A simple decoder for topological codes

James R. Wootton

Department of Physics, University of Basel, Klingelbergstrasse 82, CH-4056 Basel, Switzerland

(Dated: July 8, 2014)

Here we study an efficient algorithm for decoding the topological codes. It is based on a simple principle, which should allow straightforward generalization to complex decoding problems. It is benchmarked with the planar code for both i.i.d. and spatially correlated errors and is found to compare well with existing methods.

Quantum error correcting codes are an essential part of proposals for quantum computation. Topological error correcting codes are currently one of the most well studied examples [1]. No code is useful without a decoder, which takes the classical information produced by measurement of the code and uses it to determine how best to counteract the effects of noise. Multiple algorithms have been proposed to decode topological codes all with various advantages and disadvantages.

Here we present an algorithm whose main advantage is its simplicity, allowing straightforward generalization to a multitude of topological codes as well as more exotic decoding problems, such as non-Abelian decoding [2]. It can be considered to be an example of an HDRG decoder, which have recently begun to attract attention [3, 4], and is related to the ‘expanding diamonds’ algorithm studied in [5].

The decoder is benchmarked using one of the simplest and most well studied topological codes: the planar code. This is equivalent to the toric code up to boundary conditions. Note that methods already exist to decode this with high accuracy and with optimal behaviour in some regimes [5–8]. We therefore use the planar code only as a sandbox for our decoder, and do not intend our decoder to be applied to surface codes in practice. Rather, we intend it to be applied to problems for which the means to achieve near optimal decoding is not known.

THE PLANAR CODE

The planar code is the planar variant of the surface codes introduced by Kitaev [9] [10]. It is defined on a spin lattice, as in Fig. 1, with a spin-1/2 particle on each vertex. There are two types of plaquette, labelled s and p, for which we define Hermitian operators as follows,

\[ A_s = \prod_{i \in s} \sigma_i^x, \quad B_p = \prod_{i \in p} \sigma_i^z. \]

(1)

These operators mutually commute, and are the stabilizer operators of the code. Their eigenvalues can be interpreted in terms of anyonic occupations, with no anyon present for an eigenvalue of +1. Eigenvalues of −1 are interpreted as so-called flux anyons on p-plaquettes and charge anyons on s-plaquettes. The stabilizer space of the code therefore corresponds to the anyonic vacuum, and the anyon configuration is the syndrome of the code.

Edge operators may also be defined which commute with the stabilizers. These can be interpreted in terms of the anyon occupations of the edges, with flux anyons corresponding to a −1 eigenvalue of the left or right edge and charge anyons corresponding to a −1 eigenvalue of the top or bottom. The edge occupations define the state of the logical qubit stored in the code. The X (Z) basis of the logical qubit can be defined such that the |+\rangle (|0\rangle) state corresponds to vacuum on the top (left) edge and the |−\rangle (|1\rangle) state corresponds to a flux (charge) anyon. Errors acting on the spins of the code will create and move anyons, and so disturb the stored information when they cause anyons to move off the edges. The job of decoding is to remove the anyons in a way that yields no net anyon moved off any edge. A logical error results when the decoding does this incorrectly.

FIG. 1. The spin lattice of an \( L \times L \) planar code. The s-plaquettes are shown in blue and the p-plaquettes are shown in white. A spin-1/2 particle is situated at each vertex. In this example the linear size is \( L = 5 \). The linear size \( L \) is also the distance of the code.
THE DECODING ALGORITHM

Anyons in the planar code are created in pairs by strings of bit flip errors. Removal of the anyons, required by error correction to return the code to its stabilizer space, is similarly achieved by finding pairs of anyons (or pairing an anyon with an edge) and annihilating them using bit flips. Ideally the latter will mimic the former, choosing to annihilate the anyons using the pairs they were created in. It is not possible to do this perfectly, due to ambiguities in the syndrome information. However, we can instead aim to do it with high probability.

The decoder is designed to find pairs of anyons that are likely to have been created by the same error string. The probability of this decays exponentially with their separation. As such, anyons that are mutual nearest neighbours will be a good candidate. The decoder therefore finds and pairs such mutual nearest neighbours. The method used to do this is as follows.

1. Loop through all plaquettes to find anyons.
2. For each anyon, search through all plaquettes at a Manhattan distance of $k$, using $k = 1$ initially.
3. If either an anyon or the edge is found at this distance, pair them. Annihilate the anyons by performing bit flips on the connecting qubits. If there are multiple possibilities at this distance, pair with the first found.
4. If there are still anyons present, repeat the process for $k = k + 1$.

Once all the anyons are removed, the total pattern of bit flips used to remove them is considered. The correction procedure is a success if this belongs to the same equivalence class as the pattern of bit flips that occurred in error. Otherwise, the correction procedure results in a logical error.

An upper bound for the run time in the worst case scenario can be easily determined. The $k$th iteration requires $O(k)$ plaquettes to be searched per anyon. The number of anyons present during each iteration is at most $O(L^2)$. There will be no more than $L$ iterations since no anyon is more than this distance from the edge. The total complexity is then never more than $O(L \sum_{k=1}^{L} kL^2) = O(L^4)$. This upper bound on the complexity is clearly polynomial with system size, and a moderately low ordered polynomial also. The algorithm therefore allows for fast and efficient decoding.

MINIMUM REQUIREMENTS FOR LOGICAL ERRORS

For any code and decoder, one important benchmark of performance is the minimum number of errors required to cause a logical error. We will use $\epsilon$ to denote the value of this number realized by an exhaustive decoder, and $\epsilon'$ to denote that for the decoder described above.

For the planar code $\epsilon = L/2$ for even $L$ and $(L + 1)/2$ for odd $L$. This is the minimum number of spin flips required to create a pair of anyons such that it takes less flips to pair them with opposite edges than with each other. Ignoring $O(1)$ corrections, as we will continue to do in the following, we can state this simply as $\epsilon = L/2$.

To find $\epsilon'$ we introduce the following terminology.

- A string is a collection of errors such that either an anyon or the edge exists at each endpoint.
- A cluster is a collection of strings, and also of their corresponding endpoints.
- The width, $w$, of a cluster is the maximum distance between a pair of its endpoints.
- The distance, $d$, between non-overlapping clusters is the minimum distance between an endpoint from each. The clusters are said to overlap if there exists two endpoints $i$ and $j$ from one cluster and $k$ from the other such that $d_{ij} > d_{ik}$.
- An isolated cluster is one whose width is smaller than the distance to any other cluster or to the edge.
- A spanning cluster is one that contains both edges.
- A level-$n+1$ cluster is a collection of level-$n$ clusters such that the each is no further from another than its own width. A level-0 cluster is a single error.

Isolated clusters are defined such that no anyon within the cluster must look further than the width of the cluster to find a nearest neighbour before it is annihilated. Also no anyon within the cluster will be considered for pairing by any outside before it is annihilated. The algorithm will therefore annihilate the anyons in isolated clusters with each other (or with the edge if also included) without reference to any others.

Given a set of errors, consider all level-0 clusters. Any of these that are isolated clusters can all be ignored without affecting further analysis, because they have no effect on the other anyons.

By definition, all level-0 clusters that remain will be part of at least one level-1 cluster. Let us then consider all possible level-1 clusters. Any of these that are isolated will again be dealt with by the algorithm independently of the rest, and so can be ignored.

This procedure can repeated for higher level clusters until no errors remain. If no spanning cluster was ever considered, the algorithm will have certainly corrected the errors. The existence of a spanning cluster is therefore a necessary, but not sufficient, condition for a logical
error. Determining the minimum number of errors required to create such a spanning cluster will then give a lower bound on \( \epsilon' \).

For this it is clear that any errors that will end up being ignored before the cluster becomes spanning are superfluous. As such we require a set of errors in which none are ignored. All level-0 clusters (errors) must therefore be part of a level-1 cluster, all level-1 clusters part of a level-2 cluster, and so on.

Let us consider the uniform case, where each level-\( n+1 \) cluster contains \( m \) level-\( n \) clusters. In order to become spanning as soon as possible, it is clear that the best option is for all errors to be along a single line across the code. To make each level-1 cluster as long as possible there should be as many gaps between the \( m \) errors as possible while maintaining the level-1 cluster. This would correspond to the \( m \) errors being evenly spaced with a gap of one spin without an error between each, and so \( m - 1 \) gaps in total. The width of each level-1 cluster will then be \( w_1 = 2m - 1 \).

Similarly, to make each level-2 cluster as long as possible there should be \( m - 1 \) gaps of length \( w_1 \) on which there are no errors. The width of each level-2 cluster will then be \( w_2 = (2m - 1)w_1 = (2m - 1)^2 \). Continuing in this way will lead to each level-\( n \) cluster having a width of \( w_n = (2m - 1)^n \). Note also that each level-\( n \) cluster will contain \( m^n \) errors.

The minimum means to form a spanning cluster is to have a level-\( n \) cluster where the left-most endpoint is closer to the left edge than to the right-most endpoint, and the right-most endpoint is closer to the right edge than to the left (or the same with left and right interchanged). The former ensures that the the cluster is not isolated from the left edge, and the latter ensures that the both combined are not isolated from the right edge. The minimum width of a level-\( n \) cluster required to satisfy this is \( L/4 \). This requires a cluster of level \( n = \frac{\ln(L/4)}{\ln(2m - 1)} \). The required number of errors is then \( m^n = (L/4)^n m/\ln(2m - 1) \). The exponent reaches its minimum value of \( \beta = \log_3 2 \approx 0.63 \) for \( m = 2 \).

It is clear that the non-uniform case where different level-\( n+1 \) clusters can contain different numbers of level-\( n \) clusters, will not allow a cluster to become spanning with fewer errors. As such \( L^\beta \) is the minimum number of errors required for a spanning cluster, and hence forms a lower bound on \( \epsilon' \).

For an upper bound on \( \epsilon' \) we can find a set of errors that causes the algorithm to fail. One such example can be constructed similarly to the above. However, instead of the two level-\( n \) clusters within a level-\( n+1 \) one being a distance \( w_n \) apart, we instead put them \( w_n - 1 \) apart. This will cause the algorithm to repeatedly pair the wrong anyons, and so cause a logical error.

Given this procedure it is easy to see that \( w_n = 3w_{n-1} - 1 = (3^n + 1)/2 \). This requires a cluster of level \( n = \log_3(L/2 - 1) \) for a logical error, and so \( (L/2 - 1)^\beta \) errors. Combining this with the lower bound, we find that \( \epsilon' = (cL)^\beta \) for \( 1/4 \leq c \leq 1/2 \). Note that this is not linear with \( L \), as we would ideally like, but its power law divergence should still provide good error suppression. Note that the set of errors found here are related to the Cantor set \([5, 14]\).

**RESULTS**

**Uncorrelated errors**

We first consider the error model most commonly used for benchmarking. This consists of independent bit and phase flips and noiseless syndrome measurements. Bit flips disturb the syndrome only on \( p \)-plaquettes, and phase flips only on \( s \)-plaquettes. The decoding for these errors may then be considered independently. We therefore restrict to studying only bit flips without loss of generality. These occur independently on each spin with probability \( p \).

The decoder is benchmarked using the logical error rate, \( P \), that it achieves for different bit flip error rates, \( p \), and linear system sizes, \( L \). This is done by randomly generating error configurations for the qubits of the code according to the noise model, applying the decoder to the resulting anyons, and then determining whether or not a logical error occurred for each sample. For each case we use the number of samples, \( n \), required in order for \( 10^3 \) logical errors to occur. The logical error rate is then \( P = 10^3/n \). The data was obtained to determine :

(a) The threshold error rate, \( p_c \), for which \( P \) decays with \( L \) when \( p < p_c \);

(b) The minimum system size required for each \( p \) such that \( P < p \) (and so the minimum system size required for a code stored qubit to be less noisy than one stored on a single physical spin);

(c) The decay of the logical error rate for bit flip error rates well below threshold, to show that effective error correction occurs;

(d) The values of \( \alpha(p) \) for each \( p \), when the above data is fitted to a function \( P = 0(e^{-\alpha(p)(cL)^\beta}) \).

The results can all be found in Fig. 2. These suggest that the threshold is around 7.25%. This is of the same order as the upper bound on the threshold of around 11%. It also compares well to the threshold of 7.8% obtained by the renormalization decoder of [11]. It is larger than the 6.7% threshold obtained by the Bravyi-Haah algorithm [3], but is less than the 8.4% threshold of the modified version in [4]. Note that these decoders deal with the toric code, which is identical to the planar code in the bulk but differs in its boundary conditions. Since
the planar code boundary is known to lead to worse decoding behaviour \cite{12}, it is remarkable that this decoder compares so well in its performance.

It is found that logical error rates of $P < p_c$ can be obtained using a small code of size $L = 4$ for all $p < p_c / 2$. The decoder therefore does not require a great deal of physical qubits in order to get good results. However, the required system size rises steeply for higher $p$. For $p < p_c / 2$ we also find that the logical error rate decays very quickly as $L \to \infty$.

The decay below threshold will be of the form $P = 0(e^{-\alpha(p)cL^3})$. Here the scaling can be expected to be governed by the minimum requirements for a logical error. This means $\beta = \log_3 2$ and, to be conservative, $c = 1 / 4$. The values of the decay rate $\alpha(p)$ are then determined by fitting the data to a function of this form. Note that this differs from the method used in \cite{13}, but is morally the same.

**Correlated errors**

We also consider errors that are not generated through a simple i.i.d. noise model, as above, but instead have spacial correlations. Such error models are arguably more realistic, and have recently attracted attention \cite{15, 17}.

We consider a model for which an error occurs on each spin with probability $p'$. For each of these errors, an additional one will occur on a nearest neighbour with probability $q$. This error model is therefore equivalent to that above for $p' = p$ and $q = 0$, but has nearest neighbour correlations between errors when $q > 0$.

Using the same method of numerical benchmarking described above we study the threshold, $p_c'$, and the minimum system size required for error correction, $L^*(p)$, for the case of $q = 0.5$. The results are shown in Fig. 3.

To compare the results with those for the uncorrelated case, let us consider the expected fraction of spins to have suffered an error in both cases. For the uncorrelated case this is simply $p$, whereas in this case it will be approximately $(1 + q)p'$. We may therefore expect that the behaviour of the decoder for given values of $p'$ and $q$ in this case will correspond to the behaviour for $p = (1 + q)p'$ in the uncorrelated case. This gives us an estimate of $p_c' \approx p_c / 1.5 = 4.8\%$ for $q = 0.5$.

The threshold is found numerically to be around 4.75%, and so agrees well with the above expectations. However the logical errors rates found in this case with a given $p'$ fall well below those for $p = (1 + q)p'$ in the uncorrelated case. For example, uncorrelated $p = 6\%$ corresponds to correlated $p' = 4\%$, however for $L = 600$ we find $P = 1.6 \times 10^{-4}$ in the former case and $P = 2.4 \times 10^{-3}$ in the latter. The correlated case therefore gives worse logical error rates by an order of magnitude. Also the minimum system size for error correction to be evident is $L^* = 38$ in the former case but $L^* = 130$ in the latter. This decrease in performance is likely due to the fact that the correlations increase the typical length of error strings.

A system size of at least $L = 7$ is required for logical error rates of $P < p'$ for even as low as $p' = 5 \times 10^{-4}$. However, this small size remains sufficient until $p' = 1\%$, after which the required system size rises steeply as $p' \to p_c$.

**CONCLUSIONS AND OUTLOOK**

The version of the decoder presented here is very much the ‘vanilla’ version. There are many changes that one could imagine making in order to improve the performance of the method. For example, rather than pairing with the first anyon found in each case, one can compare all anyons found at the same distance and make an informed choice on which to pair with. Another possibility is the Markov chain Monte Carlo techniques enhancements of \cite{6}. All such modifications will depend on the code and error model considered. This paper only aims to demonstrate the core method, so such application specific modifications were not studied here.

Despite its simplicity, the method is shown to behave very well as a decoder. It achieves respectable benchmarks for the standard testing ground of the planar code with independent bit and phase flip noise: the threshold is similar to other decoders and the theoretical maximum; error correction is evident for small system sizes; and the logical error rate is found to decay strongly. Good results were also found for the case of spatially correlated errors. The decoder can therefore be used to gain good results for topological codes as well as more exotic decoding problems, before more complex tailor-made methods are developed.

**Acknowledgements:** The author would like to thank Benjamin Brown for discussions, Andrew Landahl for pointing out the prior existence of a variant of this decoder, and the Swiss NF, NCCR Nano and NCCR QSIT for support.

\begin{thebibliography}{99}

\bibitem{1} J. R. Wootton, J. Mod. Opt. \textbf{20}, 1717 (2012).
\bibitem{2} J. R. Wootton, J. Burri, S. Iblisdir, D. Loss, Phys. Rev. X \textbf{4}, 011051 (2014).
\bibitem{3} S. Bravyi, J. Haah, Phys. Rev. Lett. \textbf{111}, 200501 (2013).
\bibitem{4} H. Anwar, et al., New J. of Phys. \textbf{16} 063038 (2014).
\bibitem{5} E. Dennis, arXiv:quant-ph/0503169 (2005).
\bibitem{6} A. Hutter, J. R. Wootton and D. Loss, Phys. Rev. A \textbf{89}, 022326 (2014).
\bibitem{7} A. G. Fowler, arXiv:1310.0863 (2013).
\bibitem{8} S. Bravyi, M. Suchara, A. Vargo arXiv:1405.4883 (2014).
\bibitem{9} A. Kitaev, Annals Phys. \textbf{303}, 2 (2003).
\bibitem{10} E. Dennis, A. Kitaev, A. Landahl, J. Preskill, J. Math. Phys. \textbf{43}, 4452 (2002).
\bibitem{11} G. Duclos-Cianci, D. Poulin, Phys. Rev. Lett. \textbf{104} 050504 (2010).
\end{thebibliography}
FIG. 2. Results for uncorrelated errors. (a) Graph of logical error rate, $P$, against bit-flip error rates, $p$, around threshold. (b) Graph of minimum linear system size required for $P < p$, $L^*$, against $p$. (c) Graph of $P$ against linear system size, $L$, for various $p$ well below threshold. Lines are guides to the eye. (d) A graph of $\alpha(p)$ against $p$.

FIG. 3. Results for correlated errors. (a) Graph of logical error rate, $P$, against bit-flip error rates, $p$, around threshold. (b) Graph of minimum linear system size required for $P < p$, $L^*$, against $p$. 

[12] A. G. Fowler, Phys. Rev. A 87, 062320 (2013).
[13] S. Bravyi and A. Vargo, Phys. Rev. A 88, 062308 (2013).
[14] A. Landahl, private communication (2013).
[15] E. Novais and E. R. Mucciolo, Phys. Rev. Lett. 110, 10502 (2013).
[16] A. Hutter and D. Loss, Phys. Rev. A 89, 042334 (2014).
[17] A. G. Fowler, J. M. Martinis, Phys. Rev. A 89, 032316 (2014).