Degenerate Quantum LDPC Codes With Good Finite Length Performance

Pavel Panteleev and Gleb Kalachev

Faculty of Mechanics and Mathematics, Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation

We study the performance of medium-length quantum LDPC (QLDPC) codes in the depolarizing channel. Only degenerate codes with the maximal stabilizer weight much smaller than their minimum distance are considered. It is shown that with the help of OSD-like post-processing the performance of the standard belief propagation (BP) decoder on many QLDPC codes can be improved by several orders of magnitude. Using this new BP-OSD decoder we study the performance of several known classes of degenerate QLDPC codes including hypergraph product codes, hyperbicycle codes, homological product codes, and Haah's cubic codes. We also construct several interesting examples of short generalized bicycle codes. Some of them have an additional property that their syndromes are protected by small BCH codes, which may be useful for the fault-tolerant syndrome measurement. We also propose a new large family of QLDPC codes that contains the class of hypergraph product codes, where one of the used parity-check matrices is square. It is shown that in some cases such codes have better performance than hypergraph product codes. Finally, we demonstrate that the performance of the proposed BP-OSD decoder for some of the constructed codes is better than for a relatively large surface code decoded by a near-optimal decoder.

1 Introduction

Quantum error-correcting codes are considered as an essential component in the current architectures of quantum computers due to the inherently faulty nature of the quantum hardware. Topological quantum codes [1, 2, 3] are among the quantum codes with the highest known noise thresholds. These codes have sparse parity-check matrices and thus belong to the class of quantum LDPC (QLDPC) codes. Moreover, they are highly degenerate, which means that the minimum distance is much higher than the weight of the stabilizers. It is important that these codes also have decoding algorithms with near to optimal performance [1, 4, 5]. Though the thresholds of topological codes are relatively high, their dimensions are usually much smaller than for general QLDPC codes of the same length (it is constant for the surface and color codes). There are also interesting classes of topological quantum codes with very large dimensions (e.g., hyperbolic codes [6] have a constant rate). However, such codes usually have much smaller minimum distances compared to the surface and color codes.

Recently there have been proposed a number of interesting families of degenerate QLDPC codes (e.g., hypergraph product codes [7] and homological product codes [8]) with very good asymptotic parameters. Nevertheless their practical error-correcting performance for relatively small code lengths ($n < 1000$) is largely unexplored, and it is not clear whether their performance is competitive to the best known topological codes. From our point of view, the difficulty of constructing degenerate QLDPC codes with good practical performance is mostly related to the following two issues.

1. Asymptotically good constructions may not necessarily produce the best QLDPC codes for relatively small code lengths. Indeed, in [9, 10] the construction of hypergraph product codes was further improved and generalized. Although the asymptotic characteristics of the improved codes are the same as before, their parameters such as the rate and the minimum distance are much better for smaller lengths.

2. The performance of the known QLDPC codes is far from optimal under the state-of-the-art decoders (including the binary and non-binary BP decoders, and their modifications). The performance degradation is usually attributed [11] to the unavoidable 4-cycles in the corresponding Tanner graphs and to a large number of degenerate errors. While the number of 4-cycles can be significantly reduced by using CSS codes [12] without 4-cycles in the parity-check matrices H_X and H_Z, the number of degenerate errors can not be easily reduced for highly-degenerate codes.

In this paper, we try to address both mentioned issues. In the first part of the paper, we introduce a new enhancement of the standard BP decoder for QLDPC codes (both binary and non-binary versions are allowed) using a variant of the well-known decoding algorithm for short classical codes called the *ordered statistics decoding (OSD)* [13, 14]. This new post-processing algorithm works only if the BP decoder fails to find a recovery Pauli operator that gives the correct syndrome. We suppose here that the QLDPC parity-check matrix H is represented in the binary form[1]. The algorithm starts from finding a reliable information set [15] for H based on the soft decisions obtained by the BP decoder. Then it makes hard decisions for the bits from this information set and flips the w most unreliable of them in order to find the 2^w corresponding recovery Pauli operators that return the corrupted quantum state to the coding space. Finally, it selects a recovery operator with the minimum weight and applies it to the corrupted quantum state. Thus this new combined BP-OSD decoder, in contrast to the standard BP, *always* returns the recovery operator that moves the corrupted codeword back to the code space. We show that with the help of this OSD-like post-processing the performance of the standard BP decoder on many degenerate QLDPC codes can be improved by several orders of magnitude. As we can see from its brief description above, it is not restricted to CSS codes and can also be used in conjunction with any decoder, different from BP, that provides soft decisions.

In the second part of the paper, we construct a number of new relatively small codes using two families of degenerate codes: the *generalized bicycle* (*GB*) codes introduced in [10] and a new large family[2] of QLDPC codes introduced in this paper. We call this new family *generalized hypergraph product* (*GHP*) codes. It contains the GB codes and the class of hypergraph product (HP) codes [7], where one of the two parity-check matrices used in the product is square. We also derive new formulas for the dimension of GB and GHP codes and show how to design codes of high dimension with good error correction performance. It is interesting to note that some of the constructed GB codes have an additional property that their syndromes are protected by small BCH codes, which may be useful for the fault-tolerant syndrome measurement.

We also study the performance of the proposed BP-OSD decoder on many known classes of degenerate QLDPC codes, including the already mentioned hypergraph product codes, hyperbicycle codes [9, 10], the homological product codes [8], and Haah's cubic codes [20]. We compare their performance with the performance of the codes, constructed in this work. We show that in many cases the new codes with similar performance have better parameters such as the code length and the rate. Besides that, we compare the new BP-OSD decoder with other known modifications of the BP decoder such as the random perturbation [21], the enhanced feedback [22], and the matrix augmentation [23] algorithms. We compare all the above mentioned algorithms on the new $[[1270, 28]]$ code from the class of GHP codes mentioned earlier[3]. We show that the performance of the new decoding algorithm on this code is significantly better. Moreover, we also show that the performance of this code under the BP-OSD is even better than the performance of the $[[1201, 1, 25]]$ surface code under the near-optimal MPS-based decoder proposed in [5]. We also demonstrate that the performance under BP-OSD of one of Haah's cubic codes, which have local stabilizers in 3D, is also very good.

The remainder of the paper is organized as follows. Section 2 contains some background material, where we review standard definitions and fix notations. In Section 3, we describe the BP-OSD algorithm and compare its performance with other known modifications of BP. In Section 4, we study GB codes and construct some new codes with good performance. In Section 5, we introduce and study a new large family of GHP codes and show that it generalizes the class of hypergraph product codes in the case when one of the parity-check matrices is square. In the last section, we give some final remarks. The paper also contains three appendixes. Appendix A has some supplementary material on the ring of circulants. Appendix B contains a description of all the codes used in the simulations. Finally, we show some additional simulations in Appendix C.

2 Basic facts and definitions

In this section, we fix notations and briefly recall some standard definitions related to classical and quantum LDPC codes. See [11] for a good review of these topics.

2.1 Classical codes

Let n be a natural number. In what follows, we denote by $[n]$ the set $\{1, \dots, n\}$. Consider a finite field $\mathbb{F}_q$

[1] For a QLDPC code on n qubits with m stabilizers the matrix H is an $m \times 2n$ binary matrix.

[2] Since the first version of the current work was released, the codes from this family have been shown to have very large minimal distances [16, 17, 18] and found interesting applications in geometry [19]. In [17] they were further generalized and called *lifted product* codes.

[3] Since this GHP code is quasi-cyclic, it can also be obtained (up to some qubit permutations) as a special case of the hyperbicycle code construction.

and an n-dimensional vector space $\mathbb{F}_q^n$ over $\mathbb{F}_q$. A *linear* $[n,k]_q$ *code* is a k-dimensional subspace $\mathcal{C} \subseteq \mathbb{F}_q^n$, where the parameters n and k are called the *length* and the *dimension* of $\mathcal{C}$, respectively. We denote the dimension k of the code $\mathcal{C}$ by $\dim \mathcal{C}$. The *rate* of the code $\mathcal{C}$ is equal to k/n. The elements of $\mathcal{C}$ are called *codewords*. The *Hamming distance* $d(v,v')$ between vectors $v, v' \in \mathbb{F}_q^n$ is the number of positions in which they differ. The parameter

$$d(\mathcal{C}) = \min\{d(c,c') \mid c \neq c';\ c,c' \in \mathcal{C}\}$$

is called the *minimal distance* of $\mathcal{C}$. By definition, we put $d(\mathcal{C}) = \infty$ when $k = 0$. It is easy to see that $d(\mathcal{C})$ is equal to the minimal weight $|c|$ of non-zero codewords, where the *weight* $|c|$ is the number of non-zero components in c. When $d(\mathcal{C}) = d$ for a linear $[n,k]_q$ code $\mathcal{C}$, we say that $\mathcal{C}$ is an $[n,k,d]_q$ code. A linear $[n,k,d]_q$ code is usually defined either as the row space of a matrix G called the *generator matrix* or as the kernel of a matrix H called the *parity-check matrix*. It is easy to see that $GH^{\mathrm{T}} = \mathbf{0}$, $\operatorname{rk} G = k$, and $\operatorname{rk} H = n-k$. In what follows we mostly consider *binary* linear codes (i.e., $q = 2$), in which case we use a shorter notation: $[n,k]$ or $[n,k,d]$.

2.2 Quantum stabilizer codes

The quantum analogs of classical linear codes are quantum stabilizer codes introduced in [24]. To define them we need a number of supporting definitions. Consider an *n-qubit Hilbert space* $\mathbb{C}^{2^n} = (\mathbb{C}^2)^{\otimes n}$. A *Pauli operator on n qubits* is an operator $P = \alpha P_1 \otimes \cdots \otimes P_n$ on the space $\mathbb{C}^{2^n}$, where $\alpha \in \{\pm 1, \pm i\}$, $P_i \in \{I, X, Y, Z\}$. Here I is the identity and X, Y, Z are the Pauli 2×2 matrices. The *weight* $\operatorname{wt}(P)$ of a Pauli operator P is the number of non-identity components in the tensor product. The set $\mathcal{P}_n$ of all Pauli operators P on n qubits is a non-commutative group under the operator multiplication called the *n-qubit Pauli group*. While the group $\mathcal{P}_n$ is non-commutative, if we forget about the global phase factors α of Pauli operators we obtain the quotient group $\mathcal{P}_n/\{\pm\mathbf{I}, \pm i\mathbf{I}\}$, where $\mathbf{I}$ is the identity operator on $\mathbb{C}^{2^n}$. This quotient group is isomorphic to the commutative group $\mathbb{Z}_2^{2n}$, and the isomorphism is given by the following map:

$$\alpha \bigotimes_{i=1}^{n} X^{x_i} Z^{z_i} \mapsto (x_1, \ldots, x_n \mid z_1, \ldots, z_n). \tag{1}$$

It is well known that two n-qubit Pauli operators P and P' commute **iff** for their binary representations $(x|z), (x'|z') \in \mathbb{F}_2^{2n}$ we have

$$\langle x, z' \rangle + \langle z, x' \rangle = 0, \tag{2}$$

where $\langle a, b \rangle = \sum_i a_i b_i$ is the dot product in $\mathbb{F}_2^n$.

Denote by $\mathcal{P}_n^*$ the subset of Pauli operators $P \in \mathcal{P}_n$ with the global phase factor $\alpha = 1$. A *stabilizer group* $\mathcal{S}$ is a commutative subgroup of the Pauli group $\mathcal{P}_n$ such that $-\mathbf{I} \notin \mathcal{S}$. The group $\mathcal{S}$ is usually generated by m Pauli operators $S_1, \ldots, S_m \in \mathcal{P}_n^*$ called *stabilizers generators*, i.e., $\mathcal{S} = \langle S_1, \ldots, S_m \rangle$. We say that $S_1, \ldots, S_m$ are *independent* if none of them can be obtained (up to a global phase factor α) from the others by the group multiplication.

Let us recall that a *quantum stabilizer* $[[n,k,d]]$ *code* is a 2^k-dimensional subspace of the n-qubit space $\mathbb{C}^{2^n}$ defined as the common $+1$-eigenspace for a set of m stabilizers $S_1, \ldots, S_m \in \mathcal{P}_n^*$:

$$\mathcal{C} = \{|\psi\rangle \in \mathbb{C}^{2^n} \mid S_i|\psi\rangle = |\psi\rangle, i = 1, \ldots, m\},$$

where the group $\mathbb{S}(\mathcal{C}) = \langle S_1, \ldots, S_m \rangle$ is called the *stabilizer group of* $\mathcal{C}$. It can be shown that $m \geqslant n-k$. But if the stabilizers are independent, we get $m = n-k$. Here the parameter $d = d(\mathcal{C})$ is called the *minimal distance*[4] of $\mathcal{C}$ and is equal to the minimal possible weight of a Pauli operator $P \in \mathcal{P}_n^*$ that commutes with all the stabilizers $S_1, \ldots, S_m$ but $P \notin \mathbb{S}(\mathcal{C})$.

A Pauli operator $P \in \mathcal{P}_n^*$ is usually interpreted as an error operator (called a *Pauli error*) that can corrupt a quantum system and cause it to go from a state $|\psi\rangle$ to $P|\psi\rangle$. However, for every quantum stabilizer code $\mathcal{C}$, it is not hard to show that $P|\psi\rangle = |\psi\rangle$ for all $|\psi\rangle \in \mathcal{C}$ **iff** $P \in \mathbb{S}(\mathcal{C})$. Hence we see that in this case, not all Pauli errors $P \in \mathcal{P}_n^*$ can harm the state $|\psi\rangle$. We call a Pauli error $P \in \mathcal{P}_n^*$ *degenerate* for a code $\mathcal{C}$ if $P|\psi\rangle = |\psi\rangle$ for all $|\psi\rangle \in \mathcal{C}$ and *non-degenerate* otherwise. We see that the elements from $P \in \mathbb{S}(\mathcal{C})$ are precisely the degenerate Pauli errors and the minimum distance $d(\mathcal{C})$ is the minimal possible weight of a non-degenerate Pauli error. A stabilizer code $\mathcal{C}$ is called *degenerate* if it has degenerate Pauli errors P of weight $\operatorname{wt}(P) < d(\mathcal{C})$.

If we apply the binary mapping (1) to the stabilizer generators $S_1, \ldots, S_m$ of an $[[n,k,d]]$ code $\mathcal{C}$, we obtain the $m \times 2n$ binary matrix

$$H = (H_X \mid H_Z) \tag{3}$$

called the *parity-check matrix* of $\mathcal{C}$, where each row corresponds to a stabilizer generator. We do not require for the matrix H to be full rank (i.e, $\operatorname{rk} H = n-k \leqslant m$). Using (2) it is not hard to see that the null space of H is exactly the set of all vectors $(z,x) \in \mathbb{F}_2^{2n}$ such that (x,z) is the binary mapping of a Pauli error $P \in \mathcal{P}_n^*$ that commutes with all the stabilizer generators $S_1, \ldots, S_m$.

We see that the matrix $H = (H_X \mid H_Z)$ is not an arbitrary binary $m \times 2n$ matrix since the stabilizer generators $S_1, \ldots, S_m$ corresponding to its rows should com-

[4] We write $[[n,k]]$ if the minimal distance is not known.

mute with each other. From equation (2) it easily follows that this restriction can be formulated as the following *commutativity condition*:

$$H_X H_Z^{\mathrm{T}} + H_Z H_X^{\mathrm{T}} = \mathbf{0}. \tag{4}$$

A very important subclass of quantum stabilizer codes is *Calderbank-Shor-Steane (CSS) codes* introduced in [25, 26]. A quantum CSS code is a stabilizer code, where non-identity components in the tensor product of each stabilizer generator are either all equal to X or all equal to Z. Hence the parity-check matrix H of a CSS code of length n can be represented in the following form:

$$H = \left(\begin{array}{c|c} H_X & \mathbf{0} \\ \mathbf{0} & H_Z \end{array}\right),$$

where H_X, H_Z are binary matrices with n columns.

In this special case, commutativity condition (4) can be rewritten as:

$$H_X H_Z^{\mathrm{T}} = \mathbf{0}. \tag{5}$$

We can easily verify that the dimension k of the corresponding CSS code of length n is given by the formula:

$$k = n - \operatorname{rk} H_X - \operatorname{rk} H_Z. \tag{6}$$

2.3 Classical and quantum LDPC codes

A classical *low density parity check (LDPC) code* [27] is a linear code defined by a sparse binary parity-check matrix $H = (h_{ij})_{m\times n}$. The sparseness usually means that the weights of all rows and columns in H are upper bounded by some universal constant as the code length n grows in an infinite family of codes.

When we consider an LDPC code defined by a parity-check matrix H, it is helpful to define the bipartite graph $\mathcal{T} = \mathcal{T}(H)$ called the *Tanner graph* [28]. In this graph the first part of nodes $v_1, \ldots, v_n$ (called the *v-nodes*) corresponds to the columns of H (the *variables*), the second part of nodes $c_1, \ldots, c_m$ (called the *c-nodes*) corresponds to the rows of H (the *checks*), and we connect a v-node v_i with a c-node c_j whenever $h_{ij} = 1$, $i \in [m]$, $j \in [n]$. If the parity-check matrix H is (w_c, w_r)-*regular* (i.e., each column has weight w_c and each row has weight w_r), then the corresponding Tanner graph is also (w_c, w_r)-*regular* (i.e., each v-node has degree w_c and each c-node has degree w_r). We say that an LDPC code is w-*limited* if the degree of each node in its Tanner graph is upper bounded by w. It is obvious that any LDPC code with (w_c, w_r)-regular parity-check matrix is $\max(w_c, w_r)$-limited.

There are a number of decoding algorithms for classical LDPC codes, but the most frequently used one is the *belief propagation (BP) decoder* [27], also known as the *message passing decoder* or the *sum-product decoder* [29]. It assigns the *a priori* probability distributions of individual bits in the codeword (obtained from the channel) to the v-nodes of the Tanner graph and iteratively updates the *posterior* probability distributions for each bit. Once some maximal *iteration number limit* is reached, the decoder combines the a priori and the calculated posterior probability distributions (the *soft decisions*) to produce the optimal binary decision (called the *hard decision*) for each individual bit.

An important property of a parity-check matrix H defining an LDPC code is the *girth* of the corresponding Tanner graph $\mathcal{T} = \mathcal{T}(H)$, which is equal to the length of the shortest cycle in $\mathcal{T}$. It is well known that short cycles in the Tanner graph degrade the performance of the BP decoder. At the same time, it was observed that LDPC codes without 4-cycles (i.e., when the girth is at least 6) perform very well in practice. However, this practical observation is not fully investigated from theoretical point of view.

A *quantum LDPC (QLDPC)* is a stabilizer $[[n, k, d]]$ code with a sparse parity-check matrix H. We can also introduce the Tanner graph $\mathcal{T} = \mathcal{T}(\mathcal{S})$ for any stabilizer $[[n, k, d]]$ code $\mathcal{C}$ defined by a set of stabilizer generators $\mathcal{S} = \{S_1, \ldots, S_m\}$. In the case of stabilizer codes, the v-nodes correspond to n qubits and the c-nodes to the stabilizer generators $S_1, \ldots, S_m$, and we connect a c-node with a v-node if the corresponding stabilizer acts nontrivially on the corresponding qubit. As in the case of classical LDPC codes, we say that a QLDPC code is w-*limited* if the degree of each node in its Tanner graph is upper bounded by w. This property is much more important in the quantum case due to the faulty nature of the current quantum hardware. It is clear that any CSS code with (w_c, w_r)-regular matrices H_X and H_Z is $\max(2w_c, w_r)$-limited.

3 OSD-like post-processing for BP

In this section, we describe a new OSD-like post-processing algorithm that can be used after the BP decoder for QLDPC codes. Before we give its detailed description we consider two simple modifications of the OSD decoder for *classical* linear codes. These modifications will be used as the main components in the OSD-like post-processing algorithm for quantum codes. We should warn the reader that these modifications of the standard OSD decoder are not intended to improve its performance for classic LDPC codes. We introduce them because these algorithms eventually will be used in the OSD post-processing algorithm (called qOSD) for *quantum* codes described in Section 3.2. For example, one of the main differences between classical and quantum codes is that we have to use the syndrome decoder

in the quantum case. Hence we consider only syndrome OSD decoders here since we are going to use them as components of the decoder for quantum codes.

3.1 Syndrome OSD post-processing algorithm

Starting from this section we will often use the following notations:

- If M is a matrix, then M_i denotes its i-th column.
- If $I = \{i_1, \dots, i_k\}$ is some index set, $i_1 < \dots < i_k$, and $\pi \in \mathbf{S}_n$ is a permutation, then for every vector $v = (v_1, \dots, v_n)$ and matrix $M = (M_1, \dots, M_n)$ we define:

$$\begin{aligned} v_I &= (v_{i_1}, \dots, v_{i_k}), \\ M_I &= (M_{i_1}, \dots, M_{i_k}), \\ \pi(v) &= (v_{\pi(1)}, \dots, v_{\pi(n)}), \\ \pi(M) &= (M_{\pi(1)}, \dots, M_{\pi(n)}). \end{aligned}$$

- If $I \subseteq [n]$ is an index set, then the index set $\bar{I} = [n] \setminus I$ is called its *complement*.

Now let us recall the definition of an information set [30] of a code, which has an important role in the OSD decoding. A set of indices $I \subseteq [n]$ is called an *information set* of a classical linear $[n, k]$ code $\mathcal{C}$ if $\mathcal{C}_I = \{c_I \mid c \in \mathcal{C}\} = \mathbb{F}_2^k$. Clearly, I is an information set of $\mathcal{C}$ **iff** $|I| = k$, and for any two codewords $c, c' \in \mathcal{C}$ such that $c_I = c'_I$ we always have $c = c'$. Therefore for any information set I the corresponding k bits can be used to *uniquely* recover any vector $v \in \mathbb{F}_2^n$ if we also know its syndrome $s = Hv$. Hence we can consider the corresponding *encoding map* $\mathcal{E}_I^s \colon \mathbb{F}_2^k \to \mathbb{F}_2^n$ such that for any u, v, and s we have:

$$v_I = u,\ Hv = s \quad \Longleftrightarrow \quad v = \mathcal{E}_I^s(u).$$

It is easy to verify that $\mathcal{E}_I^{\mathbf{0}}$ is the systematic encoding map for the code $\mathcal{C}$ where the information bits u are at the positions corresponding to I.

Remark. If G is a generator matrix, and H is a parity-check matrix of a linear code $\mathcal{C}$; then it can be shown that a k-element index set I is an information set of $\mathcal{C}$ **iff** $\operatorname{rk} G_I = \operatorname{rk} G$ **iff** $\operatorname{rk} H_J = \operatorname{rk} H$, where $J = \bar{I}$. Thus if $\mathcal{B}(M)$ denotes the family of indices I such that the collection of columns $\{M_i\}_{i \in I}$ is a basis for the column space of M; then the family of all information sets of $\mathcal{C}$ coincides with $\mathcal{B}(G)$, while the family of their complements coincides with $\mathcal{B}(H)$. Since the set of all linearly independent columns of any matrix gives us a matroid[5], then for any positive real numbers $w_1, \dots, w_k$ we can efficiently find

$$\arg\max_{I \in \mathcal{B}(G)} \sum_{i \in I} w_i = \arg\min_{J \in \mathcal{B}(H)} \sum_{i \in J} w_i \tag{7}$$

in a greedy fashion [31, 32]. Here in the right part we used that $I \in \mathcal{B}(G)$ **iff** $J = \bar{I} \in \mathcal{B}(H)$.

The main idea of the syndrome OSD decoder is as follows. Consider a linear $[n, k]$ code $\mathcal{C}$ defined by a parity-check matrix H. Let $c' = c + e$ be a corrupted version of a codeword $c \in \mathcal{C}$, where $e \in \mathbb{F}_2^n$ is the corresponding random *error vector*. Given the syndrome $s = Hc' = He$ we want to find the error vector e and thus recover the codeword $c = c' - e$. Suppose that in addition we are given an estimate[6] of the error probability $p_i = \mathbf{P}(e_i = 1)$ for each $i \in [n]$. From these estimates our best guess of the error vector e would be the *hard decisions* vector $\hat{e}$, where $\hat{e}_i = 1$ when $p_i > 1/2$, and $\hat{e}_i = 0$ otherwise[7]. However, when we only use the error probability estimates, we often have that $H\hat{e} \neq s$, and thus $\hat{e} \neq e$. Nevertheless, even in such cases, some components of $\hat{e}$ are equal to e, and we can still try to use $\hat{e}$ to find e if we also take into account that

$$He = s. \tag{8}$$

Therefore, to recover e, one may try to traverse through different information sets I in the hope that for one of them we have $\hat{e}_I = e_I$, and thus

$$e = \mathcal{E}_I^s(\hat{e}_I). \tag{9}$$

Unfortunately we do not know for which indices $i \in [n]$ we have $\hat{e}_i = e_i$. Hence it makes sense to find an information set I with the indices that are as reliable as possible, where the *reliability* of an index $i \in [n]$ is the probability

$$\rho_i = \mathbf{P}(\hat{e}_i = e_i) = \max(p_i, 1 - p_i).$$

If we assume that the components in the random error vector e are mutually independent, then it follows that the probability of successful decoding $\mathbf{P}(e = \mathcal{E}_I^s(\hat{e}_I))$ for I is equal to $\rho(I) = \prod_{i \in I} \rho_i$. Thus if we want to maximize this probability we need the *most reliable information set* I, i.e., the one[8] with the maximal possible value of $\rho(I)$. Finding such a set may at first sight look like a prohibitively hard task. However, if instead of $\rho(I)$ we consider $\ln \rho(I) = \sum_{i \in I} w_i$, where $w_i = \ln \rho_i$; then

[5] A *matroid* is defined by a non-empty collection $\mathcal{I}$ of subsets (called the *independent sets*) from some set E (called the *ground set*) such that: (1) $\mathcal{I}$ is closed under taking subsets; (2) for any two $A, B \in \mathcal{I}$ if $|A| < |B|$ then $A \cup \{b\} \in \mathcal{I}$ for some $b \in B \setminus A$. Any maximal (by inclusion) independent set is called a *basis*.

[6] For example, one can use a syndrome BP decoder for this purpose (see Section 3.3).

[7] When $p_i = 1/2$ we can set $\hat{e}_i$ randomly to either 0 or 1 with equal probability.

[8] If there are several such sets, we can use any of them.

Algorithm 1: Syndrome OSD-w algorithm

Input: target weight function $\mathrm{wt}(\cdot)$,
binary parity-check matrix H,
syndrome vector $s \in \mathbb{F}_2^m$,
vector of hard decisions $\hat{e} \in \mathbb{F}_2^n$;
Output: error vector $e \in \mathbb{F}_2^n$ such that $He = s$;

1 $J \leftarrow \varnothing$;
2 **for** $i \leftarrow 1$ **to** n **do**
3 $J' \leftarrow J \cup \{i\}$;
4 **if** $\mathrm{rk}\, H_{J'} > \mathrm{rk}\, H_J$ **then**
5 $J \leftarrow J'$;
6 **end**
7 **end**
8 $\hat{x} \leftarrow \arg\min_{x \in \mathbb{F}_2^w} \mathrm{wt}\big(\mathcal{E}_I^s(\mathcal{R}_{[w]}^x \hat{e}_I)\big)$, where $I = \bar{J}$;
9 **return** $e = \mathcal{E}_I^s(\mathcal{R}_{[w]}^{\hat{x}} \hat{e}_I)$;

Algorithm 2: Fast syndrome OSD-0 algorithm

Input: binary parity-check matrix H,
syndrome vector $s \in \mathbb{F}_2^m$,
vector of hard decisions $\hat{e} \in \mathbb{F}_2^n$;
Output: error vector $e \in \mathbb{F}_2^n$ such that $He = s$;

1 $J \leftarrow \varnothing$;
2 $s' \leftarrow s + H\hat{e}$;
3 **for** $i \leftarrow 1$ **to** n **do**
4 **if** $\mathrm{rk}\,[H_J, s'] = \mathrm{rk}\, H_J$ **then**
5 **break**;
6 **end**
7 $J' \leftarrow J \cup \{i\}$;
8 **if** $\mathrm{rk}\, H_{J'} > \mathrm{rk}\, H_J$ **then**
9 $J \leftarrow J'$;
10 $s' \leftarrow s' + \hat{e}_i H_i$;
11 **end**
12 **end**
13 $x \leftarrow$ the solution of $H_J\, x = s'$;
14 **return** $e = \mathcal{R}_J^x \hat{e}$;

we can see that the most reliable information set I is given by (7), and, as we mentioned earlier, it is possible to find I using a greedy algorithm [31, 32].

This greedy algorithm is the main part of the OSD decoder. Since in our case the code is defined by a parity-check matrix H, it is more convenient to use the right part of (7), which means that we want to find the index set J corresponding to the *least reliable basis* $\{H_i\}_{i \in J}$ for the column space of H, i.e., $J \in \mathcal{B}(H)$ with the smallest $\rho(J)$. For simplicity, we assume that *the columns of H are already rearranged* such that the reliability of the corresponding positions increases: $\rho_1 \leqslant \cdots \leqslant \rho_n$. In this case, it is not hard to see that the collection $\{H_i\}_{i \in J}$ of the first $r = \mathrm{rk}\, H$ linearly independent columns is the least reliable basis [32]. We can find J by applying Gaussian elimination to equation (8), which gives us, in time $O(n^3)$, the following equation:

$$\tilde{H}e = \tilde{s}. \tag{10}$$

Then J is the index set of the first r pivot columns in $\tilde{H}$, i.e., $\tilde{H}_J = (\delta_{ij})_{r \times r}$. Moreover, from (7) it follows that the index set $I = \bar{J}$ is the most reliable information set, and we can find the error vector e from (10) in a very straightforward way since $\tilde{H}_J$ is the identity matrix.

The described above syndrome decoding algorithm is usually called *order-0 OSD decoder*. As we see, it first finds the most reliable information set I and then recovers the unique vector $e \in \mathbb{F}_2^n$ subject to the following conditions:

$$e_I = \hat{e}_I, \quad He = s. \tag{11}$$

In fact, we can further improve the error-correcting performance by using *order-w OSD decoder* (abbreviated as OSD-w). In this modification, after we find the most reliable information set I, we look at all error vectors e obtained from equation (10) by setting the first w least reliable positions from I to all possible values $x \in \mathbb{F}_2^w$ (we can obtain them using the encoding operator $\mathcal{E}_I^s$). Finally, we select an error vector e of the minimal weight $\mathrm{wt}(e)$ among all the 2^w obtained error vectors. Here the weight function $\mathrm{wt}(\cdot)$ depends on the specific channel we use. In the case of depolarizing noise, this weight function is just the weight of the corresponding Pauli operator.

A simplified pseudocode of the above OSD-w algorithm is shown in Algorithm 1. Here in the two last lines, we used a shorthand notation $\mathcal{R}_I^x v$ for the result of the replacement of the subvector v_I in the vector v by $x \in \mathbb{F}_2^{|I|}$. Therefore, $\mathcal{R}_{[w]}^x \hat{e}_I$ is obtained from $\hat{e}_I$ if we replace its first w bits by $x \in \mathbb{F}_2^w$. Since we assume that the columns of H are already sorted according to the reliabilities, these first w positions in I are the least reliable ones. We should also note that the main cycle of this algorithm, which finds J (lines 1–7), can be efficiently implemented using Gaussian elimination, as already discussed above.

Remark. There are some differences between the standard order-w OSD decoder from [13, 14] and the proposed OSD-w post-processing algorithm. For example, we try all 2^w bit flips of the w least reliable information bits (line 8), while in the standard OSD we try all the $\sum_{i \leqslant w} \binom{k}{i}$ bit flips of no more than w bits.

As we will see in Section 3.3, if the OSD-w decoder is used as a post-processing algorithm after the BP decoder for QLDPC codes, then it can radically improve the error correcting performance. In fact, in many cases even OSD-0 is enough, and thus the computational cost of the decoding is $O(n^3)$. Moreover, in the case of the

OSD-0, there is no need to do full Gaussian elimination for H. Let I be the most reliable information set found by the OSD-0 algorithm, then we can stop extending the index set J in Algorithm 2 when $\operatorname{rk}[H_J, s'] = \operatorname{rk} H_J$, and put $e = \mathcal{R}^x_J \hat{e}$, where $s' = s + H_{\bar{J}}\hat{e}_{\bar{J}}$, and x is the unique[9] solution of $H_J x = s'$. Indeed, we get

$$He = H_J x + H_{\bar{J}}\hat{e}_{\bar{J}} = s' + H_{\bar{J}}\hat{e}_{\bar{J}} = s.$$

Since we also have $I \subseteq \bar{J}$, then e satisfies conditions (11) and thus is the error vector found by the OSD-0 algorithm. This observation gives us a faster version of the OSD-0 algorithm (see Algorithm 2).

3.2 Modified OSD post-processing algorithm for stabilizer codes

In this subsection, we show how to adapt the OSD decoder from the previous subsection to a scenario where it is used as a post-processing algorithm after the BP decoder for QLDPC codes. There are a number of quantum noise models. In this paper, we consider the depolarizing channel only. However many of our ideas may be used for other memoryless quantum noise models. In the *depolarizing channel* model with *error probability* p, a quantum state $|\psi\rangle \in \mathbb{C}^{2^n}$ is subject to a random Pauli error

$$E = E_1 \otimes \cdots \otimes E_n \in \mathcal{P}_n^*,$$

where all E_i are i.i.d, and for all $i \in [n]$ we have:

$$\mathbf{P}(E_i = X) = \mathbf{P}(E_i = Y) = \mathbf{P}(E_i = Z) = p/3.$$

In what follows, it is convenient for our goals to represent (with a small abuse of notation and terminology) the set of matrices $\mathcal{P}_1^* = \{I, X, Y, Z\}$ by the elements of the finite field $\mathbb{F}_4$, where I is represented by $0 \in \mathbb{F}_4$, and the Pauli matrices X, Y, Z by the three non-zero elements from $\mathbb{F}_4$. To distinguish these finite field elements from the corresponding matrices we denote the former using a different font: $\mathrm{I}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}$. Further, we represent a Pauli vector from $\mathcal{P}_n^*$ by the corresponding vector from $\mathbb{F}_4^n$, which we also call a *Pauli vector*. Using this conventions, we represent the binary $m \times 2n$ parity-check matrix of a stabilizer code $\mathcal{C}$ (see equation (3)) by the corresponding $m \times n$ matrix over $\mathbb{F}_4$ and call it the *stabilizer matrix* of $\mathcal{C}$.

Since we consider the depolarizing channel, for the best performance, it is better to use the non-binary version of the syndrome BP decoder (see [21], [11, Algorithm 1]), which also takes into account the correlations between X and Z errors in qubits. To describe the OSD algorithm in this case we need some extra notations.

[9]The solution is unique since $\operatorname{rk} H_J = |J|$.

- If $v \in \mathbb{F}_4^n$ is a Pauli vector, and $v_i = v_i^X \mathrm{X} + v_i^Z \mathrm{Z}$, where $v_i^X, v_i^Z \in \mathbb{F}_2$, $i \in [n]$; then we can define the binary vectors:

$$\mathbb{b}(v) = (v_1^X, v_1^Z, v_2^X, v_2^Z, ..., v_n^X, v_n^Z) \in \mathbb{F}_2^{2n},$$
$$\mathbb{b}^*(v) = (v_1^Z, v_1^X, v_2^Z, v_2^X, ..., v_n^Z, v_n^X) \in \mathbb{F}_2^{2n}.$$

  We also need the inverse mapping $\mathbb{b}^{-1}(\cdot)$ for $\mathbb{b}(\cdot)$ that maps vectors from $\mathbb{F}_2^{2n}$ back to $\mathbb{F}_4^n$.

- If $\mathcal{H}$ is an $m \times n$ stabilizer matrix over $\mathbb{F}_4$, then $\mathbb{b}(\mathcal{H})$ denote the $m \times 2n$ binary matrix obtained by mapping each row h from $\mathcal{H}$ to the row $\mathbb{b}^*(h)$.

The main motivation for these, not very standard, notations is as follows. If $e \in \mathbb{F}_4^n$ is a Pauli error, then it is easy to check that

$$s = \mathbb{b}(\mathcal{H})\mathbb{b}(v)$$

is the corresponding *syndrome* vector, i.e., for every $i \in [m]$ we get $s_i = 0$ **iff** the i-th stabilizer from $\mathcal{H}$ commutes with the Pauli error e. Let us emphasize that the binary representations $\mathbb{b}(v)$ and $\mathbb{b}(\mathcal{H})$ can be also obtained from the binary representations (1) and (3) by a permutation of indices and columns, respectively.

Example. Let us illustrate the above notations on the well-known non-CSS $[[5, 1, 3]]$ code [33] defined by the parity-check matrix

$$H = (H_X|H_Z) = \left(\begin{array}{ccccc|ccccc} 1&0&0&1&0&0&1&1&0&0\\ 0&1&0&0&1&0&0&1&1&0\\ 1&0&1&0&0&0&0&0&1&1\\ 0&1&0&1&0&1&0&0&0&1 \end{array}\right),$$

which corresponds to the matrices

$$\mathcal{H} = \begin{pmatrix} \mathrm{X}&\mathrm{Z}&\mathrm{Z}&\mathrm{X}&\mathrm{I}\\ \mathrm{I}&\mathrm{X}&\mathrm{Z}&\mathrm{Z}&\mathrm{X}\\ \mathrm{X}&\mathrm{I}&\mathrm{X}&\mathrm{Z}&\mathrm{Z}\\ \mathrm{Z}&\mathrm{X}&\mathrm{I}&\mathrm{X}&\mathrm{Z} \end{pmatrix}, \mathbb{b}(\mathcal{H}) = \begin{pmatrix} 0&1&1&0&1&0&0&1&0&0\\ 0&0&0&1&1&0&1&0&0&1\\ 0&1&0&0&0&1&1&0&1&0\\ 1&0&0&1&0&0&0&1&1&0 \end{pmatrix}.$$

If we have the Z error in the second qubit, then we get the error vector $e = (\mathrm{I}, \mathrm{Z}, \mathrm{I}, \mathrm{I}, \mathrm{I}) \in \mathbb{F}_4^5$, its binary representation $\mathbb{b}(e) = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0) \in \mathbb{F}_2^{10}$, and the corresponding syndrome vector $s = (0, 1, 0, 1)$.

Since the depolarizing channel is non-binary, we need some further adjustments in the syndrome OSD decoder (Algorithms 1 and 2) to use it as a post-processor after the non-binary BP decoder. We call this modified version the *order-w qOSD* algorithm, which is defined via the order-w OSD algorithm from the previous section as follows:

$$\mathrm{qOSD}_w(\mathcal{H}, s, \hat{e}) = \mathbb{b}^{-1}\big(\mathrm{OSD}_w(|\cdot|_{\mathbf{Sp}}, \mathbb{b}(\mathcal{H}), s, \mathbb{b}(\hat{e}))\big).$$

Here we use the *symplectic weight*[10] $|\cdot|_{\mathbf{Sp}}$ as the target weight function $\operatorname{wt}(\cdot)$ for the OSD algorithm, which is

[10]It is easy to see that the symplectic weight $|x|_{\mathbf{Sp}}$ is equal to the weight $\operatorname{wt}(E)$ of the Pauli error $E \in \mathcal{P}_n^*$ represented in the binary form by the vector $x \in \mathbb{F}_2^{2n}$.

defined as:

$$|x|_{\mathbf{Sp}} = \sum_{i=1}^{n} (x_{2i-1} \vee x_{2i}).$$

As we see, the input for the qOSD algorithm includes the $m \times n$ stabilizer matrix $\mathcal{H}$ over $\mathbb{F}_4$, the syndrome $s \in \mathbb{F}_2^m$, and the error vector $\hat{e} \in \mathbb{F}_4$, which is the hard decisions vector for the result of the non-binary BP decoder (see the next subsection).

Remark. The symplectic weight $|\cdot|_{\mathbf{Sp}}$ is used above for the depolarizing channel. For the variant of the depolarizing channel, where the X and Z errors are independent, the standard binary Hamming weight $|\cdot|$ should be used instead. In general, the optimal choice of the target weight function $\mathrm{wt}(\cdot)$ is the one, where the most probable errors are of the least possible weight.

3.3 Main decoding algorithm (BP-OSD)

In this subsection, we describe our main decoding algorithm (see Algorithm 3) called the *BP-OSD* or *BP-OSD-w* (if we want to emphasize the OSD order w). It consists of two stages: the non-binary BP decoding (lines 1–4) and the OSD post-processing (lines 6–8). In fact, at the first stage, any decoding algorithm may be used instead of BP if it can provide soft decisions. The goal of this stage is either to find the error vector $e \in \mathbb{F}_4^n$ or to obtain the probabilities $\mathbf{P}(e_i = E)$, $i \in [n]$, for each type of the Pauli error $E \in \{\mathrm{I}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$.

First, we run the BP decoder and obtain the *soft decisions* (line 1), i.e., for ever $i \in [n]$ we get the 4 numbers $(p_{i,\mathrm{I}}, p_{i,\mathrm{X}}, p_{i,\mathrm{Y}}, p_{i,\mathrm{Z}})$, where $p_{i,E}$ can be interpreted as the probability $\mathbf{P}(e_i = E)$ of the error $E \in \{\mathrm{I}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$ in the i-th qubit. Next, we find the *hard decisions* for all qubits by the formula (line 2):

$$\hat{e}_i = \underset{E \in \{\mathrm{I},\mathrm{X},\mathrm{Y},\mathrm{Z}\}}{\arg\max}\ p_{i,E}, i \in [n].$$

If the BP decoding is successful (i.e., we have the correct syndrome vector), then there is no need in the post-processing, and the BP-OSD decoder returns the result of the BP decoder (line 4).

After the first stage, the probabilities $p_{i,E}$ are also used in the BP-OSD algorithm to sort the qubits in the order of increasing reliability $\rho(p_{i,\mathrm{I}}, p_{i,\mathrm{X}}, p_{i,\mathrm{Y}}, p_{i,\mathrm{Z}})$, where we can define the *reliability* of the i-th qubit as

$$\rho(p_{i,\mathrm{I}}, p_{i,\mathrm{X}}, p_{i,\mathrm{Y}}, p_{i,\mathrm{Z}}) = \mathbf{P}(\hat{e}_i = e_i) = \max_{E \in \{\mathrm{I},\mathrm{X},\mathrm{Y},\mathrm{Z}\}} p_{i,E}.$$

However, in all our simulations we used the formula

$$\rho(p_{i,\mathrm{I}}, p_{i,\mathrm{X}}, p_{i,\mathrm{Y}}, p_{i,\mathrm{Z}}) = p_{i,\mathrm{I}},$$

which gives similar error-correcting performance, but it is a little bit easier to calculate. After we sorted the qubit positions (line 7), the qOSD algorithm from the previous subsection is used to recover the errors in the least reliable qubits from the hard decisions for the remaining qubits (line 8).

Algorithm 3: BP with OSD-w post-processing

Input: stabilizer matrix $\mathcal{H}$,
syndrome vector $s \in \mathbb{F}_2^m$;
Output: error vector $e \in \mathbb{F}_4^n$;

1 $(p_{i,\mathrm{I}}, p_{i,\mathrm{X}}, p_{i,\mathrm{Y}}, p_{i,\mathrm{Z}})_{i=1}^n \leftarrow \mathrm{BP}(s)$;
2 Make hard decisions:
  $\hat{e}_i \leftarrow \arg\max_{E \in \{\mathrm{I},\mathrm{X},\mathrm{Y},\mathrm{Z}\}} p_{i,E}$, $i \in [n]$;
3 **if** $\mathbb{b}(\mathcal{H})\mathbb{b}(\hat{e}) = s$ **then**
4 **return** $e = \hat{e}$;
5 **else**
6 Calculate the reliabilities:
   $\rho_i \leftarrow \rho(p_{i,\mathrm{I}}, p_{i,\mathrm{X}}, p_{i,\mathrm{Y}}, p_{i,\mathrm{Z}})$, $i \in [n]$;
7 Sort the qubits by their reliabilities:
   $\rho_{\sigma(1)} \leqslant \rho_{\sigma(2)} \leqslant \ldots \leqslant \rho_{\sigma(n)}$, $\sigma \in S_n$;
8 **return** $e = \sigma^{-1}\big(\mathrm{qOSD}_w(\sigma(\mathcal{H}), s, \sigma(\hat{e}))\big)$;
9 **end**

In Algorithm 3, we used the qOSD algorithm, which is a good choice for the standard depolarizing channel. If we have a channel with independent X and Z errors, then for a CSS code defined by a pair of parity-check matrices H_X, H_Z we can also use the standard OSD (Algorithms 1 and 2) for the X and Z components of the error vector e separately. In this case, lines 6–8 in Algorithm 3 can be replaced by the following steps:

1. Calculate the X and Z error probabilities:

$$p_i^X = p_{i,\mathrm{X}} + p_{i,\mathrm{Y}},$$
$$p_i^Z = p_{i,\mathrm{Y}} + p_{i,\mathrm{Z}}.$$

2. Sort the qubits in the *decreasing* order of their X error and Z error probabilities and let $\sigma_X, \sigma_Z \in S_n$ be the corresponding permutations such that:

$$p^X_{\sigma_X(1)} \geqslant p^X_{\sigma_X(2)} \geqslant \ldots \geqslant p^X_{\sigma_X(n)},$$
$$p^Z_{\sigma_Z(1)} \geqslant p^Z_{\sigma_Z(2)} \geqslant \ldots \geqslant p^Z_{\sigma_Z(n)}.$$

3. Let $\hat{e} = \hat{e}_X\mathrm{X} + \hat{e}_Z\mathrm{Z}$, $s_X = H_X\hat{e}_Z$, and $s_Z = H_Z\hat{e}_X$. It is clear that $s = [s_X, s_Z]$. Run the OSD decoder separately for the X and Z components of $\hat{e}$:

$$e_X = \sigma_X^{-1}\big(\mathrm{OSD}_w(|\cdot|, \sigma_X(H_Z), s_Z, \sigma_X(\hat{e}_X))\big),$$
$$e_Z = \sigma_Z^{-1}\big(\mathrm{OSD}_w(|\cdot|, \sigma_Z(H_X), s_X, \sigma_Z(\hat{e}_Z))\big).$$

4. Return $e = e_X\mathrm{X} + e_Z\mathrm{Z}$.

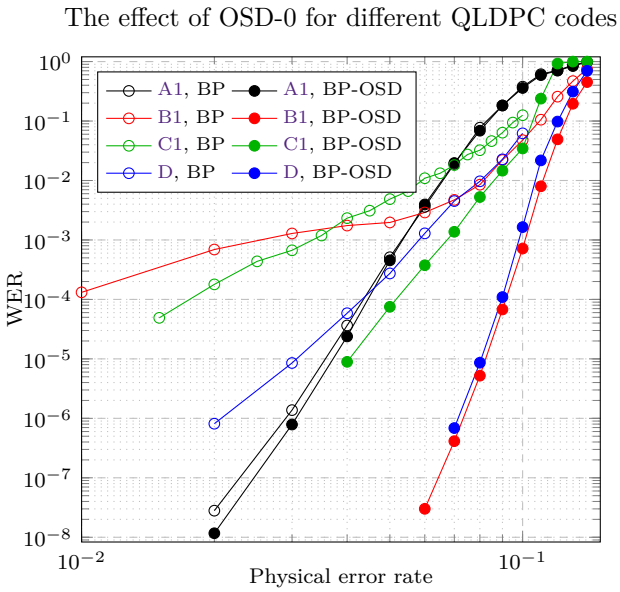

Figure 1: The WER performance of several QLDPC codes under the BP and BP-OSD-0 decoders, where: A1 is the 10-limited generalized bicycle [[254,28]] code; B1 is the 6-limited generalized hypergraph product [[882,24]] code; C1 is the 10-limited hypergraph product [[7938,578,16]] code; D is the 8-limited Haah's cubic [[1024,30]] code (see Appendix B).

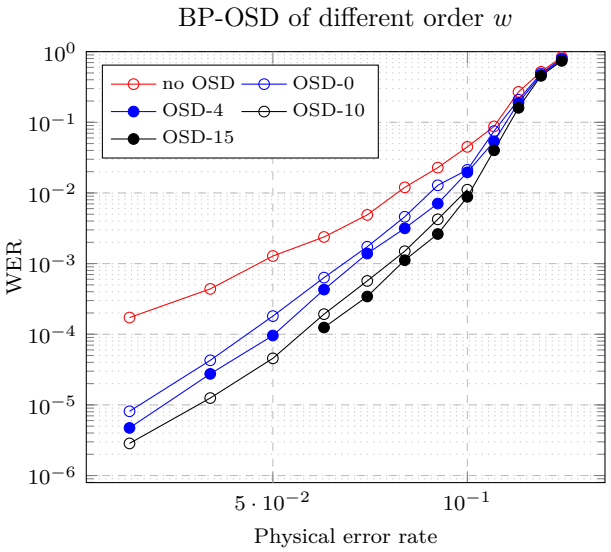

Figure 2: The impact of the OSD order w on the WER performance of the 8-limited [[882,48,16]] code B2 in Appendix B.

For all our simulations we used the normalized min-sum (NMS) decoder with the normalization factor 0.625, which approximates the non-binary BP decoder from [21], [11, Algorithm 1] in log-domain and is more numerically stable in some cases. The maximal number of iterations was set to 32. We used the layered scheduling in order to increase the convergence speed of the decoder by approximately two times[11]. For a good review of practical aspects of the BP decoder implementation, see [35, Chapter 4]. The error-correcting performance in our simulations is measured either in terms of the *logical error rate* or the *word error rate*[12] (*WER*). We should also stress that in many cases we use only the OSD-0 algorithm (Algorithm 2), which has complexity $O(n^3)$, while the complexity in the general case is $O(n^3+n2^w)$.

In Fig. 1 we show the effect of the OSD-0 post-processing after the BP decoder for different QLDPC codes, including some known and new codes described in the next sections. As you can see, the gain of the BP-OSD over the BP decoder in terms of WER for some codes is up to 5 orders of magnitude (code B1). However, for code A1 the difference between BP-OSD and BP is quite small. We think that this is because the column weight of the matrices H_X and H_Z is quite high (it is equal to 5), and hence the performance of the BP decoder is very good even without the post-processing.

Remark. It is interesting to note that 8-limited Haah's [[1024,30]] code, which has local stabilizers in 3D, also performs very well under the BP-OSD decoder. To the best of our knowledge, this is the first such demonstration on the depolarizing channel. From our point of view, this observation implicitly suggests that some of Haah's cubic codes may have very good minimum distances. In fact, even after very long runs of the BP-OSD decoder on this [[1024,30]] code, we have not found any non-degenerate codewords of weight less than 32.

In Fig. 2 we show the impact of the OSD order w on the WER performance of the 8-limited [[882,48,16]] code (Appendix B, code B2) under the BP-OSD decoder. We see that the WER performance in this case can be further improved by increasing the OSD order w. At the same time, from our experiments with the OSD post-processing on different QLDPC codes, we observed that in many cases the impact of the OSD order on the WER performance is quite small.

3.4 Different post-processing algorithms

In this section, the OSD post-processing algorithm is compared against the other known post-processing methods that also significantly improve the performance of the BP decoder. We assume that the prob-

[11] Since the first version of the current work was released, the layered (serial) schedule of the BP decoder for QLDPC codes was also studied in [34].

[12] Recall that the *word error rate*, also known as the *block error rate* or *frame error rate*, is the rate of codewords where the decoder does not give a correct answer (i.e., it either fails to decode or we have at least one logical error after the decoding).

ability distributions in the BP decoder are represented in terms of the log-likelihood ratios (LLRs). Below you can find a brief description of these methods, where each method can be applied repeatedly after the BP decoder until all the parity-checks are satisfied or the maximal number of attempts n_a is reached.

- **Random perturbation** [21]. If the syndrome is non-zero after the BP decoding, then we randomly choose an unsatisfied c-node and randomly change the initial LLRs for all the v-nodes adjacent to this c-node. The main parameter of this method is the variance of the perturbation magnitude. After the changes are done, the BP decoder runs with the perturbed input LLRs.

- **Enhanced feedback** [22]. It is similar to the previous approach but the perturbations are not random and calculated using the previous BP output. If after the BP decoding the parity-checks are not satisfied, we randomly choose an unsatisfied c-node. Then for all the v-nodes adjacent to this c-node, we set the initial LLRs using the BP decoder output for these nodes. After this is done the BP decoder runs with the perturbed input LLRs.

- **Matrix augmentation** [23]. In this method instead of modification of the input LLRs, the parity-check matrix itself is modified by random duplication of some rows. The fraction δ of the duplicated rows is called the *augmentation density*. Then a new decoding attempt with the augmented matrix is performed.

To compare the performance of all the described methods with the BP-OSD decoder we use the 6-limited $[[1270, 28]]$ QLDPC code (B3 in Appendix B). This code belongs to the new class of codes described in Section 5. For all these methods we used $n_a = 100$. We can see in Fig. 3 that the OSD post-processing outperforms all the above-mentioned post-processing methods and also outperforms the 4-limited $[[1201, 1, 25]]$ surface code on the MPS decoder from [5], which is almost optimal for this code. Let us note that all the other post-processing methods also have the WER gain that is more than 10^3 over the BP decoder for code B3. In fact, we observed a similar WER gain for many other codes from the class of $(3, 6)$-regular CSS QLDPC codes with a sufficiently large minimum distance. We think that this is mainly because the classical $(3, 6)$-regular LDPC codes defined by the matrices H_X and H_Z of such CSS codes themselves have many harmful trapping sets. Thus these $(3, 6)$-regular QLDPC codes additionally have a lot of degenerate codewords of very low weight starting from 6.

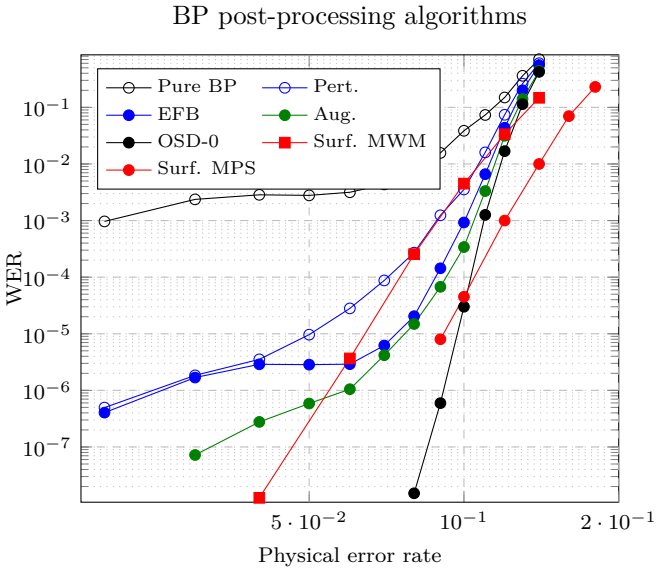

Figure 3: The WER performance of different post-processing algorithms for the BP decoder on the 6-limited [[1270,28]] QLDPC code (B3 in Appendix B). The red curves are for the 4-limited [[1201,1,25]] surface code under the minimum weight matching (MWM) and the MPS-based decoders [36, 5].

4 New generalized bicycle codes

4.1 Ansatz with commuting matrices

The commutativity conditions such as (4) and (5) are a serious obstacle to designing good QLDPC codes using random-like constructions similar to the constructions used for classical LDPC codes. Thus it makes sense to consider large families of matrices of some particular form called *ansatz*, where the commutativity conditions are always satisfied. One such quite general ansatz for CSS codes was proposed in [10] as a generalization of the bicycle QLDPC codes [37]. Let us briefly remind this ansatz. Consider two commuting binary $n \times n$ matrices A and B, i.e., $AB = BA$. Let

$$H_X = [A, B] \text{ and } H_Z = [B^{\mathrm{T}}, A^{\mathrm{T}}]. \tag{12}$$

Then we see that $H_X H_Z^{\mathrm{T}} = AB + BA = \mathbf{0}$, and the commutativity condition (5) is always satisfied. It was proposed in [10] to use binary circulant matrices A and B since they always commute. The corresponding class of codes is called the *generalized bicycle (GB) codes*, where the bicycle codes [37] are obtained as a special case when $B = A^{\mathrm{T}}$.

4.2 Ring of circulants

Let us recall that an $\ell \times \ell$ circulant matrix A over $\mathbb{F}_q$ takes the form

$$A = \begin{pmatrix} a_0 & a_{l-1} & \dots & a_1 \\ a_1 & a_0 & \dots & a_2 \\ \dots & \dots & \dots & \dots \\ a_{\ell-1} & a_{\ell-2} & \dots & a_0 \end{pmatrix},$$

where $a_0, \dots, a_{\ell-1} \in \mathbb{F}_q$. It is readily seen that the matrix A can be represented in the form

$$A = a_0 I + a_1 P + \dots a_{\ell-1} P^{\ell-1},$$

where I is the $\ell \times \ell$ identity matrix and

$$P = \begin{pmatrix} 0 & 0 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

is the $\ell \times \ell$ permutation matrix representing the *right* cyclic shift by *one* position. Since $P^\ell = I$, we see that the ring of all $\ell \times \ell$ circulant matrices over $\mathbb{F}_q$ is isomorphic to the ring $\mathbb{F}_q^{\langle \ell \rangle} = \mathbb{F}_q[x]/(x^\ell - 1)$ of polynomials over $\mathbb{F}_q$ modulo the polynomial $x^\ell - 1$.

Hence the circulant matrix A can be uniquely represented by the polynomial $a(x) = a_0 + a_1 x + \dots + a_{\ell-1} x^{\ell-1}$ and the product $C = AB$ of two circulant matrices represented by polynomials $a(x), b(x) \in R_\ell$ corresponds to the polynomial

$$c(x) = a(x)b(x) \bmod x^\ell - 1 \tag{13}$$

which is called the *cyclic convolution* of $a(x)$ and $b(x)$. Likewise, if we want to find a matrix-vector product $c = Ab$, where $b = (b_0, \dots, b_{\ell-1})$ and $c = (c_0, \dots, c_{\ell-1})$ are (column) vectors corresponding to $b(x)$ and $c(x)$, we can also use the cyclic convolution (13).

4.3 Dimension of generalized bicycle codes

As we saw before, to define two binary circulant $\ell \times \ell$ matrices A and B we need to provide two binary polynomials $a(x), b(x) \in \mathbb{F}_2^{\langle \ell \rangle}$. The dimension[13] k of the corresponding CSS $[[2\ell, k]]$ code is given by the following proposition.

Proposition 1. *The dimension k of the the generalized bicycle $[[2\ell, k]]$ code defined by $a(x), b(x) \in \mathbb{F}_2^{\langle \ell \rangle}$ is given by the formula:*

$$k = 2 \deg g(x), \tag{14}$$

where $g(x) = \gcd(a(x), b(x), x^\ell - 1)$.

[13] Let us point out that this dimension formula was given in the paper [10, Theorem 2] in a slightly more complex form. A similar formula was proved only in the special case of single generator codes.

To prove this formula we show that $\operatorname{rk} H_X = \operatorname{rk} H_Z = n - \deg g(x)$ and use CSS dimension formula (6).

Let us first find the rank of the matrix $H_X = [A, B]$, which is equal to the dimension of its column space. It is easy to see that the column space of H_X (called its *syndrome space*) is equal to the following set:

$$\{Au + Bv \mid u, v \in \mathbb{F}_2^\ell\}.$$

Using the described above polynomial representation of column vectors and circulant matrices we can consider this set as the following set of polynomials from $\mathbb{F}_2^{\langle \ell \rangle}$:

$$\{a(x)u(x) + b(x)v(x) \mid u(x), v(x) \in \mathbb{F}_2^{\langle \ell \rangle}\}.$$

It is easy to verify that this set is the principal[14] ideal of the ring $\mathbb{F}_2^{\langle \ell \rangle}$ generated by $g(x)$. Hence it is the cyclic code $\mathcal{C}_g$ with generator polynomial $g(x)$, and we proved that $\operatorname{rk} H_X = \dim \mathcal{C}_g = n - \deg g(x)$. We call this cyclic code $\mathcal{C}_g$ the *syndrome code* of H_X since its codewords are precisely the polynomial representations of the syndrome space of H_X.

To complete the proof we also need to show that $\operatorname{rk} H_Z = n - \deg g(x)$. Using similar arguments as above we can consider the syndrome code for H_Z and show that it is generated by the "transposed" polynomial $g^*(x) = g(x^{-1})$. Though the codes generated by the polynomials $g(x)$ and $g^*(x)$ may differ, they always have the same dimension since the corresponding circulant matrices G and G^{T} have the same rank. Hence we also proved that $\operatorname{rk} H_Z = n - \deg g(x)$, and the proof of formula (14) is complete.

4.4 Construction methods

The proof of Proposition 1 gives us also some valuable information on how to find generalized bicycle codes of high dimension. If we fix the circulant size ℓ then all possible dimensions k of the generalized bicycle codes with this circulant size are characterized by the degrees of all possible factors of the polynomial $x^\ell - 1$. Indeed, for each factor $g(x)$ of the polynomial $x^\ell - 1$ we can always choose polynomials $a(x), b(x) \in \mathbb{F}_2^{\langle \ell \rangle}$ such that:

$$a(x) \bmod g(x) = b(x) \bmod g(x) = 0 \tag{15}$$

since these polynomials are just two codewords from the cyclic code $\mathcal{C}_g$ generated by $g(x)$, which we called the syndrome code of H_X. To produce a w-limited QLDPC we just need to find low weight polynomials $a(x)$ and $b(x)$ that are the codewords of $\mathcal{C}_g$. In practice, this can be accomplished by several methods. If the circulant size ℓ is relatively small, we can find $a(x), b(x)$ by

[14] An ideal I in a ring R is *principal* if $I = \{au \mid u \in R\}$ for some $a \in R$.

an exhaustive search over all polynomials of the given weight.

Another alternative is to generate random polynomials of a given weight from $\mathbb{F}_2^{\langle \ell \rangle}$ until we find a pair of polynomials that satisfies condition (15). Since the probability that a random polynomial of a given weight belongs to $\mathcal{C}_g$ is equal approximately to $2^{-\deg g(x)}$; then if we test more then $2^{\deg g(x)}$ random polynomials we will find a polynomial from $\mathcal{C}_g$ with high probability.

When we find a pair of polynomials $a(x), b(x)$ we also need to check that the corresponding code has good error correcting performance. This can be done by a simulation of the corresponding generalized bicycle code.

All the generalized bicycle codes from Appendix B were found by the described above methods.

4.5 GB codes with syndrome protection

Another important observation, made in the proof of Proposition 1, is that the syndrome codes of the parity-check matrices H_X and H_Z are the cyclic codes with the generator polynomials $g(x)$ and $g^*(x)$, respectively. Let us mention that the syndrome code of a parity-check matrix is precisely the set of all possible syndromes for it. Hence if we use generator polynomials $g(x)$ and $g^*(x)$ that define cyclic codes with minimum distance d, we see that the syndromes for matrices H_X and H_Z are protected by these cyclic codes. Since the syndrome measurements for quantum codes are performed by faulty hardware, some additional protection of the syndromes may be used to improve the reliability of the syndrome measurements [38, 39]. Let us also mention that the polynomials $g(x)$ and $g^*(x)$ always produce cyclic codes $\mathcal{C}_g$ and $\mathcal{C}_{g^*}$ with the same minimum distance since the "transpose" map

$$c(x) = \sum_{i=0}^{\ell-1} c_i x^i \mapsto c^*(x) = \sum_{i=0}^{\ell-1} c_i x^{\ell-i} \bmod x^\ell - 1$$

is an automorphism of the ring $\mathbb{F}_2^{\langle \ell \rangle}$ that respects the weight of the polynomials, and therefore we have that $\mathcal{C}_{g^*} = \{c^*(x) \mid c(x) \in \mathcal{C}_g\}$.

Hence we can use any cyclic code with generator polynomial $g(x)$ and minimum distance d to construct a CSS code, where the syndromes for H_X and H_Z are protected by cyclic codes of minimum distance d. It is important to note that since $a(x), b(x) \in \mathcal{C}_g$, the weight of the polynomials $a(x), b(x)$ can not be smaller than this minimum distance d.

4.6 Comparison with other codes

In this subsection, we consider several new examples of generalized bicycle codes and compare their performance under the BP-OSD decoder against some other already known QLDPC codes.

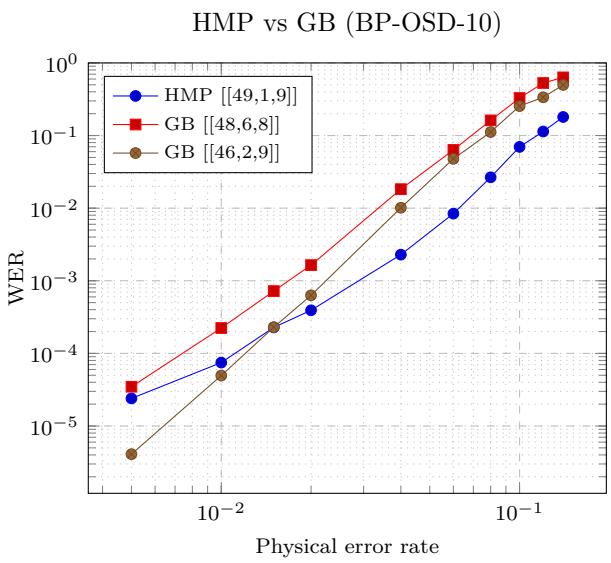

Figure 4: The WER performance of the 8-limited generalized bicycle (GB) codes (A3 and A4 in Appendix B) and the 8-limited $[[49, 1, 9]]$ homological product (HMP) code from [8] under the BP decoder with the OSD-10 post-processing.

Example 1. Let us consider the primitive narrow-sense BCH $[127, 14, 5]$ code $\mathcal{C}_g$ with the generator polynomial:

$$g(x) = (x^7 + x + 1) \cdot (x^7 + x^5 + x^3 + x + 1).$$

If we set $a(x) = 1 + x^{15} + x^{20} + x^{28} + x^{66}$ and $b(x) = 1 + x^{58} + x^{59} + x^{100} + x^{121}$, then we obtain the 10-limited generalized bicycle $[[254, 28]]$ code. Its minimum distance is not available, but the performance of this code (see Fig. 1, code A1) is almost the same as the performance of the 10-limited hypergraph product $[[7938, 578, 16]]$ code[15] obtained from the two identical circulant parity-check matrices H of the cyclic code $[63, 17, 16]$ code defined by the polynomial $h(x) = 1 + x^3 + x^{34} + x^{41} + x^{57}$. This particular code was chosen in order to match the high rate of the $[[254, 28]]$ code. It is important to note that both codes do not have 4-cycles in matrices H_X and H_Z, and they have the same weight of stabilizers. It is also interesting that the performance of the $[[254, 28]]$ code is almost the same even under the classical BP decoder without OSD post-processing. The reason of such good performance with the BP decoder is not fully understood. One of the possible explanations is related with the trapping set structure of the $[[254, 28]]$ code. Since its syndrome code has minimum distance 5, it can not have (a, b) trapping sets[16] with

[15] The definition of these codes is given in Section 5.

[16] An (a, b) *trapping set* or a *near-codeword* for a parity-check

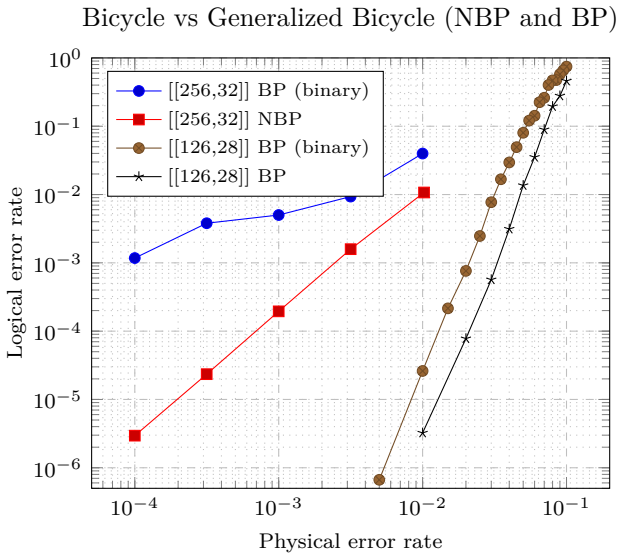

Figure 5: The logical error rate performance of the 16-limited $[[256, 32]]$ bicycle code under the neural BP (NBP) decoder from [41] and the 10-limited $[[126, 28, 8]]$ generalized bicycle code (A2 in Appendix B) under the BP decoder (binary and non-binary) *without* the OSD post-processing.

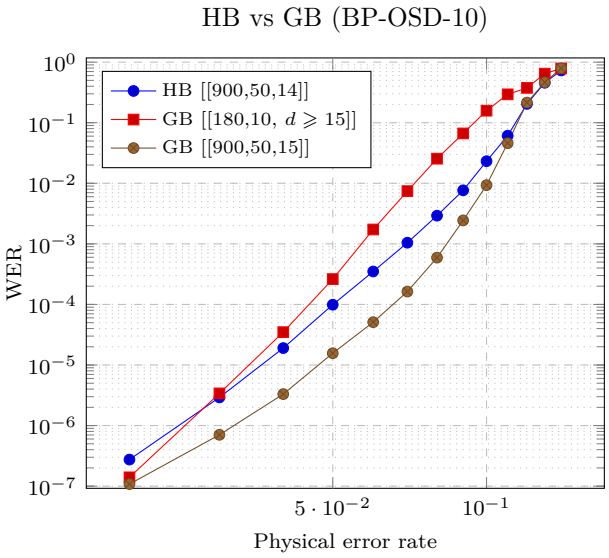

Figure 6: The WER performance of the 8-limited hyperbicycle (HB) [[900,50,14]] code and two 8-limited generalized bicycle (GB) codes (A5 and A6 in Appendix B) under the BP-OSD-10 decoder.

$b < 5$. It is very well known [40] that (a, b) trapping sets with small a and b may greatly decrease the performance of the BP decoder. And the $[[254, 28]]$ can not have the most harmful trapping sets.

Example 2. Let us consider the cyclic $[63, 14, 5]$ code with the generator polynomial

$$g(x) = (x^2 + x + 1) \cdot (x^6 + x^5 + 1) \cdot (x^6 + x^5 + x^4 + x + 1)$$

If we set $a(x) = 1 + x + x^{14} + x^{16} + x^{22}$, $b(x) = 1 + x^3 + x^{13} + x^{20} + x^{42}$ we obtain the 10-limited generalized bicycle $[[126, 28]]$ code. Its performance with the standard BP decoder (binary and non-binary) is shown in Fig. 5, code A2. We compared its performance with the performance of the neural BP decoder for the bicycle $[[256, 32]]$ code constructed in [41]. Such a big difference in the performance is mostly because the QLDPC $[[256, 32]]$ code used in [41] has a small minimum distance compared to the $[[126, 28]]$ code, which minimum distance is 8. This minimum distance was found by an exhaustive search (see Appendix B). Another possible reason is that the neural BP decoder proposed in [41] was based on the binary BP, which usually has worse performance than its non-binary version.

Example 3. In this example we constructed two very small 8-limited generalized bicycle codes (the $[[48, 6, 8]]$ code A3 and the $[[46, 2, 9]]$ code A4, see Appendix B). We compared their performance (see Fig. 4) with the performance of an 8-limited $[[49, 1, 9]]$ homological product (HMP) code from [8] using the BP with OSD-like post-processing. As we can see the performance of the newly constructed codes is similar to the $[[49, 1, 9]]$ code. At the same time, their rates are higher.

Example 4. In Fig. 6 we compared the performance of the 8-limited hyperbicycle [[900, 50, 14]] code from [9, 10] with two new 8-limited generalized bicycle codes (A5 and A6 in Appendix B). We can see that the performance of the generalized bicycle $[[180, 10, d]]$ code, $15 \leqslant d \leqslant 18$, is similar to the hyperbicycle [[900, 50, 14]] code. At the same time, it has the same weight of stabilizers, and its code length is 5 times smaller.

5 Generalization of HP codes

5.1 Hypergraph product (HP) codes

In this section, we propose a new generalization of hypergraph product codes [7] in the case when one of the parity-check matrices in the product is square. Let us first remind the definition of these codes in a matrix form [9]. Suppose we have an $[n_a, k_a, d_a]$ linear code $\mathcal{C}_a$ and an $[n_b, k_b, d_b]$ linear code $\mathcal{C}_b$ defined by parity-check matrices[17] $a \in \mathcal{M}_{m_a \times n_a}(\mathbb{F}_2)$ and $b \in \mathcal{M}_{m_b \times n_b}(\mathbb{F}_2)$ respectively. Then the *hypergraph product* code is the

matrix H is a vector v of weight a such that the corresponding syndrome $s = Hv$ has weight b.

[17] The parity-check matrices are not necessary full rank.

CSS $[[N, K, d]]$ code with $H_X = (a \otimes I_{m_b}, I_{m_a} \otimes b)$ and $H_Z = (I_{n_a} \otimes b^{\mathrm{T}}, a^{\mathrm{T}} \otimes I_{n_b})$, where $N = n_a m_b + n_b m_a$, $K = 2k_a k_b - k_a(n_b - m_b) - k_b(n_a - m_a)$. As it was shown in [7], the minimum distance d of the hypergraph product code $\mathcal{C}$ satisfies the following lower bound:

$$d \geqslant \min(d_a, d_b, d_a^{\mathrm{T}}, d_b^{\mathrm{T}}),$$

where d_a^{T} and d_b^{T} are the minimal distances of the "transposed" codes C_a^{T} and C_b^{T} defined by the parity-check matrices a^{T} and b^{T} respectively. It is important to note that if the matrices a and b are w-limited, then the corresponding CSS code $\mathcal{C}$ is $2w$-limited. Hence, using known asymptotically good families of classical LDPC codes with (w_c, w_r)-limited parity check-matrices, it is possible [7] to construct w-limited CSS codes with asymptotically non-zero rate and $d = \Theta(\sqrt{n})$ as $n \to \infty$. In [9] the hypergraph product construction was further improved, and it was shown that one can construct good hypergraph product codes using square parity-check matrices a and b. In fact, many of the best-known small-length hypergraph product codes are constructed using square parity check matrices of cyclic codes (see [9, 10]). In [10] hyperbicycle CSS codes, which generalize both generalized bicycle and hypergraph product codes, were proposed. Here we consider another generalization of hypergraph product codes where the matrix b is square.

5.2 Generalized hypergraph product codes

In what follows by a *ring* we always mean a ring with identity. Let R be a ring. We denote the ring of all $m \times n$ matrices over R by $\mathcal{M}_{m\times n}(R)$ or by $\mathcal{M}_n(R)$ when $m = n$. If R is the ring of $\ell \times \ell$ matrices over some field $\mathbb{F}$ we identify the elements of $\mathcal{M}_{m\times n}(R)$ with the corresponding block matrices from $\mathcal{M}_{m\ell\times n\ell}(\mathbb{F})$.

Consider a binary matrix $b \in \mathcal{M}_\ell(\mathbb{F}_2)$. We say that the matrix b and a ring $R \subseteq \mathcal{M}_\ell(\mathbb{F}_2)$ *commute* if all matrices from R commute with b.

Example 5. Consider $b \in \mathcal{M}_\ell(\mathbb{F}_2)$ and $R = \{\mathbf{0}, I\}$, where $\mathbf{0}$ and I are the zero and the identity matrices from $\mathcal{M}_\ell(\mathbb{F}_2)$ respectively; then b and R always commute.

Example 6. Let b be a binary circulant matrix and R be the ring of all binary circulant matrices of the same size; then b and R always commute.

Suppose that a matrix $b \in \mathcal{M}_\ell(\mathbb{F}_2)$ and a ring $R \subseteq \mathcal{M}_\ell(\mathbb{F}_2)$ commute. Consider a matrix $A = (a_{ij})_{m\times n} \in \mathcal{M}_{m\times n}(R)$. We denote by $\mathcal{C}(A, b)$ the CSS code called a *generalized hypergraph product* (*GHP*) code with the following parity-check matrices[18]:

$$H_X = [A, bI_m], H_Z = [b^{\mathrm{T}} I_n, A^*], \tag{16}$$

[18]Let us warn the reader that we understand H_X and H_Z as the corresponding binary block matrices (not as matrices over $\mathcal{M}_\ell(\mathbb{F}_2)$).

where $A^* = (a_{ji}^T)_{n\times m}$, and I_m, I_n are the identity matrices over R of size m and n respectively. The correctness of this definition follows from the following:

$$H_X H_Z^{\mathrm{T}} = [A, bI_m] \begin{bmatrix} I_n b \\ A \end{bmatrix} = Ab + bA = \mathbf{0}.$$

The code length N of the CSS code $\mathcal{C}(A, b)$ is given by $N = (m + n)\ell$. We will show later how to find the dimension K of the code $\mathcal{C}(A, b)$ in a special case.

One can easily verify that if we take a matrix $b \in \mathcal{M}_\ell(\mathbb{F}_2)$ and the ring $R = \{\mathbf{0}, I\}$ (as in Example 5) then the CSS code $\mathcal{C}(A, b)$ is the hypergraph product code defined by the binary $m \times n$ matrix $\tilde{a} = (\tilde{a}_{ij})_{m\times n}$ and the binary $\ell \times \ell$ matrix b, where for all $i \in [m]$, $j \in [n]$ we have:

$$\tilde{a}_{ij} = \begin{cases} 0, & \text{if } a_{ij} = \mathbf{0}; \\ 1, & \text{if } a_{ij} = I. \end{cases}$$

We can also see that the ansatz with two commuting matrices A and B given in (12) is also a special case of the new ansatz described in (16), where the matrix A is a 1×1 block matrix.

5.3 Quasi-cyclic generalized hypergraph product codes

In this subsection, we describe quasi-cyclic (QC) GHP codes. In fact, it can be shown that this particular subclass of GHP codes is equivalent (up to some permutation of qubits) to a special case of hyperbicycle codes from [10, Eq. (19), $\chi = r_2 = n_2 = 1$]. However, we believe that the concise language of polynomial matrices adopted in the current work is much more suitable for our examples, which are essentially sparse random QC GHP codes subject to some additional constrains.

Now, let us take b and R as in Example 6. Hence b is a binary circulant matrix, and R is the ring of all binary circulant matrices of the same size. In this case, the matrices H_X and H_Z defined by (16) are block matrices, where each block is a binary circulant matrix of size ℓ. Such matrices are called *quasi-cyclic*. Let us note that quasi-cyclic (QC) matrices are well known in classical coding theory. In fact, most of the best-known practical classical LDPC codes have QC parity-check matrices. We will show in this section how to find the dimension of generalized hypergraph product codes defined by (16). For simplicity, we consider here only the case when the circulant size ℓ is odd. Before we can provide the formula for the dimension we need some supplementary definitions from algebra.

Here we adopt the polynomial representation of QC matrices used in [42, 43]. For any polynomial $p(x) \in \mathbb{F}_q[x]$ of degree d we consider the ring $\mathbb{F}_q[x]/(p(x))$ of polynomials $f_0 + f_1 x + \cdots + f_{d-1}x^{d-1} \in \mathbb{F}_q[x]$ with addition and multiplication modulo $p(x)$. By $\mathbb{F}_q^d$ we denote the d-dimensional space of the $d \times 1$ column vectors over $\mathbb{F}_q$. We also identify an element $f(x) \in \mathbb{F}_q[x]/(p(x))$ with the corresponding column vector $f = (f_0, \ldots, f_{d-1}) \in \mathbb{F}_q^d$.

Let us recall that by $\mathbb{F}_2^{\langle \ell \rangle}$ we denote the ring of circulants $\mathbb{F}_2[x]/(x^\ell - 1)$. We use the standard identification of the circulant $\ell \times \ell$ matrices over $\mathbb{F}_2$ with the elements of the ring $\mathbb{F}_2^{\langle \ell \rangle}$ (see Subsection 4.2), where a column vector $a \in \mathbb{F}_2^{\langle \ell \rangle}$ corresponds to the circulant matrix with the first column equal to a. Using this identification we can consider an $m\ell \times n\ell$ QC matrix over $\mathbb{F}_2$ of circulant size ℓ as an $m \times n$ matrix over the ring $\mathbb{F}_2^{\langle \ell \rangle}$. We also consider $n \times 1$ column vectors over $\mathbb{F}_2^{\langle \ell \rangle}$ as $n\ell \times 1$ column vectors over $\mathbb{F}_2$. Given the above identification we consider multiplication of an $m\ell \times n\ell$ QC matrix by an $n\ell \times 1$ column vector over $\mathbb{F}_2$ as multiplication of an $m \times n$ matrix by an $n \times 1$ column vector over $\mathbb{F}_2^{\langle \ell \rangle}$.

The algebraic structure of the ring $\mathbb{F}_2^{\langle \ell \rangle}$ is very well studied in the literature (see Appendix A for further details). Since we consider the case when ℓ is odd, the polynomial $x^\ell - 1$ factors into a product of irreducible polynomials over $\mathbb{F}_2$:

$$x^\ell - 1 = f_1(x) \cdots f_s(x). \tag{17}$$

Hence the ring $\mathbb{F}_2^{\langle \ell \rangle}$ is isomorphic to the direct product of finite fields:

$$\mathbb{F}_2^{\langle \ell \rangle} \cong F_1 \times \cdots \times F_s, \tag{18}$$

where the field $F_i = \mathbb{F}_2[x]/\big(f_i(x)\big)$ has the size 2^{d_i}; $d_i = \deg f_i(x)$, $i \in [s]$. Let us consider the maps $\varphi_i \colon \mathbb{F}_2^{\langle \ell \rangle} \to F_i$ given by the formula:

$$\varphi_i \colon u(x) \mapsto u(x) \bmod f_i(x), i \in [s].$$

We also naturally extend this map to any vectors and matrices over $\mathbb{F}_2^{\langle \ell \rangle}$. The following lemma is the key to the dimension formula for the QC generalized hypergraph product codes.

Lemma 1. *Let $A \in \mathcal{M}_{m\times n}(\mathbb{F}_2^{\langle \ell \rangle})$ be a binary QC matrix. Then its rank (over $\mathbb{F}_2$) is given by:*

$$\operatorname{rk}_{\mathbb{F}_2} A = \sum_{i=1}^{s} d_i \operatorname{rk}_{F_i} \varphi_i(A)$$

Proof. The lemma easily follows from the isomorphism shown in (18) between $\mathbb{F}_2^{\langle \ell \rangle}$ and the direct product of the fields $F_1, \ldots, F_s$. Indeed, let $r_i = \operatorname{rk}_{F_i} \varphi_i(A)$, $i \in [s]$. If $A_1, \ldots, A_n$ are the columns of A, then each vector v that belongs to the column space of the matrix A can be represented as follows:

$$v = u_1 A_1 + \cdots + u_n A_n,$$

where $u_1, \ldots, u_n \in \mathbb{F}_2^{\langle \ell \rangle}$. Therefore we see that the cardinality of the column space of the non-binary matrix $\varphi_i(A)$ over the field F_i is equal to $(2^{d_i})^{r_i} = 2^{d_i r_i}$. Hence using isomorphism (18) we conclude that the number of different vectors in the column space of the matrix A is equal to

$$\prod_{i=1}^{s} 2^{d_i r_i} = 2^{\sum_{i=1}^{s} d_i r_i},$$

and we proved that $\operatorname{rk}_{\mathbb{F}_2} A = \sum_{i=1}^{s} d_i r_i$. □

The following proposition provides the formula for the dimension of the QC CSS code $\mathcal{C}(A, b)$ if ℓ is odd.

Proposition 2. *Let $b(x) \in \mathbb{F}_2^{\langle \ell \rangle}$, $A \in \mathcal{M}_{m\times n}(\mathbb{F}_2^{\langle \ell \rangle})$, where ℓ is odd. Let $g(x) = \gcd(b(x), x^\ell - 1) = \prod_{i\in S} f_i(x)$, $S \subseteq [s]$, where $f_i(x)$ are some irreducible polynomials from (17), and $F_i = \mathbb{F}_2[x]/(f_i(x))$, $i \in S$, are the corresponding finite fields. Then $\mathcal{C}(A, b)$ is a CSS $[[N, K]]$ code, where $N = (m + n)\ell$ and*

$$K = \sum_{i\in S} \deg f_i(x)(m + n - 2 \operatorname{rk}_{F_i} \varphi_i(A)).$$

Proof. The proof idea is the following. Let us mention that $\varphi_i(H_X) = [\varphi_i(A), \mathbf{0}]$ for all $i \in S$. Hence we have $\operatorname{rk}_{F_i} \varphi_i(H_X) = \operatorname{rk}_{F_i} \varphi_i(A)$, for all $i \in S$. At the same time for $i \in [s] \setminus S$, we obtain

$$\operatorname{rk}_{F_i} \varphi_i(H_X) = \operatorname{rk}_{F_i} [\varphi_i(A), \varphi_i(b) I_m] = m,$$

since for all $i \in [s] \setminus S$ we have that $\varphi_i(b) \neq 0$ and therefore the non-binary matrix $\varphi_i(H_X)$ is full rank. Hence by applying Lemma 1 to the matrix H_X we have:

$$\begin{aligned} m\ell - \operatorname{rk} H_X &= \sum_{i=1}^{s} \deg f_i(x)(m - \operatorname{rk}_{F_i} \varphi_i(H_X)) \\ &= \sum_{i\in S} \deg f_i(x)(m - \operatorname{rk}_{F_i} \varphi_i(A)). \end{aligned}$$

To complete the proof we need also to find the rank of the matrix H_Z. It is easier to find the rank of the matrix H'_Z obtained from H_Z by the application of the "transpose" map $u \mapsto u^*$ to each element. Since the transpose map is an automorphism on the ring $\mathbb{F}_2^{\langle \ell \rangle}$ we see that the number of vectors in the row space of H_Z and the row space of H'_Z is the same. Hence $\operatorname{rk} H_Z = \operatorname{rk} H'_Z$. The rank of the matrix $H'_Z = [bI_n, A^T]$, can be found in the same way as for the matrix H_X:

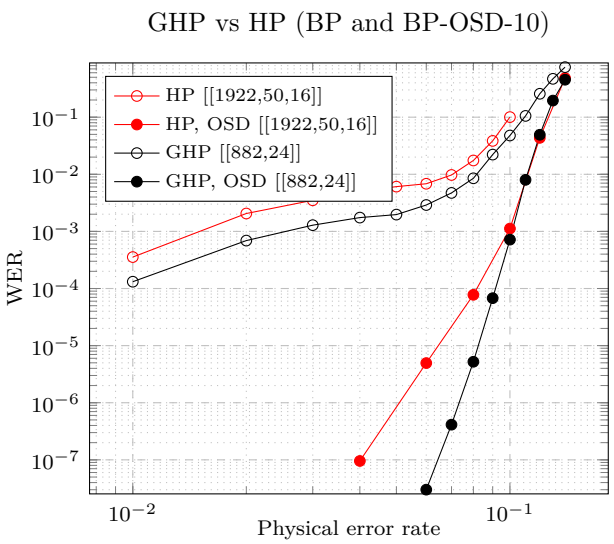

Figure 7: The WER performance of the 6-limited $[[1922, 50, 16]]$ hypergraph product (HP) code and the 6-limited $[[882, 24]]$ generalized hypergraph product (GHP) code (C2 and B1 in Appendix B). The HP code has an error floor even under the BP-OSD-10 decoder.

$$\begin{aligned} n\ell - \operatorname{rk} H'_Z &= \sum_{i=1}^{s} \deg f_i(x)(n - \operatorname{rk}_{F_i} \varphi_i(H'_Z)) \\ &= \sum_{i\in S} \deg f_i(x)(n - \operatorname{rk}_{F_i} \varphi_i(A^{\mathrm{T}})) \\ &= \sum_{i\in S} \deg f_i(x)(n - \operatorname{rk}_{F_i} \varphi_i(A)). \end{aligned}$$

Now we apply formula (6) for the CSS dimension and obtain:

$$\begin{aligned} K &= N - \operatorname{rk} H_X - \operatorname{rk} H_Z \\ &= (m\ell - \operatorname{rk} H_X) + (n\ell - \operatorname{rk} H'_Z) \\ &= \sum_{i\in S} \deg f_i(x)(m + n - 2\operatorname{rk}_{F_i} \varphi_i(A)). \end{aligned}$$

This concludes the proof. □

Let us note that if the polynomial $b(x)$ is such that $g(x) = \gcd(b(x), x^\ell - 1)$ is an irreducible factor of $x^\ell - 1$, then we can give a more elegant formula for the dimension of the code $\mathcal{C}(A, b)$ in some special cases. Indeed, in this case we have:

$$K = \deg g(x)(m + n - 2\operatorname{rk}_F \varphi(A)), \tag{19}$$

where F is the finite field $\mathbb{F}_2[x]/(g(x))$ and $\varphi(A)$ is the F-image of A under the action of the map $\varphi\colon u(x) \mapsto u(x) \bmod g(x)$. Since $\varphi(A)$ is a matrix over F, it defines, as a parity-check matrix, a non-binary linear code over F of dimension

$$k_A = n - \operatorname{rk}_F \varphi(A).$$

At the same time, $b(x)$ defines the cyclic code

$$\mathcal{C}_b = \mathcal{C}_g = \{g(x)u(x) \mid u(x) \in \mathbb{F}_2^{\langle \ell \rangle}\}$$

of dimension $k_b = \deg g(x)$ as a check polynomial. Thus formula (19) gives us the dimension K of $\mathcal{C}(A, b)$ in terms of the dimensions k_A and k_b for the two special cases shown below:

1. if A is a square matrix, i.e. $m = n$, we have
$$K = 2k_A k_b; \tag{20}$$
2. if A is a full rank matrix, i.e. $m = \operatorname{rk}_F \varphi(A)$, we have
$$K = k_A k_b = (n - m)k_b. \tag{21}$$

In the current work, we construct only codes that correspond to case 1 above. Such codes are defined by an irreducible factor $b(x)$ of $x^\ell - 1$ and a square matrix A over $\mathbb{F}_2^{\langle \ell \rangle}$. To construct all the examples of QC GHP codes given in Appendix B (codes B1, B2, and B3), we used the following procedure. First we fix a low weight irreducible polynomial $b(x) \in \mathbb{F}_2[x]$ such that $b(x) \mid x^\ell - 1$, and then randomly choose a polynomial matrix A subject to some constraints. Since we want to obtain sparse parity-check matrices H_X and H_Z for the code $\mathcal{C}(A, b)$, we restrict each entry of A to be either 0 or a monomial x^i, $0 \leqslant i < \ell$. When the number of non-zero entries in each row and each column of A is bounded above by w, this restriction guaranties that both H_X and H_Z are $(w + \deg b(x))$-limited matrices. Another restriction is related to the girth of the Tanner graphs $\mathcal{T}_X$, $\mathcal{T}_Z$ for H_X, H_Z. As you can see from Tab. 1 in Appendix B, the girth of the Tanner graphs $\mathcal{T}_X$, $\mathcal{T}_Z$ for all the constructed QC GHP codes is equal to 6, which means that $\mathcal{T}_X$, $\mathcal{T}_Z$ do not contain 4-cycles. This restriction helps to improve the error-correcting performance under the BP decoder.

Example 7. In Fig. 7 you can see the WER performance (under the BP-OSD decoder) of the two codes: the 6-limited [[1922,50,16]] hypergraph product (HP) code (C2 in Appendix B) and the 6-limited [[882,24]] generalized hypergraph product (GHP) code (B1 in Appendix B). You can see from Fig. 7 that the HP [[1922,50,16]] code has some error floor even under the BP-OSD decoder. We believe that this is due to a large amount of low-weight non-degenerate codewords in this code.

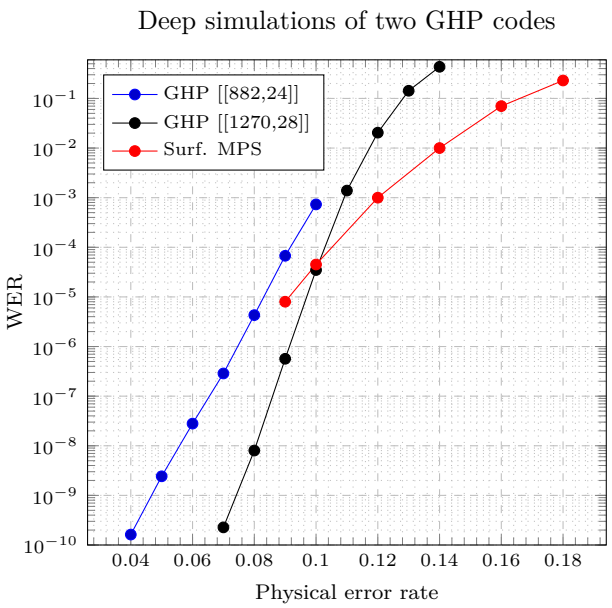

Figure 8: A deep simulation on the depolarizing channel of the 6-limited [[882,24]] and [[1270,28]] QC GHP codes (B1 and B3 in Appendix B) under the BP-OSD-0 decoder. These codes do not have error floor down to WER $= 10^{-10}$. The red curve is for the [[1201,1,25]] surface code on the MPS-based decoder from [5].

In Fig 8 we can see the WER performance of the 6-limited [[882,24]] and [[1270,28]] QC GHP codes (codes B1 and B3 in Appendix B) under the BP-OSD-0 decoder. We see that these codes do not have the error floor down to WER $= 10^{-10}$. The red curve shows the corresponding WER performance of the 4-limited [[1201,1,25]] surface code under the almost optimal MPS-based decoder from [5].

6 Conclusion

We proposed new OSD-like post-processing for the BP decoder that shows on some codes much better performance than all the modifications known to the authors. We also constructed several new generalized bicycle codes that show very good performance compared to the other known codes with similar parameters. We proposed a new ansatz for quantum CSS codes and showed how to estimate the dimension of such codes in some special cases. Unfortunately, we have not found any nontrivial general lower bound on the minimum distance of such codes. We think that to find such a bound is an interesting open problem[19] since this class contains some of the best known QLDPC codes, and their practical performance under the BP-OSD decoder is also quite good. Finally, we compared the performance of one of our codes from the new family and showed that it has better performance than a relatively large surface code of similar code length even if this code is decoded by a near-optimal decoder.

[19] Since the first version of this work appeared in arXiv, some lower bounds on the minimal distance have been obtained in [16, 17, 18] for special cases of QC GHP codes $\mathcal{C}(A, 1+x)$.

Acknowledgments

The authors would like to thank Dr. Xuecang Zhang from Huawei Technologies for the support of this work and for useful discussions. This work was also supported by the Ministry of Science and Higher Education of the Russian Federation (Grant № 075-15-2020-801).

References

[1] 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.

[2] Michael H. Freedman and David A. Meyer. Projective plane and planar quantum codes. *Foundations of Computational Mathematics*, 1(3):325–332, Jul 2001. doi:10.1007/s102080010013.

[3] H. Bombin and M. A. Martin-Delgado. Topological quantum distillation. *Phys. Rev. Lett.*, 97:180501, Oct 2006. doi:10.1103/PhysRevLett.97.180501.

[4] David S. Wang, Austin G. Fowler, and Lloyd C. L. Hollenberg. Surface code quantum computing with error rates over 1%. *Phys. Rev. A*, 83:020302, Feb 2011. doi:10.1103/PhysRevA.83.020302.

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

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

[7] J. Tillich and G. Zémor. Quantum LDPC codes with positive rate and minimum distance proportional to $n^{1/2}$. In *2009 IEEE International Symposium on Information Theory*, pages 799–803, June 2009. doi:10.1109/ISIT.2009.5205648.

[8] Sergey Bravyi and Matthew B. Hastings. Homological product codes. In *Proceedings of the*

Forty-sixth Annual ACM Symposium on Theory of Computing, STOC '14, pages 273–282, New York, NY, USA, 2014. ACM. doi:10.1145/2591796.2591870.

[9] A. A. Kovalev and L. P. Pryadko. Improved quantum hypergraph-product LDPC codes. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 348–352, July 2012. doi:10.1109/ISIT.2012.6284206.

[10] Alexey A. Kovalev and Leonid P. Pryadko. Quantum kronecker sum-product low-density parity-check codes with finite rate. *Phys. Rev. A*, 88:012311, Jul 2013. doi:10.1103/PhysRevA.88.012311.

[11] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo. Fifteen years of quantum LDPC coding and improved decoding strategies. *IEEE Access*, 3:2492–2519, 2015. doi:10.1109/ACCESS.2015.2503267.

[12] M. Hagiwara and H. Imai. Quantum quasi-cyclic LDPC codes. In *2007 IEEE International Symposium on Information Theory*, pages 806–810, June 2007. doi:10.1109/ISIT.2007.4557323.

[13] M. P. C. Fossorier and Shu Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, 41(5):1379–1396, Sep. 1995. doi:10.1109/18.412683.

[14] M. P. C. Fossorier. Iterative reliability-based decoding of low-density parity check codes. *IEEE Journal on Selected Areas in Communications*, 19(5):908–917, may 2001. doi:10.1109/49.924874.

[15] B. Dorsch. A decoding algorithm for binary block codes and j-ary output channels (corresp.). *IEEE Transactions on Information Theory*, 20(3):391–394, may 1974. doi:10.1109/TIT.1974.1055217.

[16] Matthew B. Hastings, Jeongwan Haah, and Ryan O'Donnell. Fiber bundle codes: Breaking the $n^{1/2}$ polylog(n) barrier for quantum LDPC codes. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1276–1288. Association for Computing Machinery, New York, NY, USA, June 2021. doi:10.1145/3406325.3451005.

[17] Pavel Panteleev and Gleb Kalachev. Quantum LDPC codes with almost linear minimum distance. *IEEE Transactions on Information Theory*, pages 1–1, 2021. doi:10.1109/TIT.2021.3119384.

[18] Nikolas P. Breuckmann and Jens N. Eberhardt. Balanced product quantum codes. *IEEE Transactions on Information Theory*, 67(10):6653–6674, October 2021. doi:10.1109/TIT.2021.3097347.

[19] Michael Freedman and Matthew Hastings. Building manifolds from quantum codes. *Geometric and Functional Analysis*, June 2021. doi:10.1007/s00039-021-00567-3.

[20] Jeongwan Haah. Local stabilizer codes in three dimensions without string logical operators. *Phys. Rev. A*, 83:042330, Apr 2011. doi:10.1103/PhysRevA.83.042330.

[21] D. Poulin and Yeojin C. On the iterative decoding of sparse quantum codes. *Quantum Info. Comput.*, 8(10):987–1000, November 2008. doi:10.5555/2016985.2016993.

[22] Y. Wang, B. C. Sanders, B. Bai, and X. Wang. Enhanced feedback iterative decoding of sparse quantum codes. *IEEE Transactions on Information Theory*, 58(2):1231–1241, Feb 2012. doi:10.1109/TIT.2011.2169534.

[23] Alex Rigby, J. C. Olivier, and Peter Jarvis. Modified belief propagation decoders for quantum low-density parity-check codes. *Physical Review A*, 100(1):012330, July 2019. doi:10.1103/PhysRevA.100.012330.

[24] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, 1997. doi:10.7907/rzr7-dt72.

[25] A. R. Calderbank and Peter W. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54:1098–1105, Aug 1996. doi:10.1103/PhysRevA.54.1098.

[26] A. M. Steane. Error correcting codes in quantum theory. *Phys. Rev. Lett.*, 77:793–797, Jul 1996. doi:10.1103/PhysRevLett.77.793.

[27] R. G. Gallager. *Low-Density Parity-Check Codes*. M.I.T. Press, Cambridge, MA, 1963. doi:10.7551/mitpress/4347.001.0001.

[28] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, September 1981. doi:10.1109/TIT.1981.1056404.

[29] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001. doi:10.1109/18.910572.

[30] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, September 1962. doi:10.1109/TIT.1962.1057777.

[31] Jack Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, December 1971. doi:10.1007/BF01584082.

[32] M.P.C. Fossorier, Shu Lin, and J. Snyders. Reliability-based syndrome decoding of linear block codes. *IEEE Transactions on Information Theory*, 44(1):388–398, January 1998. doi:10.1109/18.651070.

[33] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correcting code. *Physical Review Letters*, 77(1):198–201, July 1996. doi:10.1103/PhysRevLett.77.198.

[34] Kao-Yueh Kuo and Ching-Yi Lai. Refined belief propagation decoding of sparse-graph quantum codes. *IEEE Journal on Selected Areas in Information Theory*, 1(2):487–498, August 2020. doi:10.1109/JSAIT.2020.3011758.

[35] Marc Fossorier, David Declercq, and Ezio Biglieri. *Channel Coding: Theory, Algorithms, and Applications*. Academic Press, July 2014. doi:10.1016/C2011-0-07211-3.

[36] Jack Edmonds. Paths, Trees, and Flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965/ed. doi:10.4153/CJM-1965-045-4.

[37] D. J. C. MacKay, G. Mitchison, and P. L. McFadden. Sparse-graph codes for quantum error correction. *IEEE Transactions on Information Theory*, 50(10):2315–2330, Oct 2004. doi:10.1109/TIT.2004.834737.

[38] H Bombin, R W Chhajlany, M Horodecki, and M A Martin-Delgado. Self-correcting quantum computers. *New Journal of Physics*, 15(5):055023, may 2013. doi:10.1088/1367-2630/15/5/055023.

[39] Yuichiro Fujiwara. Ability of stabilizer quantum error correction to protect itself from its own imperfection. *Phys. Rev. A*, 90:062304, Dec 2014. doi:10.1103/PhysRevA.90.062304.

[40] Tom Richardson. Error-floors of LDPC codes. In *Proceedings of the 41st Annual Conference on Communication, Control and Computing*, pages 1426–1435, 2003.

[41] Ye-Hua Liu and David Poulin. Neural belief-propagation decoders for quantum error-correcting codes. *Physical Review Letters*, 122(20):200501, May 2019. doi:10.1103/PhysRevLett.122.200501.

[42] Kristine Lally and Patrick Fitzpatrick. Algebraic structure of quasicyclic codes. *Discrete Applied Mathematics*, 111(1):157–175, July 2001. doi:10.1016/S0166-218X(00)00350-4.

[43] Roxana Smarandache and Pascal O. Vontobel. Quasi-cyclic LDPC codes: Influence of proto- and tanner-graph structure on minimum hamming distance upper bounds. *IEEE Transactions on Information Theory*, 58(2):585–607, February 2012. doi:10.1109/TIT.2011.2173244.

[44] San Ling, Harald Niederreiter, and Patrick Solé. On the algebraic structure of quasi-cyclic codes IV: Repeated roots. *Designs, Codes and Cryptography*, 38:337–361, 2006. doi:10.1007/s10623-005-1431-7.

[45] I. Dumer, A. A. Kovalev, and L. P. Pryadko. Distance verification for classical and quantum LDPC codes. *IEEE Transactions on Information Theory*, 63(7):4675–4686, 2017. doi:10.1109/TIT.2017.2690381.

A Algebraic structure of the ring $\mathbb{F}_q^{\langle \ell \rangle}$

Let $\mathbb{F}_q$ be a finite field of characteristics 2. The algebraic structure of the ring $\mathbb{F}_q^{\langle \ell \rangle}$ is well studied in the coding literature (see, e.g., [44]). Below we briefly review it.

First, let us consider the special case when ℓ is odd. In this case the polynomial $x^\ell - 1$ factors into a product of different irreducible polynomials over $\mathbb{F}_q$

$$x^\ell - 1 = f_1(x) \cdots f_s(x). \tag{22}$$

This is true, since

$$\gcd\left((x^\ell - 1)', x^\ell - 1\right) = \gcd\left(\ell x^{\ell-1}, x^\ell - 1\right) = 1,$$

and the polynomial $x^\ell - 1$ is square-free.

In the general case, we have $\ell = 2^e \ell'$, where ℓ' is odd. Hence it follows that

$$x^\ell - 1 = x^{2^e \ell'} - 1 = (x^{\ell'} - 1)^{2^e}.$$

Moreover, since ℓ' is odd, we can apply factorization (22) to the polynomial $x^{\ell'} - 1$ and obtain that

$$x^\ell - 1 = \big(f_1(x)\big)^{2^e} \cdots \big(f_s(x)\big)^{2^e}. \tag{23}$$

Since the polynomials $\big(f_1(x)\big)^{2^e}, \ldots, \big(f_s(x)\big)^{2^e}$ are pairwise coprime, from the Chinese remainder theorem it follows that the ring $\mathbb{F}_q^{\langle \ell \rangle}$ is isomorphic to the direct product

$$R_1 \times \cdots \times R_s \tag{24}$$

of the rings $R_i = \mathbb{F}_q[x]/\big(f_i(x)\big)^{2^e}$, $i \in [s]$.

When ℓ is odd we have $e = 0$ and the rings $R_1, \ldots, R_s$ are in fact fields, since the polynomials $f_1(x), \ldots, f_s(x)$ are irreducible over $\mathbb{F}_q$.

B Matrices used for simulations

All matrices for generalized hypergraph product and generalized bicycle codes that we used for simulations have the form $H_X = [A, B]$, $H_Z = [B^T, A^T]$ where A and B are quasi-cyclic matrices. Thus, to define the code, we specify matrices A and B in the polynomial form as matrices over $\mathbb{F}_2^{\langle \ell \rangle}$.

For all codes presented here, we also provide lower and upper bounds on the minimum distance obtained either by methods similar to the ones from [45] or by a straightforward reduction of the minimum distance problem to a mixed integer linear program and using the GNU Linear Programming Kit, Version 4.63, http://www.gnu.org/software/glpk/glpk.html.

A. **Generalized bicycle (GB) codes.** The matrices A and B have form $A = (a(x))$, $B = (b(x))$, so here we specify the polynomials $a(x)$, $b(x)$, and the circulant size ℓ.

A1) $[[254, 28, d]]$ code ($\ell = 127$), $14 \leqslant d \leqslant 20$.

$$a(x) = 1 + x^{15} + x^{20} + x^{28} + x^{66},$$
$$b(x) = 1 + x^{58} + x^{59} + x^{100} + x^{121}.$$

A2) $[[126, 28, 8]]$ code ($\ell = 63$).

$$a(x) = 1 + x + x^{14} + x^{16} + x^{22},$$
$$b(x) = 1 + x^3 + x^{13} + x^{20} + x^{42}.$$

A3) $[[48, 6, 8]]$ code ($\ell = 24$).

$$a(x) = 1 + x^2 + x^8 + x^{15},$$
$$b(x) = 1 + x^2 + x^{12} + x^{17}.$$

A4) $[[46, 2, 9]]$ code ($\ell = 23$).

$$a(x) = 1 + x^5 + x^8 + x^{12},$$
$$b(x) = 1 + x + x^5 + x^7.$$

A5) $[[180, 10, d]]$ code ($\ell = 90$), $15 \leqslant d \leqslant 18$.

$$a(x) = 1 + x^{28} + x^{80} + x^{89},$$
$$b(x) = 1 + x^2 + x^{21} + x^{25}.$$

A6) $[[900, 50, 15]]$ code ($\ell = 450$).

$$a(x) = 1 + x^{97} + x^{372} + x^{425},$$
$$b(x) = 1 + x^{50} + x^{265} + x^{390}.$$

B. **Generalized hypergraph product (GHP) codes.** Here the matrix B is diagonal; $B = b(x)I_n$, where I_n is the $n \times n$ identity matrix over the ring $\mathbb{F}_2^{\langle \ell \rangle}$.

B1) $[[882, 24, d]]$ code, $18 \leqslant d \leqslant 24$. The matrices H_X and H_Z are (3,6)-regular ($\ell = 63$).

$$A = \begin{pmatrix} x^{27} & 0 & 0 & 0 & 0 & 1 & x^{54} \\ x^{54} & x^{27} & 0 & 0 & 0 & 0 & 1 \\ 1 & x^{54} & x^{27} & 0 & 0 & 0 & 0 \\ 0 & 1 & x^{54} & x^{27} & 0 & 0 & 0 \\ 0 & 0 & 1 & x^{54} & x^{27} & 0 & 0 \\ 0 & 0 & 0 & 1 & x^{54} & x^{27} & 0 \\ 0 & 0 & 0 & 0 & 1 & x^{54} & x^{27} \end{pmatrix},$$
$$B = (1 + x + x^6)I_7.$$

B2) $[[882, 48, 16]]$ code. Half of the columns for both H_X and H_Z matrices have weight 3, another half have weight 5. All the rows have weight 8 ($\ell = 63$).

$$A = \begin{pmatrix} x^{27} & 0 & 0 & 1 & x^{18} & x^{27} & 1 \\ 1 & x^{27} & 0 & 0 & 1 & x^{18} & x^{27} \\ x^{27} & 1 & x^{27} & 0 & 0 & 1 & x^{18} \\ x^{18} & x^{27} & 1 & x^{27} & 0 & 0 & 1 \\ 1 & x^{18} & x^{27} & 1 & x^{27} & 0 & 0 \\ 0 & 1 & x^{18} & x^{27} & 1 & x^{27} & 0 \\ 0 & 0 & 1 & x^{18} & x^{27} & 1 & x^{27} \end{pmatrix},$$
$$B = (1 + x + x^6)I_7.$$

B3) $[[1270, 28, d]]$ code, $16 \leqslant d \leqslant 46$. The matrices H_X and H_Z are (3,6)-regular ($\ell = 127$).

$$A = \begin{pmatrix} 1 & 0 & x^{51} & x^{52} & 0 \\ 0 & 1 & 0 & x^{111} & x^{20} \\ 1 & 0 & x^{98} & 0 & x^{122} \\ 1 & x^{80} & 0 & x^{119} & 0 \\ 0 & 1 & x^5 & 0 & x^{106} \end{pmatrix},$$
$$B = (1 + x + x^7)I_5.$$

C. **Hypergraph product (HP) codes.** Each hypergraph product code in our simulations is constructed from a single cyclic code defined by its parity polynomial $h(x)$ and the length ℓ.

C1) $[[7938, 578, 16]]$ code. The matrices H_X and H_Z are (5,10)-regular, and we have:

$$\ell = 63, \quad h(x) = 1 + x^3 + x^{34} + x^{41} + x^{57}.$$

C2) $[[1922, 50, 16]]$ code. The matrices H_X and H_Z are (3,6)-regular, and we have:

$$\ell = 31, \quad h(x) = 1 + x^2 + x^5.$$

D. **Haah's cubic codes.** We used the $[[1024, 30, d]]$ Haah's cubic code on the $8 \times 8 \times 8$ lattice from [20, Code 1], $13 \leqslant d \leqslant 32$.

E. **Hyperbicycle (HB) codes.** We used the $[[900, 50, 14]]$ hyperbicycle code from [9, Example 8], [10, Example 6].

F. **Homological product (HMP) codes.** We used one of the randomly constructed $[[49, 1, 9]]$ homological product codes from [8].

Table 1: Codes parameters

Code	N	K	d	rate	w_r	w_c	girth
A1	254	28	14–20	0.110	10	5	6
A2	126	28	8	0.222	10	5	4
A3	48	6	8	0.125	8	4	4
A4	46	2	9	0.043	8	4	4
A5	180	10	15–18	0.056	8	4	6
A6	900	50	15	0.056	8	4	6
B1	882	24	18–24	0.027	6	3	6
B2	882	48	16	0.054	8	3,5	6
B3	1270	28	16–46	0.022	6	3	6
C1	7938	578	16	0.073	10	5	6
C2	1922	50	16	0.026	6	3	6
D	1024	30	13–32	0.029	8	4	4
E	900	50	14	0.056	8	4	4
F	49	1	9	0.020	6,8	6,8	4

C Additional Simulations

In this appendix, we show some additional simulation results of the BP-OSD decoder on the depolarizing channel.

In Fig. 9, we demonstrate the WER performance of several 6-limited generalized bicycle (GB) codes with the parameters $[[2^{s+1}-2, 2s]]$, $s \in \mathbb{N}$, under the BP-OSD-0 decoder. As one can see from the curves, these codes have the threshold $p_{\mathrm{GB}} \approx 15\%$, which is quite close to the corresponding threshold $p_{\mathrm{S}} \approx 18\%$ of the surface codes from [5]. The codes from this family were constructed in the same way as the GB codes from Subsection 4.6, where for each $s \in \mathbb{N}$ the corresponding syndrome code $\mathcal{C}_g$ is the cyclic Hamming $[2^s-1, s, 3]$ code defined by some irreducible polynomial $g(x) \in \mathbb{F}_2[x]$ of degree s.

In all our previous examples, we considered only CSS codes. In fact, as it is shown in Subsection 3.3, the BP-OSD decoder can be applied to any stabilizer code. In Fig. 10, we demonstrate the WER performance of the BP-OSD-0 decoder on the 5-limited cyclic non-CSS [[126,2,12]] code. This code is defined by the parity-check matrix $H = [H_X \mid H_Z]$, where H_X and H_Z are the 126×126 circulant matrices represented respectively by the polynomials: $1 + x^{71} + x^{55}$ and $1 + x^{40} + x^{86}$. As we can see, the WER gain of OSD-0 post-processing in this case is at least three orders of magnitude.

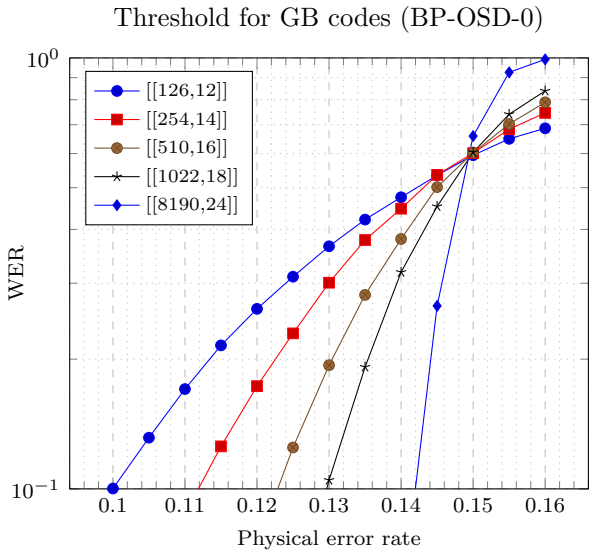

Figure 9: The WER performance on the depolarizing channel for five 6-limited GB codes under the BP-OSD-0 decoder. The threshold is $p_{\mathrm{GB}} \approx 15\%$. The corresponding threshold of the 4-limited surface codes [5] is $p_{\mathrm{S}} \approx 18\%$.

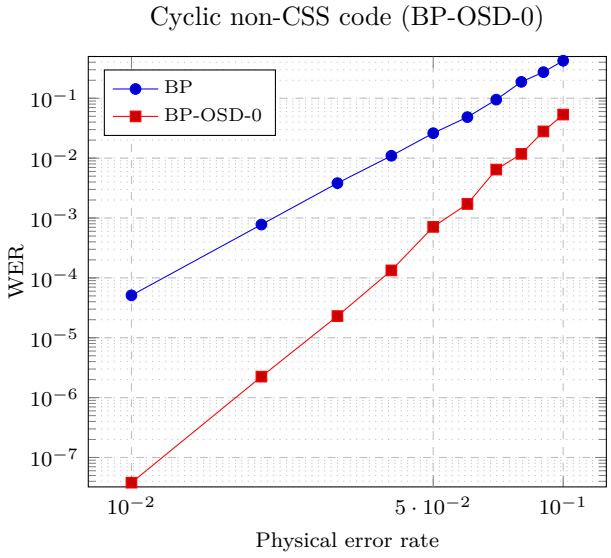

Figure 10: The WER performance of the BP-OSD-0 decoder on the 5-limited cyclic non-CSS [[126,2,12]] code defined by the parity-check matrix $H = [H_X \mid H_Z]$, where H_X and H_Z are the 126×126 circulant matrices represented respectively by the polynomials: $1 + x^{71} + x^{55}$ and $1 + x^{40} + x^{86}$.