Check-hybrid GLDPC Codes Without Small Trapping Sets

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 construct a class of check-hybrid generalized low-density parity-check (GLDPC) codes which are free of small trapping sets. The approach is based on converting some selected check nodes involving a trapping set to super checks corresponding to a shorter error-correcting component code. Specifically, we follow two main purposes to construct the check-hybrid codes: First, replacing single parity checks by super checks is done based on the knowledge of the trapping sets of the global LDPC code. We show that by converting some single checks to super checks in a trapping set, the decoder corrects the errors on a trapping set and hence eliminates the trapping set. Second, the rate-loss caused by replacing the super checks is reduced through finding the minimum number of such critical checks. We first present an algorithm to find possible critical checks in a trapping set. We then provide some upper bounds on the minimum number of such critical checks such that the decoder corrects all error patterns on certain trapping sets in the Tanner graph of the global LDPC code. We also provide a potential fixed set for a class of constructed check-hybrid codes.**

***Index Terms*—Check-hybrid GLDPC, Critical set, Low-density parity-check (LDPC) codes, Parallel Bit Flipping (PBF) algorithm, Splitting number, Trapping set.**

I. Introduction

In [1], Tanner proposed a method to construct long error correcting codes from shorter error correcting codes. His method is based on replacing super checks corresponding to a shorter error correcting component code at single parity checks of an LDPC code as the global code. The codes are assumed as the generalization of LDPC codes and thus are called generalized low-density parity-check (GLDPC) codes. Tanner also provided lower bounds on the minimum distance and rate of GLDPC codes and proposed different hard-decision algorithms for the decoding of GLDPC codes. Since then several methods and approaches have been introduced to construct various classes of these codes; all focus on strengthening the parity check and/or variable nodes of the Tanner graph of an LDPC code using shorter and stronger error correcting codes. Lentmaier *et al.* [2] studied the GLDPC codes based on the Hamming codes as the component codes and derived a lower bound on the minimum distance of the GLDPC codes. Miladinovic and Fossorier derived bounds on the error correction capability of GLDPC codes based on Reed-Solomon and BCH codes as component codes and defined the "generalized stopping set" as the configurations responsible for decoding failure of GLDPC codes over the BEC [3]. Chillapagari *et al.* studied the failures of the Parallel Bit Flipping (PBF) algorithm on GLDPC codes using the expansion property of codes [4] and provided a lower bound on the error correction capability of GLDPC codes.

Some hybrid constructions of GLDPC codes were introduced to strengthen the codes by converting only some single checks to super checks. Chen and Tanner [5] studied the check-hybrid codes with Hamming codes as component codes on the Gilbert-Elliott channel. Liva and Ryan [6] 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.* [7], 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. In another work [8], 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.

There are two common features in the methods given in the previous work: (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 binary symmetric channel (BSC) and under the 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 the analysis of failures of iterative decoding algorithms of LDPC

Part of this work has been submitted to the International Symposium on Information Theory (ISIT) 2014.

codes. Richardson showed that the failure of iterative decoders is due to existence of harmful structures in the Tanner graph of LDPC codes called "trapping sets" [9]. 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 [10], 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 single parity 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 can be reduced. The minimum number of such critical super checks is called the *splitting number* and will be defined precisely in Section IV. For some trapping sets, we provide upper bounds on the splitting number. Moreover, 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 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. We also give a summary on LDPC, GLDPC codes and the trapping sets. In Section III, we show the effect of super checks to correcting errors on a trapping set. In Section IV, we provide our main results on constructing check-hybrid GLDPC codes without small trapping sets. We will give upper bounds on the minimum number of super checks needed to be replaced such that the Tanner graph will be free of some certain trapping sets. In Section V, we show the performance of the constructed codes on the BSC and under the Gallager B decoding algorithm. Section VI concludes the paper and gives possible directions for future work.

II. Preliminaries

In this section, we first establish the notations and then give a brief summary on the definitions and concepts of LDPC and GLDPC codes. We also define trapping sets and fixed sets for the iterative decoding algorithms.

A. Graph Theory Notations

Let $G(U, E)$ be an undirected simple graph with the set of vertices U and the set of edges E. An edge e is an unordered pair (u_1, u_2). The edge $e = (u_1, u_2)$ is said to be incident on u_1 and u_2 and the two vertices u_1 and u_2 are said to be adjacent (neighbors). The set of neighbors of the vertex u is shown by $\mathcal{N}(u)$. The degree of each vertex $d(u)$ is defined as the number of vertices in its neighborhood. The length of the shortest cycle is called the girth of the graph and is denoted by g. A bipartite graph $G(V \cup C, E)$ is graph with two disjoint sets of vertices; variable nodes V and check nodes C. An edge e is incident on a variable node $v \in V$ and a check node $c \in C$. A bipartite graph is called (γ, ρ)-regular if the degree of each variable node is γ and the degree of each check node is ρ. The girth of a bipartite graph is even. The parity check matrix H of a linear code C can be represented with a bipartite graph called the Tanner graph. Each column in the parity check matrix is shown by a variable node and each row is denoted by a check node in the Tanner graph. A variable node v_j and a check node c_i are adjacent if and only if $H_{i,j} = 1$. A vector $\mathbf{v} = (v_1, v_2, ..., v_n)$ is a codeword if and only if $H\mathbf{v}^T = \mathbf{0} \pmod 2$. A linear code is called (γ, ρ)-regular if its parity check matrix is (γ, ρ)-regular. This code has rate $r \geq 1 - \frac{\gamma}{\rho}$ [11].

B. LDPC codes, GLDPC and check-hybrid GLDPC codes

LDPC codes were first introduced by Gallager in his landmark work [11] where he proposed different methods for constructing parity check matrices of LDPC codes and provided different hard decision algorithms for decoding of LDPC codes. He also showed that the performance of these codes under iterative decoding algorithms is very close to Shannon's limit. LDPC codes are usually defined by their Tanner graphs. A (γ, ρ, g) LDPC code is a (γ, ρ)-regular code of girth g.

GLDPC codes were introduced by Tanner in [1] 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 $\{v_{i_1}, v_{i_2}, ..., v_{i_n}\}$ 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. The parity check matrix of GLDPC codes are constructed using the parity check matrix of the longer code also known as the global code and the parity check matrix of the component code. To construct the parity check matrix of the GLDPC code, it is enough to replace each one in each row of the parity check matrix of the global code by one column of the parity check matrix of the component code. Each zero in each row will be replaced by a zero-column in the global parity check matrix. 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 codeword of the component code, while the single parity check is satisfied when the modulo-2 sum of its neighboring variable nodes is zero. The component codes in GLDPC and check-hybrid GLDPC codes can be chosen from different codes. However, in this paper, GLDPC and check-hybrid GLDPC codes are constructed from the same component code and

the global codes are chosen from the family of (γ, ρ)-regular codes.

C. 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 notation of "trapping sets" was first introduced by Richardson [9] as the structures in the Tanner graph of LDPC codes responsible for failures of decoders. Before we characterize the trapping sets of bit flipping decoding algorithm, we provide definitions and assumptions. In this paper, we consider transmission over the 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 $\mathbf{x} = (x_1, x_2, ..., x_n)$ denoted by $\mathrm{supp}(\mathbf{x})$ 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 $\mathbf{y} = (y_1, y_2, ..., y_n)$ be a received vector after transmitting the all-zero codeword and let $\mathbf{y^{(l)}} = (y_1^{(l)}, y_2^{(l)}, ..., y_n^{(l)})$ 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 $l \geq L$, $v \notin \mathrm{supp}(\mathbf{x}^l)$. The decoder fails on decoding $\mathbf{y}$ if there does not exist $l \leq M$ such that $\mathrm{supp}(\mathbf{x}) = 0$. For the received word $\mathbf{y}$, the set of variable nodes which are not eventually correct is called a trapping set and is denoted by $T(\mathbf{y})$. If $T(\mathbf{y}) \neq \emptyset$, then then $T(\mathbf{y})$ is called an (a, b) trapping set and is denoted by $\mathcal{T}(a, b)$ if the number of variable nodes in $T(\mathbf{y})$ equals a and the number of odd degree check nodes in the subgraph induced by $T(\mathbf{y})$ is b. For the trapping set $T(\mathbf{y})$, $\mathrm{supp}(\mathbf{y})$ is an induced set. $\mathcal{T}(a, b)$ 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 b check nodes of degree one.

Chilappagri *et al.* [12] introduced the notion of "critical number" as the minimum number of variable nodes on a trapping set that need to be initially in error such that the decoder fails. They showed that the harmfulness of a trapping set depends on its critical number; the smaller the critical number, the more harmful a trapping set. In this paper, we say that a trapping set is *harmful* if the decoder fails to decode at least one error pattern on the trapping set; Otherwise, it is called *harmless*. 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. A vector $\mathbf{y}$ is called a fixed point if $\mathrm{supp(y)} = F$. From definition of the fixed set and trapping set, it is clear that a fixed is always a trapping set while a trapping set is not necessarily a fixed set. Fixed sets of an LDPC code with the column-weight γ are the set of variable nodes $\mathcal{I}$ such that every variable node in $\mathcal{I}$ is connected to at least $\lceil \gamma/2 \rceil$ of check nodes of even-degree and no $\lfloor \gamma/2 \rfloor$ check nodes of odd-degree share a variable node outside $\mathcal{I}$ [10]. Chilappagari *et al.* defined a potential fixed set for the PBF algorithm of GLDPC codes as follows:

Fact 1:([4] Theorem 6) Let $\mathcal{C}$ be a GLDPC code with (γ, ρ)-regular global code and a t-error correcting component code. Let $\mathcal{I}$ be a subset of variable nodes with the following properties: (a) The degree of each check node in $\mathcal{I}$ is either 1 or $t+1$; (b) Each variable node in $\mathcal{I}$ is connected to $\lceil \gamma/2 \rceil$ checks of degree $t+1$ and $\lfloor \gamma/2 \rfloor$ check nodes of degree 1; and (c) No $\lfloor \gamma/2 \rfloor + 1$ checks of degree $t+1$ share a variable node outside $\mathcal{I}$. Then, $\mathcal{I}$ is a fixed set.

III. Super checks and trapping sets

Let us start by some observations on the effect of replacing single parity checks by super checks. In fact, we show how trapping sets responsible for the failure of the PBF are not harmful anymore when only some selected single checks are replaced by super checks. 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 $\mathcal{C}$ be a linear block code with the minimum distance d and let $\mathbf{y}$ be a received word. The BDD detects all error patterns of weight $w \leq \frac{d-1}{2}$ and decodes $\mathbf{y}$ to a codeword. While if there is more than $w > \frac{d-1}{2}$ errors in $\mathbf{y}$, the BDD keeps $\mathbf{y}$ as the decoded word.

Algorithm 1 The PBF 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.

Let $\mathcal{C}$ be a $(3, \rho, 8)$ LDPC code. Fig. 1 shows some small trapping sets of a column-weight three LDPC codes of girth $g = 8$ namely the $(4, 4)$ trapping set, the $(5, 3)$ trapping set and a $(6, 4)$ trapping set. In this paper, $\circ$ denotes a variable node and $\square$ denotes a check node. A list of small trapping sets of $\mathcal{C}$ under the PBF and the Gallager A/B algorithm is given in [13]. It can be easily seen that if all single parity checks in the Tanner graph corresponding to the parity check

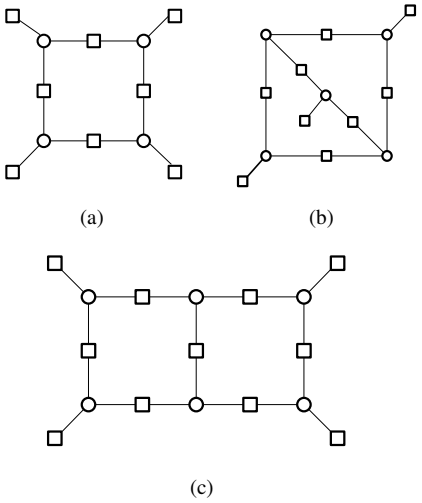

Fig. 1. Tanner graph representation of trapping sets for column-weight three and girth $g = 8$ LDPC codes; (a) the (4,4) trapping set, (b) the (5,3) trapping set, (c) a (6,4) trapping set.

matrix of $\mathcal{C}$ are replaced by super checks of a 2-error correcting component code, then the PBF decoding 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. Fig. 2 shows how the PBF corrects all errors located on the (5,3) trapping set when all single checks are replaced by super checks. In this paper, a ■ denotes a super check. 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 some selected 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 consider the (5,3) trapping set. Fig. 3 shows how the PBF algorithm can correct all errors located on the trapping set in which only two single parity checks of degree 2 are replaced by super checks.

It should be noted that not all pairs of super checks in the (5,3) trapping set can be helpful for the decoder to correct the errors on the (5,3) trapping set. Fig. 4 shows three possible cases that by replacing the super checks the trapping sets remain harmful. In Fig. 4(a) and 4(b) only the variable node v_5 will be corrected, while in Fig. 4(c) all variable nodes will remain incorrect. In fact, it can be seen that all pairs of super checks can be helpful for the decoder except when at least

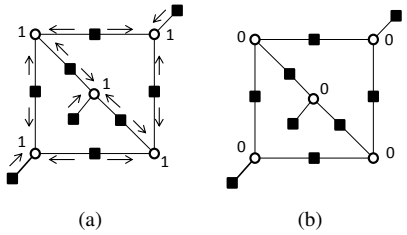

Fig. 2. The (5,3) trapping set is not harmful anymore when all single parity checks are replaced by super checks corresponding to a 2-error correcting component code. (a) flip messages from super checks to corrupt variable nodes in the first iteration of the PBF algorithm, (b) all variable nodes are corrected after the first iteration.

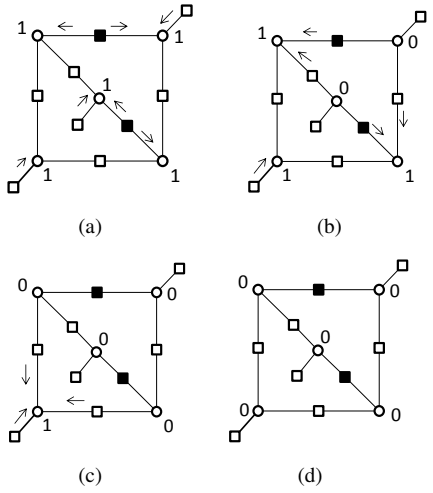

Fig. 3. The (5,3) trapping set in a column-weight three code that becomes harmless if two super checks corresponding to a 2-error correcting component code are replaced. Arrows show flip messages from check nodes to corrupt variable nodes in each iteration of the PBF algorithm: (a) flip messages from checks in the first iteration, (b) flip messages from checks to 3 variable nodes that are still in error, (c) flip messages from checks to the only one corrupt variable, (d) all variable nodes are corrected after the third iteration.

one of the single parity checks of degree one is replaced by a super check or both two super checks are replaced in the place of single parity checks of degree-2 connected to v_5 in Fig. 4.

The above examples show that not only the number of super checks, but also the position of super checks in a trapping set is important for the decoder to successfully correct the errors or fail on decoding. Since the rate of the GLDPC codes decreases by replacing single parity checks by super checks, we are interested in replacing the minimum number of super checks such that the resulting Tanner graph will be free of small trapping sets. In the next section, we first provide an algorithm to find a set of such critical checks in a trapping set and then we present upper bounds on the minimum number of super checks that need to be replaced in the parity check matrix such that the resulting Tanner graph will be free of small trapping sets.

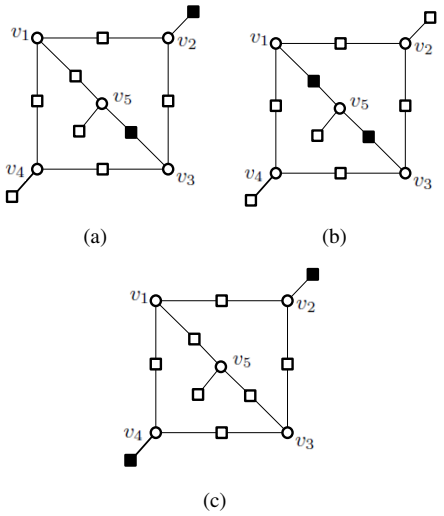

Fig. 4. Possible super-check replacements which are not helpful for the decoder to correct all errors on the (5,3) trapping set.

IV. Critical sets and the splitting number

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 will be eliminated. As shown in Section III, 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 $\mathcal{T}(a,b)$ be an elementary trapping set. Let $C_K = \{c_1, c_2, ..., c_k\}$ be a set of check nodes of degree 2 in $\mathcal{T}$. A set $S \subseteq C_K$ is called critical if by converting the single parity checks in S to the 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 $\mathcal{T}(a,b)$ be an elementary trapping set. The minimum size of a critical set in $\mathcal{T}$ is denoted by $s_{(a,b)}(\mathcal{T})$.*

As an example, $s_{(4,4)}(\mathcal{T}) = 1$ and $s_{(5,3)}(\mathcal{T}) = 2$. 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 sends a flip message to a neighboring variable node if and only if the variable node is corrupt. Thus, each super check plays the role of a single parity check for each of connected variable nodes. Breaking the cycles in a trapping set by splitting the super check into two single parity checks is the base of finding a critical set in Algorithm 2.

Fig. 5 shows an alternative view of the effect of a super check to eliminating a trapping set.

As we explained, the number of cycles in a trapping set plays a key role to find the number of critical checks of a

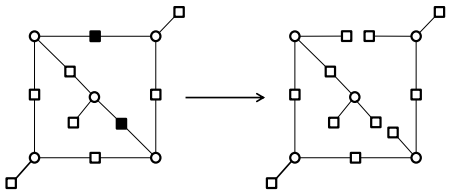

Fig. 5. Super checks corresponding to a 2-error correcting component code can be considered as two single parity checks of degree-1. These replacements break the cycles responsible for the failure of decoding.

Algorithm 2 Finding a critical set in a trapping set $\mathcal{T}(a,b)$.

initialization: Let $\mathcal{T}' = \mathcal{T}$ be the (a,b) trapping set.
while Number of variable nodes in $\mathcal{T}'$ is greater than 0 **do**
  if there exists a variable node v in $\mathcal{T}'$ 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. Split the super check into two single checks. Remove the variable node v and all edges connected to it.
  else
    Choose a variable node v in $\mathcal{T}'$. 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

trapping set. This fact helps us to find the number of critical checks in some trapping sets without using Algorithm 2. If a trapping set $\mathcal{T}'(a',b')$ has been obtained by adding some variable and check nodes to another trapping set $\mathcal{T}(a,b)$ such that the new variable and check nodes do not create a new cycle, then $s_{(a',b')}(\mathcal{T}')$ and $s_{(a,b)}(\mathcal{T})$ are equal. To be more precise, we first provide the following definitions.

Definition 3. *A subdivision of a simple graph G is a graph resulting from the subdivision of edges in G. In other words, a subdivision of a graph is a graph obtained by adding at least one vertex on an edge of the graph.*

Fig. 6 shows a simple graph (Fig. 6(a)) and its subdivision in Fig. 6(b). We define a graph induced by the set of the variable nodes of a bipartite graph and then we generalize the definition of subdivision of a graph for bipartite graphs.

Definition 4. *Let $G(V \cup C, E)$ be a bipartite graph. The simple graph $G'(V, E')$ induced by the set of variable nodes V is a graph with $|V|$ vertices in which two vertices v_1 and v_2 are connected to each other if and only if there exists a check node*

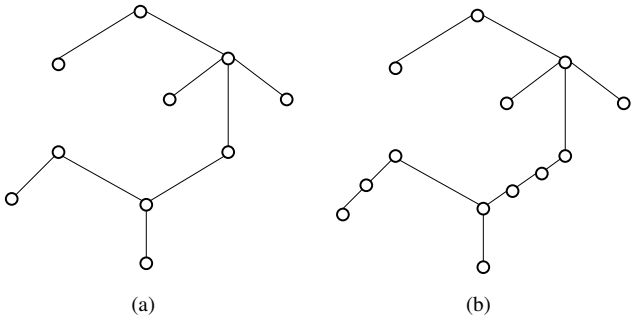

Fig. 6. (a) A simple graph, (b) a subdivision of the graph given in (a).

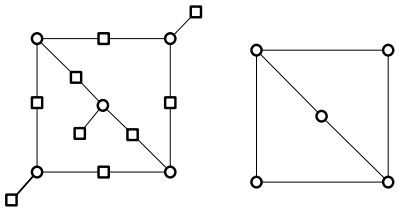

Fig. 7. (a) The (5,3) trapping set as a bipartite graph, (b) the simple graph induced by the 5 variable nodes of the (5,3) trapping set.

c in C such that v_1 and v_2 are neighbors of c.

As an example, consider the (5,3) trapping set as a bipartite graph. The simple graph induced by the set of variable nodes of the (5,3) trapping set is shown in Fig. 7.

Definition 5. *Let $\mathcal{T}(a,b)$ be a trapping set. The trapping set $\mathcal{T}'(a+1,b+1)$ is called a subdivision of $\mathcal{T}$ if the simple subgraph induced by the set of variable nodes of $\mathcal{T}'$ is a subdivision of the simple graph induced by the set of variable nodes of $\mathcal{T}$.*

Fig. 8 shows two trapping sets, a (6,4) trapping set and a (7,5) trapping set, in which the (7,5) trapping set is a subdivision of the (6,4) trapping set.

Corollary 1. *Let $\mathcal{T}'(a+1,b+1)$ be a trapping set which is a subdivision of the trapping set $\mathcal{T}(a,b)$. Then $s_{(a+1,b+1)}(\mathcal{T}') = s_{(a,b)}(\mathcal{T})$.*

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 that are required to be replaced in a Tanner graph of an LDPC code such that the decoder can correct all error patterns on all (a,b) trapping sets.

Definition 6. *Let $\mathcal{C}$ be a $(3,\rho,8)$-LDPC code with the parity check matrix H and let $\mathcal{T}(a,b)$ 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 $s_{(a,b)}(H)$.*

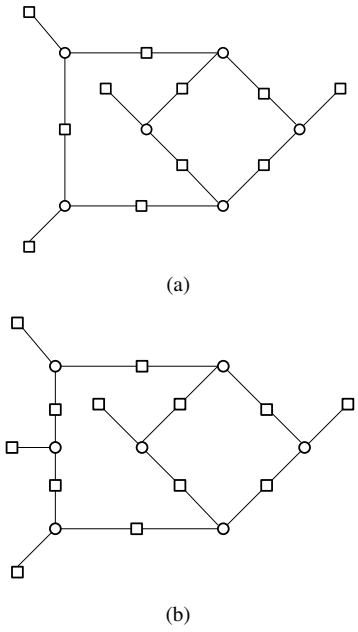

Fig. 8. (a) A (6,4) trapping set, (b) a (7,5) trapping set which is a subdivision of the (6,4) trapping set given in (a).

Now, we provide some upper bounds on the splitting number of trapping sets in the parity check-matrices based on permutation matrices. Permutation-based LDPC codes are (γ,ρ)- regular codes constructed from permutation matrices. A permutation matrix is any square matrix in which the weight of each row and each column is one. If the permutation matrix is cyclic, the permutation matrix is called a circulant permutation matrix. The parity check matrix of a quasi-cyclic LDPC code can be represented by an array of circulant permutation matrices as follows [14]:

$$H = \begin{bmatrix} I_0 & I_0 & \cdots & I_0 \\ I_0 & I_{p_{1,1}} & \cdots & I_{p_{1,\rho-1}} \\ \vdots & & \ddots & \vdots \\ I_0 & I_{p_{\gamma-1,1}} & \cdots & I_{p_{\gamma-1,\rho-1}} \end{bmatrix}$$

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

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

Proof: The proof is given in [15].

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 fixed set for the PBF algorithm for the check-hybrid GLDPC code in the case that the super checks

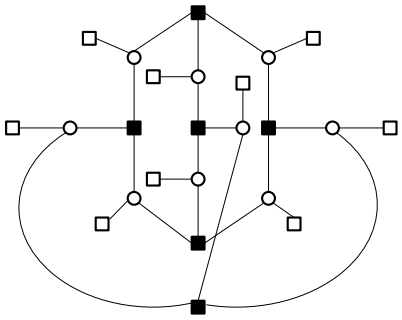

Fig. 9. A potential fixed set for a $(3, \rho, 8)$-LDPC code in which each variable node is connected to exactly two super checks.

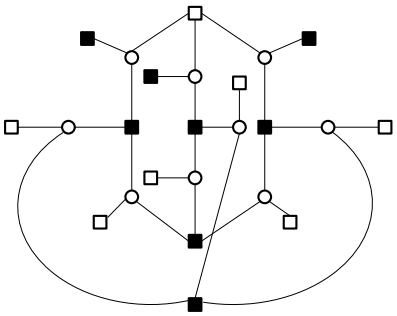

Fig. 10. An example of a subgraph in a $(3, \rho, 8)$-LDPC code which satisfies all conditions of Theorem 1 except the condition (b). This structure is not harmful for the PBF algorithm.

have been replaced such that each variable node is connected to exactly two super checks.

Theorem 1. *Let $\mathcal{T}$ be a subset of variable nodes with the induced subgraph $\mathcal{I}$. Then, $\mathcal{T}$ is a fixed set if (a) The degree of each check node in $\mathcal{I}$ is either 1 or 3 and; (b) Each variable node in $\mathcal{I}$ 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 share a variable node outside $\mathcal{I}$.*

Proof: The proof is given in [15].

Fig. 9 shows a potential fixed set in a $(3, \rho, 8)$-LDPC code in which each variable node is connected to exactly 2 super checks. We note that conditions (a) and (c) are similar to the corresponding conditions in Fact 1. The main difference is in condition (b) where in Theorem 1, the constraint on the position of super checks is a stronger condition on $\mathcal{I}$ to be a fixed set. We also note that if this condition is not satisfied, $\mathcal{I}$ may not be either a trapping set or a fixed set. Fig. 10 shows a subgraph satisfying all conditions of Theorem 1 except the condition (b) which is not a trapping set nor a fixed set.

Although all elementary trapping sets are eliminated when each variable node is connected to two super checks, there are trapping sets that are eliminated if each variable node is connected to exactly one super check. Fig. 11 depicts a possible way for replacing super checks in $\mathcal{T}(5, 3)$, such that

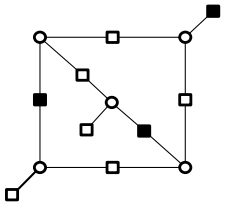

Fig. 11. The (5,3) trapping set can be eliminated if each variable node is connected to exactly one super check.

each variable node is connected to exactly one super check and the trapping sets are not harmful anymore.

Thus, for a permutation-based LDPC code $\mathcal{C}(3, \rho, 8)$ with the parity-check matrix H, if the parity checks corresponding to the first p rows of H are replaced by super checks, then all $(5, 3)$ trapping sets are eliminated and hence $s_{(5,3)}(H) \leq p$.

It is easy to see that the smallest trapping set, the (4,4) trapping set, may not be eliminated if each variable node is connected to exactly one super check. In fact, the $(4, 4)$ trapping set will remain harmful if the single parity checks of degree-1 are replaced by super checks (Fig. 12). The following Theorem provides a condition on the parity check matrix H in which all $(4, 4)$ trapping sets are eliminated if each variable node is connected to exactly one super check.

Theorem 2. *Let $\mathcal{C}$ be a $(3, \rho, 8)$ QC-LDPC code with the parity check matrix H. 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: The proof is given in [15].

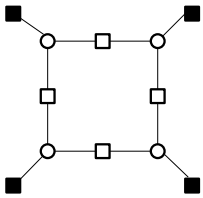

Fig. 12. The (4,4) trapping set is still harmful if each variable is connected to exactly one super check where have been replaced instead of degree-1 single parity checks in the trapping set.

We finish this section by providing a lower bound on the rate of the check-hybrid GLDPC codes.

Theorem 3. *Let $\mathcal{C}$ be a (γ, ρ)-regular LDPC code with the parity-check matrix $H_{M\times N}$. Let C be a t-error correcting component code of rate r with a full-rank parity-check matrix $H'_{m\times\rho}$. If κ be the number of single parity checks in H that are replaced by super checks corresponding to C, then the rate of the check-hybrid GLDPC code*

$$R \geq 1 - \frac{\gamma}{\rho} - \kappa\lambda(1 - r)$$

where $\lambda = \frac{\rho}{N}$.

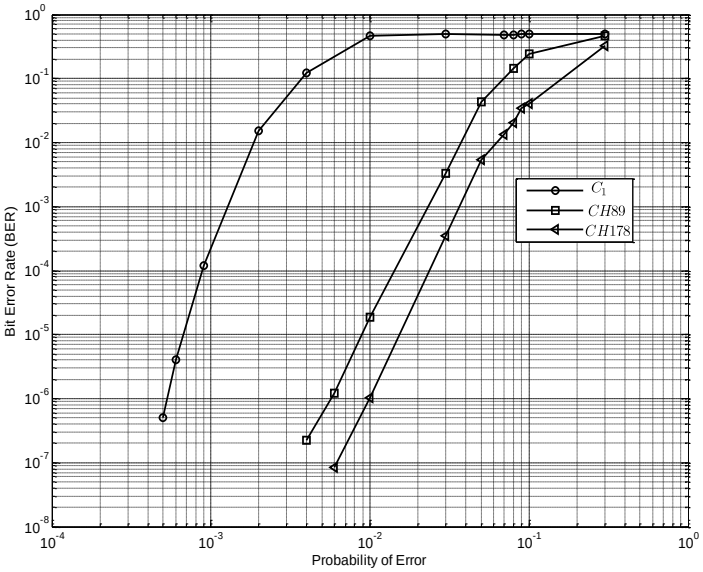

Fig. 13. The performance of the check-hybrid GLDPC codes using a $(3, 15, 8)$-regular code as a global code and the BCH(15,7) as the component code on the BSC and under the Gallager B decoding algorithm.

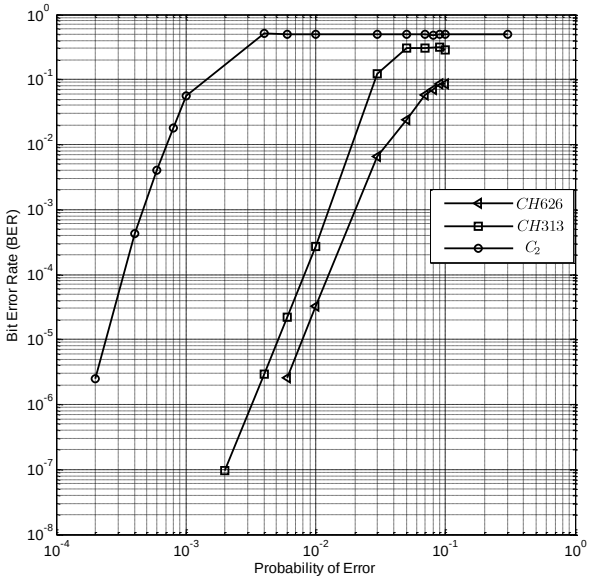

Fig. 14. The performance of the check-hybrid GLDPC codes using a $(3, 31, 8)$-regular code as a global and the BCH(31,21) as the component code on the BSC and under the Gallager B decoding algorithm.

Proof: If κ be the number of super checks that are replaced in H, then there will be $(\kappa m + (M - \kappa))$ rows in the parity check matrix of the check-hybrid GLDPC codes. Thus, the rate of the check-hybrid GLDPC code is:

$$\begin{aligned} R &\geq 1 - \frac{(\kappa m + (M - \kappa))}{N} \\ &\geq 1 - \frac{\gamma}{\rho} - \kappa(1 - r)\frac{\rho}{N} \end{aligned}$$

where the last inequality follows from the fact that $1 - \frac{M}{N} = 1 - \frac{\gamma}{\rho}$ and $m - 1 < \rho(1 - r)$. Assuming $\lambda = \frac{\rho}{N}$ proves the result. *Q.E.D.*

Corollary 2. *Let $\mathcal{C}$ be a (γ, ρ)-regular LDPC code and let $H_{M\times N}$ be the parity-check matrix based on permutation matrices with size p. Let C be a t-error correcting component code of rate r with a full-rank parity-check matrix $H'_{m\times\rho}$. If $\kappa = \alpha p$ for $0 \leq \alpha \leq \gamma$ be the number of single parity checks in H that are replaced by super checks corresponding to C, then the rate of the check-hybrid GLDPC code is:*

$$R \geq 1 - \frac{\gamma}{\rho} - \alpha(1 - r).$$

V. Numerical Results

In this section, we provide the numerical results for the performance of different check-hybrid GLDPC codes on the BSC. Although the PBF is a simple algorithm with low complexity and suitable for analysis, it is not as strong as message passing algorithms such as the Gallager B and the belief propagation (BP) algorithms. Thus, for the simulations, we use the modification of the Gallager B decoding algorithm. The Gallager B decoding algorithm for the check-hybrid GLDPC codes has the same updating rules at variable nodes and check nodes as the Gallager B decoding and in addition, there is a BDD at super checks.

We also use the permutation-based $(3, \rho, 8)$ LDPC codes [16] as the global codes and 2-error correcting BCH codes for the component codes. The first class of constructed check-hybrid GLDPC codes has a $C_1(3, 15)$-regular LDPC code of length 1335 with the permutation matrix of size $p_1 = 89$ as its global code and the 2-error correcting BCH(15,7) as the component code. The first check-hybrid code, $CH89$ has $p_1 = 89$ super checks and 178 single parity checks and the second check-hybrid code, $CH178$ has $2p_1 = 178$ super checks and 89 single parity checks. The performance of these codes under the Gallager B algorithm is provided in Fig. 13. We have also shown the performance of the $(3, 15)$-regular LDPC code, C_1, in Fig. 13.

The second class of check-hybrid codes has $C_2(3, 31, 8)$-regular permutation-based LDPC code of length 9703 as the global code and the 2-error correcting BCH(31,21) as the component code. The size of the permutation matrix of the global code is $p_2 = 313$. We show the performance of two check-hybrid codes which are constructed by replacing 313 and then 626 super checks in the parity check matrix of C_2. The performances of $CH313$ with 313 super checks and $CH626$ with 626 super checks are shown in Fig. 14. The performance of C_2 is also depicted in Fig. 14.

VI. 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 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 minimizing the rate

loss caused by adding more constraints on check nodes of the component code.

Future work includes improving the upper bounds on the splitting number of trapping sets and extending the results to other ensembles of LDPC codes as the global codes and different t-error correcting component codes. Furthermore, it is of interest to study the harmful structures for the check-hybrid GLDPC codes and find guaranteed error correction capability of the check-hybrid GLDPC codes under the PBF and the Gallager A/B decoding algorithms.

VII. Acknowledgments

The authors would like to thank E. Dupraz for her helpful comments. This work is funded by the Seagate Technology and in part by the NSF under grants CCF-0963726 and CCF-1314147.

References

[1] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT27, no. 5, pp. 533–547, Sept. 1981.

[2] M. Lentmaier and K. S. Zigangirov, "On generalized low-density parity-check codes based on Hamming component codes," *IEEE Commun. Letters*, vol. 3, no. 8, pp. 248–250, Aug. 1999.

[3] N. Miladinovic and M. P. C. Fossorier, "Generalized LDPC codes and generalized stopping sets," *IEEE Trans. Commun.*, vol. 56, no. 2, pp. 201–212, Feb. 2008.

[4] S. K. Chilappagari, D. V. Nguyen, B. Vasic, and M. W. Marcellin, "On trapping sets and guaranteed error correction capability of LDPC codes and GLDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1600–1611, Apr. 2010.

[5] J. Chen and R. M. Tanner, "A hybrid coding scheme for the Gilbert-Elliott channel," *IEEE Trans. Commun.*, vol. 54, no. 10, pp. 1787–1796, Oct. 2006.

[6] G. Liva and W. E. Ryan, "Short low-error-floor Tanner codes with Hamming nodes," in *Proc. IEEE Milcom*, vol. 1, Atlantic City, NJ, Oct. 17–20 2005, pp. 208–213.

[7] G. Liva, W. E. Ryan, and M. Chiani, "Quasi-cyclic generalized LDPC codes with low error floors," *IEEE Trans. Commun.*, vol. 56, no. 1, pp. 49–57, Jan. 2008.

[8] E. Paolini, M. F. Flanagan, M. Chiani, and M. P. C. Fossorier, "Spectral shape of check-hybrid GLDPC codes," in *Proc. IEEE Int. Conf. Commun.*, Cape Town, May 23–27 2010, pp. 1–6.

[9] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annual Allerton Conf. on Commun., Control, and Computing*, Allerton House, Monticello, IL, USA, Oct. 1–3 2003, pp. 1426–1435.

[10] B. Vasic, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery, "Trapping set ontology," in *Proc. Allerton Conf. on Commun., Control, and Computing*, Monticello, IL, USA, Sept. 30-Oct. 2 2009, pp. 1–7.

[11] R. G. Gallager, "Low density parity check codes," *IRE Trans. Inf. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[12] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. IEEE Int. Conf. Commun.*, vol. 3, Istanbul, Turkey, Jun. 11–15 2006, pp. 1089–1094.

[13] "Trapping set ontology." [Online]. Available: http://www2.engr.arizona.edu/~vasiclab/ProjectsHome.php

[14] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.

[15] V. Ravanmehr, D. Declercq, and B. Vasic, "Check-hybrid GLDPC codes:systematic elimination of trapping sets by super checks," in *Submitted to Int. Symp. Inf. Theory*, 2014.

[16] D. V. Nguyen, S. K. Chilappagari, M. W. Marcellin, and B. Vasic, "On the construction of structured LDPC codes free of small trapping sets," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2280–2302, Apr. 2012.