Quantum Information and Computation, Vol. 10, No. 9&10 (2010) 0780-0802


# GRAPHICAL ALGORITHMS AND THRESHOLD ERROR RATES FOR THE 2D COLOR CODE

DAVID S. WANG[a], AUSTIN G. FOWLER, CHARLES D. HILL, LLOYD C. L. HOLLENBERG
*Centre for Quantum Computer Technology*
*School of Physics, University of Melbourne*
*Victoria 3010, Australia*

Received July 10, 2009
Revised July 20, 2010

Recent work on fault-tolerant quantum computation making use of topological error correction shows great potential, with the 2d surface code possessing a threshold error rate approaching 1% [1, 2]. However, the 2d surface code requires the use of a complex state distillation procedure to achieve universal quantum computation. The color code of [3] is a related scheme partially solving the problem, providing a means to perform all Clifford group gates transversally. We review the color code and its error correcting methodology, discussing one approximate technique based on graph matching. We derive an analytic lower bound to the threshold error rate of 6.25% under error-free syndrome extraction, while numerical simulations indicate it may be as high as 13.3%. Inclusion of faulty syndrome extraction circuits drops the threshold to approximately $0.10 \pm 0.01\%$.

*Keywords*:

*Communicated by*: I Cirac & B Terhal

## 1 Introduction

The development of quantum error correcting codes in 1995 [4–6] is a major milestone in the journey towards realising a quantum computer that is able to outperform classical computers for large problems. Error correction allows the suppression of decoherence during a quantum algorithm, allowing one to perform lengthy calculations such as Shor's algorithm for prime number factorisation [7] with high fidelity results. The threshold theorem [8] states that, provided all gates are constructed with a failure rate below some threshold error rate, arbitrary length quantum computation can be achieved by employing quantum error correction with polylogarithmic overhead.

The act of concatenation, the recursive grouping of logical qubits into successively higher level logical qubits, is one method to form codes with a threshold. However, this concatenation procedure creates non-local stabilizers involving an ever increasing number of physical qubits. As such, threshold error rates for codes formed in this manner suffer when one is limited to local interactions in few dimensions. For example, the 7-qubit Steane code has a threshold of $p_{\text{th}} = 1.85 \times 10^{-5}$ [9] when restricted to a 2d lattice with only nearest-neighbor couplings, and the Bacon-Shor code performs similarly, $p_{\text{th}} = 2.02 \times 10^{-5}$ [10]. On the other hand,

[a]dswang@physics.unimelb.edu.au

topological error correcting codes are designed with such locality constraints in mind and hence are particularly well adapted to these architectures. It has been shown that the 2d surface code [11] possesses a threshold error rate approaching 1% [1, 2]. Additionally, use of defect braiding permits for long-range, multi-qubit interactions [12, 13].

The major drawback of the 2d surface code lies in its use of state distillation in performing $S$ and $T$ gates to achieve universal quantum computation. This method requires one be able to mass produce logical qubits approximating the states

$$|Y\rangle = (|0\rangle + i\,|1\rangle)/\sqrt{2}, \qquad |A\rangle = (|0\rangle + e^{i\pi/4}\,|1\rangle)/\sqrt{2}. \tag{1}$$

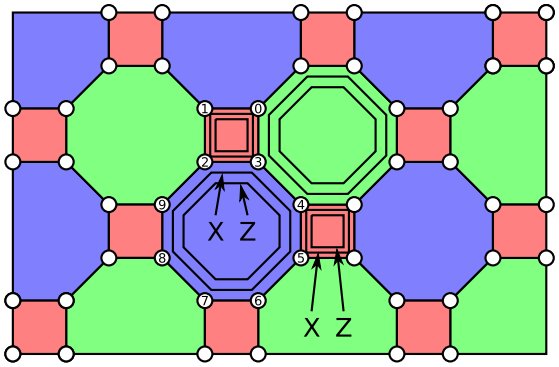
Several approximations to one of these states are run through a distillation circuit to produce a single state better approximating that state. For example, the $|Y\rangle$ state distillation circuit (corresponding to implementing $S$ gates) takes seven states approximating $|Y\rangle$ as input to produce a better approximate to $|Y\rangle$. Through several levels of iteration, one can arrive at the desired state with arbitrary accuracy. This procedure requires a large number of qubits, potentially several orders of magnitude greater than the rest of the computer, dedicated to producing such states.

This motivates work towards finding a topological scheme which bypasses the issue of state distillation. In this paper we will consider the *color code* of [3], which may be adapted to incorporate the desirable features of the surface code, whilst preserving its ability to implement $S$ gates transversely [14]. Although the issue of state distillation remains for $T$ gates, it is significant progress towards a distillation free topological code. At present what is missing is a determination of the error threshold of the color code.

Due to the similarity between the surface code and the color code, and the relative sizes of their stabilizers, one can argue that the threshold for the color code should be between $10^{-4}$ and $10^{-3}$, but this has yet to be demonstrated numerically. Recently, the threshold for a different color code scheme featuring a honeycomb lattice has been found to be $p_{\text{th}} = 0.109(2)$ by mapping to a random 3-body Ising model [15]. In these topological codes, one typically makes heavy use of classical computation in order to diagnose the sources of errors from a given syndrome. This task is non-trivial and one must have some efficient algorithm when applying these codes in real life. In order to determine the threshold for such a code, one in fact makes use of exactly this procedure, for it is necessary to restore the state back into a codeword and hence determine the logical failure.

Following our work on the surface code, we devise a method that may be applied in real life without modification to correct errors on the color code based on finding approximate hypergraph matching solutions. Using this approach, we find the average time to failure of an encoded quantum memory under both error-free (*ideal*) and error-prone (*non-ideal*) syndrome extraction. We also determine the threshold analytically under ideal syndrome extraction by combinatoric arguments, without the burden of performing the hypergraph matching. We find the threshold for the 2d color code under ideal syndrome extraction to be lower bounded by $p_{\text{th}} \geq 6.25\%$, and numerical simulations indicate it may be as high as $p_{\text{th}} = 13.3\%$ comparable to the honeycomb color code [15] and the surface code [2, 16] under similar circumstances. However, inclusion of the syndrome extraction circuits drops the threshold error rate to approximately $0.10 \pm 0.01\%$.

This paper is organized as follows. Section 2 reviews the color code and briefly discusses




$$\langle\psi| Z_0Z_1Z_2Z_3 |\psi\rangle = +1$$

$$\begin{aligned}&\langle\psi| X_0Z_0Z_1Z_2Z_3X_0 |\psi\rangle \\ &= -\langle\psi| X_0X_0Z_0Z_1Z_2Z_3 |\psi\rangle \\ &= -1\end{aligned}$$

Fig. 1. 2d lattice of qubits for the color code. White circles represent data qubits. Ancilla qubits located within the plaquettes are not shown. The stabilizer generators are the tensor products of $X$ and $Z$ on the qubits around each plaquette: $X_0X_1X_2X_3$, $Z_0Z_1Z_2Z_3$, $X_2X_3X_4X_5X_6X_7X_8X_9$, $Z_2Z_3Z_4Z_5Z_6Z_7Z_8Z_9$, etc. The state $|\psi\rangle$ is initialized to the simultaneous $+1$ eigenstate of all the generators. A single $X$ error, $|\psi\rangle \to X|\psi\rangle$, will be observed as an eigenvalue change on the adjacent $Z$-stabilizers due to the commutation relations between the Pauli matrices.

how error correction is performed, assuming that some logical state has been encoded into the surface. The discussion of logical qubits and logical operations is deferred until section 3. We return to the details of error correction in section 4. The simulation procedure and the results under non-ideal syndrome extraction are presented in section 5. Section 6 presents threshold estimates using two different methods under ideal syndrome extraction, firstly by simulation and then by combinatorics.

## 2 Error correction on the 2d color code

Consider the 2d lattice of qubits arranged as shown in figure 1, assuming for now that the lattice extends indefinitely. Each plaquette is associated with two generators of the stabilizer group; the tensor product of the Pauli-$X$ matrix, $X$, on the qubits around its perimeter, and the tensor product of the Pauli-$Z$ matrix, $Z$, on those same qubits. Neighboring plaquettes always share two qubits, ensuring all stabilizers commute. Assigned to each stabilizer is a color, red, green or blue, such that each qubit belongs to exactly one $X$-stabilizer and one $Z$-stabilizer of each color. The plaquette colors shown in figure 1 are not inherent to the system, merely a device to aid error correction. For the purposes of this paper, red stabilizers will always be synonymous with square stabilizers.

In the absence of errors, the state of the system is the simultaneous $+1$ eigenstate of each stabilizer. An $X$-error on one qubit causes the state to toggle between the $\pm 1$ eigenvalue states of the adjacent $Z$-stabilizers due to the commutation relations. Ancilla qubits located within each plaquette allow the eigenvalue to be measured locally. The configuration of eigenvalue changes measured forms the *syndrome*, from which the physical location of the errors can be inferred. Similarly, $Z$-errors are detected by measuring the eigenvalues of the $X$-stabilizers. Figure 2 illustrates more complex possible syndromes.

Combinations of errors often conspire in such a way to conceal the intermediate eigenvalue flips of many plaquettes. These sets of errors, or *error chains*, may be seen as the primitives generating the observed syndrome instead of the independent errors on individual qubits. We

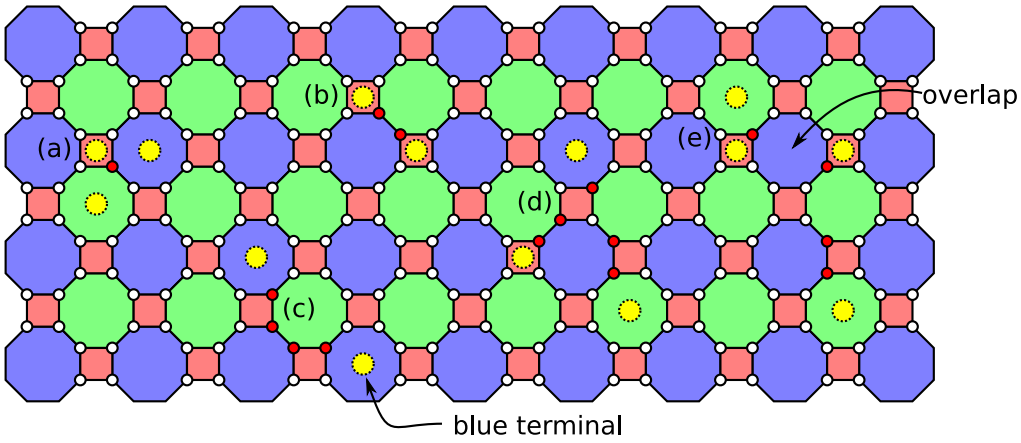



Fig. 2. Examples of syndromes produced by error chains (color online). A red circle indicates a physical error on the given data qubit. A yellow circle at a site indicates a change in eigenvalue from its previous measurement. (a) a single error toggles three plaquette eigenvalues. (b) 2-chain generating two red terminals. (c) 2-chain generating two blue terminals. (d) 3-chain generate three different color terminals. (e) many-terminal error chains can be decomposed into 2-chains and 3-chains with overlapping terminals.

will also regard single errors as error chains. The eigenvalue changes an error chain induces are its *terminals*. The color of a terminal is the color of the plaquette whose eigenvalue it alters. Error chains may have two same color terminals (2-chains), three terminals with one of each color (3-chains), or more terminals. Any chain having in excess of three terminals may be decomposed into superpositions of lower order chains, with some terminals overlapped (figure 2e). Indeed any 2-chain (figure 2b) may be derived from a pair of 3-chains, however they are treated as a primitive due to their relative simplicity. Since all observable syndromes are generated by superpositions of error chains, the identification of error chains is equivalent to locating errors. Given an arbitrary syndrome, error chain identification is carried out using algorithms from graph theory as follows.

A *matching*, $M$, of an undirected graph, $G$, is a subgraph of $G$ such that each node in $M$ has exactly one incident edge. A *perfect matching* is a matching where all nodes in $G$ belong to $M$. A minimum-weight perfect matching is an element from the set of perfect matchings, whose sum of edge weights is minimized. Many minimum-weight perfect matchings may be possible, in which case we recover just one. Polynomial-time matching algorithms for graphs exist [17–19], and are often found in the error correction procedure of toric codes. The color code lattice requires a hypergraph matching algorithm, for which efficient implementations are not known. A *hypergraph* is a generalisation of a graph, where edges are promoted to *hyperedges*, sets of arbitrary numbers of vertices. The *rank* of the hypergraph is the maximum cardinality hyperedge. We will see that one requires only a rank-3 hypergraph matching algorithm—not one of arbitrary rank—and in section 4 we describe an approximate but efficient implementation based on matching specially constructed graphs. For the present discussion, we assume the ability to determine the hypergraph matching, efficiently or otherwise.

Given a syndrome, the eigenvalue changes observed translate to nodes on a hypergraph.

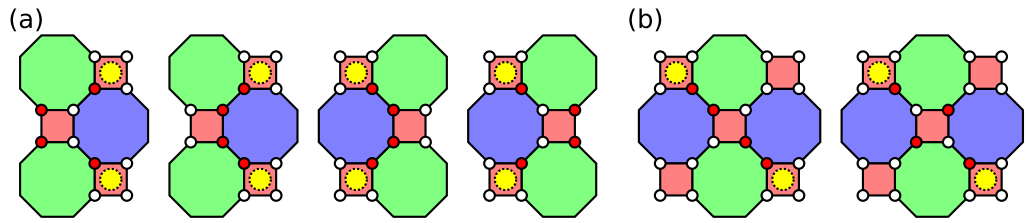



Fig. 3. Two different syndromes caused by four errors. Case (a) is more probable than case (b) as it can be generated by more length-4 error chains, and one should weight it accordingly in the graph to recover the matching of maximum likelihood. Our simulations do not consider such fine details, instead we choose only one of the possibilities covering each case. Incorporating these alternatives could potentially increase the threshold.

Any possible 2-chain required to generate a pair of terminals is represented by an edge joining the corresponding pair of nodes. Similarly, a hyperedge is added for every possible 3-chain. A matching on this hypergraph then represents a corresponding set of error chains which together will partially generate the syndrome, because each terminal belongs to exactly one error chain. Thus a perfect matching will reproduce the entire syndrome, allowing error correction to be performed.

Many matchings can be found for a given syndrome. Since the syndrome arises from physical errors which have a low probability of occurring, one should rig the weights of the edges and hyperedges such that the matching algorithm finds the matching of maximum likelihood. In general, the weight of an edge between some terminals should take into consideration all of the different possible error chains generating those terminals. For example, although the terminals in both cases shown in figure 3 are equidistant, figure 3a should be weighted as more probable than figure 3b purely because there are more length-4 error chains generating the former setup. However, in order to simplify matters, we will not take this into consideration; when many error chains are possible, we choose the graph edge to represent only one of the minimum-length possibilities. Under this approximation, the weight of an edge is taken to be the error chain length (which is proportional the logarithm of the probability), so that a chain formed by 2 errors is weighted identically to two independent single error chains. Although a minimum-weight perfect matching under this approximation will *not* necessarily be the matching of maximum likelihood, it will nevertheless reproduce the observed syndrome using the fewest number of errors; the hypergraph is constructed in a way that the weight-sum of its minimum-weight matching is the minimum number of errors required to reproduce the syndrome under ideal syndrome extraction.

The question immediately arises as to what happens when one corrects along a path of qubits which did not suffer errors. The discussion requires a formal introduction to logical qubits and logical gates, so we only make some brief remarks. The distance of a code, $d$, is defined to be the minimum number of physical operations that must be applied to the physical qubits encoding some logical state in order to interchange between the two logical states without generating any observable syndrome. Under ideal syndrome extraction, a code can in principle correct any $\lfloor \frac{d-1}{2} \rfloor$ error events. This holds true for the color code when one corrects by minimum-weight hypergraph matching. If fewer than $\lfloor \frac{d-1}{2} \rfloor$ errors occur, error correction will always produce rings of operators instead of chains of operators connecting

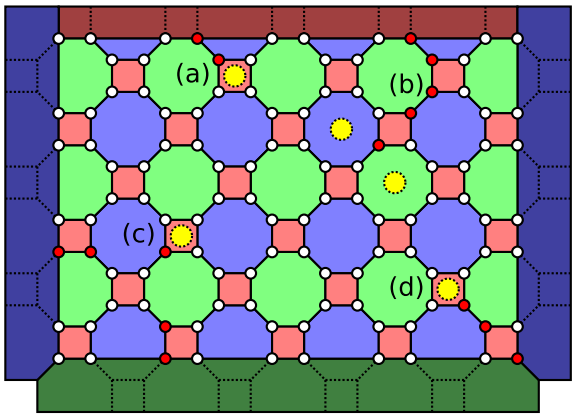



Fig. 4. Examples of syndromes on finite lattices. Dark plaquettes show the different color boundaries, where eigenvalues are not measured. (a) 2-chain with masked red terminal. (b) 3-chain with masked red terminal. (c) 3-chain with two masked terminals. (d) the interface between a green and a blue boundary can also be considered a red boundary.

boundaries. The rings of operators are stabilizers of the code, thus the logical state is restored.

As described, many-terminal error chains such as in figure 2e are not handled, resulting in misidentification of the causes of syndromes; this particular syndrome can be produced by 4 errors, however it currently appears to the error corrector as a 10 error event (generated by a pair of 2-chains). One solution is to add these error chains into the hypergraph in the form of higher cardinality hyperedges. However, this not only increases the number of hyperedges exponentially, but it also increases the complexity of the matching algorithm. There is a more elegant solution which rests on the fact that the plaquette eigenvalue observed determines only the parity, not the exact number of terminals, at that location; one can artificially insert a pair of *dummy nodes* into the graph for a plaquette suspected of harboring overlapping terminals, exactly as if two terminals had been observed there. Misplaced dummy pairs can in the worst case be matched to one another by a weight-0 edge, whence matching continues as if the pair had not been introduced. Our simulations need not anticipate such instances: as we have access to the number of toggles at each plaquette, we introduce a pair whenever an eigenvalue changes twice or more. However, in a real implementation such information is not available so one should devise some clever algorithm to identify these overlaps and introduce these nodes only as required. The benefit from introducing dummy pairs is that one eliminates the need to match arbitrary rank hypergraphs (only rank-3 is necessary). Note that even the introduction of dummy pairs on every plaquette still incurs only a polynomial-time overhead.

A physical implementation of this code must be done on a finite lattice, and hence the presence of boundaries which can hide otherwise visible terminals (figure 4). There are three boundary colors. A red boundary is the interface shielding red terminals from being observed, so that, for example, a lone red node can be produced by a 2-chain from a red boundary. Green and blue boundaries are defined equivalently. Note that the interface between green and blue boundaries can also be considered a red boundary. The case of a 3-chain producing exactly two mixed-color terminals and one masked terminal is easily accommodated for by adding an edge between those two terminals in the hypergraph, with the implicit prescription

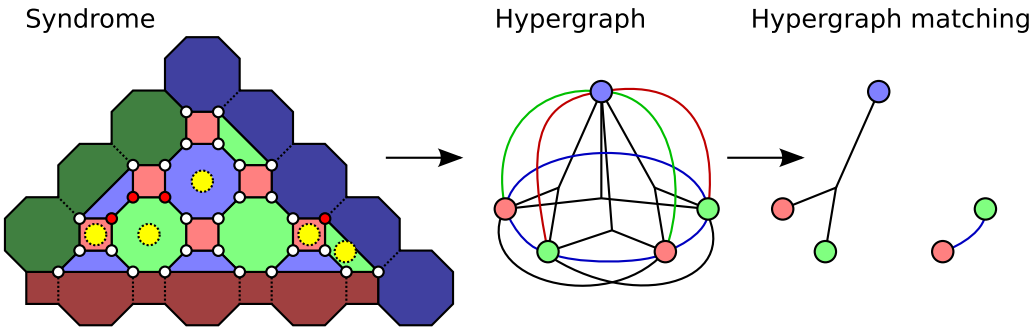



Fig. 5. The hypergraph constructed from the observed syndrome. Edge weights and boundary nodes have not been included. An edge between two different colored terminals is colored only to serve as a reminder that it denotes a 3-chain to that colored boundary. The hypergraph matching identifies a corresponding set of error chains which, once corrected, restores the state of the system into a codeword state.

that this edge denotes generating these two terminals by joining them to the closest boundary by a 3-chain. Indeed, any edge or hyperedge between some arbitrary number of nodes can denote whatever means is necessary to generate exactly those terminals, independently of all other terminals observed.

The single terminal case is more involved. Let $G$ denote the hypergraph one constructs as described so far. The intent is to create beside each node, $A$, an associated *boundary node*, $A'$. Each node is joined to its own boundary node, corresponding to generating that terminal independently, for example by a 2-chain to the closest same color boundary. However, this change by itself permits only one perfect matching: because the degree of every boundary node is exactly 1, every node must be matched to its boundary. If two regular nodes, $A$ and $B$, are matched together, their respective boundary nodes, $A'$ and $B'$, are unmatched and thus this is not a valid perfect matching. The resolution is to create a subhypergraph within the boundary nodes mirroring $G$ with only weight-0 edges and hyperedges, so that when $A$ is matched to $B$ (or $B$ and $C$ by a hyperedge), then $A'$ is readily matched to $B'$ ($B'$ and $C'$) at no extra cost. While it is not true that the boundary node's matching will always reflect that of the regular node's, this is of no concern; error correction works with only edges and hyperedges in the matching involving at least one regular node. This extra change successfully deceives the matching algorithm into behaving as desired, and error correction on finite lattices may be performed.

Figure 5 illustrates the error correction methodology described here. It is not strictly limited to this color code, and may be adapted to other similarly oriented 2d topological schemes, such as the honeycomb color code. Indeed, the error correction methodology of the 2d surface code is a specialization of the same technique, whereby one forms and matches only rank-2 hypergraphs (ie. graphs).

## 3 Logical qubits and logical gates

So far we have described the error correction procedure on the color code, which preserves some quantum state encoded onto the surface. In order to perform computation, logical qubits must be introduced. Following the original paper [3], each logical qubit is encoded onto a

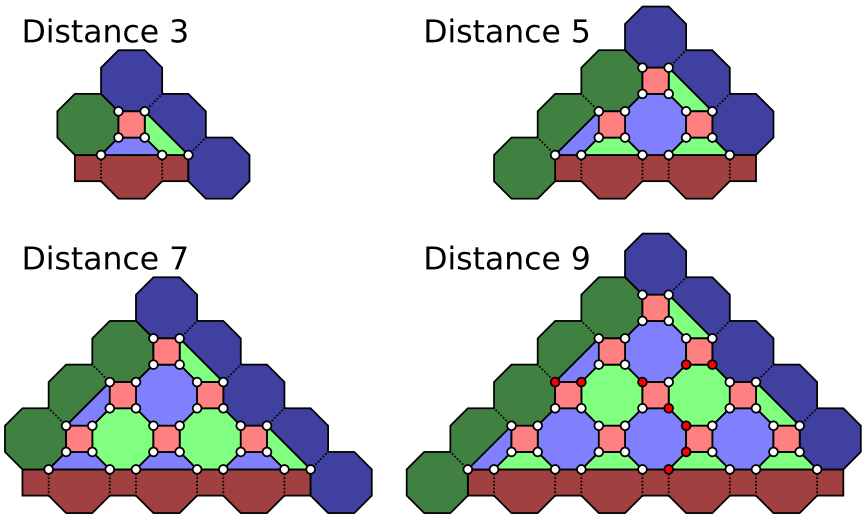



Fig. 6. Triangular lattices permitting Clifford group gates to be performed transversely. Shown are distance 3, 5, 7 and 9 logical qubits, and the different colored boundaries in dark. Also shown on the distance-9 lattice is an example of a logical-$X$ (or logical-$Z$) operation; a completely masked 3-chain.

triangular lattice, as shown in figure 6.

The logical-$X$ operation is defined to be any 3-chain of $X$-operations on the data qubits connecting together all three color boundaries, or any 2-chain from a qubit along one boundary to the opposite same colored boundary. The symmetry between $X$ and $Z$-stabilizers constrains the logical-$Z$ operation to be the same 3-chains but with $Z$ applied to the sites. Furthermore, these chains must be odd in length to yield the correct commutation relations. The minimum length of these chains defines the *distance* of the code. Therefore all distances $d$ will be assumed to be odd hereafter. Note that none of these operations change the syndrome because all terminals are masked by boundaries.

The nature of the $X$ and $Z$ stabilizers allows logical Hadamard operations to be performed transversely; when applying the Hadmard gate on every qubit, the identity $Z = HXH$ implies that $X$-chains will transform to $Z$-chains, and vice-versa.

The primary interest in using the color code is its ability to perform transversal $S$ gates. Since each plaquette comprises four or eight qubits, and neighboring plaquettes share an even number of qubits, when an $S$ gate is applied to every qubit, every $|0_L\rangle$ state acquires the same phase, and similarly for every $|1_L\rangle$ state. Furthermore, for all odd distance codes, an odd number of qubits is enclosed within the triangle lattice, ensuring that $|0_L\rangle \rightarrow |0_L\rangle$ and $|1_L\rangle \rightarrow \pm i\,|1_L\rangle$.

Finally, controlled-not gates may be implemented transversely between two different sheets of triangular lattices; this operation will propagate $X$-chains from control qubit to target qubit, and $Z$-chains from target to control.

It is possible to adapt the color code so that the entire quantum computer shares a single code [14], creating logical qubits by introducing defects into the surface — contiguous regions where the eigenvalues of particular stabilizer generators are no longer measured, thus

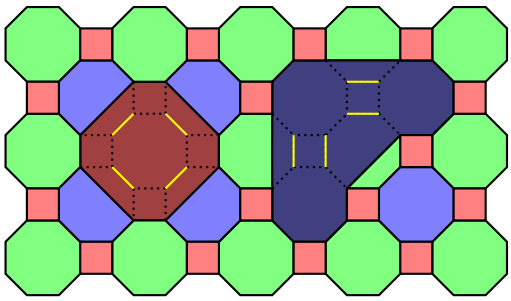


Fig. 7. Examples of defects on the color code. Green and blue defects share the same form. Logical qubits are formed by one defect of each color, allowing the color code to adopt many of the benefits of the 2d surface code, with the additional benefit of transversal $S$ gates [14].

providing extra degrees of freedom — in a similar vein to surface codes [20]. In this manner, it is possible to recover many of the features of surface codes such as long range gates. In addition, one can isolate the triangular lattices above using a defect of each color, examples of which are shown in figure 7, thus permitting transversal $S$ gates. The details go beyond the scope of this paper.

## 4 Hypergraph mimicry and pair assignment

An essential requirement for the color code error correction procedure as formulated in section 2 is the existance of an efficient rank-3 hypergraph weighted perfect matching algorithm. Whether an efficient algorithm exists is unclear; additional information from, for example, the specialized structure and the geometry may assist the problem. Recovering the minimum-weight hypergraph matching, which presumably produces better results than matchings of higher weight-sum, is not strictly necessary and may be relaxed to achieve an efficient error corrector.

While we have formulated the error correction problem such that any syndrome can be represented as a hypergraph matching, let us clarify that we never match the hypergraph directly; they are strictly limited to discussion. Our procedure is to reduce the initial hypergraph problem down to a simpler but approximate graph matching problem which we can solve. A solution to the graph problem may be mapped back to give a hypergraph matching and thus is a candidate for error correction, although it will not necessarily have as low a weight-sum as the minimum-weight hypergraph matching.

Consider the following scenario: Alice and Bob each have the hypergraph ahead of time, to which Alice has found the minimum-weight matching with weight-sum $W_{\text{hyper}}$. Alice wishes to communicate enough information to Bob, such that Bob can still reproduce her result in polynomial time. She can opt to send Bob all of the hyperedges in her matching. Bob, knowing the remaining nodes are matched by edges, can remove all hyperedges from the initial hypergraph to form a graph, which he can match efficiently to recover the remainder of Alice's matching (figure 8).

It is possible for Alice to communicate only two of the three nodes in each matched hyperedge, and still have Bob recover her minimum-weight hypergraph matching efficiently. To do so, Bob must collapse the indicated pairs into a single node. For each of Alice's matched

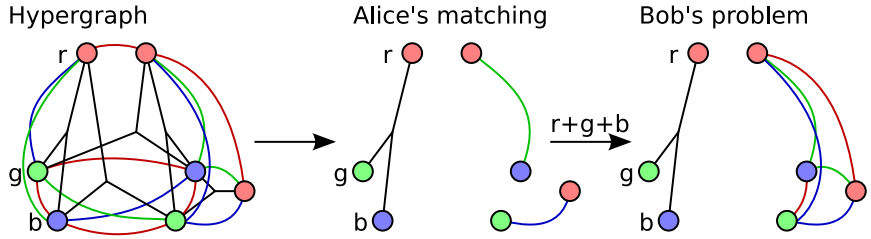



Fig. 8. Alice informs Bob exactly which nodes match as hyperedges. Using this information, Bob factors out those nodes from the hypergraph. Since he also knows the remaining nodes are matched by edges, he also eliminates the hyperedges, reducing the hypergraph down to a graph which can be matched in polynomial time.

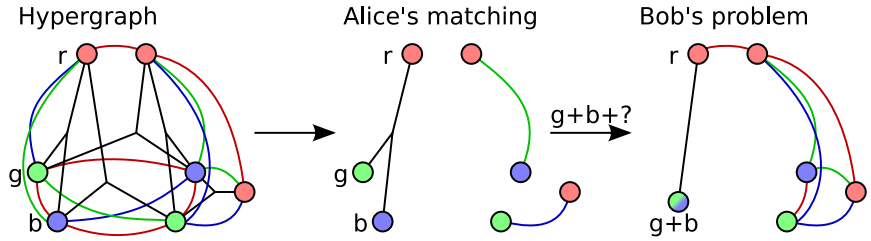



Fig. 9. Alice informs Bob which green and blue nodes together form hyperedges. In order for Bob to recover Alice's matching, he collapses the indicated blue and green nodes into a single node, so that hyperedges involving both blue and green nodes become edges, and all other edges are discarded. Since Bob also knows that all other nodes do not form triplets, the remaining hyperedges may also be discarded. Thus Bob's problem reduces to a graph matching problem.

hyperedges, Bob replaces the two nodes indicated by a single node, and the hyperedges connecting these two nodes in the original hypergraph are replaced by edges. Because he knows no other nodes are matched by hyperedges, any remaining hyperedges can be removed and thus Bob's problem reduces again to matching a graph (figure 9).

Obviously we do not have the luxury of Alice's input, and so in both scenarios illustrated we would instead perform an exhaustive search over all of Alice's possible inputs. Such a search exhibits exponential behaviour which may be counteracted by introducing approximations; trying only a small subset of the potential inputs at the expense of finding the true minimum-weight hypergraph matching. In order to scale up this approximation to higher distance codes, we will devise a speculative algorithm following Alice's second technique for it has a smaller domain.

Our method to assign together pairs of nodes is also achieved by matching a specialized *mimic graph*, which we now construct. Let us temporarily neglect all red nodes and build the hypergraph formed by just the green and blue nodes. Since we have not included red nodes, in reality this is simply a graph. In order to account for the $O(n_r n_g n_b)$ rank-3 hyperedges without actually introducing hyperedges, one must insert $O(n_g n_b)$ extra nodes. This can be done by substituting each edge connecting green and blue nodes with a series of edges, via four newly introduced intermediate nodes: $\bar{g}$, $p$, $p'$, $\bar{b}$ (figure 10).

This transformation in itself does not affect matchings on this graph; the choice of matching

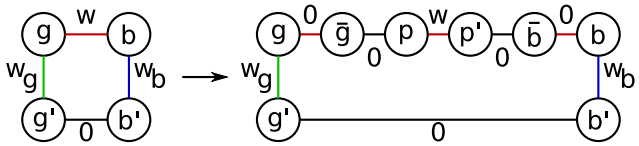



Fig. 10. Recall that edges connecting green and blue nodes in the hypergraph imply a connection to a red boundary by a 3-chain. Construction of the mimic graph begins by replacing each of these edges with a series of edges, and introducing four new nodes: $\bar{g}$, $p$, $p'$, $\bar{b}$. The newly introduced $p$ act as $g$ and $b$ combined, and $p'$ its boundary. The edge between $p$ and $p'$ carries the weight of the original edge. The $\bar{g}$ and $\bar{b}$ nodes are necessary ensure only $g$ and $b$ are not used in two error chains; once independently and once via the $p$ node.

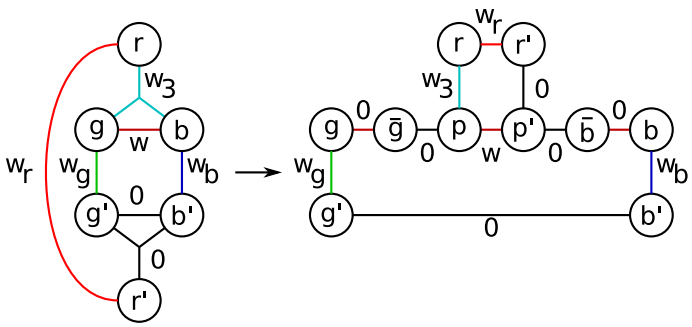



Fig. 11. Red nodes in the hypergraph are incorporated into the mimic graph. The hyperedge between $r$, $g$, $b$ in the original hypergraph becomes an edge between $r$ and $p$ in the mimic graph. The red boundary node, $r'$, is also joined to the pair boundary node, $p'$, by a weight-0 edge.

the edge $g, b$ becomes a choice of matching an alternating set of edges. However, we have the new interpretation of $p$ being the amalgam of $g$ and $b$, and $p'$ its boundary. The special nodes $\bar{g}$ and $\bar{b}$ are *disables*, which always have degree 2. They serve to ensure each node does not participate in two error chains, once as an individual and once as a pair. The interpretation of $p$ as the pair formed by combining $g$ and $b$ allows us to add the hyperedges into the mimic graph; edges joining red nodes to pair nodes. As before, we join the respective boundaries together to fool the perfect matching condition (figure 11).

We still need to incorporate the case of generating red and green terminals by connecting to a blue boundary, and similarly for the red and blue terminals connecting to the closest green boundary. We can join red nodes to green nodes (and their respective boundary nodes) with the implicit assumption that those two terminals join to the closest blue boundary to form a 3-chain. However, for our approximate method it is best to minimize the degree of the nodes. We instead choose to insert additional blue nodes along the blue boundary plaquettes, used solely for forming these 3-chains. Whether or not the boundary actually masked a terminal is unimportant; an introduced node can at worst be matched to its own boundary node by a weight-0 edge and matching continues as if they had not been introduced, akin to the introduction of dummy pairs. Similarly, we introduce green nodes along the green boundary to account for joining red and blue terminals by a 3-chain. The procedure incurs a polynomial overhead and can be optimized to minimize the number of extra nodes.

This mimic graph emulates some of the properties of the hypergraph and can be matched efficiently. In particular, all matchings on the hypergraph are matchings on the mimic graph. However, the converse if *not* true; matchings of mimic graphs may not necessarily be translated into hypergraph matchings. This is due to an implicit demand for one of three particular patterns within the triplet region (figure 12), when including the red nodes. The matching algorithm knows nothing of such patterns, nor does the placement of weights assist it. Thus what we extract from the mimic graph matching itself is *not* a correction in general; only when it does not contain any malformed patterns is this true. However, we can use the matching as a choice for pair assignment to then produce a correction. We identify the matching of $r$ to $p$ and the subsequent matching of $g$ to $\bar{g}$ (due to $\bar{g}$ having degree-2) as the assignment of $r$ to $g$.

There are two reasons for this choice. First, such combinations will always arise if a hypergraph matching were translated into a mimic graph matching. Second, one can often rotate the right hand side by a weight-0 alternating cycle, forming the desired patterns (figure 13). An alternating cycle on a graph is a simple cycle with edges alternating between included and excluded from the matching. The weight of an alternating cycle is the sum of the weights of edges not in the matching minus the weight of edges in the matching, so that the new matching will have weight $W + w$. One could correct the mimic matching by searching for lowly weighted alternating cycles, though this can be inefficient as one must be careful about breaking existing well-formed patterns. We observe that for these simple cycles, one can instead rematch with $r$ and $g$ collapsed into a single node, thus avoiding these complicated searches.

From this choice of pair assignments, we collapse the rank-3 hypergraph with dummy nodes down to a *trial graph*, as Bob had done previously after receiving Alice's partial input. This trial graph may be matched efficiently to give a trial solution with some weight-sum $W_{\text{trial}}$. A

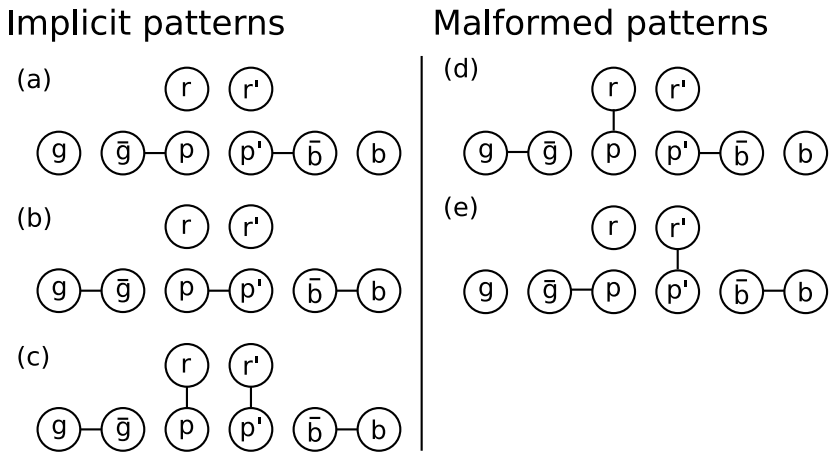



Fig. 12. Left shows the three (partial) patterns implicitly demanded in the mimic matching if it is to always be interpretable as a correction. Malformed patterns may arise in the mimic matching, leaving this interpretation invalid. (a) extra nodes introduced were unused. (b) green and blue connect together to closest red boundary. (c) red, green and blue form 3-chain.

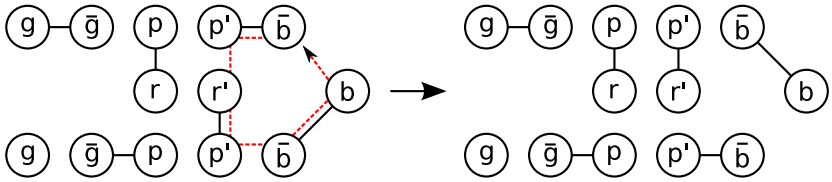



Fig. 13. Malformed matchings can often be corrected by adding a weight-0 alternating cycle, shown in dashed lines. Adding an alternating cycle to a matching toggles the edges between included and excluded from the matching, yielding a new matching.

single choice of assignment can go amiss, so we take the solution with the lowest weight-sum over several initial assignments. In addition, there are six simple mimic graph constructions possible — one for each permutation of red, green and blue[b]— and, in general, the minimum-weight matching of each can have a different weight-sum. The variants with the *highest* weight-sum matching, $W_{\text{mimic}}$, are taken to yield matchings that closest approximate the minimum-weight hypergraph matching, and hence give better choices of pair assignment and trial solutions. Only the matchings from these variants are used as choices of pair assignment. This probabilistic method appears to yield near-optimal matchings even as the codeword distance increases. The reason we assume that higher weight-sum mimic graph matchings give better trial solutions is due to the relative ordering of the weight-sums:

$$W_{\text{mimic}} \leq W_{\text{hyper}} \leq W_{\text{trial}} \tag{2}$$

In general, we do not know the weight-sum of the minimum-weight rank-3 hypergraph matching, $W_{\text{hyper}}$. However, as we have constructed the mimic graph to encompass possible hypergraph matchings, the mimic matching weight-sum is upper bounded by the minimum-weight hypergraph matching weight-sum. Extra freedom from the mimic matching not necessarily being a correction allows for its weight-sum to be lower: $W_{\text{mimic}} \leq W_{\text{hyper}}$. Similarly, after deciding upon a choice of assignment, many potential hypergraph matchings are discarded. Thus all trial matchings must have weight-sum $W_{\text{trial}} \geq W_{\text{hyper}}$. Notice that if $W_{\text{trial}} = W_{\text{mimic}}$, the heuristic has not introduced any additional errors over the minimum-weight hypergraph matching; the trial matching itself is a minimum-weight hypergraph matching.

It is the number of uncertain matchings we use to gauge the quality of the approximation. We find that using up to $6 \times 25$ initial trials, this method has a 95% probability of recovering the minimum-weight hypergraph matching with certainty from a single time step with $\frac{d+1}{2}$ errors scattered for large lattices. Taking $6 \times 50$ trials does not significantly increase the probability. From the remaining uncertain cases, the final correction applied typically has a weight-sum only one greater than the mimic matching weight-sum, so that if it were not the hypermatching it at least falls very close. For our simulations, we have chosen $6 \times 25$ initial trials.

## 5 Simulation results

Simulations take place on the triangular lattices of figure 6. In our simulations we determine the average number of syndrome extraction cycles a quantum state encoded in a color code endures before error correction results in a logical failure. The simulations trace only the propagation of $X$-errors throughout the machine.

A single simulation instance proceeds as follows. First, the quantum computer is initialized perfectly in the simultaneous $+1$ eigenstate of every stabilizer generator. At each timestep, each qubit has a probability $p$ of a memory error, then the syndrome information is extracted simultaneously over the entire surface. Red syndrome is extracted by preparing

[b] For the mimic graph permutation we have worked with, we can crudely identify the red-to-pair edge in the mimic graph as the 3-chain. The mimic matching may not be a correction as it allows for blue terminals to be either not corrected, corrected once, or corrected twice (by a 2-chain and a 3-chain). Red and green terminals do not suffer such problems; they will always be corrected exactly once. Thus even the interchange of green and blue can produce different results.

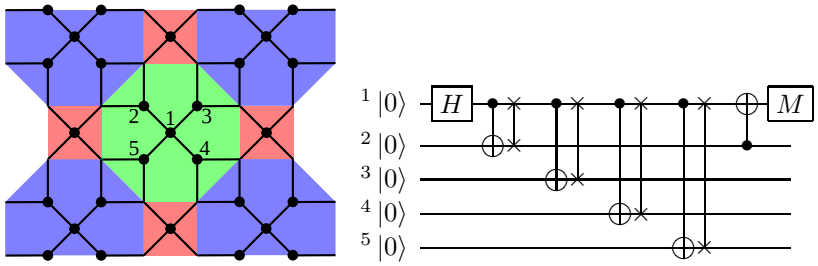


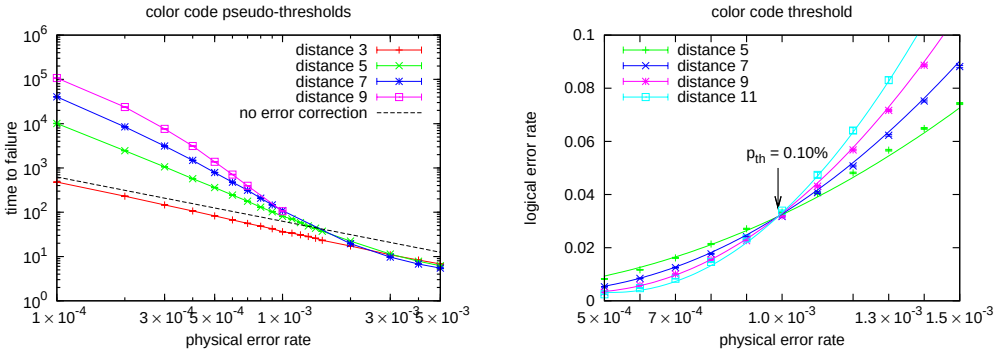

Fig. 14. The layout of ancilla qubits on the lattice (dots) and the controllable two-qubit interactions (lines) used in the simulations. When extracting the parity from the octogonal plaquettes, one first prepares a four qubit cat-state (in qubits 2–5) using the circuit shown, then interacts each qubit with two data qubits. The four measurement results are finally combined together.

the ancilla positioned within each red stabilizer in the $|0\rangle$ state ($|+\rangle$ state for $Z$-syndromes). Subsequently four controlled-nots are directed inwards (outwards) from the surrounding data qubits to the ancilla, which are then measured in the $Z$-basis ($X$-basis). A colored plaquette syndrome is extracted by first preparing a 4-qubit cat-state using the circuit shown in figure 14. Each qubit in the cat-state interacts with two distinct data qubits in that stabilizer, and is measured independently. These four sub-processes are executed simultaneously, yielding four measurements which combine together to give the parity of that 8-qubit stabilizer. Further details of syndrome extraction are available in [14].

A memory error on a qubit is a probability $p/3$ of incurring either an $X$, $Z$ or $Y = XZ$ error. Preparation of $|0\rangle$ and $|+\rangle$ states each have a probability $p$ of resulting in the $|1\rangle$ and $|-\rangle$ states respectively. Similarly, measurement in the $X$ and $Z$-bases have a probability $p$ of obtaining the incorrect result. Finally, the two-qubit gates first perform the desired operation ideally, followed by an equal probability $p/15$ of each of the 15 non-trivial tensor products of $I$, $X$, $Y$ and $Z$.

After each timestep, the syndrome information is used to correct the state and check for logical failure. Errors during syndrome extraction are taken into account by collating syndrome information over time, forming a 3d structure of eigenvalue changes. Error chains are permitted to span through time, with those segments denoting an error during syndrome extraction in the prior timestep. In our simulations, an error chain segment spanning through time is weighted equally to one of the same length spanning though space. In addition, we also collect one final syndrome ideally before error correction and checking for logical failure. Should error correction fail using this augmented syndrome information, we record the failure time. Otherwise, we continue to the next timestep recollecting this syndrome non-ideally. Using this procedure, the average life expectency of a logical state is shown figure 15(a). The error bars shown are the standard deviation in the value after averaging over a large number of failure times. Also shown is the average lifespan of a single non-error-corrected qubit.

There are two features of significance. Firstly, the gradients of the curves are observed to converge to the same value for low error rates for distance-5 and above. As with the surface code, correlated errors during the syndrome extraction cycle have caused a distance-$d$ code to correct fewer than the expected $E(d) = \lfloor \frac{d-1}{2} \rfloor$ errors. Despite the distance-3 color code




(a) The average time to failure of a quantum memory in the color code. The vertical axis is proportional to real time. Error bars represent the uncertainty in the average time taken over a large number of instances.

(b) The logical error rate as a function of physical error rate $p$. The threshold error rate is approximately $p_{\text{th}} = 0.10 \pm 0.01\%$.

Fig. 15. Threshold error rate for the 2d color code.

being identical to the 7-qubit Steane code, the topological method of dealing with errors during syndrome extraction coupled to our rather simplistic weighting of error chain segments results in the color code offering no benefits over unerror-corrected qubits. A combination of these correlated errors during syndrome extraction and our approximate error correction method are responsible for distance 5, 7 and 9 codes ultimately being able to correct the same number of errors. However, higher distance codes can be seen to improve upon lower distances; there are fewer combinations of two errors causing a distance-7 code to fail than for distance-5, and fewer yet again for distance-9. In light of this, we assert that higher distance codes will eventually be able to correct more errors under the error correction method presented.

Secondly, the intersections of successive distance codes is highly mobile. Furthermore, the intersections move leftwards thus do not give a lower bound on the threshold. This in part may be due to the presence of boundaries of the lattice; our simulations of the toric code and the surface code show similar trends [2], though admittedly are much less extreme. We will revisit this issue in section 6.2. Due to computing limitations, we are unable simulate higher distance codes to observe a convergence under the time to failure approach. The existing results indicate a threshold of approximately 0.1%.

We typically choose to simulate time to failure as it shows the true error correcting potential in the low $p$ limit. However, it is also possible to obtain a threshold by simulating for exactly $d$ timesteps and determining the logical error rate $p_L$. The results are shown in figure 15(b). Following the work of [21], quadratic curves are fit around the threshold error rate, obeying

$$p_L = A + B(p - p_{\text{th}})d^{-\nu} + C\left[(p - p_{\text{th}})d^{-\nu}\right]^2. \tag{3}$$

This second approach provides a threshold of $p_{\text{th}} = 0.10 \pm 0.01\%$ (the error bar comes about from the final fit). Close observation again shows wildly fluctuating pseudo-thresholds.

## 6 Ideal threshold error rate

We now seek to determine the threshold under ideal syndrome extraction. In this limit, there is no need to consider error chains spanning through time, making theoretical calculations much more accessible. We proceed in two directions: firstly by direct simulation, and secondly by counting the number of dangerous syndrome patterns.

### *6.1 Direct simulation*

We already have the means to simulate the color code logical failure rates under ideal syndrome extraction. One point of note is that because syndrome information is always correct, we no longer need to consider error chains spanning through time; error chain segments spanning through time are now weighted infinitely greater than those spanning through space. As such, each timeslice is corrected independently of all other timeslices. This implies that the total number of possible syndromes is finite, thus one can determine the logical error rate per timestep by summing over all possible dangerous syndrome combinations:

$$p_L^{(d)}(p_0) = \sum_{k=0}^{Q} A_d(k) p^k (1-p)^{(Q-k)}, \quad p = \frac{2}{3} p_0 \tag{4}$$

Here $p_0$ is the physical error rate, $Q(d)$ is the number of data qubits, and $A_d(k)$ is the number of dangerous syndromes resulting from $k$ errors in the distance-$d$ code. The factor of $\frac{2}{3}$ is due to our error model: only two of the three possibilities $X$, $Y$, $Z$ may lead to logical-$X$ failures. For later convenience, we will make the change of variable $k \to (F+k)$, where $F(d) = \frac{d+1}{2}$ is the minimum number of errors required for a distance-$d$ code to fail under true minimum-weight hypergraph matching and ideal syndrome extraction. We will always reference a prefactor $A_d(F+k)$ by its offset $k$ from the expected leading order term.

$$p_L^{(d)}(p_0) = p^F \sum_{k=-F}^{Q-F} A_d(F+k) p^k (1-p)^{(Q-F)-k}, \quad p = \frac{2}{3} p_0 \tag{5}$$

The exact value for $A_d(F+k)$ depends on the details of the matching algorithm used, and can be obtained by running the error corrector for each possibility. Since the total number of $(F+k)$-error configurations to test is large, $\binom{Q}{F+k}$, we approximate $A_d(F+k) \approx \binom{Q}{F+k} r_k$ by determining only the ratio of $(F+k)$-error failures, $r_k$, from a large random sample. The results in the ideal syndrome extraction limit by direct simulation of the error correction procedure are shown in figure 16. The data points are obtained from simulations of the code at certain error rates, while the curves are given by equation 5, using our matching algorithm to estimate the prefactors $A_d(F+k)$. We observe a threshold of $p_{\text{th}} = 13.3\%$.

### *6.2 Dangerous syndrome coverage*

One can place an upper bound on $A_d(F+k)$, the number of dangerous syndromes as a result of $F+k$ errors, under true minimum-weight hypergraph matching. We will assume hypergraph matching hereafter, in particular with regards to dangerous syndromes, and simply refer to it as the *matching*. By construction of the hypergraph, the matching will correct for all $E(d) = \frac{d-1}{2}$ error cases: $A_d(F+k) = 0, \forall\ k < 0$. Non-trivial contributions to the logical

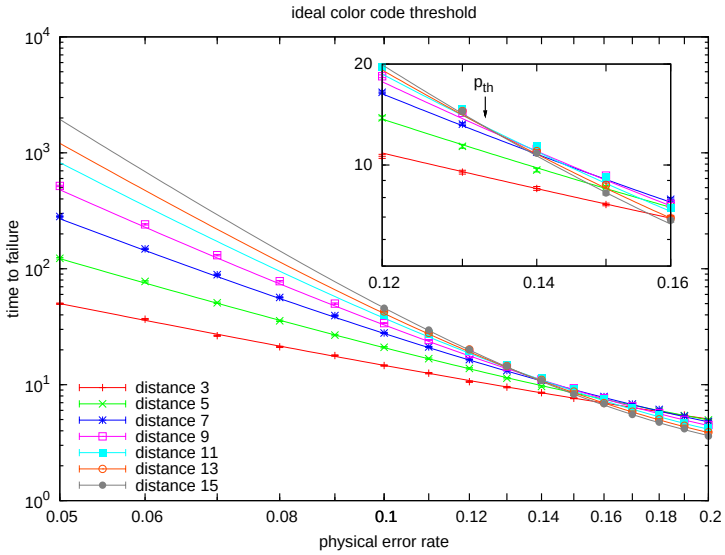



Fig. 16. Average life expectancy of a quantum state under error free syndrome extraction. The minimum error rate at which successive distance codes intersect is taken to best approximate the asymptotic threshold, $p_{\text{th}} = 13.3\%$.

failure rate start from $F(d) = \frac{d+1}{2}$.

$$p_L^{(d)}(p_0) = p^F \sum_{k=0}^{Q-F} A_d(F+k) p^k (1-p)^{(Q-F)-k}, \quad p = \frac{2}{3} p_0 \tag{6}$$

For an $F$ error syndrome to cause failure, the $F$ errors must all fall on a single length-$d$ logical operator, $\hat{O}_d$; minimum-weight matching would find it preferable to correct the state by applying corrections on the remaining $E$ qubits in $\hat{O}_d$, thus performing the logical operation. Thus an upper bound to $A_d(F)$ can be determined by enumerating all length-$d$ logical operators, then choosing $F$ qubits from each.

It is tempting to make the statement that all dangerous syndromes as a result of $F + k$ errors simply stem from a dangerous $F$-error syndrome and scattering a further $k$ errors. Unfortunately, this simplistic argument is *not* valid, as dangerous syndrome formed otherwise exist due to the presence of higher length logical operators. One can find counter-examples, and the difference in gradient between $A_d(F)$ (figure 17) and $A_d(F + k)$ (figure 18) further reinforce their presence.

We conjecture that all dangerous $F+k$ error syndromes are formed by some combination of $F$ errors on qubits belonging to a single continuous length-$(d+2\lambda)$ error chain, then scattering a further $k$ errors onto the remaining qubits, where $\lambda$ is a constant. A *continuous* error chain is one which may be derived placing a single error, generating some terminals, then shifting these terminals by the rules of figure 19. The rules are defined such that any logical operator can be formed in this way. In particular, because length-$(d+2\lambda)$ logical operators are formed in this way, the conjecture trivially holds true for the $(F + \lambda)$-error case. The conjecture is a constraint on the distribution of errors required for logical failure for a length-$d$ code. For

example, not any arbitrary placement of $F$ errors will cause a code to fail, only very select combinations, namely those where the errors occur on qubits belonging to a single length-$d$ logical operator.

One can upper bound the number of continuous length-$(d + 2\lambda)$ error chains. We start by placing an initial error on the $Q$ qubits on the lattice, generating up to three terminals. Discarding backtracking, each terminal can be shifted by one of up to 6 rules (figure 19). Some care should be taken when moving red stabilizers as one has additional options that are not listed, such as those shown in figure 3. Similarly, minor modifications are necessary when a terminal lies next to a boundary. Regardless, for a continuous length-$(d + 2\lambda)$ error chain, because each shift is accomplished by placing down pairs of errors, one must make a total of $\frac{d-1}{2} + \lambda$ shifts, shared between the three terminals. There are approximately $\left(\frac{d-1}{2}\right)^2$ ways to divide these shifts between the three terminals. Thus the maximum number of continuous length-$(d + 2\lambda)$ error chains is:

$$N_d < Q \left(\frac{d-1}{2}\right)^2 6^{(d-1)/2+\lambda} \tag{7}$$

Our conjecture then bounds the number of dangerous syndromes as a result of $F + k$:

$$A_d(F + k) < N_d \binom{d + 2\lambda}{F} \binom{Q - F}{k} \tag{8}$$

Since $F = \frac{d+1}{2}$, it follows that $\binom{d+2\lambda}{F}$ is exponentially bounded, using $\binom{\chi}{\chi/2} = \frac{\chi!}{(\chi/2)!(\chi/2)!} \leq 2^\chi$:

$$\binom{d + 2\lambda}{F} \leq \binom{d + 2\lambda}{F + \lambda} \leq 2^{d+2\lambda} \tag{9}$$

Substituting $A_d(F + k)$ back into equation 6, then simplifying using the binomial expansion:

$$p_L^{(d)}(p_0) < N_d 2^{d+2\lambda} p^F \sum_{k=0}^{Q-F} \binom{Q - F}{k} p^k (1 - p)^{(Q-F)-k}, \quad p = \frac{2}{3} p_0 \tag{10}$$

$$= N_d 2^{d+2\lambda} p^F \tag{11}$$

For the case of $A_d(F)$, the dangerous $F$-error syndromes are the various combinations of $F$ errors along the length-$d$ logical operators. For a given combination, the continuous error chain covering it is the logical operator from which it was derived. In this case, terminals must always step towards their boundary, narrowing the choices of shifting terminals down to approximately $4^{(d-1)/2}$. We can calculate $A_d(F)$ exactly for hypergraph matching by first enumerating all length-$d$ logical operators, and then find all unique combinations of $F$ errors. The logical operators themselves can be determined, for example, by deforming some given initial logical operator by the combinations of the stabilizer generators. These results are shown in figure 17, confirming our bound on $A_d(F)$ displays the correct asymptotic behaviour.

The higher order prefactors $A_d(F + 2)$ and $A_d(F + 4)$ are shown alongside the more general theoretical bound in figure 18. The counts shown in these graphs are taken from the preceeding simulations since enumerating quickly becomes difficult. The reason for choosing $F + 2$ is that in the color code, logical operators have length $d + 4n$ should only contribute to $A(F + 2)$ and higher. Our results show that the $A_d(F + k)$ gradients agree with the

| d | $A_d(F)$ |
|---|---|
| 3 | 21 |
| 5 | 332 |
| 7 | 5807 |
| 9 | 100,120 |
| 11 | 1,671,668 |
| 13 | 27,227,258 |

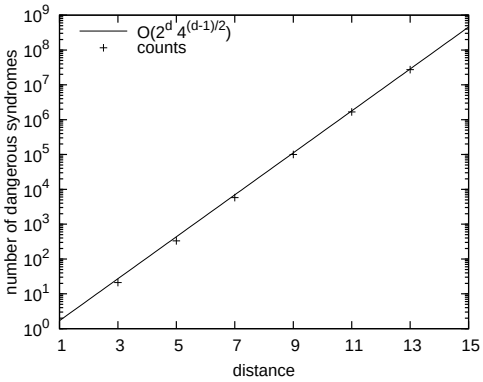



Fig. 17. The leading contribution to the logical error rate, $A_d(F)$, when using minimum-weight hypergraph matching grows exponentially with the distance of the code. These results were obtained by counting the length-$d$ logical operators and determining unique combinations of $F$ errors lying along a single logical operator.

combinatoric error rate, giving the initial conjecture further merit. Unfortunately, due to the approximate nature of our matching, higher length logical operators do in fact contribute to $A_d(F+1)$ and even $A_d(F)$, so that they too display the same $6^{(d-1)/2}$ growth rate.

It is not necessary to determine the prefactor in the logical error rate (equation 11) with a great deal of precision to calculate an asymptotic threshold; the complexity itself is sufficient. An asymptotic threshold is obtained by comparing the logical error rates of successive distance codes in the limit of large distance, $d$, whereupon all polynomial factors in $d$ disappear:

$$1 = \lim_{d\to\infty} \frac{p_L^{(d+2)}}{p_L^{(d)}} \tag{12}$$

$$p_{\text{th}} = \frac{3}{2}\frac{1}{6\cdot 2^2} = 6.25\% \tag{13}$$

Figure 20a shows the average lifespan of a quantum memory over different distances using equation 11. However, we have already observed that the prefactors $A_d(F+k)$ follow two different growth rates — $O(4^{(d-1)/2})$ for $k \leq 1$, and $O(6^{(d-1)/2})$ for $k \geq 2$ — hence one should not so readily simplify the equation. Applying equation 6 with the additional clamping of $A_d(F+k) \leq \binom{Q}{F+k}$ gives figure 20b. Interestingly, it features the same fluctuating pseudo-thresholds as observed in the simulations. While we have noted that our simulation $A_d(F)$ grows at the faster rate, it is presumed that the earlier terms are suppressed at different rates, giving rise to the same effect.

These results rest on the initial conjecture, from which we have deduced the growth rates of $A_d(F+k)$. Simulation results appear to follow the predicted growth rates. However, it remains to be rigorously proven that some constant $\lambda$ exists for all $d$, at least on this geometry, from which a lower bound to the threshold evidently follows.

## 7 Conclusion

We have described a general error correction procedure suitable for many 2d topological codes as a minimum-weight hypergraph perfect matching problem. We have also described

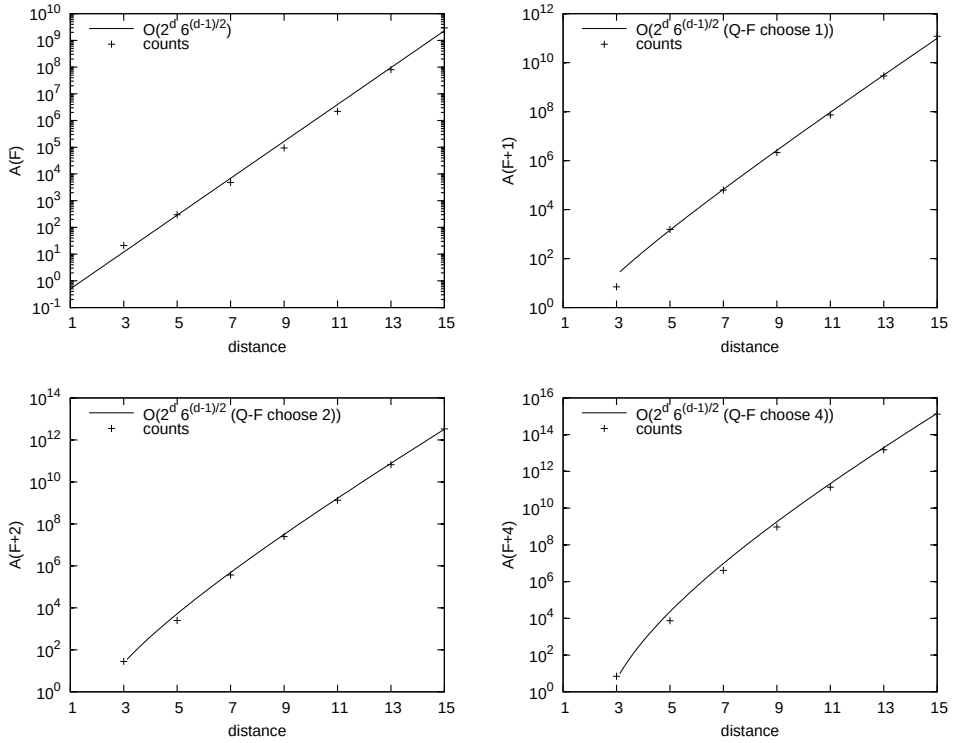



Fig. 18. The number of dangerous syndromes as a result of $F+k$ errors is $O\left(6^{(d-1)/2}\ 2^d\ \binom{Q-F}{k}\right)$. The data points were obtained using the approximate matching method outlined in section 4.

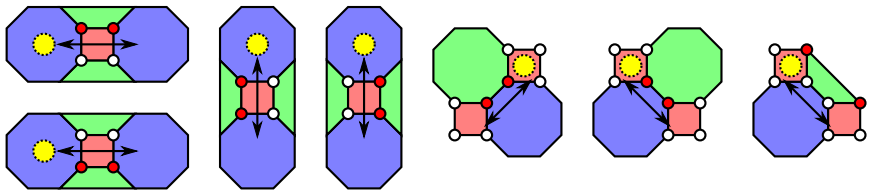


Fig. 19. Rules of shifting terminals amongst same color plaquettes. Each move requires exactly two errors, shown as the red qubits. Green and blue terminals share the same rules. Red terminals have additional multi-step rules, such as those shown in figure 3. Extra choices may be possible for terminals beside the boundary.

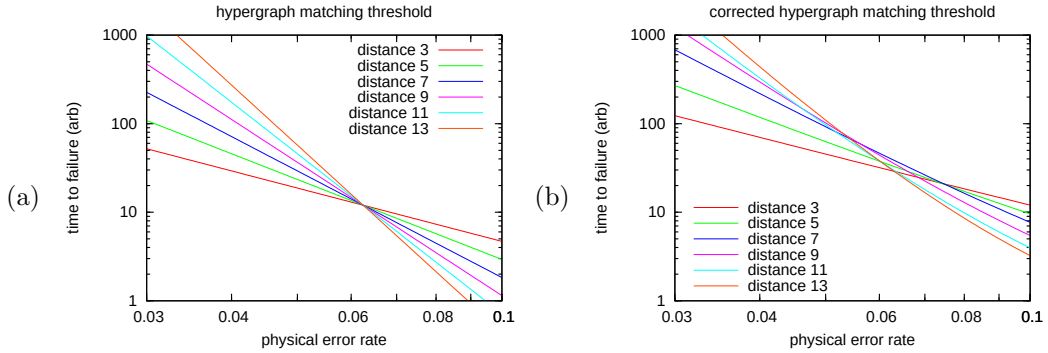



Fig. 20. (a) Lower bounds to the expected average time to failure when correcting by minimum-weight matching under ideal syndrome extraction, using equation 11 as the logical error rate. The asymptotic threshold is $p_{\text{th}} = 6.25\%$. (b) The different growth rates between $A_d(F+k)$ for $k \leq 1$ and $k \geq 2$ can be accounted for by using equation 6 to determine the logical error rate. This leads to fluctuating pseudo-thresholds. In this graph, we have also used $A_d(F+k) = \binom{Q}{F+k}$ when equation 8 exceeds $\binom{Q}{F+k}$.

an efficient but approximate method for matching rank-3 hypergraphs required for this color code, which in principle may be used for other 3-color codes. The method seeks solutions by constructing two graphs: one upper-bounded by the hypergraph solution, the other lower-bounded. When the two bounds meet, we identify that the approximation has introduced no errors. Thus this method can in principle be implemented as an initial pass before less efficient methods, which attempt to improve the code's performance, for example, to ensure that a distance-$d$ code reliably corrects $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors.

Combinatoric arguments presented here suggest that the asymptotic threshold error rate of the color code is lower bounded by $p_{\text{th}} \geq 6.25\%$ under error free syndrome extraction. Simulations using the approximate hypergraph matching method show that the lower bound on the threshold under these conditions may be as high as $p_{\text{th}} = 13.3\%$. Once faulty syndrome extraction circuits are introduced, numerical simulations indicate that the threshold may fall to approximately $0.10 \pm 0.01\%$. Unfortunately, this begins to encroach on the realm of the concatenated codes, which do not need such complex error correction procedures and some of which bypass the use of state distillation. An efficient specialized matching algorithm for the color code in the presence of boundaries may be possible, potentially raising these lower bounds on the threshold and improving its prospects.

## Acknowledgements

We thank A. M. Stephens and J. T. Anderson for helpful discussions. This work was supported by the Australian Research Council, the Australian Government, and the US National Security Agency (NSA) and the Army Research Office (ARO) under contract number W911NF-08-1-0527.

## References

1. R. Raussendorf, J. Harrington, and K. Goyal. Topological fault-tolerance in cluster state quantum

computation. *New J. Phys.*, 9:199, 2007. quant-ph/0703143.
2. D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg. Threshold error rates for the toric and planar codes. *Quant. Info. Comp.*, 10:456–469, 2010. arXiv:0905.0531.
3. H. Bombin and M. A. Martin-Delgado. Topological quantum distillation. *Phys. Rev. Lett.*, 97:180501, 2006. quant-ph/0605138.
4. P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493, 1995.
5. A. R. Calderbank and P. W. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54(2):1098–1105, 1996. quant-ph/9512032.
6. A. M. Steane. Multiple particle interference and quantum error correction. *Proc. R. Soc. Lond. A*, 452:2551–2576, 1996. quant-ph/9601029.
7. P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.
8. E. Knill, R. Laflamme, and W. H. Zurek. Accuracy threshold for quantum computation. Technical Report LAUR-96-2199, Los Alamos National Laboratory, 1996. quant-ph/9610011.
9. K. M. Svore, D. DiVincenzo, and B. Terhal. Noise threshold for a fault-tolerant two-dimensional lattice architecture. *Quant. Info. Comp.*, 7:297, 2007. quant-ph/0604090.
10. F. M. Spedalieri and V. P. Roychowdhury. Latency in local, two-dimensional, fault-tolerant quantum computing. *arXiv:0805.4213*, 2008.
11. A. Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Ann. Phys.*, 303:2–30, 2003. quant-ph/9707021.
12. R. Raussendorf, J. Harrington, and K. Goyal. A fault-tolerant one-way quantum computer. *Ann. Phys.*, 321:2242–2270, 2006. quant-ph/0510135.
13. A. G. Fowler, A. M. Stephens, and P. Groszkowski. High-threshold universal quantum computation on the surface code. *Phys. Rev. A*, 80(5):052312, Nov 2009. arXiv:0803.0272.
14. Austin G. Fowler. Universal quantum computation without state distillation on a 2-D color code. *arXiv:0806.4827v1*, 2008.
15. Helmut G. Katzgraber, H. Bombin, and M. A. Martin-Delgado. Error threshold for color codes and random three-body ising models. *Phys. Rev. Lett.*, 103(9):090501, Aug 2009. arXiv:0902.4845.
16. E. Dennis, A. Kitaev, A. Landahl, and J. Preskill. Topological quantum memory. *J. Math. Phys.*, 43:4452–4505, 2002. quant-ph/0110143.
17. J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965.
18. J. Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards*, 69(1-2):125–130, 1965.
19. W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS J. Comput.*, 11:138–148, 1999.
20. A. G. Fowler and K. Goyal. Topological cluster state quantum computing. *Quant. Info. Comp.*, 9:721–738, 2009. arXiv:0805.3202.
21. C. Wang, J. Harrington, and J. Preskill. Confinement-higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory. *Ann. Phys.*, 303(1):31–58, 2003.