Learning to Decode Trapping Sets in QLDPC Codes

Asit Kumar Pradhan, Nithin Raveendran, Narayanan Rengaswamy, Xin Xiao, and Bane Vasić
Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ, 85721 USA
Email: {asitpradhan, nithin , narayananr , 7xinxiao7 }@arizona.edu, vasic@ece.arizona.edu

2023 12th International Symposium on Topics in Coding (ISTC) | 979-8-3503-2611-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISTC57237.2023.10273526

***Abstract*—Quantum low-density parity-check (QLDPC) codes with asymptotically nonzero rates are promising candidates for fault-tolerant quantum computation. Belief propagation (BP) based iterative decoding algorithms, a primary choice for classical LDPC codes, perform poorly for QLDPC codes due to stabilizer-induced trapping sets, resulting in a high error floor. Several decoding algorithms, like post-processing decoders, normalized BP decoders, and neural decoders, have been proposed to increase the performance in the error-floor region. However, this improvement comes at the expense of an increase in the execution time of the decoder. This paper proposes a general framework for error correction for a class of QLDPC codes called lifted-product codes using recurrent neural networks (RNNs). The RNN is employed to learn message-passing rules that can decode quantum-trapping sets. Then the standard message-passing rules are used with the learned rules to improve the error floor. While training the RNN, the quasi-cyclic property of the lifted product codes is exploited to reduce the size of the training set and the number of parameters in the network. This reduction in the number of parameters makes these decoders amenable to hardware implementation. Simulation results show that the proposed decoder performs better than the existing decoders in the literature.**

I. Introduction

Quantum low-density parity-check (QLDPC) codes are a promising candidate for both quantum computing and communications, with a history of success in classical LDPC codes in admitting low-complexity decoding and near-capacity performance. As pointed out by Gottesman [1] and Kovalev and Pryadko [2], QLDPC codes are the only known class of quantum error correction (QEC) codes that permit fault-tolerant error correction with asymptotically nonzero rate. In [3], Panteleev and Kalachev propose a family of QLDPC codes with linear minimum distance and constant rate, known as *lifted-product* codes. QLDPC codes [4] based on the stabilizer formalism [5] rely on classical decoding algorithms with the syndrome measurements. In addition to the excellent distance properties of QLDPC codes, these codes have low-weight stabilizer generators; hence, their syndrome-extraction circuits have low depth, making them lucrative for fault-tolerant quantum computation.

For fault-tolerant computation, in addition to having good codes, designing low-complexity decoders is paramount. The low-complexity iterative message-passing algorithms do not perform well on the QLDPC codes, unlike their classical counterparts. This is mainly due to two types of trapping sets. First, since most of the QLDPC codes are constructed using tensor products, their corresponding Tanner graphs have inevitable short cycles, resulting in trapping sets (TSs) [6]. Second, QLDPC codes can be thought of as two dual-containing classical LDPC codes. This dual-containing property leads to a special type of trapping sets known as *symmetric stabilizer* TSs [6]. Several decoders have been proposed to address the decoder convergence due to the issues mentioned above [7]–[11]. Poulin and Chung [7] investigated heuristic methods to break the symmetric input channel values to improve decoding performance. In [12], the authors use a post-processing decoder called an ordered statistics decoder (OSD) after running a few iterations of the message-passing decoder to improve the decoding performance. In another work [9], the authors use the message-passing decoder in parts of the Tanner graph where TSs are absent and use a post-processing step to correct errors in the rest of the graph. In both approaches [9], [12], the respective post-processing step involves inverting a matrix, which may lead to high decoding complexity in some cases. In [10], authors show that normalized belief-propagation decoders with a serial schedule can avoid TSs when the normalization constant is chosen carefully. In a recent work, [13], Poulin's group used a neural network-based decoder with a different loss function instead of the generally used binary cross entropy to tackle the TSs. In [11], the authors use an overcomplete parity-check matrix to avoid decoding failures due to short cycle-related trapping sets. Both these neural network-based approaches use different message-update rules over decoding iterations, hence, are not easily amenable to hardware implementation. Also, the neural network-based approaches are unsuitable for decoding moderate-length codes (around a few thousand) due to the high training time required.

This paper focuses on designing parallel, TS-aware, message-passing decoders for QLDPC codes that do not require post-processing. The TSs corresponding to a standard message-passing decoder are collected to do so. Then, a sequence of decoders specialized in correcting error patterns that form TS is learned using recurrent neural networks (RNNs). Since our approach uses the same message-update rules across iterations, it is amenable to hardware implementation. Also, using the knowledge of TSs reduces training time, an essential criterion for learning decoders for code with moderate block-length, and helps us to achieve better performance without any post-processing step.

A. Notations

We use bold face capital letters to denote matrices and bold face small letters to vector variables. We denote cardinality of set $\mathcal{A}$ by $|\mathcal{A}|$. We will assume that vectors without transposes

are row vectors unless stated otherwise. We represent the absolute value of a scalar variable by $|\cdot|$.

II. Preliminaries

A. Depolarizing Channel

We focus on the widely studied channel model called depolarizing channel (memoryless Pauli channel), characterized by the depolarizing probability p in which the error $\mathbf{E}$ on each qubit is a Pauli operator, and error on a qubit is independent of the error on other qubits. The set of Pauli operators is given by $\mathcal{P} = \{\mathbf{I}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$. In particular, $\Pr(\mathbf{E} = \mathbf{X}) = \Pr(\mathbf{E} = \mathbf{Y}) = \Pr(\mathbf{E} = \mathbf{Z}) = p/3, \Pr(\mathbf{E} = \mathbf{I}) = 1 - p$. A Pauli error vector on the n qubits can be expressed as a binary error vector of length $2n$ by mapping the Pauli operators to binary tuples as follows: $\mathbf{I} \to (0,0), \mathbf{X} \to (1,0), \mathbf{Z} \to (0,1), \mathbf{Y} \to (1,1)$.

B. Stabilizer Formalism

Let us denote the n-qubit Pauli group by $\mathcal{P}_n = i^l\{\mathbf{I}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}\}^{\otimes n}$, $0 \leq l \leq 3$, where $\otimes n$ is the n-fold tensor product, $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ are the Pauli matrices, $\mathbf{I}$ is the 2×2 identity matrix, and i^l is the phase factor. Let $\mathcal{S} = \langle \mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_m \rangle$, $-1 \notin \mathbf{S}$, be an Abelian subgroup of $\mathcal{P}_n$ with generators $\mathbf{S}_i$, $1 \leq i \leq m$. A (n, k) quantum stabilizer code [14] is a 2^k-dimensional subspace $\mathcal{C}$ of the Hilbert space $(\mathbb{C}^2)^{\otimes n}$ given by the common $+1$ eigenspace of stabilizer group $\mathcal{S}$:

$$\mathcal{C} = \{|\psi\rangle, \quad \text{s.t. } \mathbf{S}_i |\psi\rangle = |\psi\rangle, \forall i\}. \tag{1}$$

Every element of stabilizer group $\mathcal{S}$ is mapped to a binary tuple as follows: $\mathbf{I} \to (0,0)$, $\mathbf{X} \to (1,0)$, $\mathbf{Z} \to (0,1)$, $\mathbf{Y} \to (1,1)$. This mapping gives a matrix representation of the stabilizer generators called parity-check matrix, $\mathbf{H}$, which is given by $\mathbf{H} = \left[\mathbf{H}_{\mathrm{X}} \mid \mathbf{H}_{\mathrm{Z}}\right]$, where $\mathbf{H}_{\mathrm{X}}$ and $\mathbf{H}_{\mathrm{Z}}$ represent binary matrices for bit flip and phase flip operators, respectively. Note that $\mathbf{H}$ is a $m \times 2n$ matrix. Similar to the Pauli representation, the stabilizers also commute with respect to the *symplectic inner product* in binary representation [15].

C. Protograph representation of lifted-product QLDPC codes

Lifted product codes are constructed using two classical protograph-based LDPC codes. In the next section, we briefly introduce protograph ensemble of LDPC codes.

1) Protograph LDPC codes: A protograph $G = (V \cup C, E)$ is a bipartite graph, where V (C) is the set of variable (respectively, check) nodes, and E is the set of edges that connect a variable node in V to a check node in C. The nodes and edges in the protograph are ordered, and the i-th variable node, check node, and edge in the protograph are denoted, respectively, by v_i, c_i, and e_i. The variable and check nodes connected by an edge e_i are denoted $v(e_i)$ and $c(e_i)$, respectively. A protograph can be represented by a base matrix $\mathbf{B}$ of dimension $|C| \times |V|$, whose (i,j)-th element $\mathbf{B}(i,j)$ is the number of edges between c_i and v_j. For example, consider a base matrix

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{2}$$

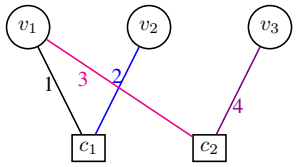

Fig. 1. Protograph for base matrix in (2).

The 4 different edges in this example are numbered, as shown in Fig. 1. The parity-check matrix of a quasi-cyclic LDPC code can be obtained from base graph $\mathbf{B}$ by replacing its entries with $l \times l$- circulants, which can be represented by the elements of quotient polynomial ring $\mathbb{R}[x]/(x^l - 1)$. This procedure is known as lifting. We denote the Tanner graph and parity-check matrix of the quasi-cyclic LDPC code obtained by lifting protograph G by G' and $\mathbf{H}$, respectively. Recall that the entries of a base (parity-check) matrix represent the edges in the corresponding protograph (Tanner graph). In the Tanner graph, the group of edges obtained by replacing the entry corresponding to edge e_i in protograph G with a $l \times l$ circulant is said to be of type e_i or, simply, type i.

2) Lifted-product codes: Given two classical base matrices $\mathbf{B}_1$ and $\mathbf{B}_2$, respectively, of size $m_{B_1} \times n_{B_1}$ and $m_{B_2} \times n_{B_2}$, two base matrices are constructed [3] as

$$\mathbf{B}_{\mathrm{x}} = \begin{bmatrix} \mathbf{B}_1 \otimes \mathbf{I}_{m_{B_2}} & \mathbf{I}_{m_{B_1}} \otimes \mathbf{B}_2 \end{bmatrix}$$
$$\mathbf{B}_{\mathrm{z}} = \begin{bmatrix} \mathbf{I}_{n_{B_1}} \otimes \mathbf{B}_2^{\mathsf{T}} & \mathbf{B}_1^{\mathsf{T}} \otimes \mathbf{I}_{n_{B_2}} \end{bmatrix}.$$

These two base matrices $\mathbf{B}_{\mathrm{x}}$ and $\mathbf{B}_{\mathrm{z}}$ can be lifted as described in Section II-C1 to obtain two parity-check matrices $\mathbf{H}_{\mathrm{x}}$ and $\mathbf{H}_{\mathrm{z}}$, respectively. Similar to classical protograph-based LDPC codes, the edges of $\mathbf{H}_{\mathrm{x}}$ and $\mathbf{H}_{\mathrm{z}}$ can be classified into different types, as described in Section II-C1. Then, $\mathbf{H}$ is the set of stabilizer generators of a lifted-product QLDPC code constructed from two classical base matrices $\mathbf{B}_1$ and $\mathbf{B}_2$. If the number of columns in $\mathbf{H}$ is N, and the number of parity-check equations is M, then $n = l(n_A n_B + m_A m_B)$ and $m = l(m_A n_A + m_B n_B)$. Assuming that $\mathbf{H}_{\mathrm{x}}$ and $\mathbf{H}_{\mathrm{z}}$ are full rank, we have a $(n, n-m)$-QLDPC code. The protograph corresponding to $\mathbf{B}_{\mathrm{x}}$ is shown in Fig. 2 when $\mathbf{B}_1$ and $\mathbf{B}_2$ are chosen as $\mathbf{B}$ given in Eq. (2).

III. Message-passing Decoder

In what follows, we assume that $\mathbf{X}$ and $\mathbf{Z}$ errors are independent. We describe only the decoding of $\mathbf{Z}$ errors for ease of exposition. Let us consider the standard message-passing decoder over a depolarizing channel with depolarizing probability p run on Tanner graph $G'_{\mathrm{x}}(V' \cup C', E')$ derived from a protograph $G_{\mathrm{x}} = (V \cup C, E)$. Given an error vector $\mathbf{x}$, the syndrome vector $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_{|C'|})$ can be obtained by $\boldsymbol{\sigma} = \mathbf{x}\mathbf{H}_{\mathrm{x}}^{\mathsf{T}}$, where $\mathbf{x}$ denotes the error pattern. We define a distinct message-passing rule for each group of edges in the lifted graph G'_{x}. Since the lifted graph G'_{x} is lifted from protograph $G_{\mathrm{x}} = (V \cup C, E)$, G'_{x} has $|E|$ edge types, one for each edge in the protograph $G_{\mathrm{x}} = (V \cup C, E)$. Therefore, we have $|E|$ message-passing rules, one for each edge type. Let

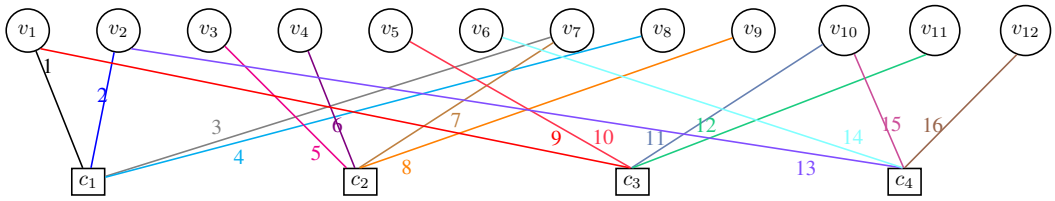

Fig. 2. Protograph for base matrix $\mathbf{B}_x$ when $\mathbf{B}_1$ and $\mathbf{B}_2$ are chosen as the base matrix in (2). Since the protograph has 16 edges, the LDPC code lifted from it will have 16 types of edges.

E'_i, $i \in \{1, 2, \ldots, |E|\}$, denote the set of type-i edges in the lifted graph $G'_{\mathrm{x}}(V' \cup C', E')$. Note that $E' = \bigcup_{i=1}^{|E|} E'_i$.

Let $m^t_{v\to c}(e)$ be the message from variable node to check node along edge e for $e \in E'_i$, $i \in \{1, 2, \ldots, |E|\}$, in the t-th iteration. Similarly, let $m^t_{c\to v}(e)$ be the message from check node to variable node in the t-th iteration. The message-passing recursion is given by

$$m^{t+1}_{c\to v}(e) = \sigma_{c_e} \prod_{e'\in E_c(e)} m^t_{v\to c}(e') \min_{e'\in E_c(e)} |m_{v\to c}(e')|, \quad (3)$$

$$m^{t+1}_{v\to c}(e) = b_i + w\left(\sum_{e'\in E_v(e)} m^{t+1}_{c\to v}(j)\right), \quad (4)$$

for $t \geq 0$, where $E_c(e) = \{e' : c(e) = c(e'), e \neq e'\}$ and $E_v(e) = \{e' : v(e) = v(e'), e \neq e'\}$ are the set of neighboring edges incident to the same check node and variable node, respectively, as the edge e, b_i, for $1 \leq i \leq |E|$, is the bias corresponding to edge type i, and w is the normalization constant. Note that we have a different update rule for each edge type at the variable node, while the update rule at the check nodes is the same for all edge types. We can recover the min-sum decoder by choosing $b_i = \log \frac{1-p}{p}$ and $w = 1$. After a pre-defined number of iterations, $t_{\max}$, of iterative syndrome decoding, we declare that the decoder failed for a particular input syndrome or error pattern if the decoder cannot find an error pattern matching the input syndrome. More precisely, a decoder failure is said to have occurred if there does not exist $t \leq t_{max}$ such that $(\hat{\boldsymbol{\sigma}}^{(t)} + \boldsymbol{\sigma}) = \mathbf{0}$, where $\hat{\boldsymbol{\sigma}}^{(t)}$ is the syndrome after the t-th iteration.

A. Quantum-trapping sets

In this section, we introduce the notion of TSs and describe how their knowledge can be used to design better decoders. During the iterative decoding, check node c is satisfied if there exists a positive integer I_{j} such that for all $t \geq I_{\mathrm{j}}$, $\hat{\sigma}^{(t)}_j = \sigma_j$. We say that variable node v_i has converged if there exists a positive integer I_{i} such that for all $t \geq I_{\mathrm{i}}$, $\hat{x}^{(t)}_i = \hat{x}^{(t-1)}_i$. Note that the $\hat{x}^{(t)}_i$ is not necessarily the correct estimate of error on the $i-$th variable node. With these definitions, we define quantum TS (QTS) as follows:

Definition 1. A trapping set $\mathcal{T}_s$ for a syndrome-based iterative decoder $\mathcal{D}_s$ is a non-empty set of variable nodes in a Tanner graph G' which have not converged or are neighbors of the check nodes that are not satisfied.

If the sub-graph $\mathcal{G}(\mathcal{T}_s)$ induced by such a set of variable nodes has a variable nodes and b unsatisfied check nodes, then $\mathcal{T}_s$ is classified as an (a, b) trapping set.

The QTSs, which are similar to the TSs in classical LDPC codes, are called *classical-type* trapping sets [6]. The second class of trapping sets is the harmful degenerate errors observed in the support of the stabilizers, called symmetric stabilizer trapping sets or stabilizer splitting errors. In such trapping sets, even though the variable nodes eventually converge to some error pattern, there exist check nodes that are not satisfied. Next, we briefly describe such trapping sets. To appreciate the behavior of *symmetric stabilizer trapping sets*, we first describe the subtle difference in decoding classical codes and quantum codes. In the classical case, the estimate of error pattern $\hat{\mathbf{x}}$ must be equal to the actual error pattern $\mathbf{x}$, whereas in quantum error-correction, the modulo-2 sum of the actual and estimated error pattern, $\mathbf{x} \oplus \hat{\mathbf{x}}$, must be a stabilizer- the space spanned by the rows of the parity-check matrix. This is because the error patterns that only differ by a stabilizer have the same effect on the codespace. Therefore, while decoding $\mathbf{H}_{\mathrm{x}}$, the decoder needs to identify any recovery operator such that $\hat{\mathbf{x}} \oplus \mathbf{x} \in$ *rowspace*$(\mathbf{H}_{\mathrm{z}})$. In decoding of quantum codes, we say error vectors $\mathbf{x}$ and $\mathbf{y}$ are degenerate if $\mathbf{x} \oplus \mathbf{y}$ is a stabilizer; hence, for successful decoding, it is sufficient to output any one of the degenerate errors as the candidate error pattern. However, some degenerate errors can be detrimental to the iterative decoding of QLDPC codes, especially when the minimum distance of the code is higher than its stabilizer weight. For example, if the stabilizer-induced sub-graph contains degenerate error patterns $\mathbf{x}$ and $\mathbf{y}$ of equal weight and has a symmetric topology, then the iterative decoder oscillates between $\mathbf{x}$ and $\mathbf{y}$, thus not matching the input syndrome. This failure can be attributed to the symmetry in the stabilizer and message-passing rules. Hence, such errors are referred to as symmetric degenerate errors, and corresponding sets of variable nodes as symmetric stabilizer trapping sets or in short, symmetric stabilizers. Although degenerate errors are typically classified as harmless while decoding quantum codes, from the above discussion, it follows that some (not all) degenerate error patterns in a symmetric stabilizer are harmful to iterative decoders.

B. Decoder Diversity

We follow the decoder diversity from [16]. Consider a set of decoders $\mathcal{D} = \{\mathcal{D}^k : k = 1, 2, \ldots, N_{\mathcal{D}}\}$ to decode a QLDPC code defined by the Tanner graph $G'(V' \cup C', E)$

with $|E|$ types of edges. Since the decoders in $\mathcal{D}$ are designed to decode a QLDPC code with $|E|$ types of edges, each $\mathcal{D}^k \in \mathcal{D}$ is parameterized by $\mathbf{b}^k = \{b_1, b_2, \ldots, b_{|E|}\}$, where b_is, $1 \leq i \leq |E|$, are biases corresponding to different edge types, as defined in Eq. (4). Given the number of decoders $N_{\mathcal{D}}$, the objective is to design a set of decoders, $\mathcal{D}$, that minimizes the logical error rate. These decoders can be used in a parallel or sequential fashion, depending on the memory and throughput constraints. In the next section, we employ RNNs to learn asymmetries in the message-passing rules to decode *symmetric stabilizer trapping sets*.

IV. Decoder Diversity by RNNs

A. A sequential framework

In this section, we describe a framework to choose a decoder set $\mathcal{D}$ that minimizes the logical-error rate when the decoders in $\mathcal{D}$ are used sequentially. We start by choosing $\mathcal{D}^0$ as the min-sum decoder. Let us denote the set of error patterns, with weight equal to or less than w, on which the min-sum decoder fails by $\mathcal{P}_w$. These error patterns can be obtained efficiently by employing the expansion-contraction method in [17], without fully simulating the min-sum decoder. To learn biases, $\mathbf{b}^1$, corresponding to decoder $\mathcal{D}^1$, the error patterns in set $\mathcal{P}_w$ are used as the training set, with the objective of correcting as many error patterns as possible from $\mathcal{P}_w$. Let us denote by $\mathcal{P}^k$ the set of error patterns from $\mathcal{P}_w$ on which decoders $\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^{k-1}$ fail. Let $\mathcal{P}_{\mathcal{D}^k}$ denote the set of error patterns from $\mathcal{P}_w$ that decoder $\mathcal{D}^k$ can correct. Define $\mathcal{P}^{k+1} = \mathcal{P}^k \setminus \mathcal{P}_{\mathcal{D}^k}$. Then, $\mathcal{P}^{k+1}$ is used as the training set for decoder $\mathcal{D}^{k+1}$. This process continues for $k = 1, 2, \ldots, N_{\mathcal{D}} - 1$.

Because of the sequential training strategy, the decoders in $\mathcal{D}$ are used in the same order as they are determined. Specifically, assume that the predefined maximum number of iterations corresponding to decoder $\mathcal{D}^k$ is I_k. To decode the error pattern corresponding to syndrome $\boldsymbol{\sigma}$, $\mathcal{D}^0$ with I_0 iterations is first applied. If $\mathcal{D}^0$ decodes successfully, the decoding process terminates, otherwise the syndrome is re-initialized to $\boldsymbol{\sigma}$ and the process switches to decoder $\mathcal{D}^1$. This process continues until a decoder $\mathcal{D}^k \in \mathcal{D}$ converges. Otherwise, all the decoders in $\mathcal{D}$ fail, and a decoding failure is declared.

B. Training RNNs

We use recurrent-neural networks with three hidden layers and depth I_k to learn decoder $\mathcal{D}_k$ in $\mathcal{D}$, where I_k is the pre-defined number of iterations for decoder $\mathcal{D}_k$. The activation function of three hidden layers represents the message-updating rules at variable nodes, check nodes, and bit-likelihood estimators, respectively. The Tanner graph of the code determines the connections between the hidden layers. To imitate the update rules of the min-sum decoder, the activation functions at variable nodes and check nodes are chosen to be of the same form as in Eq. (3)-(4). Note that the message-passing rules at check nodes are not parameterized; hence, they are fixed for all the edges and iterations. While training, the bias vector $\mathbf{b}^k$, which defines the update rules at the variable nodes, is optimized to increase error-correction capability. For a fixed edge, the re-usage of message-updating rules over iteration leads to the recurrent structure of the neural network. This is different from most of the existing NN-based decoders, wherein the update rules vary across iterations.

While training, we exploit the quasi-cyclic nature of the lifted-product codes to reduce the size of the training set and the number of parameters in the RNN-based decoder. Consider the Tanner graph $G'(V' \cup C', E')$ corresponding to a lifted-product code lifted from protograph G using $l \times l$-circulants. If a weight-w error pattern on variable nodes in $\{v_{i_1}, v_{i_2}, \ldots v_{i_w}\}$ can be estimated by decoder $\mathcal{D}^k$ in I_k iterations, then it can also decode the error pattern $\{v_{i_1+1}, v_{i_2+1}, \ldots v_{i_w+1}\}$ in the same number of iterations, where the additions in the subscript are in modulo-l. To see the above statement, consider the depth-I_k computation graphs $\mathcal{C}_{i_k}$ and $\mathcal{C}_{i_k+1}$ with v_{i_k} and v_{i_k+1}, for $k \in \{1, 2, \ldots w\}$, respectively, as their root nodes. Due to the quasi-cyclic nature of lifted-product codes, it can be concluded that $\mathcal{C}_{i_k}$ and $\mathcal{C}_{i_k+1}$ are isomorphic. Also, observe that message-passing rules at the nodes in computation graph $\mathcal{C}_{i_k}$ and its isomorphic copy $\mathcal{C}_{i_k+1}$ are the same. Therefore, the above statement is true. As a consequence, the training set of each decoder can be reduced by a factor l without compromising their error correction capability. Also, note that the training methodology reduces the number of parameters to be learnt by a factor of l, i.e., instead of learning $|E'|$ parameters, now, it is required to learn the parameter vector $\mathbf{b}^k$ with $|E| = \frac{|E'|}{l}$ elements. Since the channel is output-symmetric, and the RNN's activations preserve symmetry conditions, the all-zero codeword is assumed to be transmitted while training. The syndrome $\boldsymbol{\sigma}$ corresponding to error pattern $\mathbf{x}$ is fed into the RNN as input. All the biases are initialized to 1.

C. Loss function

We use the loss function from [13]. Define

$$\hat{\boldsymbol{\sigma}}^{\mathbf{x}} = (\mathbf{x} + \hat{\mathbf{x}})\mathbf{H}_{\perp}^{\mathsf{T}},$$

where $\mathbf{H}_{\perp}$ is the orthogonal complement of $\mathbf{H}$, $\mathbf{x}$ is the actual error pattern, and $\hat{\mathbf{x}}$ is the estimated error pattern. With these notations, the loss function is given by

$$\mathcal{L}(\hat{\sigma}^x) = \sum_i \sin\left(0.5\pi\hat{\sigma}_i^x\right). \tag{5}$$

Observe that the loss function gives zero when the actual and estimated error pattern sums to a stabilizer. Hence, unlike other popular loss functions like binary-cross entropy, the above loss function accounts for degenerate errors. Also note that the reduction in Hamming weight of the estimated syndrome, an indicator of decoder convergence, leads to decrease in loss. The loss function is averaged over a minibatch, denoted by $\mathcal{M}$, as follows

$$\mathcal{L}_{\text{avg}} = \frac{1}{|\mathcal{M}|} \sum_{x \in \mathcal{M}} \mathcal{L}(\hat{\sigma}^x), \tag{6}$$

where $\mathcal{M}$ is obtained by uniformly sampling the training set.

Decoder	Number of uncorrectable errors	
	weight-5	weight-6
$\mathcal{D}_0$	136811	80354
$\mathcal{D}_1$	10300	14327
$\mathcal{D}_2$	1125	8561
$\mathcal{D}_3$	46	4856
$\mathcal{D}_4$	6	3904

TABLE I
THE NUMBER OF UNCORRECTABLE ERROR PATTERNS WHEN EACH DECODER IS RUN FOR MAXIMUM 20 ITERATIONS.

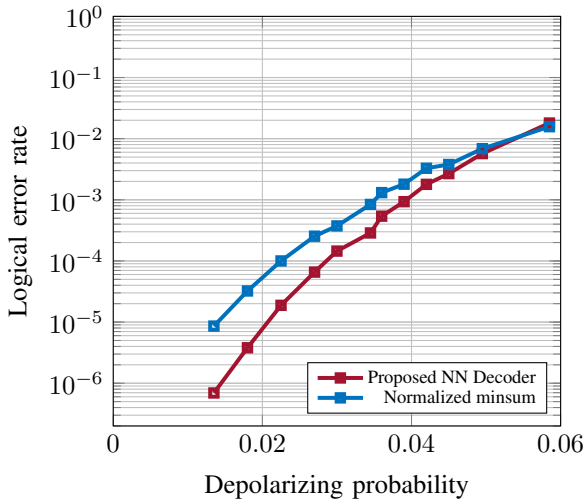

Fig. 3. This figure plots the performance of the proposed decoder and normalized min-sum decoder for 100 iterations. In the sequential approach, a decoder is used when the previous decoders fail.

V. NUMERICAL RESULTS

In this section, we consider the $[[1054, 124, 20]]$ lifted-product code constructed from the classical $(155, 62)$ Tanner code to demonstrate the superior performance of the proposed diversity-based decoder compared to the normalized min-sum decoder. We use the sub-graph expansion-contraction algorithm in [17] to find out 136811 weight five and 80321 weight six error patterns that the min-sum decoder fails to decode. We follow the sequential framework described in Section III-B to design four RNN decoders that can decode as many error patterns as possible from this list. Since the circulant size for the considered code is 31, each error pattern has 31 isomorphic copies. In the training process, we use only the non-isomorphic copies of the error patterns, which reduces the training set's size and the training duration. While training, the learning rate is set to 0.05, and the size of each minibatch is set to 200. The number of uncorrectable error patterns corresponding to the trained RNN-based decoders is given in Table I. Note that all the decoders can be run in parallel even though they are trained in a sequential fashion. When multiple decoders converge, we choose the error estimate with the lowest Hamming weight.

In 3, the logical error of the proposed decoder is compared to that of the normalized min-sum decoder when both decoders are run for 100 iterations. It can be observed that the RNN-based proposed decoder outperforms the normalized min-sum decoder and hence shows the efficacy of the proposed trapping set aware training procedure.

ACKNOWLEDGMENT

B. Vasić acknowledges the support of the NSF under grants CCF-1855879, CCF-2100013, CIF-2106189, CCSS-2027844, and CCSS-2052751. This work was also funded in part by Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and funded through JPL's Strategic University Research Partnerships (SURP) program. B. Vasić has disclosed an outside interest in Codelucida to the University of Arizona. Conflicts of interest resulting from this interest are being managed by The University of Arizona in accordance with its policies. The work of Xin Xiao was done while she was at the University of Arizona.

REFERENCES

[1] D. Gottesman, "Fault-tolerant quantum computation with constant overhead," *Quantum Information and Computation*, vol. 14, no. 15–16, pp. 1338–1372, Nov. 2014.
[2] A. A. Kovalev and L. P. Pryadko, "Fault tolerance of quantum low-density parity check codes with sublinear distance scaling," *Phys. Rev. A*, vol. 87, p. 020304, Feb. 2013.
[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, 2022.
[4] D. MacKay, G. Mitchison, and P. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2315–2330, Oct. 2004.
[5] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, 1997.
[6] N. Raveendran and B. Vasić, "Trapping sets of quantum LDPC codes," *Quantum*, vol. 5, p. 562, Oct. 2021.
[7] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *Quantum Information and Computation*, vol. 8, no. 10, pp. 987–1000, Nov. 2008.
[8] N. Raveendran, P. J. Nadkarni, S. S. Garani, and B. Vasić, "Stochastic resonance decoding for quantum LDPC codes," in *2017 IEEE Intl. Conf. on Commun. (ICC)*, May 2017, pp. 1–6.
[9] J. D. Crest, M. Mhalla, and V. Savin, "Stabilizer inactivation for message-passing decoding of quantum LDPC codes," in *2022 IEEE Information Theory Workshop (ITW)*, 2022, pp. 488–493.
[10] K.-Y. Kuo and C.-Y. Lai, "Refined belief propagation decoding of sparse-graph quantum codes," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 2, pp. 487–498, 2020.
[11] S. Miao, A. Schnerring, H. Li, and L. Schmalen, "Neural belief propagation decoding of quantum ldpc codes using overcomplete check matrices," 2023.
[12] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape," *Phys. Rev. Res.*, vol. 2, Dec 2020.
[13] Y.-H. Liu and D. Poulin, "Neural belief-propagation decoders for quantum error-correcting codes," *Phys. Rev. Lett.*, vol. 122, p. 200501, May 2019.
[14] D. Gottesman, "Class of quantum error-correcting codes saturating the quantum Hamming bound," *Phys. Rev. A*, vol. 54, no. 3, pp. 1862–1868, Sept. 1996.
[15] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. New York, NY, USA: Cambridge University Press, 2011.
[16] D. Declercq, B. Vasic, S. K. Planjery, and E. Li, "Finite alphabet iterative decoders—part ii: Towards guaranteed error correction of ldpc codes via iterative decoder diversity," *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4046–4057, 2013.
[17] N. Raveendran, D. Declercq, and B. Vasić, "A sub-graph expansion-contraction method for error floor computation," *IEEE Transactions on Communications*, vol. 68, no. 7, pp. 3984–3995, 2020.