washington, ibmq_brooklyn, Zuchongzhi, and Aspen-M-1 Rigetti. The blue dots represent the results under i.i.d., and the yellow dots reflect the performance of the code under i.n.i.d. using the conventional MWPM decoder. The green dots represent the performance of the code under the rMWPM decoder, and the red dots represent the performance of the conventional MWPM decoder but with the code undergoing the architecture optimizing method. Lastly, the purple lines indicate the performance of the code under both architecture optimizing and the rMWPM decoder. The bars correspond to the standard deviation of all the considered configurations of the specific code.

from 40 to 95% of the original i.i.d. noise scenario. Such a significant loss in performance is a direct consequence of assuming that drastically different qubits behave equally. This degradation may also have been exacerbated because we have considered the qubits with the highest and lowest relaxation parameters. Thus, the codes of lower distance have higher coefficients of variation. The average T_2 is much smaller and its coefficients of variation much higher than T_1; thus, the code's performance is restricted by the dephasing times of its qubits. Codes with relaxation parameters with low coefficients of variation will suffer less from i.n.i.d. noise, since the individual relaxation times of the qubits will tend to be closer to the average.

C. Performance of the rMWPM

Figure 5 shows how the rMWPM decoder outperforms the conventional MWPM decoder when i.n.i.d. noise is considered. For distance 3 planar codes, its standard deviation overlaps with the MWPM standard deviation in all of the scenarios we have tested. This is a result of the high probability that "bad" qubits have of being placed in pivotal positions at such low distances, which contributes to the formation of distance 3 chains and cochains operating nontrivially over the encoded state. Regardless, for distance 3 codes, the average performance of the rMWPM decoder exceeds that of the MWPM decoder by up to 104%. At distances 5 and on, the standard deviations of the rMWPM and MWPM decoder performance curves no longer overlap, but the improvement is not so significant (it ranges from 27 to 79%). This can be understood more so as a scenario change rather than a decrease in the boost provided by the rMWPM decoder. As more qubits are used to build codes with larger distances, there will be a lower coefficient of variation between qubits (more average qubits are introduced into the lattice). Consequently, the i.n.i.d. effect is not as significant as in the distance 3 scenario.

D. Performance of the architecture optimization method

Similar to the rMWPM decoder in the previous subsection, the architecture optimization method also surpasses the performance of random data qubit layouts in all of our simulations (improvements of 22 to 247%). In Fig. 5 we can see how at distance 3, under high coefficients of variance in T_2, the specific allocation of qubits within the planar code allows us to isolate the worst performing qubits. Unfortunately, because the worst qubits underperform at high rates, one-element syndromes can end up being separated by large distances, which tricks the MWPM decoder into choosing a wrong recovery operator. As higher distances are considered and the worst qubits can be further isolated within the bulk of the surface code, the improvement in performance provided by the architecture optimization method increases (it comes close to the i.i.d. scenario in particular situations).

When compared to the rMWPM decoder we observe that for low distance planar codes, the architecture optimization method yields better results because it successfully isolates the worst behaving qubits. As longer distance codes are tested, the rMWPM decoder gets closer and even ends up surpassing the architecture optimizer method. The exact arrangement of the qubits under the architecture optimization method for each processor can be found in Appendix B.

E. Performance of the combined rMWPM and optimization method

The true potential of the rMWPM decoder and the architecture optimization strategy comes to light when they are

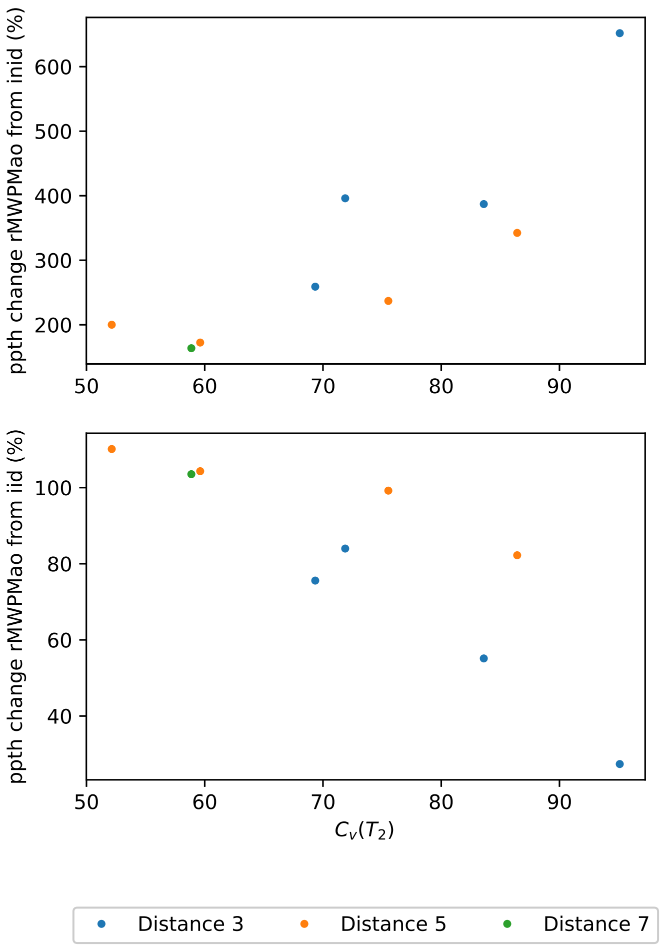

FIG. 6. Change in p_{PT} when considering the optimized architecture and the rMWPM compared to two different scenarios with respect to the variance coefficient of T_2. At top, the change in p_{PT} is compared with the i.n.i.d. case under the conventional MWPM decoder. At bottom, it is compared with the i.i.d. case.

applied together. While the architecture optimizing method ensures that the worst qubits of the planar code are surrounded by better qubits and these bad qubits in the vertical edge bipartite lattice, the rMWPM decoder accounts for the weight asymmetry in the nontrivial syndrome element graph. As a result, the combination of both of these methods produces performance increases that are 163 to 650% better than applying a conventional MWPM decoder over the i.n.i.d. model. In some cases, the amalgamation of both of these methods surpasses the performance of the MWPM decoder under i.i.d. considerations.

The performance of planar codes under both conventional MWPM decoding as well as the methods we introduce in this paper is highly correlated with the coefficient of variation of the restricting relaxation parameter, $C_v(T_2)$. As can be seen in Fig. 6, a larger $C_v(T_2)$ implies a harsher drop in performance, but it also increases the yield that our methods provide. Surpassing of the i.i.d. threshold is only achieved when $C_v(T_2)$ is lower than 60%, where $C_v(T_2) = \sigma(T_2)/\mu(T_2)$, and $\sigma(T_2)$ and $\mu(T_2)$ are the standard deviation and average of the set of T_2 values.

VII. CONCLUSION

In this paper we have proposed the i.n.i.d. noise model as an appropriate way to include the observed variance in the decoherence parameters of the qubits that make up superconducting quantum processors. Our results show how the performance of planar codes when this noise model is considered is far worse than what would be expected based on previous results obtained for the i.i.d. qubit noise model. This occurs because when the noise of the qubits of a surface code is considered to behave according to an i.n.i.d. noise channel, the qubits that are most likely to suffer errors (those with shorter T_1 and T_2) may form errors chains and cochains that are too large for the decoder to successfully estimate, which ultimately results in the manifestation of logical errors.

We have also discussed how the manner in which qubits are arranged on the planar code lattice plays an important role in the performance of the codes. We saw how the "typical" performance of planar codes over the i.n.i.d. channel is generally bad, an outcome that arises from increased likelihood that these codes have of suffering additional logical errors over the i.n.i.d. paradigm ("bad" qubits end up being located in lattice positions that cause harmful events to occur frequently). We address this issue in our paper by devising two methods that tackle the inconvenience of "low quality" qubits. The first method consists in reweighting the graph on which the minimum weight perfect matching algorithm is applied. This new weight convention is directly related to the relaxation parameters and the relaxation time. In this manner, worse qubits are prioritized (over those that have larger T_1 and T_2 values) as the potential error sources.

The second technique comes down to placing qubits on the surface code lattice in a manner that guarantees that long chains and cochains are far less probable (making use of the better qubits available). We refer to this strategy as the architecture optimizing algorithm. This method enables us to prevent the placement of the worst qubits in lattice sites that are pivotal to the performance of the code. The primary working principle of the algorithm is ranking the surface code qubits according to their noise level (T_1 and T_2). Once this best-to-worst classification is defined, the "worst" qubits are surrounded by better qubits in order to prevent the creation of chains and cochains, which ultimately leads to higher probabilities of successful decoding.

Both the rMWPM decoder and the architecture optimizing method improve the performance of the code significantly, an effect that is further exacerbated when the methods are combined. In fact, when they are applied jointly, performance can exceed results obtained for the i.i.d. scenario. Unfortunately, these improvements come at a price. On the one hand, rMWPM decoding can result in longer decoding times for large surface codes due to its increased complexity. For decoding to be practical, it must be performed in real time so that the noise that takes place actually corresponds to the measured syndrome. Methods to reduce the complexity of the MWPM have been proposed [32,37], but they also decrease its accuracy. Furthermore, recent results show that T_1 and T_2 fluctuate both intercooldown and intracooldown [7,38–42], thus any method which is based on the knowledge of the T parameters would need to be significantly faster than the lifetime of said changes. Additionally, the architecture optimization method has the particular drawback that, at this moment in time, it is likely that rearranging the position of the superconducting qubits that form real quantum processors once they are manufactured is not possible. It also seems unreasonable to attempt to use SWAP gates to reorder the physical qubits of the code [note that SWAP gates are constructed using controlled-NOT (CNOT) gates that are really noisy at this moment in time]. In any case, the results provided herein prove that the fact that each physical qubit has its own noise dynamics is critical for code performance and should not be neglected when studying quantum error correcting codes.

Moreover, even though the results in this paper have been restricted to the planar code scenario, they could be extended to other surface codes of relevance such as the toric code or the rotated planar code [43]. The decoding of different surface codes through the MWPM differs in the graph the nontrivial syndrome elements are mapped into. For the case of the toric code, the graph would be the same as the planar codes considered in this paper, but with periodic conditions on the boundaries. However, due to the periodicity of the lattice, logical operators are more prone to occur and, thus, a worse performance is expected for i.n.i.d. noise. On the other hand, the rotated planar code requires a decrease of the number of data qubits and stabilizers for the same distance. Therefore, adapting the reweighted MWPM decoder would still be a recommended step in order to mitigate the effects of i.n.i.d. quantum noise. Nevertheless, in both the toric and the planar code the argument of the sublattice which does not contain minimum distance error no longer holds. Thus, other methods should be taken into account when considering surrounding good qubits with bad qubits.

Another takeaway from our paper is the fact that in order for real planar codes (and other QEC codes for that matter) to perform at the rates promised in the literature, quantum hardware must be composed of qubits with uniform relaxation and dephasing times. Traditionally, the literature on implementation of superconducting qubits has based its elemental hardware quality claims on best-case or mean scenarios. However, it is the actual distribution of these parameters, not just the best-case or the mean values, that is most relevant to predict how good the surface codes that will operate on such hardware can be. Another possible approach is to consider that quantum systems are limited by their worst qubit and to assume that all the constituent qubits of the system behave like their “weakest link.” However, this would be a somewhat reductionist view that would miss out on the code performance improvements that can be obtained by considering qubit differences (as is done by our architecture optimization method). Thus, by designing codes for performance over the i.n.i.d. channel, one may achieve lower qubit overhead for similar code performances. This is an important outcome, since qubits are an expensive resource.

Another important issue that plays a role in the performance of real surface codes is that of gate and measurement errors in quantum hardware. We have excluded the presence of these phenomena from this paper, but they are an important source of errors that should be studied. While the literature on surface codes has already considered these error sources [27], in a similar manner to what happens for decoherence parameters, real superconducting quantum hardware will suffer different gate and measurement errors for each of their constituent qubits. This is an important problem and optimizing surface codes for such nonuniform scenarios is germane to the field of QEC.

We also believe that it is important to consider the i.n.i.d. noise model for other quantum computing tasks, not just for the purpose of QEC. For example, quantum error mitigation, which is an important approach to deal with noisy quantum algorithms running on real quantum hardware, should also operate under the framework of the i.n.i.d. noise model. As of today, qubit number and connectivity is not yet adequate to implement strong error correction strategies, and so quantum error mitigation techniques[3] are an important component of the modern quantum computing paradigm. It is possible that accounting for the nonuniformity of the noise levels that current qubits experience will lead to further improvements in mitigation techniques. Lastly, it may also be possible to improve the reliability of current quantum computers by compiling quantum algorithms specifically for the hardware that they will be executed on (accounting for the T_1 and T_2 values of individual qubits plus the individual gate and measurement error rates).

The data and code that support the findings of this paper are available from the corresponding authors upon reasonable request.

ACKNOWLEDGMENTS

This work was supported by the Spanish Ministry of Economy and Competitiveness through the ADELE project (Grant No. PID2019-104958RB-C44), by the Spanish Ministry of Science and Innovation through the project “Few-qubit quantum hardware, algorithms and codes, on photonic and solid-state systems” (Project No. PLEC2021-008251), and by the Diputación Foral de Gipuzkoa through DECALOQC Project No. E 190/2021 (ES). This work was funded in part by NSF Grant No. CCF-2007689.

J.E.M. and P.F. contributed equally. J.E.M. and P.F. conceived the research and J.E.M. proposed the model. A.dM.iO, J.E.M., and P.F. proposed the rMWPM method. A.dM.iO. proposed the architecture optimization algorithm and performed the numerical simulations. A.dM.iO., J.E.M., and P.F. analyzed the results and drew the conclusions. The paper was written by A.dM.iO., J.E.M., and P.F., and revised by P.M.C. and J.G.-F. The project was supervised by P.M.C. and J.G.-F.

APPENDIX A: RELAXATION AND DEPHASING TIMES OF REALISTIC QUANTUM HARDWARE

In the primary text we make the claim that most of the currently existing state-of-the-art superconducting quantum processors are made up of qubits the relaxation and dephasing times of which are not the same. In what follows we present

[3]Quantum error mitigation achieves error suppression by sampling available noisy intermediate-scale quantum devices many times and classically postprocessing these measurement outcomes.

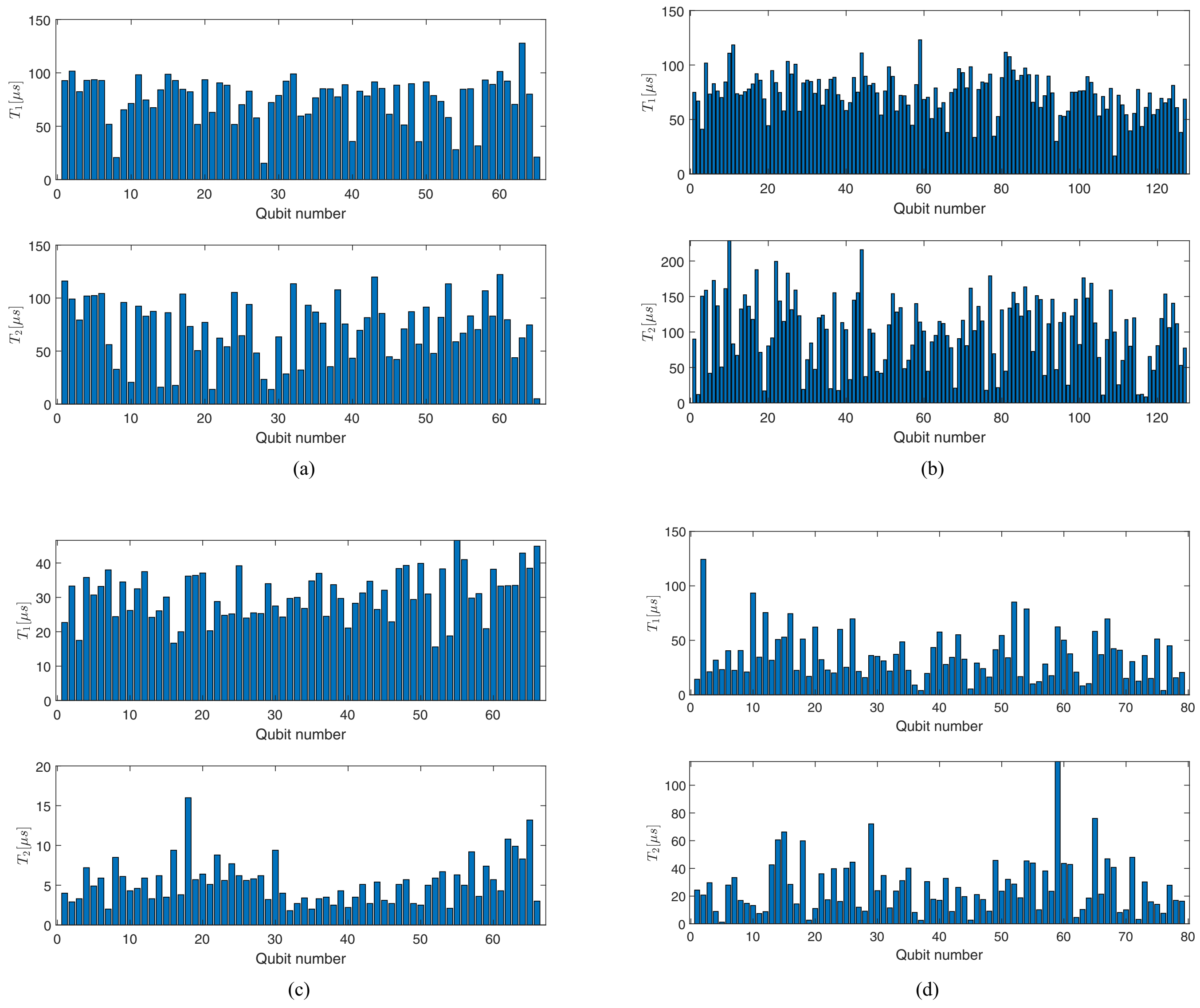

FIG. 7. Considered relaxation and dephasing times for the i.n.i.d. decoherence model. The T_1 and T_2 values associated to the individual qubits of each of the systems are presented using histograms. (a) Data for the ibmq_brooklyn machine with 65 qubits. Timestamp: 16/11/2021 10:41 (CET) [21]. (b) Data for the ibm_washington machine with 127 qubits. Timestamp: 16/11/2021 12:21 (CET) [20]. (c) Data for the Zuchongzhi machine with 66 qubits. Data are from [22]. (d) Data for the Rigetti Aspen-M-1 machine with 79 qubits. Data are obtained using the Strangeworks platform [23]. Timestamp: 30/05/2022 11:37 (CET). Note that the Aspen-M-1 quantum computer is made up of 80 qubits, but when we requested the corresponding data we were only able to obtain measurements related to its qubits.

the data that justify this claim, and we discuss how these results inspired the proposal of the i.n.i.d. decoherence model that we present in the main text.

Figure 7 shows the specific values of T_1 and T_2 that we have employed to simulate the performance of planar codes over the i.n.i.d. channel. Recall that this channel model can account for differences in the values of T_1 and T_2 of individual qubits (see the independent nonidentically distributed decoherence model in the main text) [19–23]. The measurements shown in Fig. 7 reveal how the particular relaxation and dephasing times of each individual qubit within the quantum system can vary drastically. For example, the qubits that make up the ibm_washington quantum processor exhibit a minimum relaxation time of 16.54 μs and a maximum relaxation time of 123.11 μs, i.e., there are qubits the relaxation time of which differs by an order of magnitude. This phenomenon is further exacerbated for the ibm_washington qubit dephasing times. The minimum dephasing time value is 8.58 μs and the maximum value is 228.56 μs. This behavior can be observed over all of the superconducting machines considered in this paper. We summarize the minimum and maximum T_1 and T_2, as well as their mean values, in Table I. The main takeaway here is that, within the real quantum systems, the decoherence parameters of each constituent qubit will vary significantly. Because this type of behavior must be considered when building accurate decoherence models, the i.n.i.d. noise model we propose in the main text is a relevant contribution to the field of QEC, as it can accurately reenact the real quantum noise processes that multiqubit systems can suffer.

TABLE I. Minimum, maximum, and mean T_1 and T_2 for the systems considered in our paper. The values are obtained from the data provided in Fig. 7.

Processor	min T_1 (μs)	max T_1 (μs)	μ_{T_1} (μs)	min T_2 (μs)	max T_2 (μs)	μ_{T_2} (μs)
ibmq_brooklyn	15.37	127.82	75.3554	5.06	122.19	70.4778
ibm_washington	16.54	123.11	74.2827	8.58	228.56	101.4081
Zuchongzhi	15.6	46.6	30.6045	1.8	16	5.3348
Rigetti Aspen-M-1	3.95	124.35	35.77	1.22	117.08	26.5

Some details regarding the data shown in Fig. 7 merit further discussion. To start off, notice how the T_1 and T_2 values we have considered are all timestamped (refer to Fig. 7). This is related to the fact that these values can vary between calibration rounds (intercalibration) and even during the calibration process itself (intracalibration) [38–42]. The data that quantum computing companies make available are generally updated intercalibration (from calibration to calibration), so it is important that we state that the data we have employed in our analysis strictly relate to different calibration cycles of various quantum machines. Because intracalibration fluctuations are not usually reported, even if they are important [7,39–42], it is not something that we have been able to consider in the present paper.

Additionally, we must also disclose that part of the available data do not make physical sense. For example, the data reported for qubit 3 of the ibm_washington quantum processor tell us that $T_1 = 41.09$ µs and that $T_2 = 150.47$ µs. It is physically impossible for these values to be correct, since they do not comply with the Ramsey limit $T_2 \leqslant 2T_1$. We believe that these "erroneous" readings stem from the fact that T_1 and T_2 measurements are not performed during the same time instant. Because intracalibration decoherence parameter fluctuation can take place [7,39–42], it is likely that by the time the second measurement is run, the decoherence parameters have already changed. In light of this, whenever we encounter such data readings in our paper, we have considered that the qubit in question actually saturates the Ramsey limit ($T_2 = 2T_1$) so that our simulations can be run. This also speaks to the importance of measuring the decoherence parameters of the qubits simultaneously, as this would produce more accurate data. Additionally, this also sheds light on the importance of understanding and characterizing intracalibration decoherence parameter fluctuation, as this phenomenon may also impact the performance of real quantum error correcting codes [7].

APPENDIX B: CONSIDERED PLANAR ARCHITECTURES

In this section, we present the qubit arrangements that have been considered for the planar codes in the main text. This reveals the way in which this new architecture distributes qubits according to their T_1 and T_2 values. The location of a given qubit within the lattice is given by a combination of indices, as is done by convention in the QECSIM library [17]. These indices are computed based on the equation

$$\begin{aligned}\text{index}_{r,c} =&(r\ 2) \times (\text{cols} - c\%2)+\\ &(c\ 2) + (r\%2 \times \text{rows} \times \text{cols}),\end{aligned} \tag{B1}$$

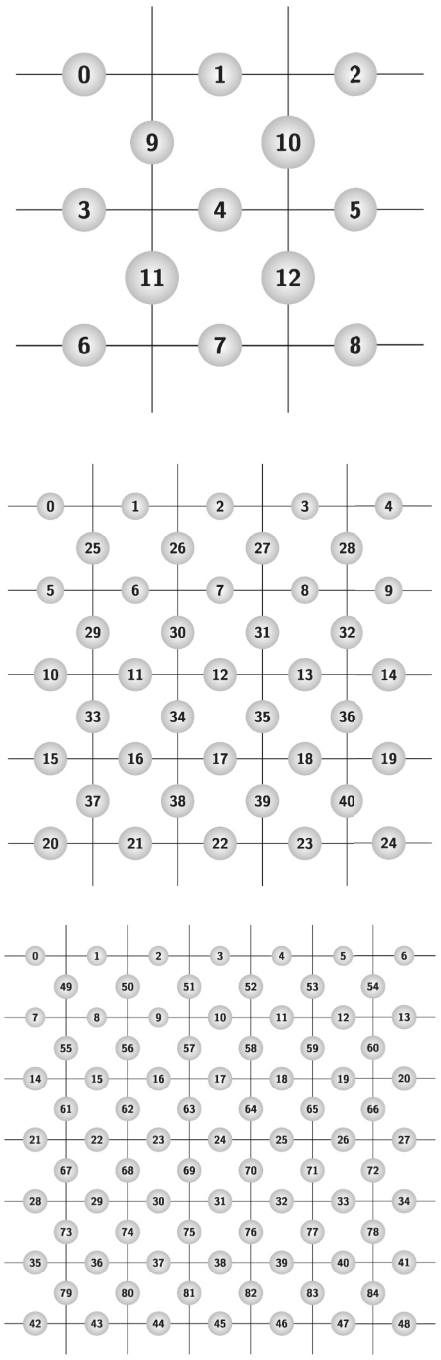

FIG. 8. Graphical representation of the data qubits of 3 × 3, 5 × 5, and 7 × 7 planar codes. Each data qubit is represented by a gray-shaded dot and a number representing its index.

TABLE II. Optimized architecture for the 3 × 3 planar code considering the ibmq_brooklyn quantum processor.

Qubit index	T_1 (μs)	T_2 (μs)
0	84.1	16.0
1	92.9	104.3
2	93.3	106.9
3	99.0	113.5
4	101.3	122.2
5	101.6	99.1
6	93.6	102.4
7	93.0	101.9
8	92.8	17.6
9	63.0	13.9
10	21.1	5.1
11	72.2	13.8
12	15.4	23.3

TABLE IV. Optimized architecture for the 3 × 3 planar code considering the ibm_washington quantum processor.

Qubit index	T_1 (μs)	T_2 (μs)
0	43.6	12.2
1	100.7	159.0
2	103.3	182.8
3	110.9	228.6
4	123.1	114.0
5	111.0	215.8
6	107.6	133.7
7	101.8	158.8
8	16.5	100.0
9	77.6	11.6
10	61.1	8.6
11	71.1	11.2
12	66.9	11.9

where *cols* and *rows* indicate the size of the given planar code, and r and c are the specific row and column of the given qubit. Additionally, the percent sign denotes the modulo-2 operation and the backslash represents the integer or floor division operation. This index-based labeling system allows us to easily distinguish between qubits that are placed on the different sublattices discussed in the main text (the horizontal edge qubits are labeled first followed by the vertical edge qubits). An example of this labeling system is shown in Fig. 8, where the qubit indices for each of the planar code lattices that have been considered in this paper are shown.

Following such indexing, the results of Fig. 1(b) in the main text are based on arranging the qubits using the same indexing of the qubits of the ibm_washington processor. The specific arrangements of the qubits obtained by the optimization algorithm are displayed in Tables II–X. The data shown in these tables correspond to the data presented in Fig. 7 of Appendix A.

Additionally to the provided T_1 and T_2, there are many additional values of relevance for experimental study of real quantum processors. This have not been discussed in the core of the paper because they are not considered in our noise model; nevertheless, they are of importance and thus we will include them in this last section. Since the quantum processors of IBM [19,20] and Aspen [23] are in continuous reparation and improvement, the values often change, thus we provide a series of tables with the most relevant values at the time of our paper in Table XI.

As can be seen in Table XI, the frequency values for the qubits of the Rigetti Aspen-M-1 are empty, which is because the information is not available to the public. Nevertheless, as mentioned in [44], superconducting qubits tend to have a frequency of the order of gigahertz, thus we consider the frequency of the Aspen-M-1 qubits will be of that order. The Zuchongzhi relevant data can be found in [22].

TABLE III. Optimized architecture for the 5 × 5 planar code considering the ibmq_brooklyn quantum processor.

Qubit index	T_1 (μs)	T_2 (μs)	Qubit index	T_1 (μs)	T_2 (μs)	Qubit index	T_1 (μs)	T_2 (μs)
0	70.5	43.8	14	93.6	102.4	28	28.1	28.8
1	78.8	47.8	15	93.0	101.9	29	71.4	20.6
2	78.4	81.6	16	92.7	116.1	30	84.1	16.0
3	92.3	79.6	17	91.6	119.9	31	63.0	13.9
4	89.2	83.0	18	89.9	87.2	32	21.1	5.1
5	84.6	103.8	19	85.4	85.5	33	72.2	13.8
6	98.7	86.2	20	85.1	83.2	34	15.4	23.3
7	91.6	91.5	21	82.9	94.0	35	92.8	17.6
8	98.2	92.4	22	82.4	79.3	36	20.7	32.8
9	92.9	104.3	23	57.8	48.2	37	92.2	28.5
10	93.3	106.9	24	61.3	44.7	38	59.6	32.2
11	99.0	113.5	25	35.8	43.3	39	35.6	56.6
12	101.3	122.2	26	85.0	35.3	40	88.5	42.1
13	101.6	99.1	27	31.6	70.3			

TABLE V. Optimized architecture for the 5 × 5 planar code considering the ibm_washington quantum processor.

Qubit index	T_1 (μs)	T_2 (μs)	Qubit index	T_1 (μs)	T_2 (μs)	Qubit index	T_1 (μs)	T_2 (μs)
0	65.5	33.0	14	107.6	133.7	28	83.5	17.9
1	34.7	69.4	15	101.8	158.8	29	68.9	17.0
2	89.8	111.6	16	98.4	161.7	30	43.6	12.2
3	96.7	90.7	17	97.2	151.2	31	77.6	11.6
4	91.1	130.0	18	93.0	116.5	32	61.1	8.6
5	94.9	91.7	19	91.7	131.4	33	71.1	11.2
6	92.0	187.7	20	91.6	179.1	34	66.9	11.9
7	95.3	155.8	21	90.7	151.2	35	16.5	100.0
8	98.3	110.1	22	90.5	122.4	36	72.8	17.6
9	100.7	159.0	23	89.7	37.2	37	83.5	19.2
10	103.3	182.8	24	33.6	101.9	38	78.0	21.0
11	110.9	228.6	25	72.3	25.6	39	57.8	25.1
12	123.1	114.0	26	52.7	21.6	40	29.9	47.0
13	111.0	215.8	27	86.9	20.2			

TABLE VI. Optimized architecture for the 7 × 7 planar code considering the ibm_washington quantum processor.

Qubit index	T_1 (μs)	T_2 (μs)	Qubit index	T_1 (μs)	T_2 (μs)	Qubit index	T_1 (μs)	T_2 (μs)
0	50.8	86.1	30	93.0	116.5	60	52.7	21.6
1	53.3	64.1	31	91.7	131.4	61	86.9	20.2
2	54.4	117.4	32	91.6	179.1	62	83.5	17.9
3	78.5	159.1	33	90.7	151.2	63	68.9	17.0
4	79.0	80.8	34	90.5	122.4	64	43.6	12.2
5	81.3	140.5	35	89.6	154.0	65	77.6	11.6
6	82.6	117.8	36	88.7	155.3	66	61.1	8.6
7	83.1	98.4	37	88.4	131.1	67	71.1	11.2
8	83.8	199.4	38	85.8	140.0	68	66.9	11.9
9	84.4	161.0	39	84.4	115.5	69	16.5	100.0
10	84.8	84.5	40	84.1	168.6	70	72.8	17.6
11	86.8	120.0	41	118.4	83.2	71	83.5	19.2
12	88.5	144.8	42	82.9	172.6	72	78.0	21.0
13	89.3	147.7	43	82.0	140.0	73	57.8	25.1
14	89.8	111.6	44	81.3	104.0	74	29.9	47.0
15	96.7	90.7	45	78.9	95.2	75	33.6	101.9
16	91.1	130.0	46	55.6	119.9	76	89.7	37.2
17	94.9	91.7	47	53.7	113.4	77	38.1	53.0
18	92.0	187.7	48	52.9	127.3	78	39.6	80.0
19	95.3	155.8	49	71.7	48.4	79	73.3	41.9
20	98.3	110.1	50	54.5	46.1	80	44.3	80.4
21	100.7	159.0	51	70.4	44.7	81	44.7	81.7
22	103.3	182.8	52	74.4	44.5	82	111.7	44.8
23	110.9	228.6	53	54.1	42.0	83	74.0	47.4
24	123.1	114.0	54	41.1	150.5	84	70.1	50.7
25	111.0	215.8	55	71.8	38.8			
26	107.6	133.7	56	38.1	95.0			
27	101.8	158.8	57	34.7	39.4			
28	98.4	161.7	58	65.5	33.0			
29	97.2	163.5	59	72.3	25.6			

TABLE VII. Optimized architecture for the 3×3 planar code considering the Zuchongzhi quantum processor.

Qubit index	T_1 (μs)	T_2 (μs)
0	21.1	2.2
1	29.8	9.2
2	27.5	9.4
3	33.4	10.8
4	36.2	16.0
5	38.5	13.2
6	33.5	9.9
7	16.7	9.4
8	33.7	2.5
9	34.8	2.0
10	29.7	1.8
11	38.0	2.0
12	18.8	2.1

TABLE IX. Optimized architecture for the 3×3 planar code considering the Rigetti Aspen-M-1 quantum processor.

Qubit index	T_1 (μs)	T_2 (μs)
0	12.5	3.2
1	78.8	45.4
2	50.7	60.6
3	52.9	66.3
4	62.3	117.1
5	58.2	76.1
6	51.2	60.0
7	69.6	47.0
8	4.0	7.6
9	16.9	2.7
10	23.2	1.2
11	3.9	2.5
12	5.37	2.7

TABLE VIII. Optimized architecture for the 5×5 planar code considering the Zuchongzhi quantum processor.

Qubit index	T_1 (μs)	T_2 (μs)	Qubit index	T_1 (μs)	T_2 (μs)	Qubit index	T_1 (μs)	T_2 (μs)
0	24.2	3.3	14	33.5	9.9	28	34.7	2.7
1	26.8	3.4	15	16.7	9.4	29	39.9	2.5
2	15.6	5.9	16	28.8	8.8	30	21.1	2.2
3	26.1	6.2	17	42.9	8.3	31	34.8	2.0
4	25.3	6.2	18	20.9	7.4	32	29.7	1.8
5	37.1	6.4	19	38.3	6.7	33	38.0	2.0
6	35.8	7.2	20	46.6	6.3	34	18.8	2.1
7	25.2	7.7	21	39.2	6.2	35	33.7	2.5
8	24.4	8.5	22	34.5	6.1	36	30.0	2.7
9	29.8	9.2	23	30.1	3.5	37	22.9	2.7
10	27.5	9.4	24	37.0	3.3	38	33.3	2.9
11	33.4	10.8	25	34.0	3.2	39	32.1	3.1
12	36.2	16.0	26	44.9	3.0	40	17.5	3.3
13	38.5	13.2	27	29.4	2.7			

TABLE X. Optimized architecture for the 5 × 5 planar code considering the Rigetti Aspen-M-1 quantum processor.

Qubit index	T_1 (μs)	T_2 (μs)	Qubit index	T_1 (μs)	T_2 (μs)	Qubit index	T_1 (μs)	T_2 (μs)
0	10.0	20.1	14	21.2	59.9	28	8.2	10.4
1	12.1	10.2	15	69.7	46.9	29	20.8	4.6
2	85.1	28.7	16	69.7	44.5	30	12.6	3.2
3	30.5	48.0	17	41.4	45.8	31	17.0	2.7
4	48.6	31.2	18	37.6	42.9	32	23.2	1.2
5	34.0	32.1	19	32.3	36.1	33	4.0	2.6
6	36.1	72.1	20	31.7	42.6	34	5.4	2.7
7	42.35	40.1	21	31.2	34.9	35	4.0	7.6
8	50.2	43.6	22	36.0	30.2	36	34.6	7.5
9	78.9	45.4	23	10.3	18.6	37	41.0	8.2
10	50.8	60.6	24	15.1	10.1	38	75.5	8.8
11	52.9	66.3	25	16.3	9.2	39	31.9	9.0
12	62.3	117.1	26	34.4	8.9	40	15.8	9.2
13	58.24	76.1	27	9.0	8.2			

TABLE XI. Average qubit frequency (f_Q), average readout error (ϵ_M), and average CNOT error (ϵ_{CNOT}) of the three quantum processors.

Processor	f_Q (GHz)	ϵ_M (%)	ϵ_{CNOT} (%)
ibmq_brooklyn	5.065	0.088	0.022
ibm_washington	4.730	0.023	0.010
Rigetti Aspen-M-1		2.7	10.6

[1] Y. Wang, Z. Hu, B. C. Sanders, and S. Kais, Qudits and high-dimensional quantum computing, Front. Phys. **8**, 589504 (2020).

[2] P. J. Low, B. M. White, A. A. Cox, M. L. Day, and C. Senko, Practical trapped-ion protocols for universal qudit-based quantum computing, Phys. Rev. Res. **2**, 033128 (2020).

[3] M. Gu, C. Weedbrook, N. C. Menicucci, T. C. Ralph, and P. van Loock, Quantum computing with continuous-variable clusters, Phys. Rev. A **79**, 062318 (2009).

[4] J. E. Bourassa *et al.*, Blueprint for a scalable photonic fault-tolerant quantum computer, Quantum **5**, 392 (2021).

[5] M. A. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, 10th anniversary ed. (Cambridge University, Cambridge, England, 2011).

[6] J. Etxezarreta Martinez, P. Fuentes, P. M. Crespo, and J. Garcia-Frías, Approximating decoherence processes for the design and simulation of quantum error correction codes on classical computers, IEEE Access **8**, 172623 (2020).

[7] J. Etxezarreta Martinez, P. Fuentes, P. M. Crespo, and J. Garcia-Frías, Time-varying quantum channel models for superconducting qubits, npj Quantum Inf. **7**, 115 (2021).

[8] D. Gottesman, Stabilizer codes and quantum error correction, Ph.D. dissertation, California Institute of Technology, 1997.

[9] D. J. C. MacKay, G. Mitchinson, and P. L. McFadden, Sparse-graph codes for quantum error correction, IEEE Trans. Inf. Theory **50**, 2315 (2004).

[10] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, Fifteen years of quantum LDPC coding and improved decoding strategies, IEEE Access **3**, 2492 (2015).

[11] P. Fuentes, J. Etxezarreta Martinez, P. M. Crespo, and J. Garcia-Frías, An approach for the construction of non-CSS LDGM-based quantum codes, Phys. Rev. A **102**, 012423 (2020).

[12] P. Fuentes, J. Etxezarreta Martinez, P. M. Crespo, and J. Garcia-Frías, Design of LDGM-based quantum codes for asymmetric quantum channels, Phys. Rev. A **103**, 022617 (2021).

[13] D. Poulin, J.-P. Tillich, and H. Ollivier, Quantum serial turbo codes, IEEE Trans. Inf. Theory **55**, 2776 (2009).

[14] M. M. Wilde, M. Hsieh, and Z. Babar, Entanglement-assisted quantum turbo codes, IEEE Trans. Inf. Theory **60**, 1203 (2014).

[15] J. E. Martinez, P. M. Crespo, and J. Garcia-Frías, Depolarizing channel mismatch and estimation protocols for quantum turbo codes, Entropy **21**, 1133 (2019).

[16] J. E. Martinez, P. Fuentes, P. M. Crespo and J. Garcia-Fras, Pauli Channel Online Estimation Protocol for Quantum Turbo Codes, in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, New York, 2020), pp. 102–108.

[17] D. K. Tuckett, Tailoring surface codes: Improvements in quantum error correction with biased noise, Ph.D. thesis, University of Sydney, Australia, 2020, https://qecsim.github.io.

[18] A. Y. Kitaev, Quantum computations: Algorithms and error correction, Russ. Math. Surv. **52**, 1191 (1997).

[19] IBM Quantum, https://quantum-computing.ibm.com/services, 2022.

[20] ibmWashington IBM Quantum, https://quantum-computing.ibm.com/services, 2022.

[21] ibmqBrooklyn IBM Quantum, https://quantum-computing.ibm.com/services, 2022.

[22] Y. Wu *et al.*, Strong Quantum Computational Advantage Using a Superconducting Quantum Processor, Phys. Rev. Lett. **127**, 180501 (2021).

[23] Rigetti Aspen-M-1, data obtained through Strangeworks, https://app.quantumcomputing.com, 2022.

[24] R. Acharya *et al.*, Suppressing quantum errors by scaling a surface code logical qubit, arXiv:2207.06431.

[25] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, Fragile boundaries of tailored surface codes and improved decoding of circuit-level noise, arXiv:2203.04948.

[26] N. Sundaresan *et al.*, Matching and maximum likelihood decoding of a multi-round subsystem quantum error correction experiment, arXiv:2203.07205.

[27] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[28] T. J. Yoder and I. H. Kim, The surface code with a twist, Quantum **1**, 2 (2017).

[29] U. Azad, A. Lipiñska, S. Mahato, R. Sachdeva, D. Bhoumik, and R. Majumdar, Surface code design for asymmetric error channels, IET Quant. Comm. **3** 174 (2022).

[30] J. Edmonds, Paths, trees, and flowers, Can. J. Math. **17**, 449 (1965).

[31] A. G. Fowler, Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $O(1)$ parallel time, Quantum Inf. Comput. **15**, 145 (2015).

[32] O. Higgott, PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching, arXiv:2105.13082.

[33] P. Fuentes, J. Etxezarreta Martinez, P. M. Crespo, and J. Garcia-Frías, Degeneracy and its impact on the decoding of sparse quantum codes, IEEE Access **9**, 89093 (2021).

[34] M. Silva, E. Magesan, D. W. Kribs, and J. Emerson, Scalable protocol for identification of correctable codes, Phys. Rev. A **78**, 012347 (2008).

[35] E. W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. **1**, 269 (1959).

[36] M. Jeruchim, Techniques for estimating the bit error rate in the simulation of digital communication systems, IEEE J. Sel. Areas Commun. **2**, 153 (1984).

[37] O. Higgott and N. P. Breuckmann, Subsystem Codes with High Thresholds by Gauge Fixing and Reduced Qubit Overhead, Phys. Rev. X **11**, 031039 (2021).

[38] M. Carroll, S. Rosenblatt, P. Jurcevic, I. Lauer, and A. Kandala, Dynamics of superconducting qubit relaxation times, npj Quantum Inf. **8**, 132 (2022).

[39] J. J. Burnett *et al.*, Decoherence benchmarking of superconducting qubits, npj Quantum Inf. **5**, 54 (2019).

[40] P. V. Klimov, J. Kelly, Z. Chen, M. Neeley, A. Megrant, B. Burkett, R. Barends, K. Arya, B. Chiaro, Y. Chen *et al.*, Fluctuations of Energy-Relaxation Times in Superconducting Qubits, Phys. Rev. Lett. **121**, 090502 (2018).

[41] S. Schlör *et al.*, Correlating Decoherence in Transmon Qubits: Low Frequency Noise by Single Fluctuators, Phys. Rev. Lett. **123**, 190502 (2019).

[42] A. Stehli *et al.*, Coherent superconducting qubits from a subtractive junction fabrication process, Appl. Phys. Lett. **117**, 124005 (2020).

[43] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. (NY) **303**, 2 (2003)

[44] National Academies of Sciences, Engineering, and Medicine, *Quantum Computing: Progress and Prospects* (National Academies, Washington, DC, 2019).