Generalizing the matching decoder for the Chamon code

Zohar Schwartzman-Nowik[1,2] and Benjamin J. Brown[3]

[1]School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel
[2]Faculty of Engineering and the Institute of Nanotechnology and Advanced Materials, Bar Ilan University, Ramat Gan, Israel
[3]IBM Denmark, Sundkrogsgade 11, 2100 Copenhagen, Denmark

Different choices of quantum error-correcting codes can reduce the demands on the physical hardware needed to build a quantum computer. To achieve the full potential of a code, we must develop practical decoding algorithms that can correct errors that have occurred with high likelihood. Matching decoders are very good at correcting local errors while also demonstrating fast run times that can keep pace with physical quantum devices. We implement variations of a matching decoder for a three-dimensional, non-CSS, low-density parity check code known as the Chamon code, which has a non-trivial structure that does not lend itself readily to this type of decoding. The non-trivial structure of the syndrome of this code means that we can supplement the decoder with additional steps to improve the threshold error rate, below which the logical failure rate decreases with increasing code distance. We find that a generalized matching decoder that is augmented by a belief-propagation step prior to matching gives a threshold of 10.5% for depolarizing noise.

1 Introduction

Quantum computers hold great potential to bring significant improvements in computation compared to their classical counterparts. One of the main obstacles for realizing these advantages is overcoming the noise obscuring the information in the quantum devices. We employ quantum error correction codes (QECCs) to overcome this obstacle [1, 2]. A QECC is a system where the information is encoded in a subspace of a larger Hilbert space, called the code space. We operate QECCs by measuring stabilizer checks to obtain syndrome data. This data is then used to determine the best way to correct the errors the system experiences. The QECCs we need to build a scalable quantum computer have a high resource cost in terms of physical hardware. The resource cost of building a quantum computer is a function of our choice of QECCs that underlie our quantum computer, and how these QECCs are operated. Finding better ways of operating new QECCs with good parameters can reduce the resource cost of a large-scale quantum computer, and bring their realization forward.

A decoder is an algorithm that takes syndrome data and decides what recovery operation is most likely to restore the logical encoded state to the state that was encoded initially. Decoding is a highly non-trivial task, and a number of different methods of decoding have been proposed and tested [3–8] that trade-off between efficiency and the likelihood that an error is corrected successfully. The minimum-weight perfect-matching (MWPM) decoder has demonstrated both good accuracy and fast decoding times [2, 9–19], where decoding can be completed by pairing individual syndrome defects in the case of the surface code [2, 20, 21].

In general, the syndrome data of a QECC may give rise to a structure that does not readily lend itself to decoding with a MWPM decoder. However, due to its demonstrated performance when used with specific codes [2, 10–12, 22–32], it is valuable to explore ways of using MWPM to decode more exotic quantum low-density parity-check (LDPC) codes, and to interrogate their performance. To this end, in this work we develop and analyse a generalization of the MWPM decoder [12] on the Chamon code [33–35]. This is a three dimensional, non-CSS LDPC code that gives rise to a non-trivial syndrome when affected

Zohar Schwartzman-Nowik: zohar.nowik@mail.huji.ac.il

by errors.

We design our decoder by exploiting code symmetries [12, 26]. These are overcomplete subsets of stabilizers of the code such that errors always create an even number of syndrome defects with respect to the stabilizers of the symmetry. Specifically, a single qubit Pauli error gives rise to at most two syndrome defects on a symmetry. This property enables us to employ MWPM subroutines to find a correction by pairing syndrome defects. Once the matching step is completed, we use the results of these subroutines to find local clusters of syndrome defects that can be corrected independently with a low-weight correction operator. A local correction is then found using an adaptation of the broom algorithm [5].

For the Chamon code, the code symmetries consist of stabilizers that lie on two-dimensional planes of the three-dimensional model. This structure raises some interesting question towards the endeavor of decoding by matching over symmetries. Given that matching is performed on a vanishingly small subset of all the stabilizers, one might ask how this affects the performance of the decoder. On the one hand, one might expect that decoding performance is improved by concentrating fallible matching subroutines on a small subset of the stabilizers. Indeed, examples have shown high performance decoding algorithms can be designed by choosing codes where a decoder can concentrate on smaller subsets of the entire syndrome [11, 25–31, 36, 37]. On the other hand, the two-dimensional matching subroutines are ignoring a significant majority of the syndrome data, and we might expect to be able to improve performance by passing the global syndrome data to the matching subroutines that operate of isolated planes of the code. This question leads us to investigate variations of the basic matching-based decoder.

The first generalization we consider is a decoder where we begin with an initial greedy correction [38, 39] before matching. Here we propose a correction if a single flip will remove four defects surrounding a qubit. We find this decoder improves the threshold of the basic matching decoder from 5% to 6%. This decoder suggests an advantage is obtained by using the structure of the syndrome data. In contrast, for the qubit implementation of the toric code a similar greedy step reduces the threshold [38].

We also consider a generalization of the decoder where we improve the matching step by first performing belief propagation over the global syndrome [40]. The messages that are passed are used to improve the matching subroutines via belief matching [41]. Importantly, this passes information from the three-dimensional syndrome to the planes where matching is conducted. Remarkably, we obtain a threshold of 10.5% with this method. This significantly exceeds that found in earlier work on the Chamon code [35] where a threshold of 4.92% is obtained. One can also compare our work to Ref. [26] where a three-dimensional QECC with similar properties to the Chamon code is studied.

As a final introductory remark, an interesting feature of the Chamon code is that its non-CSS nature enables our matching decoder to treat all types of Pauli errors equally. In contrast, without some novel types of intervention [29, 32, 42, 43], matching decoders for more conventional CSS codes treat Pauli-X and Pauli-Z errors separately [2, 29]. However, it is well understood that such treatment can lead to reduced decoding performance [4, 6]. As such, we believe the example we consider motivates further work into high-threshold decoding algorithms for other non-CSS LDPC codes, e.g. [44].

The remainder of this paper is structured as follows. In Sec. II we present the Chamon code. In Sec. III we define the generalized MWPM decoder for the Chamon code, as well as the two improved versions of the decoder by introducing different initial steps. In Sec. IV we present the results, and in Sec. V we conclude.

2 The Chamon code

2.1 The stabilizer group

The Chamon code is a stabilizer code. A stabilizer code is defined by a set of commuting Pauli operators $\{S_i\}$, [45], with eigenvalues ± 1. The stabilizers are any operators in the group spanned by the generators S_i. The logical states are all eigenstates of all stabilizers with eigenvalues 1. We also define logical operators, these are Pauli operators that commute with the elements of the stabilizer group, but are not themselves stabilizer operators. These operators generate rotations among the encoded states of the stabilizer code.

We measure the error syndrome of a stabilizer code to attempt to identify the error that has occurred. The syndrome s for an arbitrary logical state $|\psi\rangle$ is a list of bits s_i where the result of measuring stabilizer generator S_i on $|\psi\rangle$ is $(-1)^{s_i}$. Thus, the syndrome for a logical state will always be a string of zeros. When a different string is obtained, an error is detected. Any value of 1 in the syndrome is referred to as a syndrome defect.

We define the stabilizer group of the Chamon code [33–35] on a three-dimensional cubic lattice with dimensions $d \times d \times d$ and periodic boundary conditions, where d is the distance of the code, and is even. (A more general definition for non-cubic Chamon codes can be found in [33, 34].) Let us give vertices a coordinate $v = (x, y, z)$ with $1 \leq x, y, z \leq d$. Vertices are bi-coloured such that white(black) vertices lie on the odd(even) sites where the number $(x + y + z)$ determines if the vertex $v = (x, y, z)$ is odd(even). With this bicolouring we have that each white vertex represents a qubit and each black qubit represents a stabilizer, as can be seen in Fig. 1(a). Specifically, for each even vertex v we have a stabilizer

$$S_v = X_{v+\hat{x}} X_{v-\hat{x}} Y_{v+\hat{y}} Y_{v-\hat{y}} Z_{v+\hat{z}} Z_{v-\hat{z}} \qquad (1)$$

where $\hat{x}, \hat{y}, \hat{z}$ are unit translations along their respective canonical axes. An example of a Chamon code stabilizer is presented in Fig. 1(b). Operators X_v, Y_v, Z_v are standard Pauli matrices acting on qubit v where v is an odd vertex site.

Interestingly, there is a symmetry between the three Paulis - X, Y and Z - in the stabilizers of the Chamon code. Every stabilizer applies exactly two of each Pauli on its neighbors, and every single qubit Pauli error flips the stabilizer measurement of four neighboring stabilizers, as can be seen in Fig. 1(c). Due to this last property, the code is inadequate for a straightforward implementation of the MWPM decoder, and non-trivial adaptations will be needed in order to apply a matching decoder on this code.

2.2 The symmetries of the Chamon code

In this work we propose a matching decoder for the Chamon code. As such, we look for structure in the code that enables us to use matching subroutines to find correctable clusters of the syndrome. To this end we look for code symmetries [12, 26]. For the Chamon code, we find symmetries of the stabilizer group such that single-qubit Pauli errors always create two syndrome defects with respect to the stabilizers of some given symmetry (or no syndrome defects, if the Pauli error occurred far from the given symmetry). Given that all errors create defects in pairs, we can readily employ matching to group defects together to find correctable clusters.

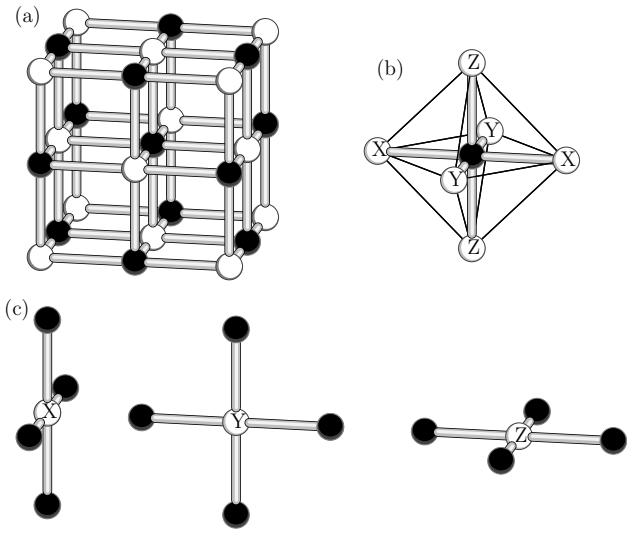

Figure 1: The Chamon code. (a) The Chamon code lattice. Every white vertex represents a qubit and every black vertex represents a stabilizer. (b) A stabilizer of the Chamon code. All stabilizers in the Chamon code are identical and apply a Pauli-X operator on the two neighboring qubits in the x axis, a Pauli-Y operator on the two neighboring qubits in the y axis, and similarly a Pauli-Z for the z axis. (c) Examples of single-qubit Pauli errors, together with their violated stabilizer operators.

The symmetries of the Chamon code that we exploit lie on two-dimensional planes of the three-dimensional code. We show examples in Fig. 2. We find the subsets of stabilizers S_v at locations $v = (x, y, z)$ that lie on the planes as follows

$$\Sigma^{\mathbf{r}}(C) = \{S_v : (r_x x + r_y y + r_z z) \,(\mathrm{mod}\ d) = \mathrm{C}\}\,, \qquad (2)$$

where the plane lies perpendicular to the vector $\mathbf{r} = (r_x, r_y, r_z)$ with $r_x, r_y, r_z = \pm 1$ and C is a constant that translates the plane along $\mathbf{r}$. Given that $-\mathbf{r}$ is parallel to $\mathbf{r}$, we have four unique plane orientations by this definition. There are $d/2$ different symmetries in each orientation, and so a total of $2d$ symmetries. One can check that for each of these subsets, the product of all the elements of this subset gives identity. It follows from this observation that any Pauli error will always produce an even number of defects among the stabilizers of a code symmetry. In fact, one

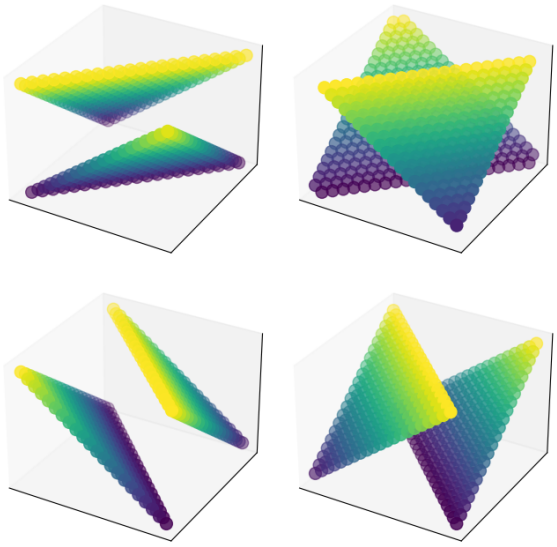

Figure 2: Examples of the symmetries of the Chamon code, which are the subgroups of stabilizers that are decoded separately. In these figures only the stabilizers included in the subgroup are depicted, with the color of the stabilizer indicative of the z value of the stabilizer location. In the case of the cubic Chamon code, these symmetries take the form of diagonally oriented sheets wrapping once around the periodic boundaries in each orientation. Representative of the subgroups in each of the four orientations are presented.

can check that errors always produce defects in pairs among each of these subsets.

3 A matching decoder for the Chamon code

Error correction requires a decoder to attempt to correct the error that has occurred. The decoder uses the syndrome to identify the recovery operation that brings the state back to its original logical state with high probability. We will assume the noise is Pauli noise for simplicity, as the stabilizer measurements project a general noise channel to a noise close to Pauli noise [46–48]. For some specific instance of a Pauli error E, which specifies which qubits suffered from which single qubit Pauli noise, the stabilizer measurement results in a syndrome s. The decoder takes the syndrome s as input and returns as output a recovery operation C which it deems most probable to bring the state back to its encoded logical state. When analyzing the logical error rate of a code via simulations where E is known, we can determine whether or not the recovery operation indeed succeeded in bringing the state back to the original logical state by observing whether CE is a logical operator, in which case the decoding failed and resulted in a logical failure, or whether CE is in the stabilizer group, in which case the decoding succeeded.

Here we describe the decoding algorithms that we implement for the Chamon code, with the source code available at [49]. All the decoders we implement are based on a matching step on symmetries that is followed by a local correction once matching is completed. We describe this first. We then go on to describe modifications to the basic implementation of the matching algorithm. We consider a greedy initialization step, as well as a belief propagation initialization step to improve the accuracy of the MWPM subroutine.

3.1 Basic matching

The decoder uses matching subroutines to identify small clusters of syndrome defects that can be corrected locally. The clusters are obtained by connecting defects that share an edge returned from the matching subroutines, where the matching is performed on all planar subsets of stabilizers as defined in Eqn. (2). A pair of defects are included in a common cluster if they share an edge on any matching subroutine. In general, a cluster contains many defects, as each defect will share edges with multiple defects over all of the matching subroutines.

We posit that the clusters obtained by connecting components of the syndrome that share an edge returned from a matching subroutine are locally correctable. It remains then to correct these defect clusters. We find that a correction can be obtained for each of these local clusters using an implementation of the broom algorithm [5] for the Chamon code. Let us now describe these steps in more detail.

3.1.1 Step 1 - Matching on symmetries

On each of the symmetries described in Sec. 2.2, every single-qubit Pauli error gives rise to up to two syndrome defects, and so can be decoded using MWPM, see Fig. 3. The first step of the matching decoder then is to apply matching, implemented using `PyMatching` [13], on each of the code symmetries, as in Eqn. (2), separately. Since we are using these decoders as a first step, in a multi-step decoder, we are not interested in the

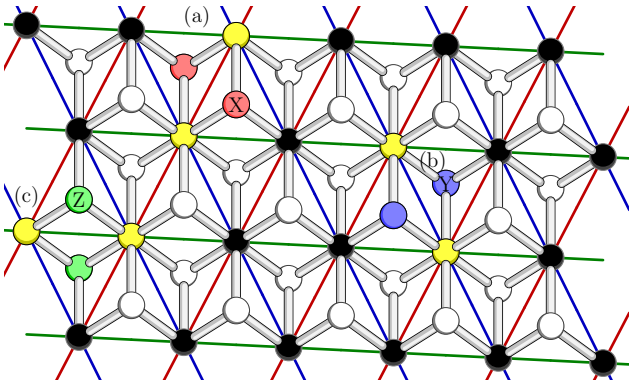

Figure 3: Stabilizers included in a planar symmetry, shown in black. Errors violate stabilizers in the plane that create defects. Defects are shown in yellow. A Pauli-X, Pauli-Y and Pauli-Z error are shown at (a), (b) and (c), respectively. All examples of errors give rise to pairs of defects. This property enables us to employ matching on individual planes. We find that there are two qubits where errors can give rise to the same pair of defects on the symmetry plane. Indeed, a Pauli-X, Pauli-Y or Pauli-Z error, respectively, on either of the red, blue or green qubits give rise to the same pair of defects on the plane. Different Pauli errors determine the orientation that defects appear on the plane. Pauli-X, Pauli-Y and Pauli-Z errors are always oriented along the red, blue and green lines that underly the lattice, respectively. In the limit that noise is biased such that we only experience one type of Pauli error, all errors give rise to pairs of defects with a common orientation. As such matching on a given plane is reduced to a one-dimensional problem. This is similar to other examples in, e.g., Refs. [11, 27, 31]. In this noise-bias limit the matching decoder we propose for the Chamon code is the same as that presented in Ref. [27] applied to planes of the code.

specific qubits it deems should be flipped. Instead, we take as output the previous step in the decoder – the pairs of syndrome defects it deems should be connected. And so the output of this first step in the decoding process is a set of connections between the syndrome defects.

3.1.2 Step 2 - dividing syndrome defects into clusters

We can now create a graph of syndrome defects and connections. The vertices of the graph are the syndrome defects, and the undirected edges are all the connections obtained from all the separate MWPM decoders of the previous step. The next step of the decoder is to identify all connected components in the graph, in order to later correct each connected component separately. This was done by using the `connected components` function in the `scipy` package [50]. After identifying the connected components, we divide the vertices of the graph - i.e. the syndrome defects - into the separate sets corresponding to the connected components, which we denote clusters. An example of such a division in presented in Fig. 4.

The next step will be decoding each cluster of defects separately using a version of the broom decoder, which will be explained in the next step. In order to apply the broom decoder [5], we need to identify the starting point and orientation to sweep the syndrome defects. To this end, we must identify the boundaries of a box enclosing each connected component. This is done by starting with a plane that crosses one of the syndrome defects in the cluster, then lowering the plane until it no longer crosses through a vertex or edge of the connected component. Repeating this for every axis and every orientation will give us a box enclosing the cluster, if one exists. If such a box does not exist we can determine with very high probability that the correction will fail. If we allow discarding samples then the best method would be to discard such a sample, and that will give better success rates. In this analysis we restrict ourselves by not allowing such post-selection. So instead, we guess random boundaries for the box, knowing that with very high probability the decoding for these cases will fail.

3.1.3 Step 3 - correcting each cluster via sweeping decoder

For low enough error rates, these clusters will not be too large. Thus, we will decode each cluster by the simple and quick sweeping decoder, which is an adaptation of the broom decoder [5] and unfolds as follows. We pick the directions to sweep, say x and z directions. We start by sweeping downward in the z direction - we pick a syndrome defect in the top of the cluster and apply the appropriate single qubit Pauli that swaps this syndrome defect by three defects in lower layers. We procceed similarly until all syndrome defects are in a double layer in the bottom of the box. Next we repeat this proccess in a different orientation, and "push" all syndrome defects from front to back. After sweeping in two directions the syndrome defects are confined to a very thin tube, and the probability of syndrome defects re-

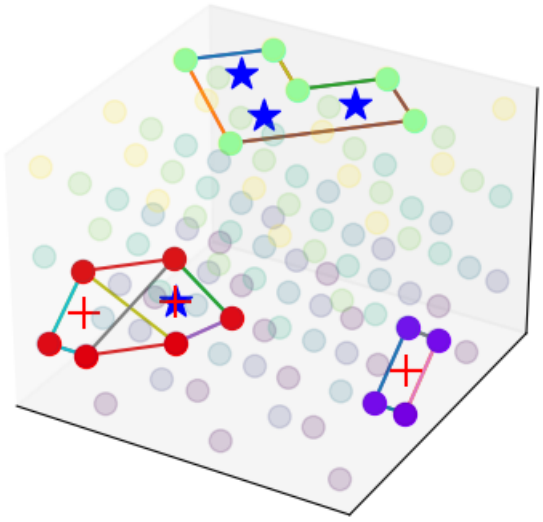

Figure 4: An example where syndrome defects are divided into clusters, together with the connections determined by all the separate matching decoders on symmetries. Syndrome defects are depicted using non-opaque balls, with different colors determined by the cluster they are included in. The solid lines between syndrome defects are the connections obtained from MWPM on the different symmetries. The Pauli errors on the qubits are also depicted, with a X error depicted as a reg cross, a Z error as a blue star, and a Y error as a combination of the two.

maining after this process, for moderate or large system sizes is extremely rare. In the negligible cases where this occurs, we deem the instance a failure of the decoder. Otherwise, the recovery process resulting from the decoder are all single-qubit Paulis that were applied in the sweeping process.

3.2 Greedy matching

We suggest an improvement on the basic version of the decoder by adding an initial greedy step to the decoding algorithm. In another context, this greedy step could be interpreted as initialization [38] or predecoding [39], or an initial stage using a small-set-flip decoder [51].

Let us give some intuition for this idea. A single qubit pauli error gives rise to exactly four syndrome defects, in the shape of a diamond in one of the orientations, as can be seen in Fig. 1(c). The greedy step we implement consists of a single round over all the syndrome defects where every such diamond pattern is identified. After all the diamond patterns are identified, the single-qubit Pauli operations that correspond to each of the diamond patterns are added to the recovery operator, eliminating the corresponding syndrome defects. And so the MWPM on symmetries is then implemented on a reduced number of syndrome defects.

3.3 Belief matching

The basic MWPM decoder on symmetries has a major drawback since every MWPM decoder applied on a symmetry has access only to the syndrome defects in that symmetry and no knowledge of the other syndrome defects. An improved version of the decoder can overcome this obstacle by adding an initial step of soft belief propagation (BP) decoding [52, 53]. Belief propagation decoding assigns initial probabilities for each qubit to suffer from each Pauli error, determined by the probabilities in the noise channel acting on the qubits. Then all qubits send messages with these probabilities to the stabilizers they interact with. The stabilizers then use these messages, together with the information on whether or not they suffer from a syndrome defect, to determine updated probabilities for each Pauli error on each qubit, and send these back as messages. The qubits then infer an updated probability for each Pauli error according to these messages. This is one round of BP. The output, after multiple rounds of BP, are updates weights assigned to each qubit and each Pauli error corresponding to the probability of such an error occurring on the qubit. These weights are then used as the initial weights for the matching decoders on each of the symmetries, thus incorporating global information in them. The initial step of soft BP decoding was implemented using the `bp decoder` in the `ldpc` package [53] with the product sum method and for $10d$ iterations for distance d of the Chamon code. This initial step improves the final result significantly since it allows for global information to affect the decoding of each of the symmetry-subset MWPM decoders.

4 Simulation and results

The threshold of a QECC is one method to determine how well the code overcomes noise. It is a property of a family of codes characterized by a parameter L. The threshold is the noise rate p_* under which the logical error rate can be

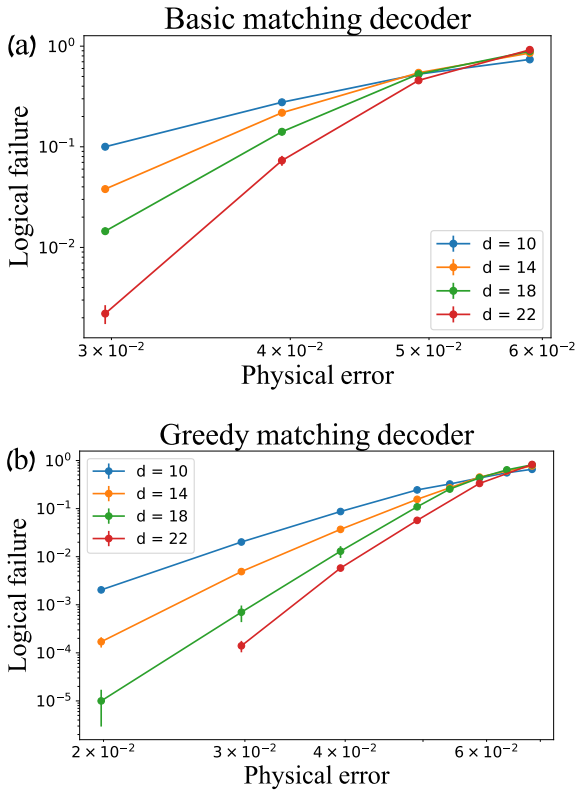

Figure 5: The total logical failure rate of any one of the considered logical qubits as a function of the physical error rate. The exponential decay of the failure rate for low errors, as well as the threshold can be observed. The standard deviation is presented in the error bars and was calculated by $\sqrt{\frac{P_{\text{fail}}-P_{\text{fail}}^2}{N}}$ where P_{fail} is the logical failure rate and N is the number of samples in the simulation. (a) The standard matching decoder. The threshold can be seen to be $\sim 5\%$. (b) The greedy matching decoder. For the improved matching decoder with an initial greedy step, the threshold is indeed slightly improved, and can be seen to be $\sim 6\%$. Also the exponential decay for low p is significantly better, and logical error rates are significantly lower.

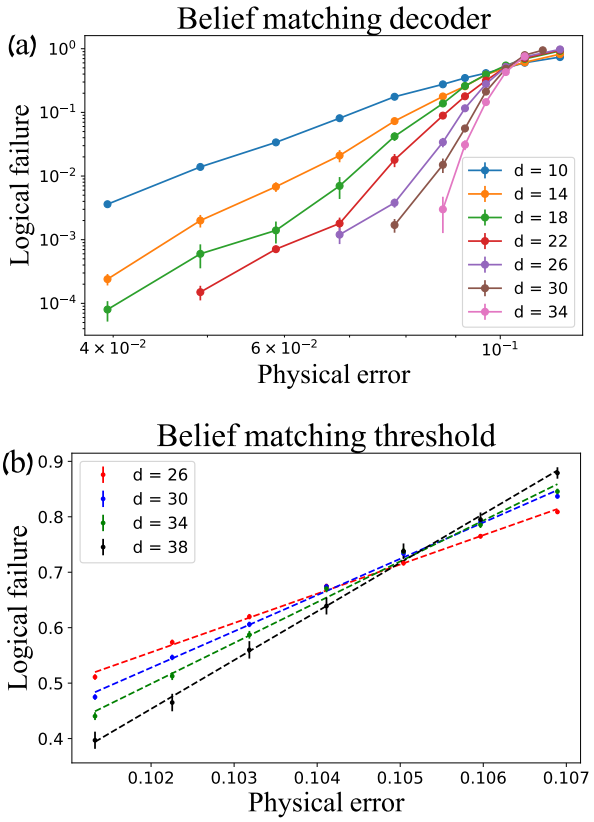

Figure 6: The total logical failure rate of any one of the considered logical qubits as a function of the physical error rate for the Belief matching decoder. The standard deviation is presented in the error bars and was calculated by $\sqrt{\frac{P_{\text{fail}}-P_{\text{fail}}^2}{N}}$ where P_{fail} is the logical failure rate and N is the number of samples in the simulation. (a) A large range of error rates shows the vast decay in logical failure rate for large system sizes. (b) A close-up on near threshold values with a linear fit of the data points reveals a threshold of 10.5%

suppressed arbitrarily by taking a representative code from the family with sufficiently large length L [54]. This is not to be confused with another definition of threshold, sometimes also called a pseudo-threshold, which relates the logical error of a code to the physical noise [1].

The threshold is a function of the noise model we choose to analyse our system. The noise model we consider here is the quantum depolarizing noise channel where all qubits of the system are affected identically by the map

$$\mathcal{E}(\rho) = (1-p)\,\rho + \frac{p}{3}X\rho X + \frac{p}{3}Y\rho Y + \frac{p}{3}Z\rho Z \quad (3)$$

where p is the noise parameter that determines the strength of the noise. This noise model is easy to analyze and assumes a symmetry between the

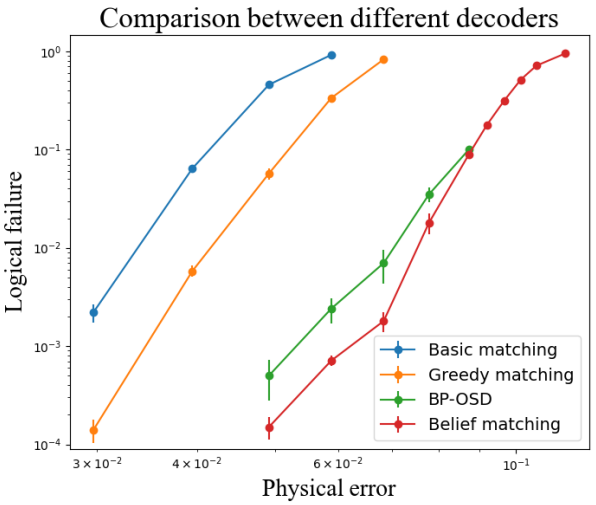

Figure 7: The total logical failure rate of any one of the considered logical qubits for the cubic $d = 22$ Chamon code as a function of physical error rate p for different decoders. The belief matching decoder presented in this paper outperforms the other versions of matching decoders as well as the BP-OSD decoder.

different types of single qubit Pauli errors, and so is a natural choice of noise channel for assessing the logical error rate for the Chamon code.

We identify a logical failure rate when the noise together with the correction do not commute with any one of the eight logical operators presented in section 3.1 of [35], which correspond to four logical qubits encoded in the code. The total logical failure rate is the rate of failure of the sweep step of the decoder (which is negligible) added to the logical failure rate. The best threshold value attained in [35] for the Chamon code is 4.92%. Already the basic implementation of generalized MWPM decoder slightly improves this threshold to a value of $\sim 5\%$, as can be seen in Fig. 5(a).

Adding an initial greedy step further improves the threshold to $\sim 6\%$ as can be seen in Fig. 5(b). Notice that in [38] a greedy initial step leads to improved decoding only for qudits with dimension 5 or larger, while for qubits and qutrits it leads to inferior decoding. In our case, though, due to the unique diamond shape of syndrome defects of a single-Pauli error, the greedy step leads to improved threshold even though we are dealing with qubits.

An initial BP step has many advantages over an initial greedy step, and so we would expect it to lead to improved decoding. It makes a better estimate for the error based on soft decisions, instead of focusing on specific greedy patterns. We expect this to make the initial BP step less prone to mistakes compared with the greedy initial step. Indeed, the belief matching decoder obtains a threshold of 10.5%, as can be seen in Fig. 6. This is a vast improvement compared to thresholds obtained with other decoders for the Chamon code.

Let us finally compare these decoders directly by concentrating on the logical error rates of a code of fixed distance, see Fig. 7. Here we show the performance of the decoders we have proposed together with a standard implementation of the belief-propagation ordered statistics decoder (BP-OSD) [52, 55, 56]. This decoder is well studied in the literature due to the fact that it is readily applicable to any code where the check matrix is known. We find that below threshold the belief matching on symmetries decoder performs best of all decoders. We will also note that its runtime was significantly faster than BP-OSD. The complexity of the BP-OSD decoder is $O(n^3)$ [55]. On the other hand, the worst-scaling part of the belief matching decoder is the MWPM subroutine which scales as $\tilde{O}(n^3)$ [13]. We run this subroutine on symmetries which include $O(n^{2/3})$ stabilizers and have $O(n^{1/3})$ such symmetries, and so in total the complexity of the belief matching decoder is $\tilde{O}(n^{7/3})$. Given that both decoders begin using a belief-propagation step, our results indicate that matching subroutines may provide a better option than ordered statistics decoding when interpreting the results of belief propagation for cases where the message passing step does not converge.

5 Summary

We presented and analyzed a generalized MWPM decoder by exploiting the symmetries of the Chamon code, allowing for this code to enjoy the benefits of a matching decoder despite the non trivial structure of its syndrome data. We also analyze two versions of improvements on this matching decoder by introducing initial decoding steps in order to incorporate the global data into the syndrome as well. The best performance was achieved by the belief matching decoder where matching subroutines are informed by messages passed during an initial BP step. This decoder has the best threshold between pre-

viously known decoders [35] and other variations of matching decoders presented in this paper. It also demonstrates the best below-threshold logical failure rates. This work demonstrates the potential in generalizing the belief-matching decoder to other QECCs with a non-trivial syndrome structure. These results suggest that generalizing belief matching decoders to quantum LDPC codes [55–57] may help to access their full error-correcting potential, and this is an interesting avenue for future work.

A central challenge in this work, and the source of its most compelling open question, lies in the difficulty of identifying suitable symmetry planes on which the matching procedure can be applied. These symmetry planes must meet two key criteria: (1) the matching decoder must be implementable on each symmetry plane, and (2) the clusters produced by the decoding processes on the symmetry planes must be independently correctable. Because of these constraints, identifying appropriate symmetry planes is a highly nontrivial task and, in this work, was carried out manually. This raises an important open question: is it possible to develop a general algorithm or systematic method for identifying such symmetry planes for arbitrary codes?

Acknowledgements

We thank D. Williamson for encouraging discussions, and J. Roffe for helpful explanations about BP decoding and their implementations. We also thank K. Sahay for comments on an earlier draft of the manuscript. BJB is grateful to the Center of Quantum Devices at the University of Copenhagen for their hospitality.

References

[1] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 176–188, 1997. URL https://doi.org/10.1145/258533.258579.

[2] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002. URL https://doi.org/10.1063/1.1499754.

[3] Antonio deMarti iOlius, Patricio Fuentes, Román Orús, Pedro M Crespo, and Josu Etxezarreta Martinez. Decoding algorithms for surface codes. *Quantum*, 8:1498, 2024. ISSN 2521-327X. URL https://doi.org/10.22331/q-2024-10-10-1498.

[4] Guillaume Duclos-Cianci and David Poulin. Fast decoders for topological quantum codes. *Phys. Rev. Lett.*, 104:050504, Feb 2010. URL https://doi.org/10.1103/PhysRevLett.104.050504.

[5] Sergey Bravyi and Jeongwan Haah. Quantum self-correction in the 3d cubic code model. *Phys. Rev. Lett.*, 111:200501, 2013. URL https://doi.org/10.1103/PhysRevLett.111.200501.

[6] James R. Wootton and Daniel Loss. High threshold error correction for the surface code. *Phys. Rev. Lett.*, 109:160503, Oct 2012. URL https://doi.org/10.1103/PhysRevLett.109.160503.

[7] Nicolas Delfosse and Naomi H Nickerson. Almost-linear time decoding algorithm for topological codes. *Quantum*, 5:595, 2021. URL https://doi.org/10.22331/q-2021-12-02-595.

[8] Sergey Bravyi, Martin Suchara, and Alexander Vargo. Efficient algorithms for maximum likelihood decoding in the surface code. *Phys. Rev. A*, 90:032326, 2014. URL https://doi.org/10.1103/PhysRevA.90.032326.

[9] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17: 449–467, 1965. URL https://doi.org/10.4153/CJM-1965-045-4.

[10] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86: 032324, 2012. URL https://doi.org/10.1103/PhysRevA.86.032324.

[11] J Pablo Bonilla Ataides, David K Tuckett, Stephen D Bartlett, Steven T Flammia, and Benjamin J Brown. The xzzx surface code. *Nature communications*, 12 (1):2172, 2021. URL https://doi.org/10.1038/s41467-021-22274-1.

[12] Benjamin J Brown. Conservation laws

and quantum error correction: towards a generalised matching decoder. *IEEE BITS the Information Theory Magazine*, 2023. URL https://doi.org/10.1109/MBITS.2023.3246025.

[13] Oscar Higgott. Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching. *ACM Transactions on Quantum Computing*, 3(3): 1–16, 2022. URL https://doi.org/10.1145/3505637.

[14] Oscar Higgott and Craig Gidney. Sparse blossom: correcting a million errors per core second with minimum-weight matching. *Quantum*, 9:1600, 2025. URL https://doi.org/10.22331/q-2025-01-20-1600.

[15] Sebastian Krinner, Nathan Lacroix, Ants Remm, Agustin Di Paolo, Elie Genois, Catherine Leroux, Christoph Hellings, Stefania Lazar, Francois Swiadek, Johannes Herrmann, Graham J. Norris, Christian Kraglund Andersen, Markus Müller, Alexandre Blais, Christopher Eichler, and Andreas Wallraff. Realizing repeated quantum error correction in a distance-three surface code. *Nature*, 605(7911), 2022. ISSN 1476-4687. URL https://doi.org/10.1038/s41586-022-04566-8.

[16] Neereja Sundaresan, Theodore J. Yoder, Youngseok Kim, Muyuan Li, Edward H. Chen, Grace Harper, Ted Thorbeck, Andrew W. Cross, Antonio D. Córcoles, and Maika Takita. Demonstrating multi-round subsystem quantum error correction using matching and maximum likelihood decoders. *Nature Communications*, 14(1):2852, 2023. URL https://doi.org/10.1038/s41467-023-38247-5.

[17] Yue Wu and Lin Zhong. Fusion blossom: Fast mwpm decoders for qec. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 01, 2023. URL https://doi.org/10.1109/QCE57702.2023.00107.

[18] Cody Jones. Improved accuracy for decoding surface codes with matching synthesis, 2024. URL https://doi.org/10.48550/arXiv.2408.12135.

[19] Google Quantum AI. Quantum error correction below the surface code threshold. *Nature*, 638:920–926, 2025. URL https://doi.org/10.1038/s41586-024-08449-y.

[20] A.Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, 2003. ISSN 0003-4916. URL https://doi.org/10.1016/S0003-4916(02)00018-0.

[21] S. B. Bravyi and A. Yu. Kitaev. Quantum codes on a lattice with boundary, 1998. URL https://doi.org/10.48550/arXiv.quant-ph/9811052.

[22] David S. Wang, Austin G. Fowler, Charles D. Hill, and Lloyd C. L. Hollenberg. Graphical algorithms and threshold error rates for the 2d color code. *Quantum Info. Comput.*, 10(9):780–802, September 2010. ISSN 1533-7146. URL https://doi.org/10.26421/QIC10.9-10-5.

[23] Georgia M. Nixon and Benjamin J. Brown. Correcting spanning errors with a fractal code. *IEEE Transactions on Information Theory*, 67(7):4504–4516, 2021. URL https://doi.org/10.1109/TIT.2021.3068359.

[24] Jonathan F. San Miguel, Dominic J. Williamson, and Benjamin J. Brown. A cellular automaton decoder for a noise-bias tailored color code. *Quantum*, 7:940, March 2023. ISSN 2521-327X. URL https://doi.org/10.22331/q-2023-03-09-940.

[25] Nicolas Delfosse. Decoding color codes by projection onto surface codes. *Phys. Rev. A*, 89:012317, Jan 2014. URL https://doi.org/10.1103/PhysRevA.89.012317.

[26] Benjamin J Brown and Dominic J Williamson. Parallelized quantum error correction with fracton topological codes. *Physical Review Research*, 2(1): 013303, 2020. URL https://doi.org/10.1103/PhysRevResearch.2.013303.

[27] David K. Tuckett, Stephen D. Bartlett, Steven T. Flammia, and Benjamin J. Brown. Fault-tolerant thresholds for the surface code in excess of 5% under biased noise. *Phys. Rev. Lett.*, 124:130501, Mar 2020. URL https://doi.org/10.1103/PhysRevLett.124.130501.

[28] Basudha Srivastava, Anton Frisk Kockum, and Mats Granath. The XYZ^2 hexagonal stabilizer code. *Quantum*, 6:698, April 2022. ISSN 2521-327X. URL https://doi.org/10.22331/q-2022-04-27-698.

[29] Kaavya Sahay and Benjamin J. Brown. Decoder for the triangular color code by matching on a möbius strip. *PRX Quantum*, 3: 010310, Jan 2022. URL https://doi.org/10.1103/PRXQuantum.3.010310.

[30] Aleksander Kubica and Nicolas Delfosse. Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders. *Quantum*, 7:929, February 2023. ISSN 2521-327X. URL https://doi.org/10.22331/q-2023-02-21-929.

[31] Eric Huang, Arthur Pesah, Christopher T. Chubb, Michael Vasmer, and Arpit Dua. Tailoring three-dimensional topological codes for biased noise. *PRX Quantum*, 4: 030338, Sep 2023. URL https://doi.org/10.1103/PRXQuantum.4.030338.

[32] Asmae Benhemou, Kaavya Sahay, Lingling Lao, and Benjamin J. Brown. Minimising surface-code failures using a color-code decoder. *Quantum*, 9:1632, 2025. ISSN 2521-327X. URL https://doi.org/10.22331/q-2025-02-17-1632.

[33] Claudio Chamon. Quantum glassiness in strongly correlated clean systems: An example of topological overprotection. *Physical review letters*, 94(4):040402, 2005. URL https://doi.org/10.1103/PhysRevLett.94.040402.

[34] Sergey Bravyi, Bernhard Leemhuis, and Barbara M Terhal. Topological order in an exactly solvable 3d spin model. *Annals of Physics*, 326(4):839–866, 2011. URL https://doi.org/10.1016/j.aop.2010.11.002.

[35] Jian Zhao, Yu-Chun Wu, and Guo-Ping Guo. Quantum memory error correction computation based on chamon model, 2023. URL https://doi.org/10.48550/arXiv.2303.05267.

[36] Naomi H. Nickerson and Benjamin J. Brown. Analysing correlated noise on the surface code using adaptive decoding algorithms. *Quantum*, 3:131, April 2019. ISSN 2521-327X. URL https://doi.org/10.22331/q-2019-04-08-131.

[37] Craig Gidney and Cody Jones. New circuits and an open source decoder for the color code, 2023. URL https://doi.org/10.48550/arXiv.2312.08813.

[38] Hussain Anwar, Benjamin J Brown, Earl T Campbell, and Dan E Browne. Fast decoders for qudit topological codes. *New Journal of Physics*, 16(6):063038, 2014. URL https://doi.org/10.1088/1367-2630/16/6/063038.

[39] Samuel C Smith, Benjamin J Brown, and Stephen D Bartlett. Local predecoder to reduce the bandwidth and latency of quantum error correction. *Physical Review Applied*, 19 (3):034050, 2023. URL https://doi.org/10.1103/PhysRevApplied.19.034050.

[40] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press, 2003. URL https://www.inference.org.uk/itila/.

[41] Oscar Higgott, Thomas C. Bohdanowicz, Aleksander Kubica, Steven T. Flammia, and Earl T. Campbell. Improved decoding of circuit noise and fragile boundaries of tailored surface codes. *Phys. Rev. X*, 13:031007, Jul 2023. URL https://doi.org/10.1103/PhysRevX.13.031007.

[42] Austin G. Fowler. Optimal complexity correction of correlated errors in the surface code, 2013. URL https://doi.org/10.48550/arXiv.1310.0863.

[43] Nicolas Delfosse and Jean-Pierre Tillich. A decoding algorithm for css codes using the x/z correlations. In *2014 IEEE International Symposium on Information Theory*, pages 1071–1075, 2014. URL https://doi.org/10.1109/ISIT.2014.6874997.

[44] Anthony Leverrier, Simon Apers, and Christophe Vuillot. Quantum XYZ Product Codes. *Quantum*, 6:766, July 2022. ISSN 2521-327X. URL https://doi.org/10.22331/q-2022-07-14-766.

[45] Daniel Gottesman. *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997. URL https://thesis.library.caltech.edu/2900/.

[46] Sergey Bravyi, Matthias Englbrecht, Robert König, and Nolan Peard. Correcting coherent errors with surface codes. *npj Quantum Information*, 4(1):55, 2018. URL https://doi.org/10.1038/s41534-018-0106-y.

[47] Andrew S Darmawan and David Poulin. Tensor-network simulations of the surface code under realistic noise. *Physical review letters*, 119(4):040502, 2017. URL

https://doi.org/10.1103/PhysRevLett.119.040502.

[48] Zohar Schwartzman-Nowik, Liran Shirizly, and Haggai Landa. Modeling error correction with lindblad dynamics and approximate channels. *Phys. Rev. A*, 111: 022613, Feb 2025. URL https://doi.org/10.1103/PhysRevA.111.022613.

[49] Zohar Schwartzman-Nowik and Benjamin J Brown. Simulation of success rates of generalized matching decoders of the cubic chamon code., 2024. URL https://github.com/zoharno/chamon_decoder.

[50] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/.

[51] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier. Efficient decoding of random errors for quantum expander codes. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 521–534, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355599. URL https://doi.org/10.1145/3188745.3188886.

[52] Joschka Roffe, David R. White, Simon Burton, and Earl Campbell. Decoding across the quantum low-density parity-check code landscape. *Physical Review Research*, 2(4), Dec 2020. ISSN 2643-1564. URL https://doi.org/10.1103/PhysRevResearch.2.043423.

[53] Joschka Roffe. LDPC: Python tools for low density parity check codes, 2022. URL https://pypi.org/project/ldpc/.

[54] Emanuel Knill, Raymond Laflamme, and Wojciech H Zurek. Resilient quantum computation: error models and thresholds. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):365–384, 1998. URL https://doi.org/10.1098/rspa.1998.0166.

[55] Pavel Panteleev and Gleb Kalachev. Degenerate Quantum LDPC Codes With Good Finite Length Performance. *Quantum*, 5:585, November 2021. ISSN 2521-327X. URL https://doi.org/10.22331/q-2021-11-22-585.

[56] Sergey Bravyi, Andrew W Cross, Jay M Gambetta, Dmitri Maslov, Patrick Rall, and Theodore J Yoder. High-threshold and low-overhead fault-tolerant quantum memory. *Nature*, 627(8005):778–782, 2024. URL https://doi.org/10.1038/s41586-024-07107-7.

[57] Antoine Grospellier, Lucien Grouès, Anirudh Krishna, and Anthony Leverrier. Combining hard and soft decoders for hypergraph product codes. *Quantum*, 5:432, April 2021. ISSN 2521-327X. URL https://doi.org/10.22331/q-2021-04-15-432.