Check-hybrid GLDPC Codes: Systematic Elimination of Trapping Sets by Super Checks

Vida Ravanmehr
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ, 85721
Email: vravanmehr@ece.arizona.edu

David Declercq
ETIS, ENSEA
University of Cergy-Pontoise
CNRS F-95000 Cergy-Pontoise, France
Email: declercq@ensea.fr

Bane Vasic
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ, 85721
Email: vasic@ece.arizona.edu

Abstract—In this paper, we propose a new approach to constructing a class of check-hybrid generalized low-density parity-check (GLDPC) codes which are free of small trapping sets. This approach is based on converting selected checks of an LDPC code involving a trapping set to super checks corresponding to a shorter error correcting component code. In particular, we follow two goals in constructing the check-hybrid GLDPC codes: First, the super checks are replaced based on the knowledge of trapping sets of the global LDPC code. We show that by converting only some single checks to super checks the decoder corrects the errors on a trapping set and hence eliminates the trapping set. Second, the number of super checks required for eliminating certain trapping sets is minimized to reduce the rate-loss. We first give an algorithm to find a set of critical checks in a trapping set of an LDPC code and then we provide some upper bounds on the minimum number of critical checks needed to eliminate certain trapping sets in the parity-check matrix of an LDPC code. A potential fixed set for a class of check-hybrid codes are also given.

I. INTRODUCTION

It has been shown that short low-rate codes with a good performance can be constructed from generalized low-density parity-check (GLDPC) codes with hybrid check nodes (e.g. [1], [2]). Liva and Ryan [1] were first who defined doping to refer to substituting some single parity checks by super checks corresponding to a stronger linear block code and constructed check-hybrid GLDPC codes using Hamming codes as component codes. In another work by Liva et al. [2], low-rate GLDPC codes are constructed by doping quasi-cyclic (QC)-LDPC codes with Hamming codes. It was shown that the constructed codes have a remarkable performance both in the waterfall and the error-floor regions on the additive white Gaussian noise (AWGN) channel. Paolini et al. [3], [4] studied the GLDPC and doubly-GLDPC codes with random component codes such as Hamming or BCH codes and proposed a method for the asymptotic analysis of the codes on the binary symmetric channel (BSC). They also considered check-hybrid GLDPC codes and showed that the asymptotic threshold of hybrid GLDPC codes outperforms that of the LDPC codes. In another work [5], Paolini et al. analyzed the asymptotic exponent of both the weight spectrum and the stopping set size spectrum for the check-hybrid GLDPC codes and provided a simple formula for the asymptotic exponent of the weight distribution of the check-hybrid GLDPC codes. Two common features of the methods given in the previous work are: (i) random choice of component codes, and (ii) significant reduction of the rate of the resulting check-hybrid GLDPC codes compared to the original LDPC code. Our approach is different in that the super checks that are replaced are chosen based on the knowledge of failures of the global LDPC code on the BSC under the Parallel Bit Flipping (PBF) algorithm. The PBF algorithm is a simple algorithm with low complexity and hence suitable for high-speed applications. This algorithm is also appropriate for analysis of failures of iterative decoding algorithms of LDPC codes. Richardson showed that the failure of iterative decoders is due to existence of the harmful structures in the Tanner graph of LDPC codes called “trapping sets” [6]. While trapping sets of the LDPC codes over the binary erasure channel (BEC) are well characterized as “stopping sets”, they are more complicated over the BSC and the AWGN channel. In [7], we identified the most harmful structures of column-weight three LDPC codes on the BSC using Gallager A/B and the PBF algorithms. We also showed that the trapping sets are short cycles or can be obtained as the union of short cycles in the Tanner graph.

To construct the parity check matrix of the check-hybrid GLDPC codes, we start from a collection of trapping sets and instead of randomly choosing super checks, we place the super checks at those check nodes so that the PBF decoder can correct the errors on a trapping set. It is also desirable to find the minimum number of super checks such that the rate loss of the constructed check-hybrid codes be reduced. The minimum number of such critical super checks is called the splitting number \(^1\). For some trapping sets, we provide upper bounds on the splitting number. Moreover, to achieve higher rate codes, we use low column-weight LDPC codes. The LDPC codes that are used in this paper are column-weight three permutation-based LDPC codes of girth 8 and the component codes are 2-error correcting codes. (The justification for this choice of global and component codes is explained in Section III). The decoder that is used for decoding the check-hybrid codes is

\(^1\)The more precise definitions (e.g. the definition of the splitting number) are given in Section III.
a modification of the PBF algorithm for GLDPC codes and differs only in the updating rule at check nodes.

The rest of the paper is organized as follows. In Section II, we provide the notations and definitions that are used throughout the paper. In Section III, we provide our main results on constructing check-hybrid GLDPC codes. Section IV concludes the paper and gives the possible directions for the future work.

II. PRELIMINARIES

In this section, we establish the notations and give a brief summary on the definitions and concepts of LDPC and GLDPC codes and trapping sets.

A. LDPC Codes, GLDPC and Check-hybrid GLDPC Codes

LDPC codes were first introduced by Gallager in his landmark work [8] and are usually defined by their Tanner graphs [9]. A Tanner graph is a bipartite graph with two disjoint sets of nodes, variable nodes \( V = \{1, 2, ..., n\} \) and check nodes \( C = \{1, 2, ..., m\} \). Each variable (check) node corresponds to a column (row) in the parity-check matrix of the code. The girth of the graph is the length of the smallest cycle in the graph and is denoted by \( g \). An LDPC code is called \((\gamma, \rho)\)-regular if the degree of each variable node is \( \gamma \) and the degree of each check node is \( \rho \). A \((\gamma, \rho, g)\) LDPC code is a \((\gamma, \rho)\)-regular code of girth \( g \).

Permutation-based LDPC codes are \((\gamma, \rho)\)-regular codes constructed from permutation matrices. We adopt the notations and definition from [10]. A permutation matrix is any square matrix which is cyclic, the permutation matrix is called a circulant permutation matrix. The parity check matrix of a QC-LDPC code can be represented by an array of circulant permutation matrices as follows:

\[
H = \begin{pmatrix}
I_0 & I_0 & \cdots & I_0 \\
I_0 & I_{p_1,1} & \cdots & I_{p_1,\rho-1} \\
\vdots & \vdots & \ddots & \vdots \\
I_0 & I_{p_{\gamma-1},1} & \cdots & I_{p_{\gamma-1},\rho-1}
\end{pmatrix}
\]

(1)

where for \(1 \leq j \leq \gamma-1\) and \(1 \leq l \leq \rho-1\), \(I_{p_{lj}}\) represents the circulant permutation matrix with a one at column \((r + p_{lj}) \mod p\) for the row \(r\) \((0 \leq r \leq p-1)\). If for \(1 \leq j \leq \gamma-1\) and \(1 \leq l \leq \rho-1\), \(I_{p_{lj}}\) is not circulant, then \(H\) is just a \((\gamma, \rho)\)-regular matrix based on permutation matrices.

GLDPC codes were introduced by Tanner in [9] where he proposed a method to construct longer error-correcting codes from shorter error-correcting codes. In GLDPC codes, each check node is satisfied if its neighboring variable nodes be a codeword of a linear code called component code. That is if \(c_i\) be a single parity check node in the Tanner graph of the global code and \(\{v_{i_1}, v_{i_2}, ..., v_{i_k}\}\) with values \(\{x_1, x_2, ..., x_n\}\) be the neighbors of \(c_i\), then in the GLDPC code, the super check corresponding to \(c_i\) is satisfied if \(\{x_1, x_2, ..., x_n\}\) be a codeword of the component code. A check-hybrid GLDPC code has two types of check nodes: single parity checks and super checks corresponding to a component code. A super check node is satisfied when its neighboring variable nodes be the codeword of the component code, while the single parity check is satisfied when the modulo-2 sum of its neighboring variable nodes is zero.

B. Decoding Algorithms and Trapping Sets

The decoding algorithms for decoding LDPC codes include a class of iterative algorithms such as bit flipping algorithms (parallel and serial) and messages passing algorithms like Gallager A/B and belief propagation decoding algorithms.

The notion of trapping sets was first introduced by Richardson [6] as the structures in the Tanner graph of LDPC codes responsible for failures of iterative decoders. Before we characterize trapping sets, we recall some definitions and assumptions. In this paper, we consider transmission over the binary symmetric channel (BSC). We also consider that the all-zero codeword is sent. Under this assumption, a variable node is said to be correct if its received value is 0; otherwise it is called corrupt. The support of a vector \(x = (x_1, x_2, ..., x_n)\), denoted by \(\text{supp}(x)\), is the set \(\{x_i \mid x_i \neq 0\}\). The decoder runs until maximum number of iterations \(M\) is reached or a codeword is found. Let \(y = (y_1, y_2, ..., y_n)\) be a received vector after transmitting the all-zero codeword and let \(y^{(l)} = (y_1^{(l)}, y_2^{(l)}, ..., y_n^{(l)})\) be the output of the decoder after the \(l\)-th iteration. A variable node \(v\) is said to be eventually correct if there exists an integer \(L > 0\) such that for all \(l \geq L\), \(v \notin \text{supp}(x^l)\). The decoder fails on decoding \(y\) if there does not exist \(l \leq M\) such that \(\text{supp}(x^l) = 0\). For the received word \(y\), the set of variable nodes which are not eventually correct is called a trapping set and is denoted by \(T(y)\). If \(T(y) \neq \emptyset\), then \(T(y)\) is called an \((a, b)\) trapping set and is denoted by \(T(a, b)\) if the number of variable nodes in \(T(y)\) equals \(a\) and the number of odd degree check nodes in the subgraph induced by \(T(y)\) is \(b\). For the trapping set \(T(y)\), \(\text{supp}(y)\) is an induced set. \(T(a, b)\) is called an elementary trapping set if the degree of each check node in the subgraph induced by the set of variable nodes is one or two and there are \(b\) check nodes of degree one. In this paper, we say that a trapping set is harmless if the decoder fails to decode at least one error pattern on the trapping set; Otherwise, it is called harmful. While trapping sets can have different induced sets, a class of trapping sets called fixed sets have the fixed induced set. A fixed set \(F\) is the set of variable nodes that are corrupt at the beginning and at the end of iterations of decoding, while other variable nodes remain correct after decoding.

III. MAIN RESULTS

In this section, we provide our main results on constructing check-hybrid GLDPC codes in which the trapping sets responsible for the failure of the PBF algorithm are not harmful anymore. We first describe the PBF algorithm for the check-hybrid GLDPC codes and use it throughout the paper for our analysis. We mention that the decoding algorithm at each super check is the bounded distance decoding (BDD) which can be simply explained as follows:
Let $C$ be a linear block code with the minimum distance $d$ and let $y$ be a received word. The BDD detects all error patterns of weight $t \leq \frac{d-1}{2}$ and decodes $y$ to a correct codeword. While if there is more than $t > \frac{d-1}{2}$ errors in $y$, the BDD keeps $y$ as the decoded word.

**Algorithm 1** The PBD algorithm for check-hybrid GLDPC codes.

*In each iteration:*

- Variable nodes send their current estimates to the neighboring single parity check and super check nodes.

**Updating rule at check nodes:**

- Each super check node performs the BDD on the incoming messages. If a codeword is found, then the check node sends flip messages to all variable nodes which differ from the codeword. If not, then the check node does not send any flip messages.
- At each single parity check, the modulo-2 sum of the incoming messages is calculated. If the sum is not zero, then the check node sends flip messages to the neighboring variable nodes. If the sum is zero, then the check node does not send any flip messages.

**Updating rule at variable nodes:**

- A variable node flips if it receives more than $\gamma/2$ flip messages.

**A. Intuition and Illustrative Examples**

Let us start by some observations on the effect of replacing single parity checks by super checks. Let $C$ be a $(3, \rho, 8)$ LDPC code. A list of small trapping sets of $C$ under PBF algorithm is given in [11]. It can be easily seen that if all single parity checks in the Tanner graph corresponding to the parity check matrix of $C$ are replaced by super checks of a 2-error correcting component code, then the PBF algorithm for GLDPC codes can correct all errors on the trapping sets. This result can be explained by the fact that in all elementary trapping sets, the degree of each check node is at most two and since they are replaced by a 2-error correcting component code, the BDD at each super check can correct all errors.

However, as we show in the following, it is not necessary to replace all super checks in a trapping set for the decoder to correct the errors. We show that a trapping set is not harmful if only two single parity checks are replaced by super checks. We say a trapping set is eliminated if by replacing super checks, the trapping set is not harmful anymore. In this paper, we only consider $(3, \rho, 8)$-LDPC codes as the global codes and 2-error correcting codes as the component codes. However, the approach can be generalized to other $t$-error correcting component codes and other LDPC-code ensembles for which their trapping sets under a certain decoding algorithm are known.

Let us now focus on the two smallest trapping sets for $(3, \rho, 8)$-LDPC codes, i.e. the $(4, 4)$ and the $(5, 3)$ trapping sets. Suppose only one check node of degree 2 in the $(4, 4)$ trapping set is replaced by a super check. In this case, the PBF algorithm can correct all errors located on the $(4, 4)$ trapping set. Fig. 1 shows how the PBF algorithm corrects the errors in 2 iterations. In this figure, denotes a variable node, □ shows a single parity check node and ■ shows a super check.

Now, consider a $(5, 3)$ trapping set. Fig. 2 shows how the PBF algorithm can correct all errors located in the trapping set in which only two single parity checks of degree 2 are replaced by super checks.

**B. Critical Sets and Decoding Analysis**

As shown in Section III-A, a trapping set can be eliminated by judiciously replacing check nodes in the original global code. A set of such checks is called a critical set and defined as follows.

**Definition 1.** Let $T(a, b)$ be an elementary trapping set. Let $C_K = \{c_1, c_2, \ldots, c_k\}$ be a subset of check nodes of degree 2 in $T$. A set $S \subseteq C_K$ is called critical if by converting the
single parity checks to super checks, the trapping set is not harmful anymore.

We note that a critical set is not unique and there are many possible critical sets with different sizes in a trapping set.

**Definition 2.** Let $T(a, b)$ be an elementary trapping set. The minimum size of a critical set in $T$ is denoted by $s(a, b)(T)$.

As an example, $s(4, 4)(T) = 1$ and $s(5, 3)(T) = 2$.

In Algorithm 2, we provide a method to find a possible critical set in a trapping set. The motivation behind finding the critical set using Algorithm 2 is based on the role of super checks in elementary trapping sets. When a single parity check of degree-2 is replaced by a super check, then the super check plays a role of a single parity check for each of connected variable nodes. Thus, each super check plays a role of a single parity check for each of connected variable nodes. Fig. 3 shows an alternative view of the effect of a super check to eliminating a trapping set.

**Algorithm 2** Finding a critical set in a trapping set $T(a, b)$.

**Initialization:** Let $T' = T$ be the $(a, b)$ trapping set.

while Number of variable nodes in $T'$ is greater than 0 do

if there exists a variable node $v$ in $T'$ which is connected to exactly one degree-1 check node and two degree-2 checks then

Replace one of the check nodes of degree-2 connected to $v$ by a super check corresponding to a 2-error correcting code. Remove the variable node $v$ and all edges connected to it.

else

Choose a variable node $v$ in $T'$. Replace one check node of degree-2 connected to $v$ by a super check and split the super check node to 2 single parity checks.

end if

while Number of variable nodes connected to at least two single parity checks of degree-1 is greater than 0 do

Remove variable nodes connected to at least two single parity checks of degree-1 and all edges connected to them.

end while

end while

As we want to reduce the rate-loss caused by converting single checks to super checks, we now study the minimum number of super checks required to replace in a Tanner graph of an LDPC code such that the decoder can correct all error patterns on all $(a, b)$ trapping sets.

**Definition 3.** Let $C$ be a $(3, \rho, 8)$-LDPC code with the parity check matrix $H$ and let $T(a, b)$ be an elementary trapping set in $H$. The minimum number of super checks corresponding to a 2-error correcting component code that are required for eliminating all $(a, b)$ trapping sets in $H$ is called the splitting number of the $(a, b)$ trapping set in $H$ and is denoted by $s(a, b)(H)$.

**Lemma 1.** Let $C$ be a $(3, \rho, 8)$ LDPC code with the parity-check matrix $H$ based on permutation matrices of size $p$. Then, $s(a, b)(H) \leq 2p$, for all $a$ and $b$.

Proof: Suppose the first $2p$ rows of $H$ are replaced by super checks. The first $2p$ rows of $H$ correspond to the first two rows of blocks in equation 1. Thus, each variable node is connected to 2 super checks and 1 single parity check. This results that each variable node receives at least 2 correct messages from its neighbors. In fact, by converting two single parity checks to super checks and then splitting each super check into two single parity check nodes, all cycles in all elementary trapping sets are eliminated. **Q.E.D.**

According to Lemma 1, all elementary trapping sets are eliminated when each variable node is connected to exactly two super checks. Thus, the trapping sets for this class of check-hybrid GLDPC codes are non-elementary trapping sets.

We now exhibit a potential fixed set for the check-hybrid GLDPC code in the case that each variable node is connected to exactly two super checks. We consider $C$ to be a $(3, \rho, 8)$-LDPC code as the global code and super checks correspond to a 2-error correcting component code.

**Theorem 1.** Let $T$ be a subset of variable nodes with the induced subgraph $I$. Then, $T$ is a fixed set if (a) The degree of each check node in $I$ is either 1 or 3 and; (b) Each variable node in $I$ is connected to 2 check nodes of degree 3 and 1 check node of degree 1 where the check nodes of degree 3 have been replaced by super checks of the 2-error correcting component code and; (c) No 2 check nodes of degree 3 share a variable node outside $I$.

Proof: Since the check nodes of degree 3 have been replaced by super checks of a 2-error correcting component code and since the decoding in the component codes is the BDD, the super checks of degree 3 do not send any flip messages to the variable nodes in $I$. Also, since any variable node in $I$ is connected to 2 super checks, it remains corrupt. Furthermore, no variable node outside $I$ can receive more than 1 flip message because no 2 check nodes of degree 3 share a variable node outside $I$. Thus, the variable nodes outside $I$ that are originally correct will remain correct. Consequently, $I$ is a fixed set. **Q.E.D.**

Fig. 4 shows a potential fixed set in a $(3, \rho, 8)$-LDPC code in which each variable node is connected to exactly 2 super checks.
is connected to exactly one super check. In fact, the trapping set, may not be eliminated if each variable node degree-1 are replaced by super checks. The following Theorem trapping set will remain harmful if the single parity checks of node is connected to exactly two super checks. Theorem 2. Suppose the first $p$ rows of $H$ are replaced by super checks. Then, $s(4,4)(H) \leq p$ if the girth of the Tanner graph corresponding to the last $2p$ rows of $H$ is 12.

Proof: As explained before, if in $T(4,4)$ the single parity checks of degree-1 are replaced by super checks, then due to the existence of a cycle of length 8, the PBF cannot correct the errors. However, if the girth of the subgraph induced by single parity checks is greater than 8, then there will not be any 8-cycle and consequently all $(4,4)$ trapping sets will be eliminated. According to Corollary 2.1 in [10], the girth of a $(2,p)$-regular QC-LDPC code is $4i$ for some integer $i > 0$. Moreover, the girth of $H$ cannot be more than 12 as shown in [10]. Thus, if the girth of the subgraph induced by the last $2p$ rows of $H$ is 12, it results that all 8-cycles in $H$ will contain at least one super check of degree 2, and henceforth the 8-cycles are not the harmful $(4,4)$ trapping sets.

IV. Discussion and Future Work

In this paper, we introduced a method for constructing check-hybrid GLDPC codes in which the super checks corresponding to a 2-error correcting component code that have been replaced are chosen based on the knowledge of trapping sets of the global code. By carefully replacing the super checks, we eliminated harmful trapping sets of the PBF algorithm while reducing the rate loss caused by adding more constraints on check nodes of the component code.

Future works include improving the upper bounds on the splitting number of trapping sets and extending the results to other classes of LDPC codes as the global codes and t-error correcting component codes.

V. Acknowledgments

This work is funded by the Seagate Technology and in part by the NSF under grants CCF-0963726 and CCF-1314147.

REFERENCES


