

Belief Propagation Decoding of Quantum LDPC Codes with Guided Decimation

Hanwen Yao^{1,2}, Waleed Abu Laban³, Christian Häger³, Alexandre Graell i Amat³, and Henry D. Pfister^{1,2,4}

¹Duke Quantum Center, Duke University, Durham, NC 27701, USA

²Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA

³Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden

⁴Department of Mathematics, Duke University, Durham, NC, USA

Quantum low-density parity-check (QLDPC) codes have emerged as a promising technique for quantum error correction. A variety of decoders have been proposed for QLDPC codes and many of them utilize belief propagation (BP) decoding in some fashion. However, the use of BP decoding for degenerate QLDPC codes is known to have issues with convergence. These issues are typically attributed to short cycles in the Tanner graph and code degeneracy (i.e. multiple error patterns with the same syndrome). Although various methods have been proposed to mitigate the non-convergence issue, such as BP with ordered statistics decoding (BP-OSD) and BP with stabilizer inactivation (BP-SI), achieving better performance with lower complexity remains an active area of research.

In this work, we propose a decoder for QLDPC codes based on BP guided decimation (BPGD), which has been previously studied for constraint satisfaction and lossy compression problems. The decimation process is applicable to both binary and quaternary BP and it involves sequentially fixing the value of the most reliable qubits to encourage BP convergence. Despite its simplicity, We find that BPGD significantly reduces the BP failure rate due to non-convergence, achieving perfor-

mance on par with BP with ordered statistics decoding and BP with stabilizer inactivation, without the need to solve systems of linear equations.

1 Introduction

In a quantum computing system, error correction is an essential building block to protect fragile quantum information against decoherence and other noise sources. The general framework of quantum stabilizer codes has been studied extensively [1–3]. Using this framework, various quantum error-correcting codes have been constructed over the past two decades including toric codes [4, 5], surface codes [6–9], and various quantum low-density parity-check (QLDPC) codes [10–17]. Among them, QLDPC codes provide a promising direction because they support multiple logical qubits and their low-weight stabilizers allow reliable syndrome measurement in practice. Until recently, researchers did not know how to construct QLDPC codes with constant rate and linear distance. But, in a series of recent advances [18–21], researchers have overcome this obstacle and there are now constructions of asymptotically *good* quantum LDPC codes with constant rate and linear minimum distance.

For classical communication, low-density parity-check (LDPC) codes were first proposed by Gallager [22] in 1962 and are now widely applied in practical communication systems [23, 24]. LDPC codes are typically decoded with the belief propagation (BP) algorithm [25], which has low complexity and can provide good performance for code rates close to the channel capacity [26–29]. However, in the quantum sce-

Hanwen Yao: hanwen.yao@duke.edu

Waleed Abu Laban: laban@student.chalmers.se

Christian Häger: christian.haeger@chalmers.se

Alexandre Graell i Amat: alexandre.graell@chalmers.se

Henry D. Pfister: henry.pfister@duke.edu

nario, the performance of BP decoding based on syndrome measurement is hindered by cycles in the Tanner graph and code degeneracy [30–32]. The first problem follows from the commutativity constraint imposed on stabilizer codes which produces unavoidable cycles in the Tanner graph that degrade the BP decoding performance [30, 31]. The second is because good QLDPC codes are recognized to be highly *degenerate*; this means that their minimum distance is much larger than the minimum weight of their stabilizers. Due to degeneracy, the syndrome (i.e., stabilizer measurements) can be used to identify a correction procedure that works with high probability even when the actual error is not uniquely identified. This hinders convergence of the BP decoding process [30, 32]. Since the initial application of BP to decode QLDPC codes by Poulin and Chung [30], significant efforts have been made to enhance its performance. Various methods have been proposed to modify the BP decoding process itself, including random perturbation [30], enhanced feedback [33], grouping check nodes as super nodes [31], parity-check matrix augmentation [34], neural BP [35–37], generalized BP [38], adaptive BP with memory [39], and BP with trapping set dynamics [40]. Alternatively, post-processing methods have also been explored to improve performance. In [15], when BP fails to converge, it was proposed to use ordered statistics decoding (OSD) to construct a syndrome-matching error pattern based on the soft information provided by BP. This is called the BP-OSD algorithm. Another post-processing approach introduced in [41] involves iteratively running BP with stabilizer inactivation (BP-SI).

In this work, we improve the BP decoding performance for QLDPC codes by combining it with guided decimation. The term “decimation” refers to the process of sequentially fixing variables to hard decisions during iterative decoding [42, 43]. To motivate BP guided decimation (BPGD), we propose a sampling decoding approach for QLDPC codes which samples an error pattern based on their posterior probabilities. Then we explain how BPGD can be used to approximate the sampling decoder. In the proposed BPGD algorithm, we incorporate the decimation method by iteratively fixing the most reliable qubit based on its marginals estimated by BP. Despite its simplicity, we show that BPGD achieves

performance on par with both order-0 BP-OSD and BP-SI. Notably, BPGD exhibits lower complexity than BP-OSD and comparable complexity to BP-SI, without the need to solve any systems of linear equations. Additionally, through a randomized version of BPGD, we shed light on how code degeneracy contributes to the non-convergence issue in syndrome BP decoding over QLDPC codes. Our experiment also shows that guided decimation improves BP convergence, and highlights how BPGD benefits from code degeneracy. Furthermore, we extend BPGD from binary symbols to quaternary symbols, demonstrating competitive performance compared to BP-OSD in the high error rate regime over depolarizing noise.

2 Preliminaries

2.1 Binary Linear Codes

Let $\mathbb{F}_2 = \{0, 1\}$ be the binary Galois field defined by modulo-2 addition and multiplication. A length- n binary linear code $\mathcal{C} \subseteq \mathbb{F}_2^n$ is a subset of length- n binary strings satisfying $w+x \in \mathcal{C}$ for all $w, x \in \mathcal{C}$. Such a code forms a vector space over \mathbb{F}_2 . A generator matrix $G \in \mathbb{F}_2^{k \times n}$ for \mathcal{C} is a $k \times n$ matrix whose rows span the code. A parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ for \mathcal{C} is an $(n-k) \times n$ matrix whose rows are orthogonal to the code.

For classical communication where the codeword $x \in \mathcal{C}$ is corrupted by an additive error vector $z \in \mathbb{F}_2^n$, the received vector is given by $y = x + z$. Then, the optimal recovery process for x given y is choosing a codeword \hat{x} that maximizes the conditional probability given y . Since $Hx = 0$, it follows that the syndrome vector $s \triangleq Hy \in \mathbb{F}_2^{n-k}$ satisfies

$$s = H(x + z) = Hz. \quad (1)$$

For the binary symmetric channel (BSC) where z is an i.i.d. vector, the optimal recovery process can also be implemented by *syndrome decoding* where the syndrome s is mapped to the minimum-weight error vector in the coset $\{u \in \mathbb{F}_2^n \mid Hu = s\}$ containing z . For more details on this well-known classical setup, see [44].

2.2 Stabilizer Formalism

An $[[n, k]]$ quantum stabilizer code is an error correction code designed to protect k logical qubits

with n physical qubits against noise. We first define the Pauli operators to establish the definition of quantum stabilizer codes. For a single qubit, its pure quantum state is represented as a unit vector in the two-dimensional Hilbert space \mathbb{C}_2 . The Pauli operators for a single qubit system are defined as the 2×2 complex Hermitian matrices

$$\begin{aligned} I &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \\ \sigma_z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \sigma_y = i\sigma_x\sigma_z = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \end{aligned} \quad (2)$$

where $i = \sqrt{-1}$, and they form a basis for all 2×2 complex matrices. For an n -qubit system, we are working in the n -fold Kronecker product of the two-dimensional Hilbert space $\mathbb{C}_2^{\otimes n}$. Given two length- n binary vectors $a = (a_1, a_2, \dots, a_n) \in \mathbb{F}_2^n$, and $b = (b_1, b_2, \dots, b_n) \in \mathbb{F}_2^n$, we define the n -fold Pauli operator $D(a, b)$ as

$$D(a, b) = \sigma_x^{a_1} \sigma_z^{b_1} \otimes \sigma_x^{a_2} \sigma_z^{b_2} \otimes \dots \otimes \sigma_x^{a_n} \sigma_z^{b_n}. \quad (3)$$

It follows that the Pauli operators $i^k D(a, b)$ with $a, b \in \mathbb{F}_2^n$ and an overall phase i^k with $k \in \{0, 1, 2, 3\}$ form the n -qubit Pauli group, denoted by \mathcal{P}_n , with the multiplication rule

$$\begin{aligned} D(a, b)D(a', b') &= (-1)^{a'b^T} D(a + a', b + b') \\ &= (-1)^{a'b^T + b'a^T} D(a', b')D(a, b). \end{aligned} \quad (4)$$

The *symplectic inner product* between length- $2n$ binary vectors (a, b) and (a', b') is defined by

$$\begin{aligned} \langle (a, b), (a', b') \rangle_s &\triangleq (a', b')\Lambda(a, b)^T \bmod 2 \\ &= b'a^T + a'b^T \bmod 2, \end{aligned} \quad (5)$$

where

$$\Lambda = \begin{bmatrix} 0 & I_n \\ I_n & 0 \end{bmatrix}. \quad (6)$$

This equals 0 if the two Pauli operators $D(a, b)$ and $D(a', b')$ commute and 1 if they anti-commute.

A *quantum stabilizer code* \mathcal{C} with n physical qubits is defined by a commutative subgroup $\mathcal{S} \subseteq \mathcal{P}_n$ with $-I_2^{\otimes n} \notin \mathcal{S}$. The subgroup \mathcal{S} is referred to as the *stabilizer group*, and the Pauli operators in \mathcal{S} are called the *stabilizers*. The code space consists of all states in $\mathbb{C}_2^{\otimes n}$ stabilized by \mathcal{S} :

$$\mathcal{C} = \{|\psi\rangle \in \mathbb{C}_2^{\otimes n} : M|\psi\rangle = |\psi\rangle, \forall M \in \mathcal{S}\}. \quad (7)$$

In other words, \mathcal{C} consists of states that are $+1$ eigenstates of all the stabilizers in \mathcal{S} . If \mathcal{S} has $n - k$ independent generators, the code space has dimension k .

The *weight* of a Pauli operator in \mathcal{P}_n is defined to be the number of elements in its n -fold Kronecker product that are not equal to I . The *distance* of a stabilizer code \mathcal{C} is defined as the minimum weight of all Pauli operators in $N(\mathcal{S}) \setminus \mathcal{S}$, where $N(\mathcal{S})$ denotes the normalizer group of \mathcal{S} in \mathcal{P} . If code \mathcal{C} has distance d , we call \mathcal{C} an $[[n, k, d]]$ quantum stabilizer code. In particular, \mathcal{C} is called *degenerate* if its distance d is larger than the minimum weight of its stabilizers.

In the symplectic representation, the stabilizer group \mathcal{S} is constructed from the rows of the stabilizer matrix

$$H = [H_x, H_z], \quad (8)$$

where $H_x, H_z \in \mathbb{F}_2^{m \times n}$ are binary matrices with m rows and n columns. In particular, each row (h_x, h_z) of H defines the stabilizer $D(h_x, h_z)$ and the set of stabilizers defined by all rows generates the stabilizer group \mathcal{S} . In this way, the constraint requiring all the stabilizers in H to commute with each other can be expressed as

$$H\Lambda H^T = H_x H_z^T + H_z H_x^T = 0. \quad (9)$$

There is an important class of stabilizer codes, known as Calderbank–Shor–Steane (CSS) codes [45, 46], where each stabilizer has the form $D(a, 0)$ (i.e., only σ_x operators) or $D(0, b)$ (i.e., only σ_z operators). In this case, following the convention in [45], we have

$$H_x = \begin{bmatrix} 0 \\ G_2 \end{bmatrix}, \quad H_z = \begin{bmatrix} H_1 \\ 0 \end{bmatrix}, \quad (10)$$

where $H_1 \in \mathbb{F}_2^{(n-k_1) \times n}$ is the parity-check matrix of a classical $[n, k_1]$ code \mathcal{C}_1 and $G_2 \in \mathbb{F}_2^{k_2 \times n}$ is the generator matrix of a classical $[n, k_2]$ code \mathcal{C}_2 . Thus, the stabilizer matrix has the form

$$H = \begin{bmatrix} 0 & H_1 \\ G_2 & 0 \end{bmatrix}. \quad (11)$$

For CSS codes, the commutativity constraint for the stabilizers given by $H\Lambda H^T = 0$ reduces to $G_2 H_1^T = 0$, which is equivalent to $\mathcal{C}_2 \subseteq \mathcal{C}_1$. Throughout this work, we will focus our discussion on the CSS codes.

2.3 Syndrome Decoding of Stabilizer Codes

In this work, we consider two types of code capacity noise models: the bit-flip noise model and the depolarizing noise model. In both of these models, the encoded state $|\psi\rangle$ of an $[[n, k]]$ stabilizer code \mathcal{C} is corrupted by an n -qubit Pauli error $E = D(x, z) \in \mathcal{P}_n$ as $|\psi\rangle \rightarrow E|\psi\rangle$. The goal of the decoder is to detect and correct this error by conducting measurements on all the stabilizers in the parity-check matrix H .

Measuring the stabilizer $M = D(a, b) \in \mathcal{S}$ reveals the symplectic inner product $\langle (a, b), (x, z) \rangle_s$, indicating whether E commutes or anti-commutes with M . When measuring all the stabilizers in H , the result can be expressed as a length- m binary syndrome vector

$$s = (x, z)\Lambda H^T. \quad (12)$$

In this work, we assume that all syndrome measurements are uncorrupted (i.e. do not contain errors).

For CSS codes, H_1 in H_x defines σ_z -stabilizers that interact with Pauli- σ_x errors and G_2 in H_z defines σ_x -stabilizers that interact with Pauli- σ_z errors. Thus, we can separate the length $m = n - k_1 + k_2$ syndrome vector into two parts $s = (s_x, s_z)$ where $s_x = xH_1^T$ has length $n - k_1$ and $s_z = zG_2^T$ has length k_2 .

Based on its syndrome, the Pauli error E that affects the qubits can be categorized as follows:

1. **Detectable Error:** E is called *detectable* if its syndrome s is not the all-zero vector. In other words, E is detectable if it anticommutes with at least one of the stabilizers in \mathcal{S} . Otherwise, E is called *undetectable*.
2. **Degenerate Error:** If E is undetectable, it is called *degenerate* if it belongs to the stabilizer group, i.e., $E \in \mathcal{S}$. In this case, E preserves the encoded state and needs no correction.
3. **Logical Error:** If E is undetectable and it does not belong to the stabilizer group, i.e., $E \in N(\mathcal{S}) \setminus \mathcal{S}$, it is called a *logical* error, and it alters the logical state of the encoded qubits.

After obtaining the syndrome s through measurement, the decoder aims to find an estimated \hat{E} with high posterior probability yielding this

syndrome. Then, a reverse operator \hat{E}^\dagger , which in our case is equal to \hat{E} since \hat{E} is Pauli, can be applied to the affected qubits as $E|\psi\rangle \rightarrow \hat{E}E|\psi\rangle$ for error correction. This decoding process has the following four possible outcomes:

1. **Failure:** The decoder fails to provide an estimated \hat{E} that yields the syndrome s .
2. **Successful (Exact Match):** The estimated error is equal to the channel error, i.e., $\hat{E} = E$.
3. **Successful (Degenerate Error):** The difference between the estimated error and the channel error is a degenerate error, i.e., $\hat{E}E \in \mathcal{S}$.
4. **Failure (Logical Error):** The difference between the estimated error and the channel error is a logical error, i.e., $\hat{E}E \in N(\mathcal{S}) \setminus \mathcal{S}$.

Since there are 2^m possible syndromes, using a look-up table for error estimation quickly becomes infeasible as the number of qubits increases because the number of stabilizers m in H typically scales linearly with n . Thus, a computationally efficient decoder is required for long stabilizer codes.

Using the terminology from [47], here we describe two decoding strategies for stabilizer codes.

Quantum Maximum Likelihood (QML) Decoding This decoding strategy aims at finding the most probable error $E = D(X, Z)$ given the syndrome, where $X, Z \in \mathbb{F}_2^n$ are random vectors drawn from some Pauli error distribution. In this model, the syndrome is a random vector defined by $S = (X, Z)\Lambda H^T$. Given the observed syndrome event $S = s$, the decoder aims to find $\hat{E} = D(\hat{X}, \hat{Z})$ where

$$(\hat{X}, \hat{Z}) = \arg \max_{(x', z') \in \mathbb{F}_2^n \times \mathbb{F}_2^n} \Pr((X, Z) = (x', z') | S = s). \quad (13)$$

For both the bit-flip noise model and the depolarizing noise model, solving (13) is at least as hard as the maximum-likelihood decoding problem in classical coding theory, which is well-known to be NP-complete [48].

Degenerate Quantum Maximum Likelihood (DQML) Decoding Denote the coset of the stabilizer group \mathcal{S} shifted by a Pauli error $E \in \mathcal{P}_n$ as $E\mathcal{S}$, and denote the quotient group of all cosets of \mathcal{S} in \mathcal{P}_n as $\mathcal{P}_n/\mathcal{S}$. In the context of a stabilizer code, where all Pauli errors in the same coset $E\mathcal{S}$ affect the logic state of encoded qubits equivalently, the optimal decoding strategy would be finding the most probable coset given the syndrome event $S = s$ as

$$\widehat{E\mathcal{S}} = \arg \max_{E\mathcal{S} \in \mathcal{P}_n/\mathcal{S}} \sum_{D(x', z') \in E\mathcal{S}} \Pr((X, Z) = (x', z') | S = s). \quad (14)$$

After that, the decoder can choose one error pattern \widehat{E} in coset $\widehat{E\mathcal{S}}$ as the error to be corrected, as the specific error selected does not affect the result due to degeneracy. This decoding problem has been shown to be $\#P$ -complete [47], which is computationally much harder than the QML decoding problem.

2.4 CSS Codes with Bit-Flip Errors

In Section 3, 4, and 5 that follow, we will focus on the syndrome decoding problem of CSS codes with the bit-flip errors, where each encoded qubit is independently affected by a Pauli- σ_x error with probability p_x . In this model, the Pauli error has the form $E = D(X, 0)$, with $X \in \mathbb{F}_2^n$ being a random vector following the distribution:

$$\Pr(X = (x_1, x_2, \dots, x_n)) = \prod_{i=1}^n p_x^{x_i} (1 - p_x)^{1-x_i} \quad (15)$$

In the following, we will use the random vector X to denote the channel error for simplicity. For CSS codes in this case, the decoder only needs to consider part of the syndrome $S_x = XH_1^T$, where H_1 is the submatrix of H_z as in (10). The QML decoding problem then becomes finding \widehat{X} where

$$\begin{aligned} \widehat{X} &= \arg \max_{x \in \mathbb{F}_2^n} \Pr(X = x | S_x = s_x) \\ &= \arg \min_{x \in \mathbb{F}_2^n : xH_1^T = s_x} \text{wt}(x) \end{aligned} \quad (16)$$

This is equivalent to the maximum-likelihood decoding problem for the BSC in classical coding theory, which is known to be NP-complete [48]. Therefore, we turn to the low-complexity belief propagation (BP) algorithm.

3 Belief Propagation Decoding

BP is a low-complexity algorithm that solves inference problems for graphical models. Originated in the 1960s, Gallager's LDPC decoder [22] already contains the essence of BP, which was later formalized by Pearl in [49, 50]. In 1993, the discovery of turbo codes [51] brought along the turbo decoding algorithm, which is now recognized as an instance of BP [52]. In the late 1990s, those ideas led to the formalization of factor graphs and the sum-product algorithm [25, 53]. Some strengths of BP decoding are that it is highly parallelizable and it can achieve near-optimal decoding of well-designed LDPC codes and turbo codes. It also has wide applications in other problems on graphical models, such as the spin glass models [54], lossy compression with the low-density generator matrix codes [55, 56], and constraint satisfaction problems such as K -SAT [43, 57].

BP was first introduced to the quantum decoding problem by Poulin and Chung in [30]. However, the performance of BP for the quantum syndrome decoding problem for the QLDPC codes is degraded due to degeneracy [30, 32]. In this section, we first review the binary BP algorithm for syndrome decoding of stabilizer codes over bit-flip errors. Then, we discuss prior improvements for BP that mitigate the non-convergence problem, including BP-OSD [15] and BP-SI [41].

3.1 Belief Propagation for Bit-Flip Errors

Consider the setup described in Section 2.4 for the syndrome decoding problem of an $[[n, k]]$ CSS code over bit-flip noise. Denote the Pauli- σ_x error as a random vector $X = (X_1, X_2, \dots, X_n) \in \mathbb{F}_2^n$ following the distribution in (15). BP is an iterative message-passing algorithm runs on a Tanner graph representing H_1 that provides an estimate of the marginal probability

$$\Pr(X_i = x_i | S_x = s_x) \quad (17)$$

for all $i \in [n]$ and $x_i \in \{0, 1\}$.

Let H_1 be the σ_z -stabilizer matrix in (10) with $m_1 = n - k_1$ rows and n columns. the Tanner graph $G = (V, C, E)$ is a bipartite graph with the variable nodes in $V = \{v_1, \dots, v_n\}$ representing the elements of $X = (X_1, \dots, X_n)$ and the check nodes in $C = \{c_1, \dots, c_m\}$ representing the σ_z -stabilizers in H_1 . A variable node v_i is connected

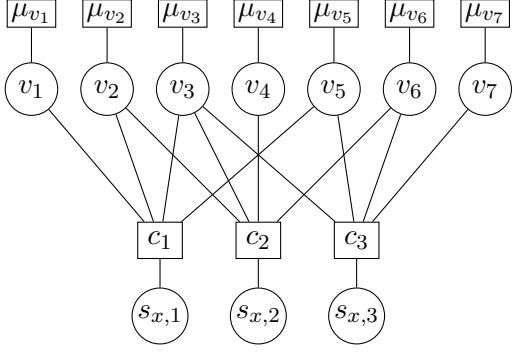


Figure 1: Tanner graph of H_1 for the $[[7, 1, 3]]$ Steane code

to a check node c_j if $H_1(i, j) = 1$. In addition, each variable node v_i is connected to a degree 1 check node that represents the source of the bit-flip error, and each check node c_j is connected to a degree 1 variable node that represents the source of a syndrome bit. An example Tanner graph representing H_1 for the $[[7, 1, 3]]$ Steane code [46] with

$$H_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \quad (18)$$

is shown in Fig. 1.

Now we describe the BP algorithm that runs on this Tanner graph. Let μ_{v_i} denote the channel log-likelihood ratio (LLR) for v_i as

$$\mu_{v_i} = \log \frac{\Pr(X_i = 0)}{\Pr(X_i = 1)} = \log \frac{1 - p_x}{p_x}. \quad (19)$$

This is the initial estimate of the overall LLR of X_i . At iteration $t = 0$, set the message from each variable node v_i to all its connected check node c_j as

$$m_{v_i \rightarrow c_j}^{(0)} = \mu_{v_i}. \quad (20)$$

Then, for iteration $t = 0, 1, \dots, T$, given the syndrome $s_x = (s_{x,1}, \dots, s_{x,m})$, the messages between variable nodes and check nodes are updated as

$$m_{c_j \rightarrow v_i}^{(t)} = (-1)^{s_{x,j}} \tanh^{-1} \left(\tanh \prod_{v_k \in \partial c_j \setminus v_i} m_{v_k \rightarrow c_j}^{(t)} \right), \quad (21)$$

and

$$m_{v_i \rightarrow c_j}^{(t+1)} = \mu_{v_i} + \sum_{c_k \in \partial v_i \setminus c_j} m_{c_k \rightarrow v_i}^{(t)}, \quad (22)$$

where $s_{x,j} \in \{0, 1\}$ is the syndrome of the σ_z -stabilizer corresponding to c_j (i.e., defined by the

j -th row of H_1), ∂c_j denotes the set of variable nodes in V connected to c_j , and ∂v_i denotes the set of check nodes in C connected to v_i . This update rule is called the sum-product algorithm [25].

In our implementation of BP, we also saturate the message to a domain $[-K, K]$ for numerical stability, where K is a large fixed number. Specifically, after every BP iteration, we clip the message $m_{v_i \rightarrow c_j}^{(t+1)}$ as

$$m_{v_i \rightarrow c_j}^{(t+1)} = \begin{cases} K & \text{if } m_{v_i \rightarrow c_j}^{(t+1)} > K \\ m_{v_i \rightarrow c_j}^{(t+1)} & \text{if } m_{v_i \rightarrow c_j}^{(t+1)} \in [-K, K] \\ -K & \text{if } m_{v_i \rightarrow c_j}^{(t+1)} < -K \end{cases} \quad (23)$$

for all edges in the Tanner graph. BP with saturation is commonly applied in practice to upper bound the soft information on the Tanner graph, and it plays a role in the error floor phenomenon and the stability of density evolution [58–61]. In the software implementation for all our simulation results shown in this work, we set $K = 25$. It is known that quantization and saturation for BP are complicated subjects, and we have yet to fully optimize for them.

The *bias* for variable node v_i after t iterations is defined to be

$$m_{v_i}^{(t)} = \mu_{v_i} + \sum_{c_k \in \partial v_i} m_{c_k \rightarrow v_i}^{(t)}. \quad (24)$$

For sufficiently large t , one approximates the log-likelihood ratio of the marginal probabilities for X_i with

$$m_{v_i}^{(t)} \approx \log \frac{\Pr(X_i = 0 | S_x = s_x)}{\Pr(X_i = 1 | S_x = s_x)}. \quad (25)$$

This approximation is exact if the Tanner graph G is a tree [25]. The sign of the bias $m_{v_i}^{(t)}$ represents the hard value toward which this variable node v_i is biased, and the absolute value of the bias is denoted by

$$\gamma^{(t)}(v_i) = |m_{v_i}^{(t)}|, \quad (26)$$

or just $\gamma(v_i)$ if t is clear from the context. This represents the *reliability* of this variable node. The larger the reliability, the more certain we are about its indicated hard value.

BP is commonly equipped with the following termination rule upon convergence [30]. After t

iterations, the estimated hard value for X_i can be computed as

$$\hat{x}_i^{(t)} = \begin{cases} 0, & m_{v_i}^{(t)} > 0 \\ 1, & m_{v_i}^{(t)} \leq 0 \end{cases}. \quad (27)$$

The BP algorithm terminates if the computed hard values

$$\hat{x}^{(t)} = (\hat{x}_1^{(t)}, \hat{x}_2^{(t)}, \dots, \hat{x}_n^{(t)}) \quad (28)$$

match the syndrome by satisfying $\hat{x}^{(t)} H_1^T = s_x$. This syndrome match event is called *convergence* [30]. The BP decoder then outputs $\hat{E} = D(\hat{x}^{(t)}, 0)$ as the estimated error operator. If none of the estimated $\hat{x}^{(t)}$ match the syndrome when t reaches a preset maximum iteration number T , then BP reports failure due to *non-convergence*.

The computational complexity of BP is $O(nT)$ without parallelism. If we assume that BP converges exponentially fast [62], a natural choice of T would be $O(\log n)$, which gives us $O(n \log n)$ for the complexity of BP.

3.2 Prior Improvements on BP

BP decoding for QLDPC codes has been known to have issues with convergence [30, 32]. Take the $[[882, 24, 18 \leq d \leq 24]]$ generalized bicycle B1 code proposed in [15] for example. We simulated its BP decoding performance with the sum-product algorithm with $T = 100$ for the bit-flip noise with error probability p_x . The performance is shown in the blue curve in Figure 4. In our simulation, we observe that almost all the block error cases are due to non-convergence.

To mitigate the non-convergence issue, Pantaleev and Kalachev [15] proposed to use ordered statistics decoding (OSD) as a post-processor when BP fails to converge. In Figure 4, the red curve shows the BP-OSD performance with order 0 for the same B1 code over the bit-flip noise. In this simulation, we employ the normalized min-sum algorithm for BP with the normalization factor $\alpha = 0.625$, matching the decoder in [15]. The BP-OSD decoder shows a significant performance improvement over BP. This gain has also been observed across various families of quantum stabilizer codes [15, 63]. However, it is important to note that OSD needs to solve a system of linear equations, which costs a higher computational complexity of $O(n^3)$ [15] compared to BP. In this

work, we take BP-OSD as the primary benchmark for evaluating the performance of our proposed decoder.

Another improvement on BP referred to as BP with stabilizer inactivation (BP-SI) is proposed by Crest, Mhalla, and Savin in [41]. It starts by sorting the σ_z -stabilizer rows in H_1 in increasing order of reliability based on BP soft information. After that, the BP algorithm is run λ times with exactly one of the λ least reliable stabilizers inactivated (i.e. punctured), with early termination upon convergence. This BP-SI approach is based on the intuition that “stabilizer-splitting” errors [32, 41] prevent BP convergence. In Figure 4, the performance of BP-SI with $\lambda = 10$ on the B1 code is shown in the yellow curve. The data points for the yellow curve are taken and translated directly from [41, Figure 2], where the decoder uses the serial message-passing scheduling for the normalized min-sum algorithm with normalization factor $\alpha = 0.9$. In [41], the worst-case complexity and the average complexity of BP-SI are claimed to be $O(\lambda_{\max} n \log n)$ and $O(\lambda_{\text{avg}} n \log n)$, respectively, assuming BP has complexity $O(n \log n)$. Here, λ_{\max} represents the maximum number of inactivated stabilizers and λ_{avg} represents the average number of inactivated stabilizers. Notably, at the end of BP-SI, one needs to solve a system of linear equations to recover the inactivated error positions.

4 Belief Propagation with Guided Decimation

In this section, we introduce a sampling approach for the syndrome decoding problem of QLDPC codes, and show how we can approximate it by combining BP with guided decimation (BPGD). First, we describe the sampling approach and show that in the low error rate regime, the sampling decoder can achieve performance close to the optimal DQML decoder. Then, we explain how BPGD can be used to approximate the sampling decoder. Following that, we show the simulated performance of BPGD on some QLDPC codes compared against BP-OSD and BP-SI under bit-flip noise, and analyze its complexity.

4.1 Sampling Decoding for Stabilizer Codes

Consider the noise model described in Section 2.3, where the encoded qubits of a stabilizer code are affected by an n -qubit Pauli error E following some distribution. We now describe a randomized version of QML decoding. Instead of selecting \widehat{E} with maximal posterior probability as in (13), this decoder draws an error \widehat{E} from a probability distribution conditioned on the observed syndrome event $S = s$. Specifically, the probability of an error $D(x, z)$ in this conditional probability distribution is given by:

$$\frac{\mathbb{1}_{(x,z)\Lambda H=s} \cdot \Pr((X, Z) = (x, z))}{\sum_{(x',z'): (x',z')\Lambda H=s} \Pr((X, Z) = (x', z'))}, \quad (29)$$

We note that, following this sampling decoding approach, each coset $E\mathcal{S}$ of the stabilizer group \mathcal{S} in \mathcal{P}_n will also be drawn according to their conditional probability distribution given syndrome s . This directly follows from the fact that the conditional probability of a coset $E\mathcal{S}$ equals the sum of the conditional probabilities for the errors in this coset. Therefore, the sampling version of the DQML decoder can be achieved by the sampling version of the QML decoder. Henceforth, we simply refer to it as the *sampling decoder*.

We remark that the sampling decoder has also been used as a theoretical proof technique in works on network information theory [64] and classical coding theory [65]. In the latter, they proved a lemma [65, Lemma 3] that upper bounds the error probability of the sampling decoder for a classical code by twice the error probability of the maximum-likelihood (ML) decoder. A similar upper bound holds for stabilizer codes as stated in the following theorem. The proof of this theorem is deferred to the Appendix.

Theorem 1 (DQML vs. sampling). *Consider decoding a stabilizer code over Pauli errors following some distribution. Denote the error probability of the DQML decoder by P_{DQML} , and denote the error probability of the sampling decoder by P_S . Then, the following inequalities hold:*

$$P_{\text{DQML}} \leq P_S \leq 2 \cdot P_{\text{DQML}} \quad (30)$$

Therefore, in the low error rate regime, the sampling decoder can achieve close-to-optimal performance. Next, we describe how we can approximate the sampling decoder with the help of BP.

4.2 Approximating the Sampling Decoder

Consider decoding an $[[n, k]]$ CSS code over the bit-flip noise. Given the syndrome event $S = s$, the sampling decoder would draw a random error vector following a conditional probability distribution. Here, we use the random vector

$$\widehat{X} = (\widehat{X}_1, \widehat{X}_2, \dots, \widehat{X}_n) \in \mathbb{F}_2^n \quad (31)$$

to denote the output of the sampling decoder.

This sampling process can be achieved by sequentially sampling the error bits in \widehat{X} as follows. First, we sample \widehat{X}_1 according to its marginal probabilities

$$\Pr(\widehat{X}_1 = \widehat{x}_1 | S = s), \quad \widehat{x}_1 \in \{0, 1\}. \quad (32)$$

Then, for i from 2 to n , we sequentially sample \widehat{X}_i conditioned on both the syndrome s and the value of its previous bits following the marginals

$$\Pr(\widehat{X}_i = \widehat{x}_i | \widehat{X}_1^{i-1} = \widehat{x}_1^{i-1}, S = s), \quad \widehat{x}_i \in \{0, 1\}. \quad (33)$$

Here, we use \widehat{X}_1^{i-1} and \widehat{x}_1^{i-1} to denote the sub-vectors of \widehat{X} and \widehat{x} with entries from 1 to $i - 1$, respectively.

The error sampled this way has the correct conditional probability according to the chain rule:

$$\Pr(\widehat{X} = \widehat{x} | S = s) = \prod_{i=1}^n \Pr(\widehat{X}_i = \widehat{x}_i | \widehat{X}_1^{i-1} = \widehat{x}_1^{i-1}, S = s). \quad (34)$$

Note that we are also free to choose an arbitrary order to sample the bits in \widehat{X} .

Since BP can be used to compute the approximated marginal probabilities for the error bits as shown in (25), the marginals in (32) and (33) can be estimated by running BP on a Tanner graph. Therefore, we can approximate the above sampling process by iterating the following steps:

- 1) Run BP to obtain estimated marginals for the remaining bits in the error vector.
- 2) Choose an \widehat{X}_i and fix its value to \widehat{x}_i according to its estimated marginals.
- 3) Update the condition to include $\widehat{X}_i = \widehat{x}_i$.

The process of sequentially fixing variables to hard decisions during iterative decoding has been referred to as guided decimation [42, 43]. We note that in [43], BP guided decimation combined

Algorithm 1: BPGD over bit flips

Input: block length n ,
 Tanner graph $G = (V, C, E)$ for H_1 ,
 syndrome s_x ,
 bit-flip error rate p_x ,
 number of BP iterations per round T

Output: estimated \hat{x} or non-convergence

```

1  $\mu_{v_i} = \log((1 - p_x)/p_x)$  for all  $v_i \in V$ 
2  $m_{v_i \rightarrow c_j}^{(0)} = \mu_{v_i}$  for all  $v_i \in V, c_j \in \partial v_i$ 
3  $V_u = V$ 
4 for  $r = 1$  to  $n$  do
5   run BP for  $T$  iterations
6    $\hat{x} \leftarrow$  hard values of the variable nodes
7   if  $\hat{x}H_1^T = s_x$  then
8     return  $\hat{x}$ 
9   else
10     $v_i = \arg \max_{v \in V_u} \gamma(v_i)$ 
11    if  $m_{v_i}^{(rT)} \geq 0$  then
12       $\mu_{v_i} = \text{llr}_{\max}$ 
13    else
14       $\mu_{v_i} = -\text{llr}_{\max}$ 
15     $V_u = V_u \setminus \{v_i\}$ 
16 return non-convergence

```

with warning propagation can also be understood to approximate the process of sampling a vector from the distribution implied by the Tanner graph for constraint satisfaction problems.

4.3 The BPGD Algorithm

Message-passing algorithms with “decimation” were first introduced for the K -SAT constraint satisfaction problem based on insights from statistical physics [42]. In such problems, there are typically many valid solutions and the goal is to find just one of them. In [42], decimation was first combined with a related message-passing algorithm called survey propagation. Later, the idea of decimation was extended to define the BP guided decimation (BPGD) algorithm [43] and related approaches were applied to the lossy compression problem [55, 56, 66].

Now, following the setup in Section 3.1, we describe the BPGD algorithm for decoding bit-flip errors in detail on a $[[n, k]]$ CSS code.

First, we initialize the channel LLR for each

variable node v_i on the Tanner graph as

$$\mu_{v_i} = \log \left(\frac{1 - p_x}{p_x} \right) \quad (35)$$

similar to BP.

Then, BPGD proceeds in rounds with r going from 1 to n . In the r -th round, we first run BP with the sum-product algorithm ((21) and (22)) on the Tanner graph for T iterations to obtain the estimated marginals for the variable nodes. If BP converges to an error matching the syndrome, then the decoder immediately terminates and outputs the hard values of the variable nodes as the estimated error.

Otherwise, out of all the variable nodes that are not yet decimated, we pick v_i with the largest reliability $\gamma(v_i)$ in (26) for decimation. We decimate this variable node v_i by updating its channel message μ_{v_i} based on its bias $m_{v_i}^{(rT)}$ as

$$\mu_{v_i} = \begin{cases} \text{llr}_{\max}, & \text{if } m_{v_i}^{(rT)} > 0 \\ -\text{llr}_{\max}, & \text{if } m_{v_i}^{(rT)} \leq 0, \end{cases} \quad (36)$$

where llr_{\max} is a large fixed number.

If we choose llr_{\max} to be infinity (i.e. $\text{llr}_{\max} = \infty$), this decimation effectively assigns a hard value for \hat{X}_i as \hat{x}_i according to the bias of v_i . Different from the sampling decoder that samples \hat{X}_i according to its marginals, here we simply fix the value of \hat{X}_i towards its bias. The new condition $\hat{X}_i = \hat{x}_i$ is also incorporated into the Tanner graph and takes effect during BP in the subsequent rounds. While letting llr_{\max} to be infinity makes sense in theory, it can introduce numerical issues in software implementation. and thus setting llr_{\max} to be a large fixed number is preferred in practice. For all our simulation results shown in this work, we set $\text{llr}_{\max} = 25$.

After decimating the most reliable variable node, we proceed to the next round $r + 1$ by continuing to run BP for another T iterations followed by decimation. This process continues until either BP converges or all the variable nodes have been decimated. If, after all variable nodes are decimated, their hard values still do not match the syndrome, then this is referred to as a *non-convergence failure* of the BPGD algorithm. The pseudo-code for the BPGD algorithm is provided in Algorithm 1.

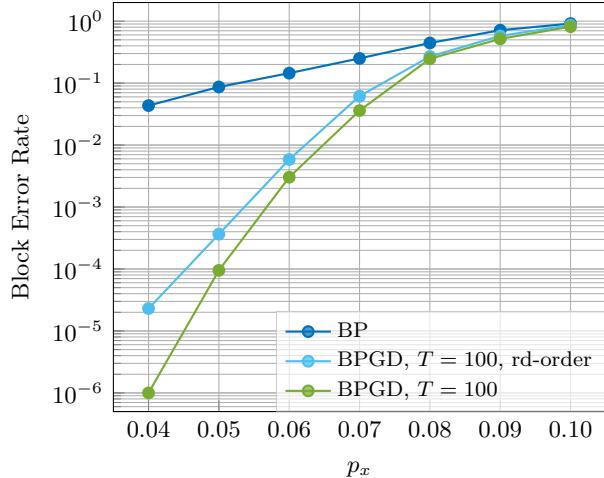


Figure 2: Performance of BPGD on the $[[882, 24, 18 \leq d \leq 24]]$ B1 code [15] over bit-flip noise following different decimation orders. The data points are collected by running simulations until we observe 100 error cases.

4.4 Numerical Results

In Section 4.3, we describe BPGD as an approximation for the sampling decoder, where we sequentially decimate variable nodes after each round of BP. Here we show simulation results of BPGD comparing two different decimation orders: sequentially decimating the most reliable variable nodes as described in Algorithm 1, versus randomly decimating variable nodes. For the second decimation order, after each round of BP, we randomly pick a variable node that hasn't been decimated and update its channel message according to its bias as in (36).

In Figure 2, we show the simulation result of BPGD for the $[[882, 24, 18 \leq d \leq 24]]$ B1 code [15] over bit-flip noise with error probability p_x . We start by setting $T = 100$ to ensure that in each round, BP is run for a sufficient number of iterations to provide accurate approximate marginal probabilities for the variable nodes. First, we observe that BPGD provides a significant performance gain compared with ordinary BP. Then, compared with random decimation order as shown in the light blue curve, BPGD which sequentially decimates the most reliable variable nodes offers better performance, as shown in the green curve.

In Figure 3, we also compare the BPGD performance with different BP iterations per round. We see that while $T = 100$ provides the best performance, reducing the number of BP iterations per round from $T = 100$ to $T = 10$, shown by the

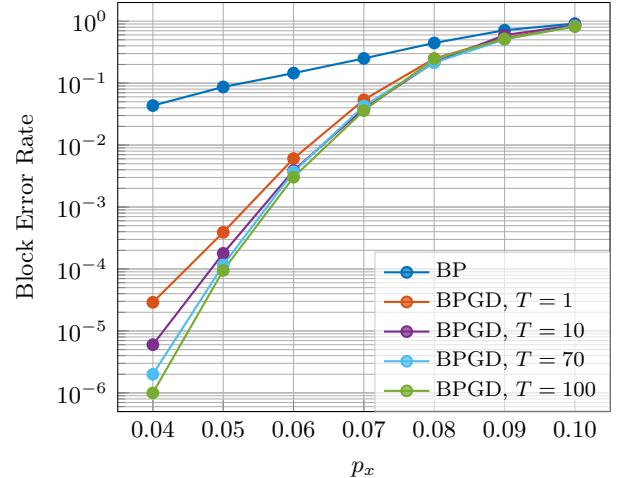


Figure 3: Performance of BPGD on the $[[882, 24, 18 \leq d \leq 24]]$ B1 code [15] over bit-flip noise with different BP iterations per round. The data points are collected by running simulations until we observe 100 error cases.

purple curve, does not significantly degrade the BPGD performance.

4.5 Performance Comparison

In Figure 4, we compare our BPGD performance with some prior works for the $[[882, 24, 18 \leq d \leq 24]]$ B1 code [15] over bit-flip noise. The performance of BPGD with $T = 10$, shown by the purple curve, is identical to the purple curve in Figure 3. For comparison, Figure 4 includes the decoding performance of BP, BP-OSD with order 0, and BP-SI with $\lambda = 10$, whose settings are given in Section 3.2. BPGD with $T = 10$ yields the best performance. It is worth noting that, in our simulations, the majority of the errors (over 90%) from the BPGD runs for both $T = 100$ and $T = 10$ are attributed to non-convergence. Therefore, similar to the BP algorithm, we rarely encounter logical errors upon convergence from running BPGD.

In Figure 5, we also show the simulation result of BPGD with $T = 10$ for the $[[1922, 50, 16]]$ hypergraph product code C2 [15] over bit-flip noise with error probability p_x . For comparison, we include the performances for BP, BP-OSD with order 0, and BP-SI with $\lambda = 10$, whose respective settings remain consistent with those in Figure 4: 1) the BP decoder runs the basic sum-product algorithm with $T = 100$; 2) the BP-OSD decoder with order 0 runs the serial normalized min-sum algorithm with normalization factor $\alpha = 0.625$; 3) the BP-SI decoder with $\lambda = 10$ runs the serial

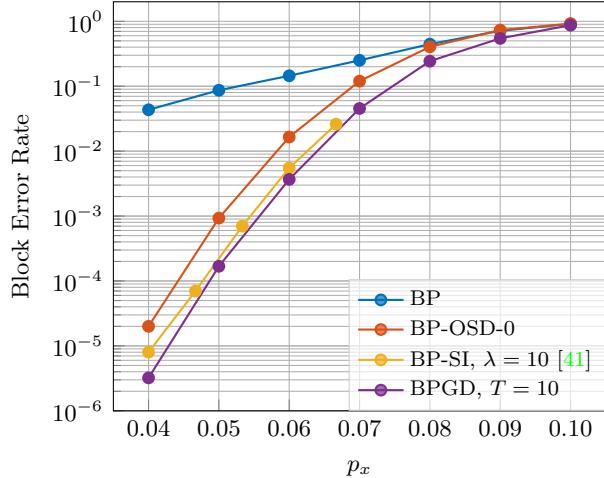


Figure 4: Performance of various decoders on the $[[882, 24, 18 \leq d \leq 24]]$ B1 code [15] over bit-flip noise. The data points of BP, BP-OSD with order 0, and BPGD decoders are collected by running simulations until we observe 100 error cases. The data points of BP-SI are taken and translated from [41, Figure 2]

normalized min-sum algorithm with normalization factor $\alpha = 0.9$, whose data points are taken and translated directly from [41, Figure 2]. We can see that for the two QLDPC codes we considered in Figure 4 and Figure 5, BPGD shows better performance compared with BP-OSD with order 0 and BP-SI with $\lambda = 10$.

4.6 Complexity Analysis for BPGD

BPGD requires at most n decimation rounds, each involving T iterations of message-passing updates of complexity $O(n)$ and a search for the most reliable qubit for decimation with complexity $O(n)$. Therefore, the worst-case complexity of BPGD is $O(Tn^2)$. This worst-case complexity scales roughly the same as BP-SI [41], and it is better compared with BP-OSD with order 0, which has complexity $O(n^3)$ [15]. Moreover, unlike BP-OSD and BP-SI, BPGD does not require solving systems of linear equations, which makes it potentially more friendly for hardware implementation.

If we assume BP in each round has complexity $O(Tn)$, then the average-case complexity of BPGD becomes $O(r_{\text{avg}}Tn)$, where r_{avg} denotes the average number of decimated variable nodes. The value of r_{avg} is highly dependent on the bit-flip error probability p_x , meaning BPGD has different average complexity in different error rate regimes. Table 1 shows the average number of

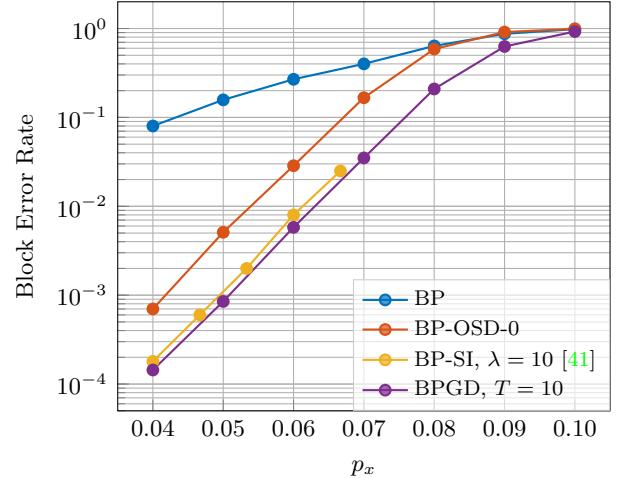


Figure 5: Performance of various decoders on the $[[1922, 50, 16]]$ C2 code [15] over bit-flip noise. The data points of BP, BP-OSD with order 0, and BPGD decoders are collected by running simulations until we observe 100 error cases. The data points of BP-SI are taken and translated from [41, Figure 2]

p_x	0.05	0.06	0.07	0.08
sim. runs	1000000	100000	100000	10000
r_{avg}	2.91	9.82	60.46	231.7

Table 1: Average number of decimation rounds of BPGD with $T = 100$ on the B1 code [15] over bit-flip noise.

decimated variable nodes for BPGD with $T = 10$ when decoding the B1 code. Note that when calculating r_{avg} in Table 1, we take into account both the convergent cases and the non-convergent cases. In the non-convergence cases, the number of variable nodes decimated by BPGD equals the block length $n = 882$. From Table 1 we can see that, in the low error rate regime such as $p_x = 0.05$, r_{avg} becomes very small, making the average complexity of BPGD approaches the BP complexity $O(Tn)$. We note that a similar observation has also been made for BP-SI concerning the average number of inactivated stabilizers in the low error rate regime [41].

One natural way to reduce the worst-case complexity of BPGD is by bounding the maximum number of variable nodes it decimates. This can be achieved by modifying line 4 in Algorithm 1 to “for $r = 1$ to R do”, where $R < n$ represents a predefined limit on the number of decimation rounds. With this modification, the worst-case complexity of BPGD becomes $O(RTn)$. In Figure 6, we present the decoding performance of BPGD with $T = 100$ and different round limits

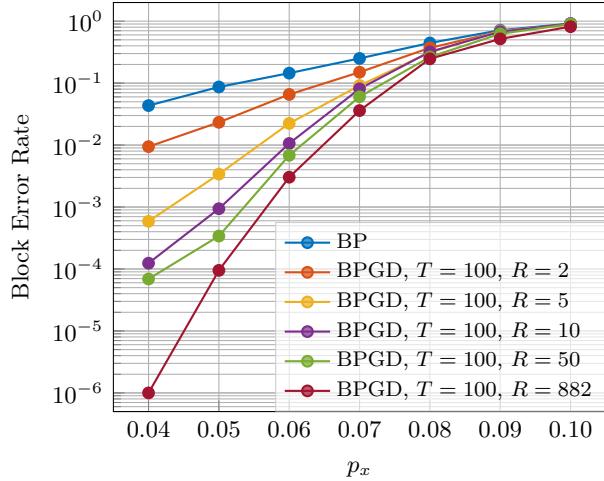


Figure 6: Performance of BPGD with different decimation round limits on the $[[882, 24, 18 \leq d \leq 24]]$ B1 code [15] over bit-flip noise.

R for the $[[882, 24, 18 \leq d \leq 24]]$ B1 code [15]. As expected, the performance of BPGD steadily improves with increasing round limit R . The optimal performance is attained when R equals the block length $n = 882$. This allows us to pick a desired trade-off between worst-case complexity and the BPGD decoding performance tailored to specific application requirements.

5 Randomized BPGD: An Experiment

For the syndrome decoding of degenerate QLDPC codes, one reason for the non-convergence issue of BP is that there may exist multiple low-weight error patterns that match the syndrome [30]. Due to code degeneracy, many of them differ only by a stabilizer. Intuitively, this is supposed to help the decoder, as it yields the same decoding result by picking any of those solutions. However, when running BP, the locally operating algorithm gets confused about the direction to proceed, resulting in non-convergence [32]. In a case study presented in [30, Section IV. A], Poulin and Chung explored this scenario for a two-qubit stabilizer code. A more comprehensive study of this phenomenon from the perspective of trapping sets can be found in the work by Raveendran and Vasić [32]. In another work, Kuo and Lai [39] examined this situation by regarding BP as an energy-minimization process, and explain non-convergence as being trapped in a local minimum of the energy function.

Here, we devise an experiment to shed light on

this situation. Consider the following randomized version of the BPGD algorithm, where in each round, instead of decimating the most reliable variable node, we identify the top few reliable variable nodes and then randomly select one for decimation. Concretely, this randomized decimation process goes as follows. Denote V_u as the set of variable nodes that have not yet been decimated. In the r -th round, after running BP for T iterations, we first find the most reliable variable node in V_u and denote its reliability as γ_{\max} . Then, we construct a set P including all $v_i \in V_u$ whose reliabilities satisfy $\gamma(v_i) \geq \gamma_{\max} - \gamma'$. Here, P is a set of many reliable variable nodes prepared for random decimation, and γ' stands for a gap such that the reliability threshold for P is $\gamma_{\max} - \gamma'$. Note that P has size at least one since it always includes the most reliable variable node in V_u . After that, we randomly select a variable node in P for decimation. The pseudo-code for the randomized BPGD algorithm, referred to as BPGD-rd, is presented in Algorithm 2. The differences between BPGD-rd and BPGD in Algorithm 1 regarding the randomized decimation process are shown in lines 10-12 in Algorithm 2.

Since we are randomly selecting variable nodes in BPGD-rd each round for decimation, this algorithm can potentially converge to many different errors. Leveraging this property, we consider the following experiment for the $[[882, 24, 18 \leq d \leq 24]]$ B1 code [15]. First, we select a bit-flip error X with weight 73. This error is randomly generated with $p_x = 0.08$. Then, we run the BPGD-rd algorithm 10000 times to decode its syndrome, each with a distinct seed for the random decimation process. The configuration for BPGD-rd during this experiment includes $T = 10$ for the number of BP iterations per round, $p_x = 0.08$ for the bit-flip error probability, and $\gamma' = 1.0$ for the reliability gap.

We observe that out of 10000 BPGD-rd runs, 9568 of them converge. Within these 9568 convergent cases, we identify a total of 111 distinct errors, whose weights range from 73 to 87. Table 2 shows the 10 most frequent errors along with their weights and distances relative to the channel error X . Notably, all 111 error patterns differ from X by a stabilizer, which means that all of them decode X successfully due to degeneracy. If we run the original BPGD algorithm in Algorithm 1 for decoding, it converges to the error

Algorithm 2: BPGD-rd over bit flips

Input: block length n ,
 Tanner graph $G = (V, C, E)$ for H_1 ,
 syndrome s_x ,
 bit-flip error rate p_x ,
 number of BP iterations per round T

Output: estimated \hat{x} or non-convergence

```

1  $\mu_{v_i} = \log((1 - p_x)/p_x)$  for all  $v_i \in V$ 
2  $m_{v_i \rightarrow c_j}^{(0)} = \mu_{v_i}$  for all  $v_i \in V, c_j \in \partial v_i$ 
3  $V_u = V$ 
4 for  $r = 1$  to  $n$  do
5   run BP for  $T$  iterations
6    $\hat{x} \leftarrow$  hard values of the variable nodes
7   if  $\hat{x}H_1^T = s_x$  then
8     return  $\hat{x}$ 
9   else
10     $\gamma_{\max} = \max_{v_i \in V_u} \gamma(v_i)$ 
11     $P = \{v_i \in V_u \mid \gamma(v_i) \geq \gamma_{\max} - \gamma'\}$ 
12    randomly select  $v_i$  from  $P$ 
13    if  $m_{v_i}^{(rT)} \geq 0$  then
14       $\mu_{v_i} = \text{llr}_{\max}$ 
15    else
16       $\mu_{v_i} = -\text{llr}_{\max}$ 
17     $V_u = V_u \setminus \{v_i\}$ 
18 return non-convergence

```

with index 2 in Table 2.

From this experiment we see that for the syndrome decoding problem of a degenerate QLDPC code such as the B1 code, there can be multiple low-weight error patterns matching the input syndrome, confusing the BP decoding process. In particular, for decoding X in this experiment, BP itself with the sum-product algorithm results in non-convergence. Moreover, it also shows that combining BP with guided decimation encourages BP convergence towards one of those errors. In this case, degeneracy plays to our advantage since decoding is successful no matter which of the 111 errors is selected.

Here we make some additional remarks regarding this experiment. Firstly, it can be observed that the distribution of the error patterns upon convergence in the BPGD-rd algorithm is not uniform. We can see from Table 2 that, for the syndrome decoding problem we investigated, a significant portion of the convergent cases focuses on four specific weight-73 errors, indexed by 1-4 in Table 2. In contrast, the original weight-73

index	frequency	weight	distance to X
1	2944	73	22
2	2727	73	26
3	1560	73	20
4	1521	73	16
5	125	73	30
6	110	73	26
7	91	73	16
8	68	73	20
9	66	73	24
10	32	75	28

Table 2: A list of 10 most frequent errors obtained out of 10000 BPGD-rd decoding runs on B1 code [15] for a bit-flip error X with weight 73.

bit-flip error X does not even appear among the 111 errors we observed.

Secondly, we note that due to the random decimation process inherent to BPGD-rd, the results presented in this section are the product of a single simulation run of this experiment. A repeat of the same experiment could yield different numerical results. Additionally, the outcomes depend on the specific choice of the channel error E_x . However, we observe that the phenomenon of multiple distinct error patterns matching the input syndrome and the uneven distribution among convergent error patterns appears to be a general characteristic of the BPGD-rd algorithm.

6 Quaternary Belief Propagation with Guided Decimation

In this section, we consider the syndrome decoding problem of a $[[n, k]]$ CSS code over depolarizing noise, and present a natural extension of the binary BPGD algorithm that gives the quaternary version.

In the depolarizing noise model with physical error rate p , each encoded qubit is independently affected by a Pauli σ_x , σ_y , or σ_z error, each occurring with a probability of $p/3$. Here, we represent the Pauli error as a random quaternary vector $Q = (Q_1, \dots, Q_n)$ with its realization $q = (q_1, \dots, q_n) \in \{0, 1, 2, 3\}^n$, with 0, 1, 2, 3 representing the absence of error, or the presence of a Pauli σ_x , σ_y or σ_z error, respectively. We define the Pauli error represented by q as $E(q)$. For

Algorithm 3: Q-BPGD for depolarizing noise

Input: block length n , Tanner graph G , syndrome s , physical error rate p , number of iterations per round T

Output: estimated \hat{q} or non-convergence

- 1 $\mu_{v_i} = (1 - p, p/3, p/3, p/3)$ for all $v_i \in V$
- 2 $V_u = V$
- 3 **for** $r = 1$ **to** n **do**
- 4 run Q-BP for T iterations
- 5 $\hat{q} \leftarrow$ hard values for the variable nodes
- 6 **if** \hat{q} matches the syndrome s **then**
- 7 **return** \hat{q}
- 8 **else**
- 9 $v_i = \arg \max_{v \in V_u} \gamma(v_i)$
- 10 **switch** $\arg \max_{j \in \{0,1,2,3\}} p_{v_i,j}$ **do**
- 11 **case 0 do** $\mu_{v_i} = (1 - \epsilon, \epsilon, \epsilon, \epsilon);$
- 12 **case 1 do** $\mu_{v_i} = (\epsilon, 1 - \epsilon, \epsilon, \epsilon);$
- 13 **case 2 do** $\mu_{v_i} = (\epsilon, \epsilon, 1 - \epsilon, \epsilon);$
- 14 **case 3 do** $\mu_{v_i} = (\epsilon, \epsilon, \epsilon, 1 - \epsilon);$
- 15 $V_u = V_u \setminus \{v_i\}$
- 16 **return** non-convergence

example,

$$E(q = (1, 0, 2, 3, 0)) = \sigma_x \otimes I \otimes \sigma_y \otimes \sigma_z \otimes I. \quad (37)$$

For depolarizing noise, we consider the quaternary BP (Q-BP) algorithm [30], also referred to as the non-binary version of the syndrome BP algorithm [31]. Notably, Q-BP offers better performance over depolarizing noise in comparison to decoding bit-flip errors and phase-flip errors separately with binary BP, as exemplified in [15, Figure 5]. For detailed descriptions of Q-BP, we refer the readers to [67] and to [31, Algorithm 1]. An efficient log domain implementation of Q-BP has also been discussed in [68].

For the syndrome decoding problem on a CSS code, Q-BP runs on a Tanner graph $G = (V, C, E)$ similar to binary BP, except that here C contains check nodes representing both the σ_x -stabilizers in H_2 and the σ_z -stabilizers in H_1 , where H_1 and H_2 are submatrices of H_x and H_z , respectively, as in (10).

In addition, the channel LLRs for the variable nodes are now become the channel message

$$\mu_{v_i} = (1 - p, p/3, p/3, p/3), \quad (38)$$

and each variable node $v_i \in V$ contains information of four normalized probabilities

$$(p_{v_i,0}, p_{v_i,1}, p_{v_i,2}, p_{v_i,3}) \quad (39)$$

which, after a sufficient number of BP iterations, approximate the marginal probabilities for Q_i conditioned on the syndrome s as

$$p_{v_i,j} \approx \Pr(Q_i = j \mid S = s), \quad (40)$$

for $j \in \{0, 1, 2, 3\}$. Following [15], we define the reliability of v_i as

$$\gamma(v_i) = \max\{p_{v_i,0}, p_{v_i,1}, p_{v_i,2}, p_{v_i,3}\}, \quad (41)$$

Similar to its binary counterpart, Q-BP can also be improved by guided decimation, denoted as the quaternary belief propagation guided decimation (Q-BPGD) algorithm. Q-BPGD iterates between Q-BP and decimation in rounds with r going from 1 to n , with early termination upon Q-BP convergence. In each round, after running Q-BP for T iterations, out of all the variable nodes not yet decimated, we pick the variable node v_i with the largest reliability $\gamma(v_i)$ and decimate it by updating its channel message to

$$\mu_{v_i} = \begin{cases} (1 - \epsilon, \epsilon, \epsilon, \epsilon), & \text{if } \gamma_i^* = 0 \\ (\epsilon, 1 - \epsilon, \epsilon, \epsilon), & \text{if } \gamma_i^* = 1 \\ (\epsilon, \epsilon, 1 - \epsilon, \epsilon), & \text{if } \gamma_i^* = 2 \\ (\epsilon, \epsilon, \epsilon, 1 - \epsilon), & \text{if } \gamma_i^* = 3 \end{cases} \quad (42)$$

where

$$\gamma_i^* = \arg \max_{j \in \{0,1,2,3\}} p_{v_i,j}. \quad (43)$$

As ϵ approaches zero, this decimation effectively assigns a hard value \hat{q}_i for Q_i in $\{0, 1, 2, 3\}$ according to the bias of v_i . In our software implementation for the simulation results in this section, we set $\epsilon = 1 \times 10^{-10}$ for numerical stability. The pseudo-code for the Q-BPGD algorithm over depolarizing noise is provided in Algorithm 3.

In Figure 7, we present simulation results of Q-BPGD decoder with $T = 10$ on the $[[180, 10, 15 \leq d \leq 18]]$ A5 code [15] over depolarizing noise. For comparison, Figure 7 includes the performances of the Q-BP decoder with $T = 100$ and the BP-OSD decoder with data points directly taken from [15, Figure 6]. For the A5 code, Q-BPGD exhibits improved performance compared to Q-BP and similar performance compared to BP-OSD with order 10.

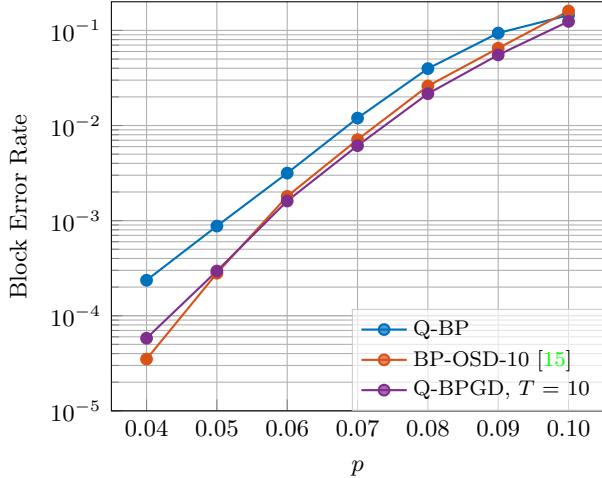


Figure 7: Performance of various decoders on the $[[180, 10, 15 \leq d \leq 18]]$ A5 code [15] over depolarizing noise. The data points of Q-BP and Q-BPGD are collected by running simulations until we observe 100 error cases. The data points of BP-OSD with order 10 are taken from [15, Figure 6]

In Figure 8, we also present the Q-BPGD performance with $T = 10$ for the $[[882, 48, 16]]$ B2 code [15]. Similarly, the comparison includes the Q-BP decoder with $T = 100$, and the BP-OSD decoder with data points directly taken from [15, Figure 2]. On the B2 code, Q-BPGD outperforms the BP-OSD decoder in the high-error regime but is surpassed by BP-OSD in the low-error-rate regime. We remark that besides the BP-OSD performance that we have been using as the main comparison benchmark in this paper, a very good decoding performance is also reported in [39, Figure 13] for the B2 code using an adaptive version of Q-BP with memory.

7 Summary

In this paper, we introduce and evaluate the use of BPGD to decode QLDPC codes to encourage convergence. For motivation, we propose a sampling decoding approach for QLDPC codes and show how we can approximate it with BPGD. BPGD shows strong performance compared with BP-OSD and BP-SI over bit-flip noise. To better understand how BPGD performs, we experiment with a randomized version of BPGD that helps illuminate the role of degeneracy in syndrome decoding. Our experiments suggest that BPGD performed well because degeneracy allows it to achieve successful decoding along many dif-

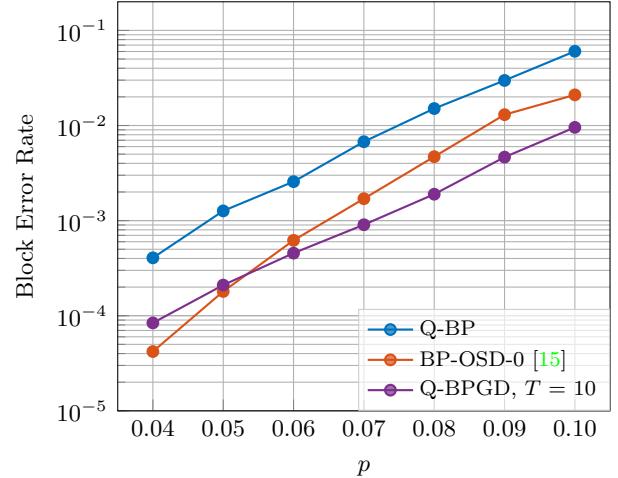


Figure 8: Performance of various decoders on the $[[882, 48, 16]]$ B2 code [15] over depolarizing noise. The data points of Q-BP and Q-BPGD are collected by running simulations until we observe 100 error cases. The data points of BP-OSD with order 0 are taken from [15, Figure 2]

ferent decimation paths. Furthermore, we extend our guided decimation from binary BP to quaternary BP, demonstrating performance competitive compared to BP-OSD in the high-error regime over depolarizing noise.

8 Funding Acknowledgement

This material is based on work supported by the NSF under Grants 2106213 and 2120757. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

- [1] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane. “Quantum error correction and orthogonal geometry”. *Phys. Rev. Lett.* **78**, 405–408 (1997).
- [2] A.R. Calderbank, E.M. Rains, P.M. Shor, and N.J.A. Sloane. “Quantum error correction via codes over $GF(4)$ ”. *IEEE Transactions on Information Theory* **44**, 1369–1387 (1998).
- [3] Daniel Gottesman. “Stabilizer codes and quantum error correction” (1997). [arXiv:quant-ph/9705052](https://arxiv.org/abs/quant-ph/9705052).

- [4] A Yu Kitaev. “Quantum computations: algorithms and error correction”. *Russian Mathematical Surveys* **52**, 1191 (1997).
- [5] A.Yu. Kitaev. “Fault-tolerant quantum computation by anyons”. *Annals of Physics* **303**, 2–30 (2003).
- [6] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. “Topological quantum memory”. *Journal of Mathematical Physics* **43**, 4452–4505 (2002).
- [7] Sergey Bravyi, David Poulin, and Barbara Terhal. “Tradeoffs for reliable quantum information storage in 2D systems”. *Phys. Rev. Lett.* **104**, 050503 (2010).
- [8] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. “Surface codes: Towards practical large-scale quantum computation”. *Phys. Rev. A* **86**, 032324 (2012).
- [9] Dominic Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. “Surface code quantum computing by lattice surgery”. *New Journal of Physics* **14**, 123011 (2012).
- [10] D.J.C. MacKay, G. Mitchison, and P.L. McFadden. “Sparse-graph codes for quantum error correction”. *IEEE Transactions on Information Theory* **50**, 2315–2330 (2004).
- [11] Jean-Pierre Tillich and Gilles Zémor. “Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength”. *IEEE Transactions on Information Theory* **60**, 1193–1202 (2014).
- [12] Alexey A. Kovalev and Leonid P. Pryadko. “Improved quantum hypergraph-product LDPC codes”. In 2012 IEEE International Symposium on Information Theory Proceedings. Pages 348–352. (2012).
- [13] Alexey A. Kovalev and Leonid P. Pryadko. “Quantum kronecker sum-product low-density parity-check codes with finite rate”. *Phys. Rev. A* **88**, 012311 (2013).
- [14] Jeongwan Haah. “Local stabilizer codes in three dimensions without string logical operators”. *Phys. Rev. A* **83**, 042330 (2011).
- [15] Pavel Panteleev and Gleb Kalachev. “Degenerate Quantum LDPC Codes With Good Finite Length Performance”. *Quantum* **5**, 585 (2021).
- [16] Siyi Yang and Robert Calderbank. “Spatially-coupled QDLPc codes” (2023). [arXiv:2305.00137](https://arxiv.org/abs/2305.00137).
- [17] Nikolas P. Breuckmann and Jens Niklas Eberhardt. “Quantum low-density parity-check codes”. *PRX Quantum* **2**, 040101 (2021).
- [18] Matthew B. Hastings, Jeongwan Haah, and Ryan O’Donnell. “Fiber bundle codes: breaking the $n^{1/2}\text{polylog}(n)$ barrier for quantum LDPC codes”. In Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing. Page 1276–1288. STOC 2021New York, NY, USA (2021). Association for Computing Machinery.
- [19] Nikolas P. Breuckmann and Jens N. Eberhardt. “Balanced product quantum codes”. *IEEE Transactions on Information Theory* **67**, 6653–6674 (2021).
- [20] Pavel Panteleev and Gleb Kalachev. “Quantum LDPC codes with almost linear minimum distance”. *IEEE Transactions on Information Theory* **68**, 213–229 (2022).
- [21] Pavel Panteleev and Gleb Kalachev. “Asymptotically good quantum and locally testable classical LDPC codes”. In Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing. Page 375–388. STOC 2022New York, NY, USA (2022). Association for Computing Machinery.
- [22] R. Gallager. “Low-density parity-check codes”. *IRE Transactions on Information Theory* **8**, 21–28 (1962).
- [23] European Telecommunications Standards Institute (ETSI). “Digital video broadcasting (DVB): Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications”. Standard. ETSI EN 302 307, V1.2.1 (2009).
- [24] Tom Richardson and Shrinivas Kudekar. “Design of low-density parity check codes for 5G new radio”. *IEEE Communications Magazine* **56**, 28–34 (2018).
- [25] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. “Factor graphs and the sum-product algorithm”. *IEEE Transactions on Information Theory* **47**, 498–519 (2001).
- [26] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman. “Effi-

- cient erasure correcting codes”. *IEEE Transactions on Information Theory* **47**, 569–584 (2001).
- [27] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman. “Improved low-density parity-check codes using irregular graphs”. *IEEE Transactions on Information Theory* **47**, 585–598 (2001).
- [28] T.J. Richardson and R.L. Urbanke. “The capacity of low-density parity-check codes under message-passing decoding”. *IEEE Transactions on Information Theory* **47**, 599–618 (2001).
- [29] T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke. “Design of capacity-approaching irregular low-density parity-check codes”. *IEEE Transactions on Information Theory* **47**, 619–637 (2001).
- [30] David Poulin and Yeojin Chung. “On the iterative decoding of sparse quantum codes”. *Quant. Inf. Comput.* **8**, 0987–1000 (2008).
- [31] Zunaira Babar, Panagiotis Botsinis, Dimitrios Alanis, Soon Xin Ng, and Lajos Hanzo. “Fifteen years of quantum LDPC coding and improved decoding strategies”. *IEEE Access* **3**, 2492–2519 (2015).
- [32] Nithin Raveendran and Bane Vasić. “Trapping Sets of Quantum LDPC Codes”. *Quantum* **5**, 562 (2021).
- [33] Yun-Jiang Wang, Barry C. Sanders, Bao-Ming Bai, and Xin-Mei Wang. “Enhanced feedback iterative decoding of sparse quantum codes”. *IEEE Transactions on Information Theory* **58**, 1231–1241 (2012).
- [34] Alex Rigby, J. C. Olivier, and Peter Jarvis. “Modified belief propagation decoders for quantum low-density parity-check codes”. *Phys. Rev. A* **100**, 012330 (2019).
- [35] Ye-Hua Liu and David Poulin. “Neural belief-propagation decoders for quantum error-correcting codes”. *Phys. Rev. Lett.* **122**, 200501 (2019).
- [36] Xin Xiao, Nithin Raveendran, and Bane Vasić. “Neural-net decoding of quantum LDPC codes with straight-through estimators”. In Proc. Information Theory and Applications Workshop (ITA). (2019).
- [37] Sisi Miao, Alexander Schnerring, Haizheng Li, and Laurent Schmalen. “Neural belief propagation decoding of quantum LDPC codes using overcomplete check matrices”. In 2023 IEEE Information Theory Workshop (ITW). Pages 215–220. (2023).
- [38] Josias Old and Manuel Rispler. “Generalized Belief Propagation Algorithms for Decoding of Surface Codes”. *Quantum* **7**, 1037 (2023).
- [39] Kao-Yueh Kuo and Ching-Yi Lai. “Exploiting degeneracy in belief propagation decoding of quantum codes”. *npj Quantum Information* **8**, 111 (2022).
- [40] Dimitris Chytas, Michele Pacenti, Nithin Raveendran, Mark F. Flanagan, and Bane Vasić. “Enhanced message-passing decoding of degenerate quantum codes utilizing trapping set dynamics”. *IEEE Communications Letters* **28**, 444–448 (2024).
- [41] Julien Du Crest, Mehdi Mhalla, and Valentin Savin. “Stabilizer inactivation for message-passing decoding of quantum LDPC codes”. In 2022 IEEE Information Theory Workshop (ITW). Pages 488–493. (2022).
- [42] M. Mézard, G. Parisi, and R. Zecchina. “Analytic and algorithmic solution of random satisfiability problems”. *Science* **297**, 812–815 (2002).
- [43] Andrea Montanari, Federico Ricci-Tersenghi, and Guilhem Semerjian. “Solving constraint satisfaction problems through belief propagation-guided decimation”. In Forty-Fifth Annual Allerton Conference. (2007).
- [44] Richard E. Blahut. “Algebraic codes for data transmission”. Cambridge University Press. (2003).
- [45] A. R. Calderbank and Peter W. Shor. “Good quantum error-correcting codes exist”. *Phys. Rev. A* **54**, 1098–1105 (1996).
- [46] Andrew Steane. “Multiple-particle interference and quantum error correction”. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* **452**, 2551–2577 (1996).
- [47] Pavithran Iyer and David Poulin. “Hardness of decoding quantum stabilizer codes”. *IEEE Transactions on Information Theory* **61**, 5209–5223 (2015).
- [48] E. Berlekamp, R. McEliece, and H. van Tilborg. “On the inherent intractability of certain coding problems (corresp.)”. *IEEE Transactions on Information Theory* **24**, 384–386 (1978).

- [49] Judea Pearl. “Reverend Bayes on inference engines: A distributed hierarchical approach”. In Proceedings of the Second AAAI Conference on Artificial Intelligence. Page 133–136. AAAI’82. AAAI Press (1982).
- [50] Judea Pearl. “Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (2nd ed.)”. San Francisco, CA: Morgan Kaufmann. (1988).
- [51] C. Berrou, A. Glavieux, and P. Thitimajshima. “Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1”. In Proceedings of ICC ’93 - IEEE International Conference on Communications. Volume 2, pages 1064–1070 vol.2. (1993).
- [52] R.J. McEliece, D.J.C. MacKay, and Jung-Fu Cheng. “Turbo decoding as an instance of Pearl’s “belief propagation” algorithm”. *IEEE Journal on Selected Areas in Communications* **16**, 140–152 (1998).
- [53] S.M. Aji and R.J. McEliece. “The generalized distributive law”. *IEEE Transactions on Information Theory* **46**, 325–343 (2000).
- [54] Marc Mézard and Giorgio Parisi. “The Bethe lattice spin glass revisited”. *The European Physical Journal B-Condensed Matter and Complex Systems* **20**, 217–233 (2001).
- [55] Tomas Filler and Jessica Fridrich. “Binary quantization using belief propagation with decimation over factor graphs of LDGM codes”. In Forty-Fifth Annual Allerton Conference on Communication, Control, and Computing. (2007).
- [56] Vahid Aref, Nicolas Macris, and Marc Vuffray. “Approaching the rate-distortion limit with spatial coupling, belief propagation, and decimation”. *IEEE Transactions on Information Theory* **61**, 3954–3979 (2015).
- [57] Amin Coja-Oghlan. “On belief propagation guided decimation for random k -sat”. In Proceedings of the 2011 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). Pages 957–966. (2011).
- [58] Xiaojie Zhang and Paul H. Siegel. “Will the real error floor please stand up?”. In 2012 International Conference on Signal Processing and Communications (SPCOM). Pages 1–5. (2012).
- [59] Christian Schlegel and Shuai Zhang. “On the dynamics of the error floor behavior in (regular) LDPC codes”. *IEEE Transactions on Information Theory* **56**, 3248–3264 (2010).
- [60] Brian K. Butler and Paul H. Siegel. “Error floor approximation for LDPC codes in the AWGN channel”. *IEEE Transactions on Information Theory* **60**, 7416–7441 (2014).
- [61] Shrinivas Kudekar, Tom Richardson, and Aravind Iyengar. “The effect of saturation on belief propagation decoding of LDPC codes”. In 2014 IEEE International Symposium on Information Theory. Pages 2604–2608. (2014).
- [62] Amir Dembo and Andrea Montanari. “Ising models on locally tree-like graphs”. *The Annals of Applied Probability* **20**, 565 – 592 (2010).
- [63] Joschka Roffe, David R. White, Simon Burton, and Earl Campbell. “Decoding across the quantum low-density parity-check code landscape”. *Phys. Rev. Res.* **2**, 043423 (2020).
- [64] Mohammad Hossein Yassaee, Mohammad Reza Aref, and Amin Gohari. “A technique for deriving one-shot achievability results in network information theory”. In 2013 IEEE International Symposium on Information Theory. Pages 1287–1291. (2013).
- [65] Shrinivas Kudekar, Santhosh Kumar, Marco Mondelli, Henry D. Pfister, and Rüdiger Urbanke. “Comparing the bit-MAP and block-MAP decoding thresholds of reed-muller codes on BMS channels”. In 2016 IEEE International Symposium on Information Theory (ISIT). Pages 1755–1759. (2016).
- [66] M.J. Wainwright and E. Maneva. “Lossy source encoding via message-passing and decimation over generalized codewords of LDGM codes”. In Proceedings. International Symposium on Information Theory, 2005. ISIT 2005. Pages 1493–1497. (2005).
- [67] 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**, 487–498 (2020).
- [68] Ching-Yi Lai and Kao-Yueh Kuo. “Log-domain decoding of quantum LDPC codes over binary finite fields”. *IEEE Transactions on Quantum Engineering* **2**, 1–15 (2021).

A Proof of Theorem 1

The inequality $P_{\text{DQML}} \leq P_S$ follows from the fact that the DQML decoder is an optimal decoder that minimizes the error probability.

To prove $P_S \leq 2 \cdot P_{\text{DQML}}$, let's denote E as a Pauli error, $E\mathcal{S}$ as a coset of the stabilizer group \mathcal{S} in $\mathcal{P}_n \setminus \mathcal{S}$, S as a syndrome, and $\widehat{E\mathcal{S}}(S)$ as the coset with the largest posterior probability output by the DQML decoder given syndrome S . Denote by $\mathbb{1}(\cdot)$ an indicator function of an event, we can then write P_{DQML} as

$$\begin{aligned} P_{\text{DQML}} &= \sum_S \Pr(S) \sum_{E\mathcal{S}} \Pr(E\mathcal{S} | S) \mathbb{1}(E\mathcal{S} \neq \widehat{E\mathcal{S}}(S)) \\ &= \sum_S \Pr(S) \left(1 - \Pr(\widehat{E\mathcal{S}}(S) | S)\right) \\ &= 1 - \sum_S \Pr(S) \cdot \Pr(\widehat{E\mathcal{S}}(S) | S), \end{aligned} \tag{44}$$

Similarly, let $\widehat{E\mathcal{S}}_s(S)$ denote the coset picked by the sampling decoder, we can write P_S as

$$\begin{aligned} P_S &= \sum_S \Pr(S) \sum_{E\mathcal{S}} \Pr(E\mathcal{S} | S) \mathbb{1}(E\mathcal{S} \neq \widehat{E\mathcal{S}}_s(S)) \\ &= \sum_S \Pr(S) \sum_{E\mathcal{S}} \Pr(E\mathcal{S} | S) (1 - \Pr(E\mathcal{S} | S)) \\ &= \sum_S \Pr(S) \left(1 - \sum_{E\mathcal{S}} \Pr(E\mathcal{S} | S)^2\right) \\ &\leq \sum_S \Pr(S) \left(1 - \max_{E\mathcal{S}} \Pr(E\mathcal{S} | S)^2\right) \\ &= \sum_S \Pr(S) \left(1 - \Pr(\widehat{E\mathcal{S}} | S)^2\right) \\ &= 1 - \sum_S \Pr(S) \Pr(\widehat{E\mathcal{S}} | S)^2 \\ &\leq 1 - \left(\Pr(S) \Pr(\widehat{E\mathcal{S}} | S)\right)^2 \\ &= 1 - (1 - P_{\text{DQML}}^2) \leq 2 \cdot P_{\text{DQML}}. \end{aligned} \tag{45}$$

This completes the proof.