An Improved Decimation Technique for Erasure Decoding of Quantum LDPC Codes

Gayathri R.*, Shobhit Bhatnagar*, Abhinav Vaishya and P Vijay Kumar
Department of ECE, Indian Institute of Science, Bengaluru
{gayatr,shobhitb97,vaishyaabhinav,pvk1729}@gmail.com

***Abstract*—Quantum low density parity-check (QLDPC) codes represent an attractive candidate for achieving fault-tolerant quantum computation. Erasure decoding of QLDPC codes has gained traction recently as many physical systems such as neutral-atom systems, photonic systems, trapped-ion systems etc. suffer from qubit erasures. Techniques for erasure decoding of QLDPC codes via belief propagation (BP) have been proposed in the literature, such as BP with guided decimation (BP-GD), where decimation refers to sequentially fixing hard values for the variable nodes. We provide an alternative decimation-based BP algorithm, which we term the BP with degree-based decimation (BP-DD) algorithm, that intelligently chooses a check node based on its degree and subsequently decimates a variable node in its neighborhood. We apply the BP-DD algorithm to two families of QLDPC codes, namely hypergraph product codes and lifted product codes, and show improved logical error performance over the BP-GD algorithm, without incurring additional complexity. Our simulations show that the BP-DD algorithm provides a versatile solution for erasure decoding of QLDPC codes, in contrast to some techniques in the literature that are applicable to only specific classes of QLDPC codes.**

I. Introduction

Quantum information processing promises an exponential computational advantage over classical information processing. However, quantum information is more sensitive to noise as compared to classical information, and quantum error correction is indispensable to realize a practical, fault-tolerant quantum information processing system. A promising family of quantum error-correcting codes to achieve fault tolerance is the family of quantum low density parity-check (QLDPC) codes, which are the quantum anlogues of classical LDPC codes. Efforts to construct quantum LDPC codes having both high rate and large minimum distance led to many non-trivial code constructions [1]–[8]. One such construction is the hypergraph product (HGP) construction by Tillich and Zémor [9] which derives a QLDPC code out of two classical LDPC codes. The HGP construction was subsequently generalized by Pantaleev and Kalachev in [10] to obtain the family of Lifted Product (LP) codes (which are also sometimes referred to as generalized hypergraph product codes). Since qubit erasures occur in many physical systems such as neutral-atom systems [11], photonic systems [12], [13], trapped-ion systems [14], superconducting qubits [15] etc., erasure decoding of QLDPC codes has gained a lot of interest among researchers recently. However, some of the techniques proposed in the literature are applicable to only specific families of QLDPC codes. For example, the vertical-horizontal decoding algorithm proposed by Connolly et al. [16] is applicable to only HGP codes. The focus of [17] is also on optimization of HGP codes via reinforcement learning and simulated annealing for quantum erasure channels. On the other hand, techniques such as pruned peeling, again proposed by Connolly et al. in [16], and belief propagation with guided decimation (BP-GD) proposed by Gökduman et al., initially for qubit errors [18] and subsequently for qubit erasures [19], are applicable to general QLDPC codes. Here, decimation means sequentially fixing variable nodes to hard values to assist BP converge. A further set of techniques that is applicable to general QLDPC codes has been proposed by Kuo et al. in [20]. These techniques are based on a combination of gradient descent and BP, and have better worst-case complexity than BP-GD, but the optimized complexity of BP-GD is smaller than the optimized complexity of these schemes. Erasure decoding for a class of quantum error-correcting codes called color codes has been dealt with in [21]–[23]. A linear time, maximum-likelihood erasure decoder for surface codes was presented by Delfosse et al. in [24], which was later extended to a union-find decoder for topological codes in [25] and general QLDPC codes in [26] with some additional complexity.

Our Contributions: We present an improved decimation technique to assist BP decoding of QLDPC codes over the quantum erasure channel, which we refer to as the belief propagation with degree-based decimation (BP-DD) algorithm. Whenever BP fails to converge, the BP-DD algorithm intelligently chooses a check node based on the number of erased variable nodes in its neighborhood and subsequently decimates one of these variable nodes. We demonstrate through simulations the improved logical error performance of the BP-DD algorithm over the BP-GD algorithm for both HGP codes and LP codes, without incurring any additional complexity.

Organization of the paper: In Section II we provide background on the stabilizer formalism for quantum error correction and briefly describe the HGP and LP constructions of QLDPC codes. We also describe syndrome-based BP

* indicates equal contribution.

decoding of QLDPC codes over the quantum erasure channel in this section. In Section III, we describe our BP-DD algorithm and provide its pseudo-code. Simulation results are presented in Section IV. Section V draws conclusions and identifies directions for future work.

II. Preliminaries

Notation: We use I_n to denote the $(n \times n)$ identity matrix. We use $\mathbb{F}_2$ to denote the finite field of two elements. For a vector $E \in \mathbb{F}_2^n$, we use E_i to deote the i-th coordinate of E, and supp(E) to denote the support of E, i.e., supp$(E) = \{i \mid E_i \neq 0,\ i \in \{1, 2, \ldots, n\}\}$. The tensor product of two matrices A and B is denoted by $A \otimes B$.

A. Stabilizer Formalism

A qubit, the fundamental unit of quantum information, is associated with a 2-dimensional vector in the complex Hilbert space $\mathbb{C}^2$. The Pauli group on n qubits is defined as

$$\mathcal{P}_n = \{\omega P_1 \otimes P_2 \otimes \cdots \otimes P_n\},$$

where $\omega = \{\pm 1, \pm i\}$ and $P_i \in \{I_2, X, Y, Z\}$, where

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},\ Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

are the Pauli matrices. An $[[n, k]]$ stabilizer code [27] is a 2^k-dimensional subspace of the Hilbert space $(\mathbb{C}^2)^{\otimes n}$ that encodes k logical qubits into n physical qubits. It is defined as the simultaneous $+1$-eigenspace of an abelian sub-group $\mathcal{S}$ of $\mathcal{P}_n$ that does not contain $-I_{2^n}$. For a stabilizer code, the stabilizer generators are a set of operators that generate $\mathcal{S}$. For a quantum stabilizer code defined via the stabilizer group $\mathcal{S}$, the minimum distance d is defined as the minimum weight of a Pauli operator in $N(\mathcal{S}) \setminus \mathcal{S}$, where $N(\mathcal{S})$ is the normalizer of $\mathcal{S}$ in $\mathcal{P}_n$, and the weight of a Pauli operator

$$P = \omega P_1 \otimes P_2 \otimes \cdots \otimes P_n$$

is the number of indices $j \in \{1, 2, \ldots, n\}$ such that $P_j \neq I_2$. This stabilizer code is then denoted as an $[[n, k, d]]$ stabilizer code.

Calderbank-Shor-Steane (CSS) codes are a popular subclass of stabilizer codes where each stabilizer is a tensor product of either only I_2-type and X-type Pauli matrices or only I_2-type and Z-type Pauli matrices [28], [29]. Consider two classical binary codes $\mathcal{C}_X$ and $\mathcal{C}_Z$ with corresponding parity check matrices H_X and H_Z, respectively, such that $\mathcal{C}_Z^{\perp} \subseteq \mathcal{C}_X$. It follows that $H_X H_Z^T = 0$. A set (not necessarily minimal) of stabilizer generators is given by $\mathcal{S}_X \cup \mathcal{S}_Z$, where

$$\mathcal{S}_X = \{X^{a_1} \otimes X^{a_2} \otimes \cdots \otimes X^{a_n} \mid [a_1, a_2, \ldots, a_n] \in \mathcal{C}_X^{\perp}\},$$

and

$$\mathcal{S}_Z = \{Z^{b_1} \otimes Z^{b_2} \otimes \cdots \otimes Z^{b_n} \mid [b_1, b_2, \ldots, b_n] \in \mathcal{C}_Z^{\perp}\}.$$

It is easy to see that any two stabilizers in $\mathcal{S}_X$ commute with each other, as is the case for any two stabilizers in $\mathcal{S}_Z$. Further, since $H_X H_Z^T = 0$, it follows that every stabilizer in $\mathcal{S}_X$ commutes with every stabilizer in $\mathcal{S}_Z$.

When X-type and Z-type errors occur independently, in the CSS framework, they are handled separately. H_X is used to correct Z-type errors and H_Z is used to correct X-type errors. The error correction happens via syndrome computation. The syndrome for X-type errors is given by $s_X = H_Z E_X^T$ and that for Z-type errors is given by $s_Z = H_X E_Z^T$, where E_X and E_Z are the corresponding error vectors for X-type and Z-type errors, respectively.

B. Quantum LDPC Codes

QLDPC codes are quantum analogues of classical LDPC codes and correspond to the case when the H_X and H_Z matrices mentioned before are sparse in nature. We will now briefly describe two popular classes of QLDPC codes known as hypergraph product [9] codes and lifted product codes [10] as we have considered these codes for our simulations.

For HGP codes, H_X and H_Z are given by

$$H_X = (H_1 \otimes I_{n_2} \quad I_{m_1} \otimes H_2^T),$$
$$H_Z = (I_{n_1} \otimes H_2 \quad H_1^T \otimes I_{m_2}),$$

where $H_1 \in \mathbb{F}_2^{m_1 \times n_1}$ and $H_2 \in \mathbb{F}_2^{m_2 \times n_2}$ are parity check matrices of two classical binary LDPC codes.

LP codes are generalizations of HGP codes, where each scalar entry in the matrices H_X and H_Z above is replaced by a circulant matrix. This is called *lifting*. Thus, the matrices H_X and H_Z for an LP code are given by

$$H_X = (\tilde{H}_1 \otimes I_{n_2} \quad I_{m_1} \otimes \tilde{H}_2^T),$$
$$H_Z = (I_{n_1} \otimes \tilde{H}_2 \quad \tilde{H}_1^T \otimes I_{m_2}),$$

where $\tilde{H}_1$ and $\tilde{H}_2$ are $(m_1 \times n_1)$ and $(m_2 \times n_2)$ matrices, respectively, whose entries are $(L \times L)$ binary circulant matrices.

C. Quantum Erasure Channel

In the quantum erasure channel model, each qubit can be lost or erased independently with a certain probability. The loss of a qubit can be detected, and an erased qubit is replaced with a maximally mixed state $\frac{I_2}{2}$, implying that each erased qubit can be interpreted as being acted upon by the Pauli matrices I_2, X, Y, and Z with equal probability. Thus erasure correction is converted into correction of errors with known locations. We will use the vector $E \in \mathbb{F}_2^n$ to represent the erasure locations. More precisely, qubit $i \in \{1, 2, \ldots, n\}$ is erased iff $i \in$ supp(E).

D. Syndrome Decoding over the Quantum Erasure Channel

BP is a well-known, low-complexity iterative algorithm for decoding classical LDPC codes [30]–[32]. It is a message-passing algorithm that passes messages along the edges of a graph known as the Tanner graph associated with an LDPC code. Let $H \in \mathbb{F}_2^{m \times n}$ be a parity check matrix associated to a binary $[n, k \geq (n - m)]$ LDPC code $\mathcal{C}$. Then the corresponding Tanner graph $\mathcal{T}(H) = (V, C, \mathcal{E})$ is a bipartite

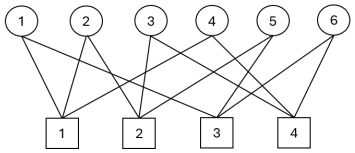

Fig. 1: Tanner graph corresponding to the parity-check matrix H in (1). Circles represent variable nodes and squares represent check nodes.

graph, where $V = \{1, 2, \ldots, n\}$ is the set of variable nodes corresponding to the n code symbols, and $C = \{1, 2, \ldots, m\}$ is the set of check nodes that correspond to the rows of H. There is an edge connecting check node i and variable node j iff $H_{i,j} = 1$, where $H_{i,j}$ denotes the (i, j)-th entry of H. The neighborhood $\mathcal{N}(v)$ of a variable node $v \in V$ is the set of all check nodes which are connected to it. Analogously, the neighborhood $\mathcal{N}(c)$ of a check node $c \in C$ is the set of all variable nodes which are connected to it. The Tanner graph for the example case when

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{1}$$

is shown in Fig. 1.

As mentioned earlier, for CSS codes, X-type and Z-type errors are handled separately. In the remainder of this paper, we will deal with only X-type errors, Z-type errors can be handled analogously. Thus we will lose the subscript and use the notation s for s_X, H for H_Z and E for E_X, so that $s = HE^T$.

Consider a quantum channel which generates X-type errors with probability p. A syndrome extraction circuit is used to obtain the syndrome $s = HE^T$. In this paper, we make the simplifying assumption that the extracted syndrome is error-free. BP tries to find an error estimate $\hat{E}$ such that $H\hat{E}^T = s$. There can be two possible outcomes:

1) Decoding is considered successful if the actual error E and the estimated error $\hat{E}$ differ by a stabilizer i.e., the Pauli operator corresponding to $(E + \hat{E})$ belongs to the stabilizer group $\mathcal{S}$.
2) The decoding is unsuccessful if either (a) BP fails to converge to an error estimate $\hat{E}$ which yields syndrome s despite having run for a predetermined maximum number of iterations or, (b) the Pauli operator corresponding to $(E + \hat{E})$ belongs to $\mathcal{N}(\mathcal{S}) \setminus \mathcal{S}$, in which case it is a logical error (an incorrectable error).

The messages exchanged between check and variable nodes in the BP algorithm are in the form of log-likelihood ratios (LLRs). As mentioned above, consider a channel where X-type errors occur with probability p. Then the channel LLRs for each variable node $v \in V$ are given by

$$\lambda_v = \ln \frac{1-p}{p}.$$

For the erasure channel, as explained before, an erased qubit is replaced with a maximally mixed state, which corresponds to the value $p = \frac{1}{2}$. Thus, the channel LLR value corresponding to erased variable nodes is 0. For unerased variable nodes, the channel LLR value is ∞ as the corresponding value of p is 0.

At the 0^{th} iteration $t = 0$ of BP, each variable node $v \in V$ passes the message $\nu_{v\to c}^{(0)}$ to each check node $c \in \mathcal{N}(v)$, where

$$\nu_{v\to c}^{(0)} = \lambda_v.$$

At the t^{th} iteration, $t \geq 1$, each check node c passes the message $\mu_{c\to v}^{(t)}$ to each variable node $v \in \mathcal{N}(c)$, where

$$\mu_{c\to v}^{(t)} = (-1)^{s_c}\ 2\ \tanh^{-1}\left(\prod_{v' \in \mathcal{N}(c)\setminus\{v\}} \tanh\left(\frac{\nu_{v'\to c}^{(t-1)}}{2}\right)\right). \tag{2}$$

Recall that $s_c,\ c \in C$, is the c^{th} bit of the syndrome vector s. Subsequently, each variable node v passes the message $\nu_{v\to c}^{(t)}$ to each check node $c \in \mathcal{N}(v)$, where

$$\nu_{v\to c}^{(t)} = \lambda_v + \sum_{c' \in \mathcal{N}(v)\setminus\{c\}} \mu_{c'\to v}^{(t)}. \tag{3}$$

Thus, apart from the 0^{th} iteration, each iteration of BP has two rounds of message passing. In the first round, messages are passed from check nodes to variable nodes according to (2), and in the second round, messages are passed from variable nodes to check nodes according to (3). After running BP for T iterations, the final message at variable node v is computed as

$$\nu_v = \lambda_v + \sum_{c' \in \mathcal{N}(v)} \mu_{c'\to v}^{(T)}.$$

To get the error estimate $\hat{E}$, we take hard decisions on the variable nodes as

$$\hat{E}_v = \begin{cases} 0, & \nu_v > 0, \\ 1, & \nu_v \leq 0. \end{cases}$$

The computational complexity of BP is $\mathcal{O}(nT)$, where a natural choice for T is $\mathcal{O}(\log n)$ [33]. The commutativity constraint on stabilizers for a stabilizer code, along with inherent degeneracy of QLDPC codes results in many short cycles in the associated Tanner graph. This hampers the convergence of BP [34], [35] and motivates the study of techniques that help BP converge.

E. The BP-GD Algorithm

The belief propagation with guided decimation algorithm was proposed in [18] to assist BP converge in the presence of qubit errors. Later, the authors modified this technique to handle qubit erasures in [19]. For the erasure case, if BP fails to converge, the soft information provided by BP is used by BP-GD to determine the most reliable variable node. Then, based on the sign of this soft information a hard value is assigned to this variable node. Following this update, BP continues to run. This procedure of decimation followed by

BP continues until either a syndrome match occurs, or there are no more variable nodes left to decimate.

In simulations, decimations are realized by appropriately updating the channel LLR of the variable node being decimated. In [19], the channel LLR for an unerased variable node v is chosen to be a fixed finite value $\lambda_v = \text{llr}_{\max} = 25$, instead of ∞. For an erased variable node v, the channel LLR is set to a small number $\lambda_v = \text{llr}_{\min} \approx 0$. If variable node v is decimated to bit 0, then its channel LLR is updated to $\lambda_v = +\text{llr}_{\max}$, and if it is decimated to bit 1, then its channel LLR is updated to $\lambda_v = -\text{llr}_{\max}$.

III. Belief Propagation with Degree-Based Decimation

In this section we will describe the BP-DD algorithm. We will first provide an overview of the algorithm and later describe it in greater detail along with the pseudo-code.

As described earlier, with channel LLRs initialized as 0 and ∞ for erased and unerased variable nodes, respectively, performing BP is equivalent to performing peeling decoding. For numerical stability of Algorithm 1 in simulations, we use the value $\text{llr}_{\max} = 25$ instead of ∞ (as in [19]) and $\text{llr}_{\min} = 10^{-5}$ instead of 0. By doing so, after we run BP for T iterations, each variable node v accumulates soft information whose absolute value gives the reliability Γ_v associated with it. Thresholding this reliability with respect to a predetermined threshold Γ yields a soft version of peeling, since for Γ close to 25, if $\Gamma_v > \Gamma$, it is reasonable to think that the variable node v is reliable and has been peeled. We will refer to such variable nodes as reliable variable nodes. Similarly, variable nodes whose associated reliabilities are lesser than or equal to Γ will be referred to as unreliable variable nodes. The set V_U in Line 15 of Algorithm 1 is the set of unreliable variable nodes.

In the BP-DD algorithm, we first run BP for T iterations to get an error estimate $\hat{E}$ and compute the corresponding syndrome $\hat{s}$. If this syndrome matches the original syndrome s, the decoder outputs $\hat{E}$ as the error estimate. If, however, $\hat{s} \neq s$, we perform the decimation step, as described next.

We search for a check node c^* having the minimum number of unreliable variable nodes in its neighborhood (note that the choice of $c^\star$ may not be unique). We denote this minimum number by $\alpha_{\min}$ (note that $\alpha_{\min}$ is the least degree of a check node in the sub-graph induced by the variable nodes in V_U). We then randomly choose one of the unreliable variable nodes in the neighborhood of c^* for decimation. The value that is assigned to this variable node is chosen to be 0 or 1 with probability $\frac{1}{2}$. In other words, we update the initial channel LLR for this variable node to either $+\text{llr}_{\max}$ or $-\text{llr}_{\max}$ with probability $\frac{1}{2}$.

We consider the following cases. In the first case, we have $\alpha_{\min} = 2$, and peeling (via BP) continues after decimation. We continue running BP for T iterations and perform the syndrome check with the new error estimate. If syndrome match does not occur, we again go to the decimation step. In the second case, we have $\alpha_{\min} > 2$, and further peeling is not possible even after decimation. In this case, we decimate a variable node connected to c^* and let the iterative decoding continue. In rare scenarios, due to the choice of the threshold Γ, we may encounter a third case where $\alpha_{\min} = 1$. In this case, the single unreliable variable node in $\mathcal{N}(c^*)$ gets decimated and the algorithm continues. The iterative decoding procedure continues until either the syndrome match happens or all the unreliable variable nodes have undergone decimation.

We emphasize here that the choice of the variable node for decimation is made to potentially enable peeling (via BP) to be continued whenever possible, which, as mentioned before, is different from the choice adopted in [19], where the most reliable variable node is chosen for decimation.

Algorithm 1 Belief Propagation with Degree-Based Decimation (BP-DD)

Input: Set V_E of erased variable nodes, syndrome s, Tanner graph $\mathcal{T}(H) = (V, C, \mathcal{E})$, Set $\mathcal{S}_C$ of check nodes connected to at least one erased variable node

Output: Estimated $\hat{E}$ or non-convergence

1: **for** $v \in V$ **do**
2: **if** $v \in V_E$ **then**
3: $\lambda_v = \text{llr}_{\min}$
4: **else**
5: $\lambda_v = \text{llr}_{\max}$
6: **end if**
7: $\nu_{v\to c}^{(0)} \leftarrow \lambda_v$ for all $c \in \mathcal{N}(v)$
8: **end for**
9: **while** $|V_E| > 0$ **do**
10: perform BP for T iterations
11: $\hat{E} \leftarrow$ hard values of all variable nodes
12: **if** $H\hat{E}^T = s$ **then**
13: **return** $\hat{E}$
14: **else**
15: Set $V_U = V_E \setminus \{v \mid \Gamma_v > \Gamma\}$
16: find $c^* = \arg\min_{c\in\mathcal{S}_C} |\mathcal{N}(c) \cap V_U|$ ▷ c^* may not be unique
17: $\alpha_{\min} = |\mathcal{N}(c^*) \cap V_U|$
18: **if** $\alpha_{\min} = 2$ **then**
19: $\mathcal{S}_C \leftarrow \mathcal{S}_C \setminus \{c^*\}$
20: **end if**
21: pick $v^* \in \mathcal{N}(c^*) \cap V_U$ uniformly at random
22: pick $u \in \{0, 1\}$ uniformly at random
23: $\lambda_{v^*} \leftarrow (-1)^u \text{llr}_{\max}$
24: $V_E \leftarrow V_E \setminus \{v^*\}$
25: **end if**
26: **end while**
27: **return** non-convergence

The pseudo-code for the BP-DD algorithm is provided in Algorithm 1. As mentioned before, for numerical stability we set the value of $\text{llr}_{\max}$ to be 25 and that of $\text{llr}_{\min}$ to be

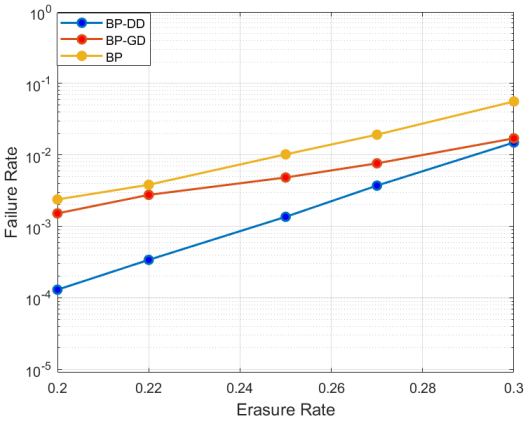

Fig. 2: Performance comparison between BP-DD, BP-GD and vanilla BP for the $[[2025, 81]]$ HGP code.

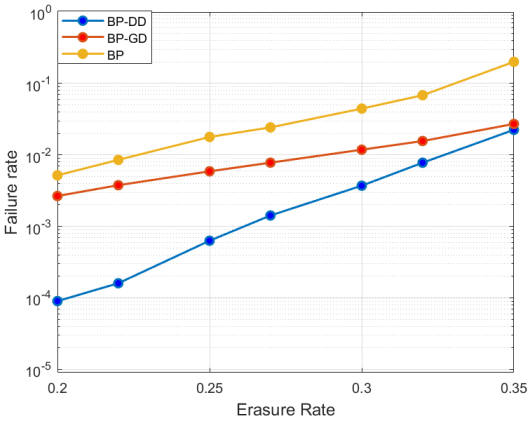

Fig. 3: Performance comparison between BP-DD, BP-GD and vanilla BP for the $[[882, 24, 18 \leq d \leq 24]]$ B1 LP code.

10^{-5}. Further, as in [18], [19], we limit the values of the messages passed from variable to check nodes as

$$\nu_{v\to c}^{(l)} = \begin{cases} 25, & \text{if } \nu_{v\to c}^{(l)} > 25, \\ \nu_{v\to c}^{(l)}, & \text{if } \nu_{v\to c}^{(l)} \in [-25, 25], \\ -25, & \text{if } \nu_{v\to c}^{(l)} < -25. \end{cases}$$

In Line 16 of Algorithm 1, we find a check node from the set $\mathcal{S}_C$, having the minimum number of unreliable variable nodes in its neighborhood, where $\mathcal{S}_C$ is the set of check nodes having at least one erased variable node in their neighborhood. As mentioned earlier, this choice may not be unique. In our simulations we sequentially go over each check node and count the number of unreliable variable nodes in its neighborhood, and then choose c^* to be the first check node that we encountered having the minimum count.

IV. Simulation Results

Figure 2 and Figure 3 show the performance comparison between the BP-DD, BP-GD and vanilla BP schemes for various erasure rates for the $[[2025, 81]]$ HGP code and the $[[882, 24, 18 \leq d \leq 24]]$ B1 LP code, respectively. The $[[2025, 81]]$ HGP code was obtained from the GitHub repository in [16] and the B1 code from Appendix B in [10]. In Fig. 2 and Fig. 3, we plot the failure rate on the Y-axis, which includes failures due to both logical errors and non-convergence (i.e., syndrome mismatch). The value of the threshold Γ used to obtain these plots is $\Gamma = 20$.

We now discuss the computational complexity of the BP-DD algorithm. The number of decimation rounds is at most $|V_E|$, which is $\mathcal{O}(n)$. As mentioned before, the complexity of running BP for T iterations is $\mathcal{O}(nT)$. Thus, the worst-case complexity of BP-DD is $O(n^2T)$. This is the same as that for BP-GD [18]. However, in our simulations we observed that the number of decimation rounds required when BP failed to converge was very small (1 or 2 rounds) in most cases. Thus, in the typical case the complexity of the BP-DD algorithm is much smaller than $\mathcal{O}(n^2T)$.

We remark here that the plot we generated for BP-GD in Fig. 2 is consistent with the corresponding plot in [19], however, the BP-GD plot for the B1 code in Fig. 3 slightly deviates from that in [19]. This could possibly be attributed to the simulation settings. For example, in our simulations we run BP for $T = \log n$ iterations, where n is the length of the code. However, changing T can result in variation in performance. Nonetheless, even when compared with the plots in [19] BP-DD has better performance.

V. Conclusion and Future Work

We presented a decimation-based BP algorithm, which we called the BP-DD algorithm, for erasure decoding of QLDPC codes. The BP-DD algorithm intelligently chooses a check node based on its degree and subsequently decimates one of its neighboring variable nodes. We simulated the BP-DD algorithm for two families of QLDPC codes, namely, hypergraph product codes and lifted product codes, and showed improved logical error performance over the BP-GD algorithm known in prior literature. Further, the complexity of the BP-DD algorithm is similar to that of the BP-GD algorithm. Also, unlike some techniques known in the literature that are applicable only to specific families of QLDPC codes, the BP-DD algorithm provides good performance for general QLDPC codes.

As mentioned in [19], modifications of the BP-GD algorithm such as adjusting the channel LLRs of each variable node by multiplying them with a suitable constant, or taking a weighed sum of the present and the previous estimate at a variable node (damping), or a combination of both adjusting and damping, result in improved performance of the BP-GD algorithm. However, the optimal values of the parameters for adjusting and damping are specific to the choice of the QLDPC code as well as the erasure rate, and determining them requires extensive simulations. The effect of incorporating such modifications to the proposed BP-DD algorithm is left for future work. Comparison between BP-DD and its suitable modifications with the gradient-descent-based approaches in [20] is also considered for future work.

REFERENCES

[1] M. B. Hastings, J. Haah, and R. 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*, 2021, pp. 1276–1288.

[2] N. P. Breuckmann and J. N. Eberhardt, "Balanced product quantum codes," *IEEE Transactions on Information Theory*, vol. 67, no. 10, pp. 6653–6674, 2021.

[3] P. Panteleev and G. Kalachev, "Quantum LDPC codes with almost linear minimum distance," *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 213–229, 2021.

[4] ——, "Asymptotically good quantum and locally testable classical LDPC codes," in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, 2022, pp. 375–388.

[5] A. Leverrier and G. Zémor, "Quantum tanner codes," in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2022, pp. 872–883.

[6] ——, "Decoding quantum tanner codes," *IEEE Transactions on Information Theory*, vol. 69, no. 8, pp. 5100–5115, 2023.

[7] I. Dinur, M.-H. Hsieh, T.-C. Lin, and T. Vidick, "Good quantum LDPC codes with linear time decoders," in *Proceedings of the 55th annual ACM symposium on theory of computing*, 2023, pp. 905–918.

[8] S. Gu, C. A. Pattison, and E. Tang, "An efficient decoder for a linear distance quantum LDPC code," in *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, 2023, pp. 919–932.

[9] J.-P. Tillich and G. Zémor, "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Transactions on Information Theory*, vol. 60, no. 2, pp. 1193–1202, 2013.

[10] P. Panteleev and G. Kalachev, "Degenerate quantum LDPC codes with good finite length performance," *Quantum*, vol. 5, p. 585, 2021.

[11] Y. Wu, S. Kolkowitz, S. Puri, and J. D. Thompson, "Erasure conversion for fault-tolerant quantum computing in alkaline earth Rydberg atom arrays," *Nature communications*, vol. 13, no. 1, p. 4657, 2022.

[12] E. Knill, R. Laflamme, and G. J. Milburn, "A scheme for efficient quantum computation with linear optics," *nature*, vol. 409, no. 6816, pp. 46–52, 2001.

[13] S. Bartolucci, P. Birchall, H. Bombin, H. Cable, C. Dawson, M. Gimeno-Segovia, E. Johnston, K. Kieling, N. Nickerson, M. Pant *et al.*, "Fusion-based quantum computation," *Nature Communications*, vol. 14, no. 1, p. 912, 2023.

[14] M. Kang, W. C. Campbell, and K. R. Brown, "Quantum error correction with metastable states of trapped ions using erasure conversion," *PRX Quantum*, vol. 4, no. 2, p. 020358, 2023.

[15] A. Kubica, A. Haim, Y. Vaknin, H. Levine, F. Brandão, and A. Retzker, "Erasure qubits: Overcoming the T_1 limit in superconducting circuits," *Physical Review X*, vol. 13, no. 4, p. 041022, 2023.

[16] N. Connolly, V. Londe, A. Leverrier, and N. Delfosse, "Fast erasure decoder for hypergraph product codes," *Quantum*, vol. 8, p. 1450, 2024.

[17] B. Freire, N. Delfosse, and A. Leverrier, "Optimizing hypergraph product codes with random walks, simulated annealing and reinforcement learning (2025)," *arXiv preprint arXiv:2501.09622*.

[18] H. Yao, W. A. Laban, C. Häger, A. G. i Amat, and H. D. Pfister, "Belief propagation decoding of quantum LDPC codes with guided decimation," in *2024 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2024, pp. 2478–2483.

[19] M. Gökduman, H. Yao, and H. D. Pfister, "Erasure decoding for quantum LDPC codes via belief propagation with guided decimation," in *2024 60th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2024, pp. 1–8.

[20] K.-Y. Kuo and Y. Ouyang, "Degenerate quantum erasure decoding," *arXiv preprint arXiv:2411.13509*, 2024.

[21] S. Lee, M. Mhalla, and V. Savin, "Trimming decoding of color codes over the quantum erasure channel," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 1886–1890.

[22] H. M. Solanki and P. K. Sarvepalli, "Correcting erasures with topological subsystem color codes," in *2020 IEEE Information Theory Workshop (ITW)*. IEEE, 2021, pp. 1–5.

[23] ——, "Decoding topological subsystem color codes over the erasure channel using gauge fixing," *IEEE Transactions on Communications*, vol. 71, no. 7, pp. 4181–4192, 2023.

[24] N. Delfosse and G. Zémor, "Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel," *Physical Review Research*, vol. 2, no. 3, p. 033042, 2020.

[25] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *Quantum*, vol. 5, p. 595, 2021.

[26] N. Delfosse, V. Londe, and M. E. Beverland, "Toward a union-find decoder for quantum LDPC codes," *IEEE Transactions on Information Theory*, vol. 68, no. 5, pp. 3187–3199, 2022.

[27] D. Gottesman, *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

[28] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, no. 2, p. 1098, 1996.

[29] A. M. Steane, "Error correcting codes in quantum theory," *Physical Review Letters*, vol. 77, no. 5, p. 793, 1996.

[30] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[31] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[32] T. Richardson and R. Urbanke, "Modern Coding Theory, 2005," *available at lthcwww. epfl. ch/mct*.

[33] A. Dembo and A. Montanari, "Ising models on locally tree-like graphs," 2010.

[34] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *arXiv preprint arXiv:0801.1241*, 2008.

[35] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen years of quantum LDPC coding and improved decoding strategies," *iEEE Access*, vol. 3, pp. 2492–2519, 2015.