Combining hard and soft decoders for hypergraph product codes

Antoine Grospellier, Lucien Grouès, Anirudh Krishna, Anthony Leverrier

Combining hard and soft decoders for hypergraph product codes

Antoine Grospellier[1], Lucien Grouès[1], Anirudh Krishna[2], and Anthony Leverrier[1]

[1]Inria, 2 Rue Simone IFF, CS 42112, 75589 Paris Cedex 12, France

[2]Université de Sherbrooke, 2500 Boulevard de l'Université, Sherbrooke, QC J1K 2R1, Canada

Hypergraph product codes are a class of constant-rate quantum low-density parity-check (LDPC) codes equipped with a linear-time decoder called small-set-flip (SSF). This decoder displays sub-optimal performance in practice and requires very large error correcting codes to be effective. In this work, we present new hybrid decoders that combine the belief propagation (BP) algorithm with the SSF decoder. We present the results of numerical simulations when codes are subject to independent bit-flip and phase-flip errors. We provide evidence that the threshold of these codes is roughly 7.5% assuming an ideal syndrome extraction, and remains close to 3% in the presence of syndrome noise. This result subsumes and significantly improves upon an earlier work by Grospellier and Krishna (arXiv:1810.03681). The low-complexity high-performance of these heuristic decoders suggests that decoding should not be a substantial difficulty when moving from zero-rate surface codes to constant-rate LDPC codes and gives a further hint that such codes are well-worth investigating in the context of building large universal quantum computers.

1 Introduction

It is imperative to make quantum circuits fault tolerant en route to building a scalable quantum computer. The threshold theorem [1, 22, 23] guarantees that it will be possible to do so using quantum error correcting codes which encode information redundantly. This redundancy serves as a buffer against errors but we need to be mindful of the trade-offs involved as the number of qubits we can control in the laboratory is limited. A relevant figure-of-merit to quantify this trade-off is the *overhead*, defined as the ratio between the number of qubits in a fault-tolerant implementation of a quantum circuit to the number of qubits in an ideal, noise-free environment.

Low-density parity-check (LDPC) codes are a natural class of codes to consider for implementations. They are families of stabilizer codes $\mathcal{C}_n = \{[\![n, k, d]\!]\}_n$ such that every stabilizer generator acts on a constant number of qubits and every qubit is involved in a constant number of generators [18]. Quantum architectures that would not satisfy these conditions would probably be very difficult to scale up, for instance because of difficulties to extract a syndrome fault-tolerantly. A stronger restriction asks for quantum LDPC codes with geometric locality, where interactions only concern neighboring qubits in a 2 or 3-dimensional setup, but it is well known that this requirement severely restricts the ability of these codes to store information [3]. On the other hand, general quantum LDPC codes can display a constant encoding rate $k/n = \Theta(1)$, while maintaining a large minimum distance $d = \Omega(\sqrt{n})$. In a breakthrough paper, Gottesman exploited this favorable encoding rate and described a construction of fault-tolerant quantum circuits with constant space-overhead [19]. This means that if we considered an ideal circuit that processes m qubits, then its fault-tolerant counterpart will only require $\Theta(m)$ qubits.

Constructing good LDPC codes is difficult because we need to balance two competing constraints – on the one hand, we want the weight of the stabilizers to be low, but on the other hand we want the stabilizers to commute. Tillich and Zémor [38] proposed a construction called the hypergraph product code which overcomes this difficulty (see also generalizations [24, 40]). This con-

struction takes two good *classical* LDPC codes and constructs a quantum LDPC code with parameters $k = \Theta(n)$ and $d = \Theta(\sqrt{n})$. In some sense, this construction generalizes the toric code and allows one to obtain a constant encoding rate while keeping the LDPC property as well as a large minimum distance. As shown by Krishna and Poulin, there exists a framework rich enough to perform gates fault tolerantly on this class of codes [27, 28]. Devising low-complexity decoding for hypergraph product codes is arguably one of the main challenges in the field right now.

Early analytic works establish and estimate the threshold of a broad class of LDPC codes with sublinear distance scaling [13, 25] without the use of an efficient decoder. In [29], Leverrier *et al.* have shown the existence of a linear-time decoder for the hypergraph product codes called `SSF` and proved it corrects errors of size $O(\sqrt{n})$ in an adversarial setting. In [15], Fawzi *et al.* showed that the `SSF` decoder corrects with high probability a constant fraction of random errors in the case of ideal syndromes and later made these results fault tolerant by showing that this decoder is robust to syndrome noise as well [14]. To be precise, they showed that `SSF` is a single-shot decoder and that in the presence of syndrome noise, the number of residual qubit errors on the state after decoding is proportional to the number of syndrome errors. These works yield a rigorous proof of existence of a threshold for this class of codes, but only provide very pessimistic bounds on the numerical value of the threshold. Beginning with the seminal work of [12], statistical mechanical models have been used to make indirect estimates of the threshold of quantum error correcting codes [2, 9]. Exploiting these ideas, Kovalev *et al.* [26] showed that certain hypergraph product codes can achieve a relatively high threshold (approximately 7×10^{-2}) with the minimum-weight decoding algorithm. Such an algorithm is too complex to be implemented in practice for general LDPC codes, however.

We note that some recent work from Panteleev and Kalachev investigated a quantum version of Ordered Statistical Decoding and obtained promising results for decoding small quantum LDPC codes [33].

Related work: Other families of quantum LDPC codes with constant rate include 2D and 4D hyperbolic codes: while the 2D version has a logarithmic minimum distance [5, 11, 17], the 4D hyperbolic codes satisfy $d = \Omega(n^c)$ for some $c > 0$ and can therefore be interesting for fault-tolerance [4, 21, 30, 31]. Variants of these codes exhibit very good features and we compare our results to earlier works on hyperbolic codes. The works [10, 41] study some properties of these codes further in the context of fault-tolerant quantum computation.

Results and outline: In this paper, we present a new, efficient decoder for hypergraph product codes. We combine the `SSF` decoder with a powerful decoder for classical LDPC codes called belief propogation (`BP`). The resulting decoders boast low decoding complexity, while at the same time yielding good peformance. The idea behind these algorithms is to first decrease the size of the error using `BP`, and then correct the residual error using the `SSF` decoder. This paper subsumes and considerably improves upon an earlier work by Grospellier and Krishna [20]. We first study the performance of `SSF` by itself, and then study the performance of the hybrid decoder `Iterative BP` + `SSF`. Our simulations use a simple error model, that of independent bit-flip and phase-flip errors. When compared to [20], the thresholds of codes are significantly improved (from 4.5% to roughly 7.5%). We mention that there is not yet a theoretical threshold for the codes we focus on (obtained as a product of $(3, 4)$-regular codes) together with the `Iterative BP` + `SSF` decoder. Hence this is formally only a pseudo-threshold. Furthermore, since the decoder is less demanding on the underlying quantum error correcting code, the weights of the stabilizers are reduced. The stabilizers weights drop from 11 to 7. We then extend this idea to decoding in the presence of syndrome noise. In this model, we assume the syndrome bits are flipped with some probability in addition to qubits being subject to bit-flip and phase-flip noise. We find that our codes perform well using a modified decoder called `First-min BP` + `SSF`. The results are compared to the toric code, 2D and 4D hyperbolic codes.

In Section 2, we begin by providing some background and establishing our notation. We first review classical codes and discuss `flip` and the sum-product version of `BP` (simply referred to as `BP`), and then proceed to review hypergraph product codes, and why naive generalizations of clas-

sical decoders `flip` and `BP` fail. We introduce the `SSF` decoder and discuss how it overcomes these issues. Section 3 then presents some results of numerical simulations. Finally in Section 4 we discuss the results of simulations for faulty syndrome measurements.

2 Background

2.1 Classical codes

In this section, we shall review aspects of classical LDPC codes pertinent to quantum LDPC codes. We begin by discussing the association between codes and graphs. We then proceed to discuss expander graphs and the decoding algorithm `flip`. Finally we discuss a particular version of belief propagation (`BP`) called the sum-product algorithm.

A classical code family $\{\mathcal{C}_n\}_n$, where $\mathcal{C}_n = \ker H_n$ is the binary linear code with parity-check matrix H_n, is said to be LDPC if the row weight and column weight of H_n are upper-bounded respectively by constants Δ_C and Δ_V independent of n. The weight of a row (or column) is the number of non-zero entries appearing in the row (or column). In other words, the number of checks acting on any given bit and the number of bits in the support of any given check is a constant with respect to the block size. These codes are equipped with iterative decoding algorithms (such as belief propagation) which have low time complexity and excellent performance. Furthermore, they can be described in an intuitive manner using the factor graph associated with the classical code and for this reason these codes are also called graph codes.

The factor graph associated with $\mathcal{C} = \ker H$ is the bipartite graph $\mathcal{G}(\mathcal{C}) = (V \cup C, E)$ where one set of nodes V represents the bits (*i.e.*, the columns of H) and the other set C represents the checks (the rows of H). For nodes $v_i \in V$ and $c_j \in C$, where $i \in [n]$ and $j \in [m]$, we draw an edge between v_i and c_j if the i-th variable node is in the support of the j-th check, or equivalently if $\mathrm{H}(i, j) = 1$. It follows that a code $\mathcal{C}$ is LDPC if the associated factor graph has bounded degree, with left degree (associated with nodes in V) bounded by Δ_V and right degree bounded by Δ_C.

Of particular interest are expander codes, codes whose factor graph corresponds to an expander graph. Let $\mathcal{G} = (V \cup C, E)$ be a bipartite factor graph such that $|V| = n$ and $|C| = m$ such that $n \geq m$. We use $\Gamma(c)$ to denote the neighborhood of the node c in the graph $\mathcal{G}$. This naturally extends to a set S of nodes; $\Gamma(S)$ includes any nodes connected to nodes in S via an edge in $\mathcal{G}$. Furthermore, $\deg(c) = |\Gamma(c)|$ is the degree of a node c.

The graph $\mathcal{G}$ is said to be (γ_V, δ_V)-left-expanding if for $S \subseteq V$,

$$|S| \leq \gamma_V n \implies |\Gamma(S)| \geq (1 - \delta_V)\Delta_V |S| \,. \quad (1)$$

Similarly, the graph is (γ_C, δ_C)-right-expanding if for $T \subseteq C$,

$$|T| \leq \gamma_C m \implies |\Gamma(T)| \geq (1 - \delta_C)\Delta_C |T| \,. \quad (2)$$

It is a *bipartite* expander if it is both left and right expanding.

In their seminal paper, Sipser and Spielman [36] studied expander codes and applied an elegant algorithm called `flip` to decode them. They showed that if the factor graph is a left expander such that $\delta_V < 1/4$, then the `flip` algorithm is guaranteed to correct errors whose weight scales linearly with the block size of the code. Furthermore, it does so in time scaling linearly with the size of the code block.

`flip` is a deceptively simple algorithm and it is remarkable that it works. We describe it here as it forms the basis for the quantum case decoding algorithm `SSF`. Let $x \in \mathcal{C}$ be a codeword and y be the corrupted word we receive upon transmitting x through a noisy channel. With each variable node v_i in the factor graph, $i \in [n]$, we associate the value y_i. With each check node c_j in the factor graph, $j \in [m]$, we associate the syndrome bit $s_j = \sum_{i:v_i \in \Gamma(c_j)} y_i \pmod 2$. We shall say that a check node c_j is unsatisfied if the syndrome is 1 and satisfied otherwise. Note that if $y \in \mathcal{C}$ is a codeword, then all the checks c_j, $j \in [m]$, must be satisfied. Informally, `flip` searches for a variable node that is connected to more unsatisfied neighbors than it is to satisfied, and flips the corresponding bit. This reduces the number of unsatisfied checks. It is stated formally in Algorithm 1 in Appendix C (comments in blue).

The algorithm can be shown to terminate in linear time. For a detailed analysis, we point the interested reader to the original paper by Sipser and Spielman [36].

`flip` is not used in practice because it requires large code blocks to be effective [35]; instead we resort to `BP`. In what follows, we shall use the sum-product algorithm and use `BP` to refer to this algorithm. This algorithm is presented in Alg. 3.

`BP` proceeds iteratively with T iterations (described in Alg. 4) further broken down into two elementary steps. The first step (Alg. 5) involves variable nodes passing messages to checks and the second step (Alg. 6) exchanges the direction, and involves check nodes passing messages to variable nodes. We introduce some notation to refer to these objects. We let:

1. p be the error probability on the variable nodes,
2. s_j be the syndrome value of the check c_j (0 if satisfied, 1 otherwise),
3. $m^t_{v_i \to c_j}$ be the message sent from variable-node i to check-node j on iteration t,
4. $m^t_{c_j \to v_i}$ be the message sent from check-node j to variable-node i on iteration t,
5. λ^t_i is the approximate log-likelihood ratio computed at iteration t for the variable-node i: $\lambda^t_i > 0$ if it is more likely that the i-th variable node is more likely to be 0 than 1, otherwise $\lambda^t_i < 0$.

On graphs with cycles, `BP` can only compute approximate values of the posterior probabilities. However, it turns out to be relatively precise when the length of the smallest cycle (the girth) is big enough. Thus the constraints of `BP` are weaker than that for `flip` and do not require expander graphs.

2.2 Quantum codes

We now review the definition of the hypergraph product. We proceed to discuss quantum expander codes, and the decoding algorithm proposed by Leverrier, Tillich and Zémor called `SSF`. We then present some earlier results of numerical simulations from [20].

CSS quantum codes are quantum error correcting codes that only contain stabilizers each of whose elements are all Pauli-X operators (and identity) or all Z [8, 37]. The hypergraph product is a framework to construct CSS codes starting from two classical codes [38]. The construction ensures that we have the appropriate commutation relations between the X and Z stabilizers without resorting to topology. If the two classical codes are LDPC, then so is the resulting quantum code. In general, the construction employs two potentially distinct bipartite graphs, but for simplicity, we shall only consider the product of a graph with itself here. Let $\mathcal{G}$ be a bipartite graph, *i.e.*, $\mathcal{G} = (V \cup C, E)$. We denote by $n := |V|$ and $m := |C|$ the size of the sets V and C respectively.

These graphs define two pairs of codes depending on which set defines the variable nodes and which set defines the check nodes. The graph $\mathcal{G}$ defines the code $\mathcal{C} = [n, k, d]$ when nodes in V are interpreted as variable nodes and nodes C are represented as checks. Note that $m \geq n - k$ as some of the checks could be redundant. Similarly, these graphs serve to define codes $\mathcal{C}^T = [m, k^T, d^T]$ if C represents variable nodes and V the check nodes. Equivalently, we can define these codes algebraically. We say that the code $\mathcal{C}$ is the right-kernel of a parity check matrix H and the code $\mathcal{C}^T$ is the right-kernel of the transpose matrix H^T.

We define a quantum code $\mathcal{Q} = [\![n_\mathcal{Q}, k_\mathcal{Q}, d_\mathcal{Q}]\!]$ via the graph product of these two codes as follows. The set of qubits is associated with the set $(V \times V) \cup (C \times C)$. The set of Z stabilizers is associated with the set $(C \times V)$ and the X stabilizers with the set $(V \times C)$. Ref. [38] establishes the following:

Lemma 1. *The hypergraph product code $\mathcal{Q}$ has parameters:*

$$[\![n^2 + m^2, k^2 + (k^T)^2, \min(d, d^T)]\!].$$

Naively generalized to the quantum realm, both `flip` and `BP` perform poorly [34]. Unlike the classical setting, we are not looking for the exact error that occurred, but for any error belonging to the most likely error class since errors differing by an element of the stabilizer group are equivalent. In the case of `flip`, there exist constant size errors (typically half a generator) for which the algorithm gets stuck, which implies that `flip` will not work well even in a random error model.

Overcoming the failure of `flip`: Leverrier *et al.* [29] devised an algorithm called small-set-flip (`SSF`) obtained by modifying `flip`. This algorithm is guaranteed to work on quantum expander codes which are the hypergraph product of bipartite expanders. The algorithm is sketched

out in Alg. 2 in Appendix C (comments in blue). For a detailed analysis of the algorithm, we point the reader to [29]. Note that this is not the full decoding algorithm – it has to be run separately for both X and Z type errors.

Let $\mathcal{F}$ denote the union of the power sets of all the Z generators in the code $\mathcal{Q}$. For $E \in \mathbb{F}_2^{n^2+m^2}$, let $\sigma_X(E)$ denote the *syndrome* of E with respect to the X stabilizers. The syndrome $\sigma_X(E) \in \mathbb{F}_2^{nm}$ is defined as $\mathrm{H}_X E$; the j-th element of this vector is 0 if and only if the j-th X stabilizer commutes with the error E. Given the syndrome σ_0 of a Z type error chain E, the algorithm proceeds iteratively. In each iteration, it searches within the support of the Z stabilizers for an error F that reduces the syndrome weight. The case of X errors follows in a similar way by swapping the role of X and Z stabilizer generators.

Ref. [29] proceeds to show that SSF is guaranteed to work if the graphs corresponding to classical codes are bipartite expanders. They prove the following theorem (Theorem 2 in [29]):

Theorem 2. *Let $\mathcal{G} = (V \cup C, E)$ be a (Δ_V, Δ_C) biregular $(\gamma_V, \delta_V, \gamma_C, \delta_C)$ bipartite expander, with $\delta_V, \delta_C < 1/6$. Further suppose that (Δ_V, Δ_C) are constants as n and m grow. The decoder* SSF *for the quantum code $\mathcal{Q}$ obtained via the hypergraph product of $\mathcal{G}$ with itself runs in time linear in the code length $n^2 + m^2$, and it decodes any error of weight less than*

$$w = \frac{1}{3(1+\Delta_C)} \min(\gamma_V n, \gamma_C m) \ . \qquad (3)$$

Overcoming the failure of BP: While BP can be adapted to decode quantum LDPC codes, it does not perform very well. The most common behaviour when BP fails at decoding quantum LDPC codes is that it does not converge: the likelihood ratios of some nodes keep oscillating. This can be explained by the existence of some symmetric patterns in the Tanner graph which prevent BP from settling on a precise error. To circumvent this, Poulin and Chung suggested some workarounds [34] such as fixing the value of some qubits whose probabilities keep oscillating, or running BP again on a slightly modified Tanner graph where we randomly change the initial error probability of one of the qubits linked to an unsatisfied check. The idea behind both of these solutions is to break the symmetry of the code. While it does exhibit improvements, the results are still far from the performance of BP in the classical case.

An other approach to improve the performance of BP is to feed its output to a second decoder, with the hope that it will converge to a valid codeword if BP cannot. This idea was recently investigated by Panteleev and Kalachev [33] who considered a quantum version of the *Ordered Statistical Decoding* algorithm OSD [16]. This algorithm was imported from the classical case where it is either used alone or after BP. The idea of this algorithm is to sort the different qubits by their log likelihood ratios, a measure of their reliability, before proceeding with a brute force approach. Once OSD has sorted the qubits, it will brute force *all* valid corrections on the w least reliable qubits, where w is some tunable parameter, and then choose the most probable of these valid corrections (or fail if there are none). If w is proportional to the block-length, the time complexity is no longer polynomial. Instead we can use the OSD-0 algorithm which is a simplified version that reduces the error floor of BP. In practice, this appears to work almost as well as OSD-w. The time complexity is then polynomial, but may remain inappropriate for large codes.

In this work, we present some heuristic algorithms where the output of BP is fed to SSF. The idea behind these algorithms is to first decrease the size of the error using BP before correcting the residual error with the SSF decoder. In practice, if BP manages to sufficiently decrease the error weight, then SSF will often reach a valid codeword without making a logical error. We highlight that these hybrid algorithms have a time complexity far lower than that of OSD.

3 Ideal syndrome extraction

In this section, we study a decoding algorithm called Iterative BP + SSF. To this end, we consider hypergraph product codes subject to a simple noise model. We use classical codes generated with the configuration model, briefly described in Appendix B. We work with an independent bit-flip and phase-flip error noise model, where each qubit is afflicted independently by an X or Z error with probability p. The advantage of studying such an error model with CSS codes is that it is sufficient to try to correct X errors only to understand the performance of the whole decod-

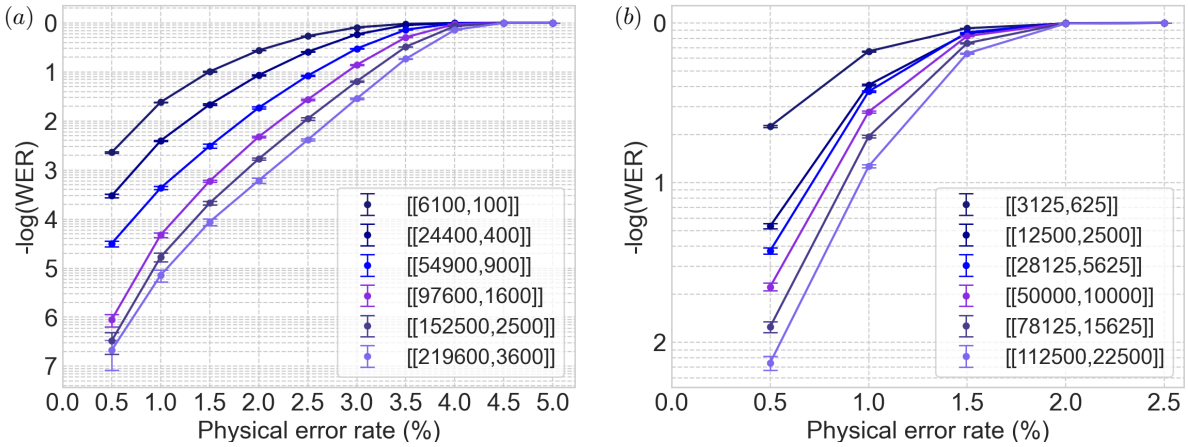

Figure 1: Variation of word error rate (WER) with the physical error for hypergraph product codes formed as product of regular (Δ_V, Δ_C)-regular graphs with the SSF decoder from [20]. Logarithms are base 10 throughout the paper. The errors bars indicate 99% confidence intervals, *i.e.*, approximately 2.6 standard deviations. (a) Codes obtained as the product of $(5, 6)$-regular graphs (encoding rate of $1/61 \approx 0.016$): we observe a threshold of roughly 4.5%. (b) Codes obtained as the product of $(5, 10)$-regular graphs (encoding rate of 0.2): we observe a threshold of roughly 2%.

ing algorithm. We focus here on ideal syndrome measurements. We will remove this assumption in the next section.

To establish a baseline, we begin by describing the performance of `SSF` as defined in [29]. Grospellier and Krishna [20] studied the performance of `SSF` on quantum codes obtained as the hypergraph product of two $(5, 6)$-regular graphs and $(5, 10)$-regular graphs.

Fig. 1 plots the logical error rate of these codes as a function of the physical error rate. In this context, the logical error rate refers to the word error rate (WER), *i.e.*, the probability that *any* logical qubit fails.

In numerical benchmarks, we found a correlation between the performance of the classical codes under `flip` and the performance of the resulting quantum codes under `SSF`. The best among these codes were chosen as representatives for the quantum case and correspond to the different curves in the figure. The $(5, 6)$-regular codes have a threshold of roughly 4.5%, whereas the $(5, 10)$-regular codes have a threshold of roughly 2%. In this context, the threshold is the physical error rate below which we find that the logical error rate decreases as we increase the block size. The error bars represent the 99% confidence intervals, *i.e.*, approximately 2.6 standard deviations. At first glance, it appears that the $(5, 10)$-regular codes perform much worse. However this can be attributed to a much higher encoding rate compared to the first code family ($1/5$ versus $1/61$).

Albeit promising, we note that `SSF` by itself requires large block sizes before it becomes effective. This is unsurprising considering its classical counterpart also exhibits the same behaviour. As mentioned in the previous section, this shortcoming of `flip` is addressed in the classical case by using instead soft decoding such as `BP`. However, used naively, `BP` fails in the quantum realm. In practice, it fails at finding a valid codeword, but still manages to get rather close in the sense that the syndrome weight can be reduced by an order of magnitude.

Our idea to exploit this property is to start the decoding procedure with `BP` and switch to `SSF` after a certain number of rounds. For such an approach to work for a noisy syndrome, we need to specify a criterion to switch between the two decoders. Here, however, we consider a noiseless syndrome extraction and can apply the following simple iterative procedure: try decoding the error with `SSF` only; if this does not work, then perform a single round of `BP` followed by `SSF`; if this still does not work, then perform 2 rounds of `BP` before switching to `SSF`; and so on until a codeword is finally found, or when a maximum number $T_{\max}$ of `BP` rounds is reached. In the latter case, we say that the decoder failed. This heuristic defines

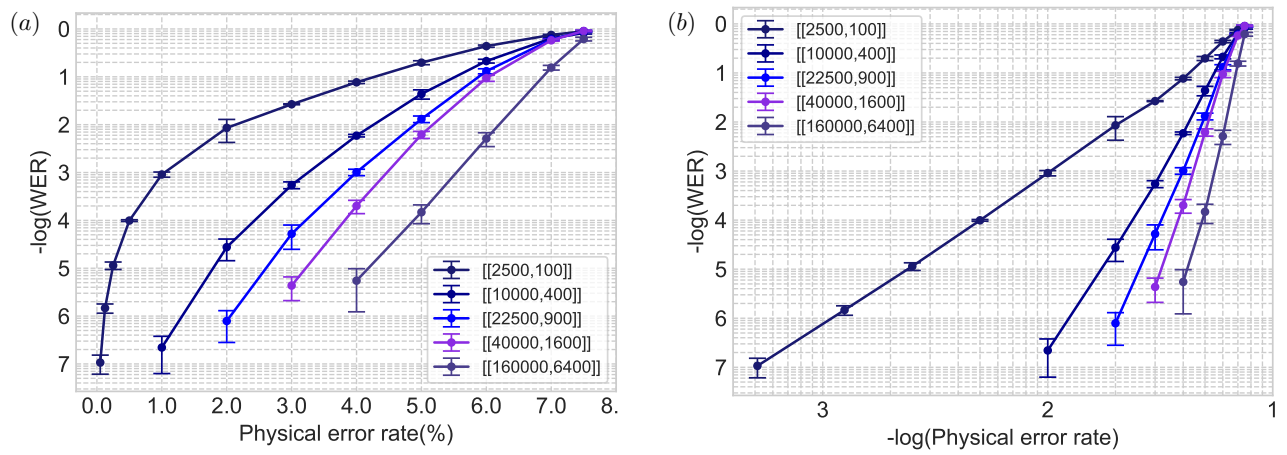

Figure 2: Variation of the word error rate (WER) with the physical error for a hypergraph product code formed as a product of $(3,4)$-regular graphs with the `Iterative BP + SSF` decoder. This code family displays an encoding rate of 0.04. The error bars denote 99% confidence intervals, *i.e.*, ≈ 2.6 standard deviations. (a) Base 10 logarithm of WER versus physical error rate. We see that the threshold is above 7%. (b) Log of WER versus log of physical error rate.

the hybrid decoder `Iterative BP + SSF` which is presented in Alg. 7 in Appendix E. To make a direct comparison with the $(5,6)$-regular codes presented above, we include the results of numerical simulations of the $(5,6)$-regular codes using the `Iterative BP + SSF` decoder in appendix F.

We now make some remarks concerning the time complexity of this algorithm. As described in Alg. 7 in Appendix E, each iteration computes one additional round of `BP`. To avoid redundancy in our computations, we save the output of the last step of each block of `BP` rounds. When we proceed to the next iteration, we only need to compute the new round. In practice most of the computation time is due to `SSF`. In our simulations, we choosed $T_{max} = 100$.

This hybrid decoder significantly improves upon the earlier results using only `SSF`. Fig. 2 shows the variation of the WER versus the physical error rate using the hybrid decoder `Iterative BP + SSF`. The threshold appears to be at roughly 7.5%: below this value of physical error, the WER reduces as we increase the block size. The log-log plot facilitates extrapolation to low noise rates.

Interestingly, the better performance of the hybrid `Iterative BP + SSF` decoder compared to the `SSF` decoder also comes with additional features such as an increased encoding rate (from 1.6% to 4%) and a reduction of the stabilizer weight. The `SSF` decoder used in [20] indeed required classical codes generated from bipartite biregular factor graphs of degrees $(5,6)$. The resulting quantum codes therefore had qubit degrees 10 and 12, and stabilizer weights 11, respectively. With the hybrid decoder, it suffices to use classical codes whose bipartite biregular factor graphs have degrees $(3,4)$. The resulting quantum codes have qubit degrees 6 and 8, and stabilizer weights 7, respectively.

This is surprising – Theorem 2 only guarantees performance of the `SSF` decoder if the graphs are sufficiently good expanders, which would require factor graphs with larger degrees that those we have considered. Our hybrid decoder seems to be able to get away with a much lower expansion, and therefore smaller degrees. This is important for physical implementations as higher degrees require more connectivity between different parts of the circuit.

Lastly, the word error rate is improved by several orders of magnitude, for a given block size. Compare the codes $[\![24400, 400]\!]$ generated from the $(5,6)$-regular family on Fig. 1 and the $[\![22500, 900]\!]$ code generated from the $(3,4)$ family on Fig. 2. The encoding rate is twice as large in the second case and the code performance is significantly better for a given noise rate. For instance at $p = 2\%$, the WER is 10^{-1} for the $[\![24400, 400]\!]$ code but only 10^{-3} for the $[\![22500, 900]\!]$ code.

Where do the $(3,4)$-regular codes with `Iterative BP + SSF` stand with respect to other codes? In Table 1, we compare their performance

Code	Asymptotic Rate	Stabilizers weight	Algorithm	Threshold
Toric code [39]	0%	4	MWPM	10.5%
4,5-hyperbolic code[5]	10%	4 and 5	MWPM	2.5%
4D-hyperbolic code [4]	18%	12	BP	$\approx 5\%$
(5,6) HGP code [20]	1.6%	11	SSF	$\approx 4.6\%$
(3,4) HGP code	4%	7	Iterative BP + SSF	$\approx 7.5\%$

Table 1: Comparing different LDPC codes and decoders when subject to independent $X - Z$ noise and assuming ideal syndrome measurements. In this model, $p_x = p_z$. The results of this work highlighted in blue.

with the toric code, the $(4, 5)$-hyperbolic code from [5], the 4D hyperbolic code from [4]. We find that the $(3, 4)$-regular codes have a competitive threshold of roughly 7.5% only behind the toric code. While the rate is not as good as that of [4], it has a higher threshold and lower stabilizer weights.

4 Dealing with syndrome noise

Although promising, the results of the previous section focus on an unrealistic problem since they assume perfect syndrome extraction. We now move on to the more relevant setting where the syndrome themselves are error prone. In addition to independent bit-flip and phase-flip noise each occurring at probability p, each of the syndrome bits is independently flipped with the same probability p. We choose the same probability for qubit and syndrome errors for simplicity. Let us immediately note that we will not be able to use the Iterative BP + SSF decoder here since it requires knowledge of whether decoding has succeeded or not (*i.e.*, whether the syndrome is null or not) in order to stop. In the case where the syndrome is noisy, there is in general no way to know whether all qubit errors have been corrected.

Analyzing the performance of decoding algorithms with a noisy syndrome is not as straightforward as in the noiseless syndrome case, and we will in particular need to adapt our metrics. We will follow the approach of Breuckmann and Terhal [5]. When the syndrome is itself prone to error, we do not expect the output of the decoding algorithm to be an error-free code state. We consider the following scenario corresponding to a quantum computation with T layers of logical gates for instance[1], and are interested in whether the final output is correct. For each of these T time steps, we consider both qubit noise (independent $X - Z$ noise with error rate p) and observe a noisy syndrome (corresponding to the ideal syndrome, with each bit further independently flipped with probability p). After each time step, we use some efficient decoder $\texttt{Dec}_1$ that returns some candidate error and apply the corresponding correction. After the T steps, we want to verify whether we are close to the correct codeword. To this end, we perform error correction with the assumption that the syndrome can be noiselessly extracted. This is because we are typically interested in a classical result and simply measure the qubits directly and compute the value of the syndrome directly (no need to measure ancilla qubits in the last round). We then perform a final decoding procedure with a potentially different decoder $\texttt{Dec}_2$. We can then estimate the threshold as a function of T, and its asymptotic value corresponds to the so-called *sustainable error rate* [7]. The idea is that if the physical error rate is below that threshold, then it means that one can perform arbitrarily long computations (or equivalently increase the lifetime of encoded information) by increasing the block length of the quantum codes. Formally the sustainable error rate is defined as the failure rate obtained as a limit of the threshold error rate as the number of rounds of error correction is increased. As mentioned above, the threshold itself is obtained as the block size of the code is increased to infinity such that the logical error rate is zero.

[1] We could alternatively consider the problem of faithfully storing a state in a quantum memory for T time steps for instance.

As alluded to above, `Iterative BP + SSF` is not a valid option for $\texttt{Dec}_1$ since we would not know when to stop the decoding in general. We have experimentally tried a number of heuristics for $\texttt{Dec}_1$ and the one that performed the best is the `First-min BP` decoder (described in Alg. 8 in Appendix E). This decoder simply implements `BP` and terminates when the syndrome size stops decreasing. To be precise, it performs a round of `BP`, computes the estimated error and sees whether correcting for this error leads to a decrease of the syndrome weight. If so, it continues with another round and otherwise, it returns the guess made at the previous round. We have numerically considered a number of variations for the stopping criterion but this one was consistently the best option. Another possibility that we investigated for $\texttt{Dec}_1$ is to add `SSF` after `First-min BP`. This leads however to worse performance (as can be observed on Fig. 3) and increases the decoder complexity.

Having described the algorithm, we now discuss what parameters are fed to `First-min BP`. Recall from Alg. 3 that `BP` is initialized with prior information on how likely it is for each qubit to have been flipped. This is done by specifying the Log-Likelihood Ratios (LLRs) $\{\lambda_i^0\}_{i=1}^n$ for every qubit. If the syndromes are perfect, this algorithm proceeds to update these LLRs over several iterations. To adapt `BP` to a fault-tolerant setting where it is employed T times, we need to

1. specify how to initialize the LLRs at each round, and

2. how to process potentially incorrect syndrome information.

In the fault-tolerant setting, qubits are subjected to i.i.d. noise only for the first round. From the second round onwards, the noise is complicated and no longer Markovian; in addition to the potential noise at a given round, the state of the qubits depends on the results of the corrections in all the previous rounds. We make a simplifying assumption: when called at round T, `BP` is only fed approximate LLRs. The LLRs are computed as if the qubits were subject to i.i.d. bit-flip and phase-flip errors with rate p, thereby ignoring all sources of noise from the previous rounds. For each qubit, these LLRs prescribe a bias $1-p$ to not being flipped, and p to being flipped.

Secondly, although `BP` was described for perfect syndromes, it can easily be adapted to take the syndrome error into account. We can simulate a Tanner graph with noisy checks by modifying the original Tanner graph. For each check node in the graph, we add a unique variable node which represents whether or not it is erroneous. We highlight some useful properties of this construction. These nodes do not create cycles, and therefore should not affect the performance of `BP`. They are treated like any other variable nodes, making this procedure easy to implement. Since they are linked to only one node, and therefore always send the same message. In particular, this adaptation of `BP` to the noisy syndrome case does not increase its time complexity.

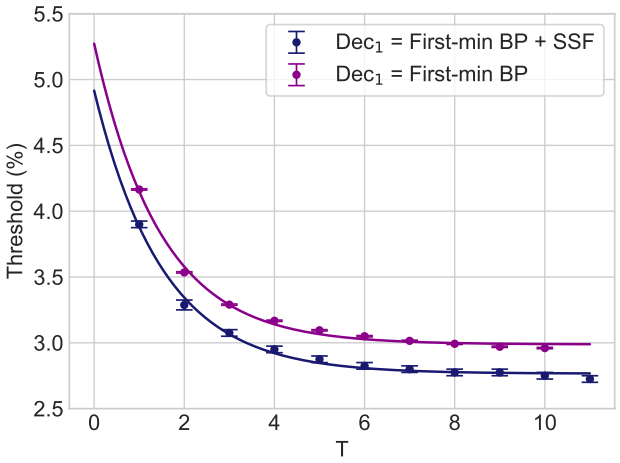

Figure 3: Evolution of the threshold as a function of T. Surprisingly, the simpler procedure `First-min BP` performs better than `First-min BP+SSF` for $\texttt{Dec}_1$ with a sustainable error rate which seems to be around 3% (compared to 2.5%).

For $\texttt{Dec}_2$ on the other hand, the `Iterative BP+SSF` decoder would be admissible in principle since we assume that the syndrome is perfect at the last step. For convenience, we chose to implement `First-min BP+SSF` instead, but either options are expected to yield the same threshold. A detailed review of the performance of `First-min BP+SSF` is presented in Appendix A, and specifically in Fig. 4.

The noisy-sampling algorithm which gives the prescription to estimate the sustainable error rate is described in Alg. 10 in Appendix G. For brevity, we denote by $\mu(p)$ the Bernoulli distribution with bias p, *i.e.*, drawing from this distribution $\Pr\{X=1\}=p$ and $\Pr\{X=0\}=1-p$.

The simulations in the noisy syndrome case are

Code	Asymptotic Rate	Stabilizers weight	Algorithm	Threshold	Single shot
Toric code [39]	0%	4	MWPM	2.9%	No
4,5-hyperbolic code[6]	10%	4 and 5	MWPM	1.3%	No
3,4 HGP code	4%	7	$\texttt{Dec}_1$: `First-min BP` $\texttt{Dec}_2$: `First-min BP` + `SSF`	$\approx 3\%$	Yes

Table 2: Independent X-Z noise with syndrome errors ($p_x = p_z = p_{check}$). The results of this work highlighted in blue.

far more time consuming than in the noiseless case since we need to plot the threshold as a function of T to estimate its asymptotic limit. To simplify the analysis, we again consider the independent $X - Z$ noise model: as before, we only need to simulate the case of X-errors only. This simplification does not affect the threshold.

Fig. 3 presents an estimate of the sustainable error rates for these algorithms. We obtain "asymptotic" values around 2.5% when $\texttt{Dec}_1 =$ `First-min BP` + `SSF` and slightly above 3% when $\texttt{Dec}_1 =$ `First-min BP`. The first algorithm seems to converge around 2.5% whereas the second one seems to converge above 3%. A plausible explanation for the worse performance for the *a priori* better algorithm `First-min BP` + `SSF` is that `BP` does not manage to correct all syndrome errors, which is an issue for the `SSF` decoder.

As shown in Table 2, we obtain good results in comparison with other main families of quantum LDPC codes. Indeed the threshold is roughly the same as the toric code (2.9% versus approx. 3%). While the stabilizer weights are lower, recall that the toric code has a zero asymptotic rate. When compared to a positive rate code such as the $(4, 5)$-hyperbolic codes, we cannot point to a clear winner: hyperbolic codes come with a better rate and lower stabilizer weights, but yield a lower threshold (1.3%).

We conclude by pointing out that the sustainable error rate does not tell the whole story. While not shown in Fig. 3, the WER is typically very high in the vicinity of the threshold. Similarly to the noiseless-syndrome case, very large codes are probably needed for the decoder to reach its full potential.

5 Conclusion

While previous studies showed that LDPC hypergraph product codes display good error suppression properties in the asymptotic regime, it is really the finite block length regime that matters for applications such as fault-tolerant quantum computation. We addressed this problem here by introducing new heuristic decoders for hypergraph product codes that combine soft information (`BP`) and hard decisions (`SSF`). Our main motivation was that `BP` typically fails to converge when applied to quantum LDPC codes and that `SSF` typically only performs well when the syndrome has low weight. Combining both decoders to let `BP` reduce the weight of the syndrome as much as possible, before turning to `SSF` to finish the decoding, leads to surprisingly good results. In the noiseless syndrome case, this combination yields an improvement from 4.6% to 7.5% for the threshold and much lower WER when compared to `SSF` alone, while at the same time relying on codes with higher encoding rate and lower stabilizer weights. In the noisy syndrome case, we studied a combination of `BP` and `SSF` where we use `BP` after each syndrome measurement to try to reduce the error and only rely on the hybrid decoder at the very last step of the procedure. The sustainable error rate that we observe in simulations is competitive with the toric code as well as hyperbolic codes.

LDPC codes are among the most versatile classical codes and come with efficient decoders with essentially optimal performance. For those codes, the hard decision decoder `flip` was successfully replaced by decoders such as `BP` that exploit soft information. It is tempting to believe that the same approach should also be true in the quantum case and that soft information decoders will convincingly replace decoders such as `SSF` in the

future. While our results are a first step in this direction, they also call for a better understanding on how to process soft information in the case of quantum LDPC codes.

Acknowledgements

We would like to thank Earl Campbell, Vivien Londe, David Poulin and Jean-Pierre Tillich for discussions. AG, LG and AL acknowledge support from the ANR through the QuantERA project QCDA. AK would like to thank the Fonds de Recherche du Québec Nature et Technologies (FRQNT) for the B2X scholarship.

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. ACM, 1997. DOI: 10.1137/S0097539799359385. URL https://doi.org/10.1137/S0097539799359385.

[2] Héctor Bombin, Ruben S Andrist, Masayuki Ohzeki, Helmut G Katzgraber, and Miguel A Martin-Delgado. Strong resilience of topological codes to depolarization. *Physical Review X*, 2(2):021004, 2012. DOI: 10.1103/PhysRevX.2.021004. URL https://link.aps.org/doi/10.1103/PhysRevX.2.021004.

[3] Sergey Bravyi, David Poulin, and Barbara Terhal. Tradeoffs for reliable quantum information storage in 2D systems. *Physical Review Letters*, 104(5):050503, 2010. DOI: 10.1103/PhysRevLett.104.050503. URL https://link.aps.org/doi/10.1103/PhysRevLett.104.050503.

[4] Nikolas P Breuckmann and Vivien Londe. Single-Shot Decoding of Linear Rate LDPC Quantum Codes with High Performance. *arXiv preprint arXiv:2001.03568*, 2020.

[5] Nikolas P Breuckmann and Barbara M Terhal. Constructions and noise threshold of hyperbolic surface codes. *IEEE transactions on Information Theory*, 62(6):3731–3744, 2016. DOI: 10.1109/TIT.2016.2555700.

[6] Nikolas P Breuckmann, Christophe Vuillot, Earl Campbell, Anirudh Krishna, and Barbara M Terhal. Hyperbolic and semi-hyperbolic surface codes for quantum storage. *Quantum Science and Technology*, 2(3):035007, 2017. DOI: 10.1088/2058-9565/aa7d3b.

[7] Benjamin J Brown, Naomi H Nickerson, and Dan E Browne. Fault-tolerant error correction with the gauge color code. *Nature communications*, 7(1):1–8, 2016. DOI: 10.1038/ncomms12302.

[8] A Robert Calderbank and Peter W Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098, 1996. DOI: 10.1103/PhysRevA.54.1098.

[9] Christopher T Chubb and Steven T Flammia. Statistical mechanical models for quantum codes with correlated noise. *arXiv preprint arXiv:1809.10704*, 2018.

[10] Jonathan Conrad, Christopher Chamberland, Nikolas P Breuckmann, and Barbara M Terhal. The small stellated dodecahedron code and friends. *Phil. Trans. R. Soc. A*, 376(2123):20170323, 2018. DOI: 10.1098/rsta.2017.0323.

[11] Nicolas Delfosse. Tradeoffs for reliable quantum information storage in surface codes and color codes. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 917–921. IEEE, 2013. DOI: 10.1109/ISIT.2013.6620360.

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

[13] Ilya Dumer, Alexey A Kovalev, and Leonid P Pryadko. Thresholds for correcting errors, erasures, and faulty syndrome measurements in degenerate quantum codes. *Physical review letters*, 115(5):050502, 2015. DOI: 10.1103/PhysRevLett.115.050502.

[14] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier. Constant overhead quantum fault-tolerance with quantum expander codes. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 743–754. IEEE, 2018. DOI: 10.1109/FOCS.2018.00076.

[15] 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*, pages 521–534. ACM, 2018. DOI: 10.1145/3188745.3188886.

[16] Marc PC Fossorier and Shu Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, 41(5):1379–1396, 1995. DOI: 10.1109/18.412683.

[17] Michael H Freedman, David A Meyer, and Feng Luo. Z2-systolic freedom and quantum codes. *Mathematics of quantum computation, Chapman & Hall/CRC*, pages 287–320, 2002.

[18] Daniel Gottesman. Stabilizer codes and quantum error correction. *arXiv preprint quant-ph/9705052*, 1997.

[19] Daniel Gottesman. Fault-tolerant quantum computation with constant overhead. *Quantum Information & Computation*, 14(15-16):1338–1372, 2014. DOI: 10.5555/2685179.2685184.

[20] Antoine Grospellier and Anirudh Krishna. Numerical study of hypergraph product codes. *arXiv preprint arXiv:1810.03681*, 2018.

[21] Larry Guth and Alexander Lubotzky. Quantum error correcting codes and 4-dimensional arithmetic hyperbolic manifolds. *Journal of Mathematical Physics*, 55 (8):082202, 2014. DOI: 10.1063/1.4891487.

[22] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997. DOI: 10.4213/rm892.

[23] Emanuel Knill, Raymond Laflamme, and Wojciech H Zurek. Resilient quantum computation: error models and thresholds. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 454, pages 365–384. The Royal Society, 1998. DOI: 10.1126/science.279.5349.342.

[24] Alexey A Kovalev and Leonid P Pryadko. Improved quantum hypergraph-product LDPC codes. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 348–352. IEEE, 2012. DOI: 10.1109/ISIT.2012.6284206.

[25] Alexey A Kovalev and Leonid P Pryadko. Fault tolerance of quantum low-density parity check codes with sublinear distance scaling. *Physical Review A*, 87(2):020304, 2013. DOI: 10.1103/PhysRevA.87.020304.

[26] Alexey A Kovalev, Sanjay Prabhakar, Ilya Dumer, and Leonid P Pryadko. Numerical and analytical bounds on threshold error rates for hypergraph-product codes. *Physical Review A*, 97(6):062320, 2018. DOI: 10.1103/PhysRevA.97.062320.

[27] Anirudh Krishna and David Poulin. Topological wormholes: Nonlocal defects on the toric code. *Phys. Rev. Research*, 2:023116, May 2020. DOI: 10.1103/PhysRevResearch.2.023116. URL `https://link.aps.org/doi/10.1103/PhysRevResearch.2.023116`.

[28] Anirudh Krishna and David Poulin. Fault-tolerant gates on hypergraph product codes. *Phys. Rev. X*, 11:011023, Feb 2021. DOI: 10.1103/PhysRevX.11.011023. URL `https://link.aps.org/doi/10.1103/PhysRevX.11.011023`.

[29] Anthony Leverrier, Jean-Pierre Tillich, and Gilles Zémor. Quantum expander codes. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 810–824. IEEE, 2015. DOI: 10.1109/FOCS.2015.55.

[30] Muyuan Li and Theodore J. Yoder. A numerical study of bravyi-bacon-shor and subsystem hypergraph product codes. pages 109–119, 2020. DOI: 10.1109/QCE49297.2020.00024.

[31] Vivien Londe and Anthony Leverrier. Golden codes: quantum LDPC codes built from regular tessellations of hyperbolic 4-manifolds. *Quantum Information & Computation*, 19(5-6):361–391, 2019. DOI: 10.26421/QIC19.5-6.

[32] Brendan D McKay and Xiaoji Wang. Asymptotic enumeration of 0–1 matrices with equal row sums and equal column sums. *Linear algebra and its applications*, 373:273–287, 2003. DOI: 10.1016/S0024-3795(03)00506-8.

[33] Pavel Panteleev and Gleb Kalachev. Degenerate quantum ldpc codes with good finite length performance. *arXiv preprint arXiv:1904.02703*, 2019.

[34] David Poulin and Yeojin Chung. On the iterative decoding of sparse quantum

codes. *Quantum Information & Computation*, 8(10):987–1000, 2008. DOI: 10.5555/2016985.2016993.

[35] Tom Richardson and Ruediger Urbanke. *Modern coding theory*. Cambridge university press, 2008.

[36] Michael Sipser and Daniel A Spielman. Expander codes. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 566–576. IEEE, 1994. DOI: 10.1109/18.556667.

[37] Andrew Steane. Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society A*, 452(1954):2551–2577, 1996. DOI: 10.1098/rspa.1996.0136.

[38] Jean-Pierre Tillich and Gilles Zémor. Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength. *IEEE Transactions on Information Theory*, 60(2):1193–1202, 2014. DOI: 10.1109/TIT.2013.2292061.

[39] Chenyang Wang, Jim Harrington, and John Preskill. Confinement-higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory. *Annals of Physics*, 303(1):31–58, 2003. DOI: 10.1016/S0003-4916(02)00019-2.

[40] Weilei Zeng and Leonid P Pryadko. Higher-dimensional quantum hypergraph-product codes with finite rates. *Physical Review Letters*, 122(23):230501, 2019. DOI: 10.1103/PhysRevLett.122.230501.

[41] Guanyu Zhu, Ali Lavasani, and Maissam Barkeshli. Instantaneous braids and dehn twists in topologically ordered states. *Phys. Rev. B*, 102:075105, Aug 2020. DOI: 10.1103/PhysRevB.102.075105.

6 Appendix

A `First-min BP+ SSF`

In the noisy syndrome setting, we cannot expect `SSF` to know whether it has succeeded and therefore cannot apply `Iterative BP + SSF` anymore. Rather, we want to find a heuristic criterion to stop `BP` after the right number of rounds, and then feed the result to the `SSF` decoder. The simplest possibility is to observe the evolution of the syndrome weight through the successive rounds of `BP`. This evolution is usually approximately periodic, displaying oscillations with the weight reaching a local minimum before increasing again.

We have empirically investigated several choices of stopping criterion, *e.g.*, first minimum of the syndrome weight, global minimum in the 100 first rounds, and found that the best option was the first one. Stopping `BP` when the weight reaches its first minimum gives rise to our heuristic decoder `First-min BP`. The `First-min BP + SSF` decoder then corresponds to the case where the output of `First-min BP` is given to `SSF`. These algorithms are described in Alg. 8 and Alg. 9 in Appendix E.

Fig. 4 shows the variation of the WER versus the physical error rate for independent bit-flip and phase-flip noise, and ideal syndrome measurements. It is interesting to note that while the WER is degraded compared to `Iterative BP + SSF`, the threshold behaviour is essentially identical for both decoding algorithms since they both yield a value around 7.5%. We had initially investigated thoroughly the `First-min BP + SSF` decoder but the error floor occurring when the physical error rate approaches 1% led us to switch to `Iterative BP + SSF`, for which this behaviour disappears. We found the good performance of `First-min BP + SSF` remarkable given the simplicity of the heuristic, and it would be worthwhile studying other variants to decide reliably when to switch from `BP` to `SSF`.

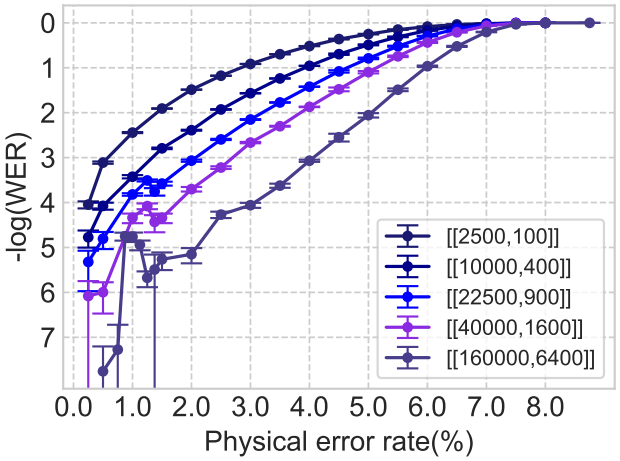

Figure 4: Word Error Rate (WER) as a function of the physical error for a hypergraph product code formed as a product of (3,4)-regular graphs with the `First-min BP + SSF` decoder. The threshold is similar to that of `Iterative BP + SSF`. The WER is worse, however, and odd patterns at low error rates.

B Code construction

We describe here how we construct the regular bipartite graphs we use in our simulations. Specifically we start with the *configuration model* and then apply some post-processing to increase the girth of the factor graphs. This algorithm generates graphs as follows:

1. We first create an 'empty' graph with the desired number of nodes. A vertex of degree Δ has Δ *ports*. In our case, check nodes will have Δ_C ports and variable nodes will have Δ_V ports.
2. We randomly draw edges between nodes by connecting a check port with a variable port. In the event we have two edges between a given pair of nodes, we randomly swap the double edges.

This algorithm could potentially generate bipartite graphs with small cycles. To avoid this, we use a post-processing algorithm that randomly swaps edges to increase the girth [32]. Although time consuming, this process yields good bipartite graphs.

C `flip` and `SSF`

Algorithm 1: `flip`

Input: Received word $y \in \mathbb{F}_2^n$
Output: $w \in \mathbb{F}_2^n$, the deduced word

$w := y$; // Update w iteratively
$\mathcal{F} = \emptyset$; // Flippable vertices
for $u \in V$ **do** // Setup phase: update bits
  If u has more UNSAT than SAT neighbors, add it to $\mathcal{F}$;
end
while $\exists u \in \mathcal{F}$ **do** // While flippable vertices exist
  flip w_u;
  Decide whether the elements of $\Gamma(u)$ are UNSAT;
  Decide whether the elements of $\Gamma(\Gamma(u))$ are in $\mathcal{F}$;
end
return w;

Algorithm 2: `SSF`

Input: A syndrome $\sigma_0 \in \mathbb{F}_2^{nm}$
Output: Deduced error $\widehat{E}$ if algorithm converges and FAIL otherwise

$\widehat{E} = 0^{n_2+m^2}$ // Iteratively maintain $\widehat{E}$
$\sigma_0 = \sigma_X(E)$ // Iteratively maintain syndrome
while $\exists F \in \mathcal{F} : |\sigma_i| - |\sigma_i \oplus \sigma_X(F)| > 0$ **do**

$$F_i = \arg\max_{F \in \mathcal{F}} \frac{|\sigma_i| - |\sigma_i \oplus \sigma_X(F)|}{|F|}$$
$$\widehat{E}_{i+1} = \widehat{E}_i \oplus F$$
$$\sigma_{i+1} = \sigma_i \oplus \sigma_X(F_i)$$
$$i = i + 1$$

end
return $\widehat{E}_i$ if $\sigma_X(\widehat{E}_i) + \sigma_0$ is zero and FAIL otherwise.

D BP and subroutines

Algorithm 3: `BP`

Input: Time steps T
Syndromes $s \in \mathbb{F}_2^m$
Error probability p
Output: Deduced error $\hat{v} \in \mathbb{F}_2^n$

Initialization: At time $t = 0$: $\forall v_i, \forall c_j \in \Gamma(v_i), m^0_{v_i \to c_j} = \lambda_i^0 = \ln\left(\frac{1-p}{p}\right)$.

for $1 \leq t \leq T$: **do**
  `sum-product-single-step`(t)
end
Terminate: `// Log-likelihood computation`
for $i \in [n]$ **do**
  $\lambda_i^t = \lambda_i^0 + \sum_{c_j \in \Gamma(v_i)} m^t_{c_j \to v_i}$
  if $\lambda_i > 0$ **then**
    $\hat{v}_i = 0$
  else
    $\hat{v}_i = 1$
  end
end
return $\hat{v}$

Algorithm 4: `sum-product-single-step`(t)

for $c_j \in C, v_i \in \Gamma(c_j)$ **do** `// Checks to bits`
  `Check-to-bit`(c_j, v_i)
end
for $v_i \in V, c_j \in \Gamma(v_i)$ **do** `// Bits to checks`
  `Bit-to-check`(v_i, c_j)
end

Algorithm 5: `Bit-to-Check`

Input: Variable node v_i, check node $c_j \in \Gamma(v_i)$
Output: $m^{t+1}_{v_i \to c_j}$

return $m^{t+1}_{v_i \to c_j} := \ln\left(\frac{1-p}{p}\right) + \sum_{c_{j'} \in \Gamma(v_i) \setminus c_j} m^t_{c_{j'} \to v_i}$

Algorithm 6: `Check-to-bit`

Input: Check node c_j, variable node $v_i \in \Gamma(c_j)$
Output: $m^{t+1}_{c_j \to v_i}$

return $m^{t+1}_{c_j \to v_i} := (-1)^{s_j} 2 \tanh^{-1}\left(\prod_{v_{i'} \in \Gamma(c_j) \setminus v_i} \tanh\left(\frac{m^t_{v_{i'} \to c_j}}{2}\right)\right)$

E Heuristics

Algorithm 7: `Iterative BP + SSF`

Input: Syndrome σ_0
maximal no. of `BP` iterations $T_{\max}$
Output: Deduced error $\widehat{E}$ if algorithm converges and FAIL otherwise.

$T = 0$
while $T \leq T_{\max}$ **do**

$\widehat{E} = \texttt{BP}(T, \sigma_0, p)$
$\sigma' = \sigma_0 + \sigma_X(\widehat{E})$
$\widehat{E} = \widehat{E} + \texttt{SSF}(\sigma')$
$T = T + 1$
return $\widehat{E}$ if $\sigma_X(\widehat{E}) + \sigma_0$ is zero and keep running otherwise.

end
return FAIL

Algorithm 8: `First-min BP`

Input: Syndrome $\sigma_0 = \sigma_X(E)$
Output: Error E' that minimizes the syndrome

$T = 0$
$\sigma_{\texttt{current}} = \sigma_0$
$E_{\texttt{BPcurrent}} = 0^n$
do

$T = T + 1$
$E_{\texttt{BPprev}} = E_{\texttt{BPcurrent}}$
$\sigma_{\texttt{prev}} = \sigma_{\texttt{current}}$
$E_{\texttt{BPcurrent}} = \texttt{BP}(T, \sigma_0, p)$
$\sigma_{\texttt{current}} = \sigma_0 + \sigma_X(E_{\texttt{BPcurrent}})$

while $|\sigma_{current}| < |\sigma_{prev}|$ `// We repeat these steps while the syndrome decreases`
$E_{\texttt{BP}} = E_{\texttt{BPprev}}$ `// We choose the previous correction.`
return $E_{\texttt{BP}}$

Algorithm 9: `First-min BP + SSF`

Input: Syndrome $\sigma_0 = \sigma_X(E)$
Output: Deduced error $\widehat{E}$

$E_{\texttt{BP}} = \texttt{First-min BP}(\sigma_0)$
$\sigma' = \sigma_0 + \sigma_X(E_{\texttt{BP}})$
$E_{\texttt{SSF}} = \texttt{SSF}(\sigma')$
$\widehat{E} = E_{\texttt{BP}} + E_{\texttt{SSF}}$
return $\widehat{E}$

F Simulating decoding $(5, 6)$-regular codes using `Iterative BP + SSF`

In this section, we present the result of simulating the hypergraph product codes obtained as a product of $(5, 6)$-regular codes using the `Iterative BP + SSF` decoder. As can be seen from the plot below, the threshold is improved, and is closer to 7% than the previously observed 5% as shown in fig. 1. On the other hand, these codes do not have as high a threshold as the $(3, 4)$-regular codes which we presented above.

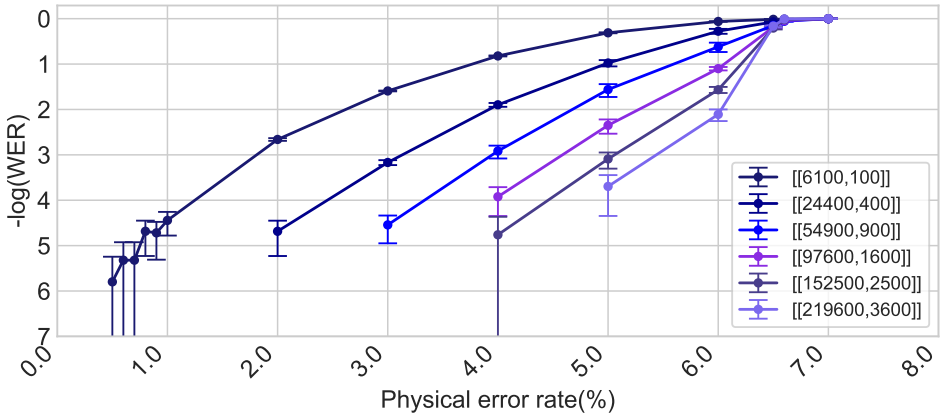

Figure 5: Word Error Rate (WER) as a function of the physical error rate for a hypergraph product code formed as a product of $(5, 6)$-regular graphs with the `Iterative BP + SSF` decoder. The threshold is markedly better than that obtained from `SSF` alone.

G Noisy sampling

Algorithm 10: `Noisy-sampling`

Input: Bias probability p
Number of faulty rounds T
Output: SUCCESS if no logical errors, and FAIL otherwise

$E = 0^n$ `// The code is initialized with no errors`
for $i \in \{1, ..., T\}$ **do**
  Sample error E_i from $\mu(p)^n$
  $E = E + E_i$
  Compute syndrome $\sigma_i = \sigma_X(E)$
  Sample syndrome noise $\widetilde{\sigma}_i$ from $\mu(p)^m$
  Let $\xi_i := \widetilde{\sigma}_i + \sigma_i$
  Compute $\widehat{E}_i = \mathtt{Dec}_1(\xi_i)$
  $E = E + \widehat{E}_i$
end
Sample error E_{T+1} from $\mu(p)^n$
$E = E + E_{T+1}$
Compute syndrome $\sigma_{T+1} = \sigma_X(E)$
Compute $\widehat{E}_{T+1} = \mathtt{Dec}_2(\sigma_{X,T+1})$
$E = \widehat{E}_{T+1} + E$
return SUCCESS if E is in the stabilizer group and FAIL otherwise