On the Construction of Structured LDPC Codes Free of Small Trapping Sets

Dung Viet Nguyen, *Student Member, IEEE*, Shashi Kiran Chilappagari, *Member, IEEE*, Michael W. Marcellin, *Fellow, IEEE*, and Bane Vasić, *Fellow, IEEE*

***Abstract*—We present a method to construct low-density parity-check (LDPC) codes with low error floors on the binary symmetric channel. Codes are constructed so that their Tanner graphs are free of certain small trapping sets. These trapping sets are selected from the trapping set ontology for the Gallager A/B decoder. They are selected based on their relative harmfulness for a given decoding algorithm. We evaluate the relative harmfulness of different trapping sets for the sum–product algorithm by using the topological relations among them and by analyzing the decoding failures on one trapping set in the presence or absence of other trapping sets. We apply this method to construct structured LDPC codes. To facilitate the discussion, we give a new description of structured LDPC codes whose parity-check matrices are arrays of permutation matrices. This description uses Latin squares to define a set of permutation matrices that have disjoint support and to derive a simple necessary and sufficient condition for the Tanner graph of a code to be free of four cycles.**

***Index Terms*—Error floor, Latin squares, structured low-density parity-check codes, trapping sets.**

I. Introduction

By now, it is well established that the error-floor phenomenon, an abrupt degradation in the error rate performance of low-density parity-check (LDPC) codes in the high signal-to-noise-ratio (SNR) region, is due to the presence of certain structures in the Tanner graph that lead to decoder failures [1]. For iterative decoding, these structures are known as trapping sets (see [2] for a list of references).

To construct LDPC codes with provably low error floors, it is essential to understand the failure mechanism of the decoders in the high SNR region as well as to fully characterize trapping sets. These prerequisites are met for decoders on the binary erasure channel (BEC), in which case trapping sets are known under the notion of stopping sets [3]. For the BEC, the definition of stopping sets is fully combinatorial and the code construction strategy is simply to maximize the size of the smallest stopping set as well as to minimize the number of the smallest stopping sets. Such a level of understanding has not been gained for other channels of interest.

On other channels, such as the binary symmetric channel (BSC) or the additive white Gaussian noise channel (AWGNC), knowledge of trapping sets is far from complete due to the complex nature of iterative decoding algorithms, such as the sum–product algorithm (SPA). As a result, code performance is typically improved by increasing the girth of the Tanner graph [4]–[8]. The basis for these approaches is mostly based on two facts. First, a linear increase in the girth results in an exponential increase of the lower bound on the minimum distance if the code has column weight $d_{\mathrm{v}} \geq 3$ [9]. Second, trapping sets containing shortest cycles in the Tanner graph are eliminated when the girth is increased. In addition, several recent results can be used to justify the construction of a code with large girth: the error correction capability under the bit flipping algorithms was shown to grow exponentially with the girth for codes with column weight $d_{\mathrm{v}} \geq 5$ [10]; and the lower bound on the minimum BSC pseudocodeword weight for linear programming decoding was also shown to increase exponentially with the girth [11]. Notably, this lower bound on the minimum BSC pseudocodeword weight of an LDPC code whose Tanner graph has girth greater than 4 was proven to be tight if and only if the minimum pseudocodeword is a real multiple of a codeword [12]. It is worth noting here that the lower bound on the minimum stopping set size also grows exponentially with the girth for codes with column weight $d_{\mathrm{v}} \geq 3$ [13].

For a given column weight d_{v}, increasing the girth of a Tanner graph requires either increasing the number of variable nodes, thus requiring a longer code, or decreasing the row weight d_{c} and increasing the number of check nodes, which lowers the code rate. In most cases, at a desirable length and code rate, the girth cannot be made large enough for the Tanner graph to be free of the most harmful trapping sets that mainly contribute to decoding failures in the error-floor region. These trapping sets dictate the size of the smallest error patterns uncorrectable by the decoder and hence also dictate the slope of the frame error rate (FER) curve [2]. To preserve the rate while lowering the error floor, a code must be optimized not by simply increasing the girth but rather by more surgically avoiding the most harmful trapping sets.

Manuscript received August 24, 2010; revised September 21, 2011; accepted September 27, 2011. Date of current version March 13, 2012. This work was supported by the National Science Foundation under Grant CCF-0963726, Grant CCF-0830245, and in part by the Advanced Storage Technology Consortium, International Disk Drive Equipment and Materials Association. The material in this paper was presented in part at the 2010 IEEE Information Theory Workshop, Dublin, Ireland, at the 47th Annual Allerton Conference on Communications, Control, and Computing, Monticello, IL, Sep. 2009, and at the 48th Annual Allerton Conference on Communications, Control, and Computing, Monticello, IL, Sep. 2010.

D. V. Nguyen, M. W. Marcellin, and B. Vasić are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721 USA (e-mail: nguyendv@ece.arizona.edu; marcellin@ece.arizona.edu; vasic@ece.arizona.edu).

S. K. Chilappagari was with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721 USA. He is now with Marvell Semiconductors, Inc., Santa Clara, CA 95054 USA (e-mail: shashickiran@gmail.com).

Communicated by P. O. Vontobel, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2011.2173733

In this paper, LDPC codes are constructed so that they are free of small harmful trapping sets. We focus our attention on regular column-weight-three codes as these codes allow low decoding complexity but exhibit high error floor if they are not designed properly. A key element in the construction of a code free of trapping sets is the choice of forbidden subgraphs in the Tanner graph, since this choice greatly affects the error performance as well as the code rate. This choice is well determined if the Gallager A/B algorithm is used on the BSC since the necessary and sufficient conditions for a code to guarantee the correction of a given number of errors are known [14], [15]. However, for the SPA on the BSC and on the AWGNC, the choice of forbidden subgraphs is not clear since the conditions on a Tanner graph for the code to achieve a given guaranteed error correction capability have not been characterized. In a series of papers [16]–[18], we used the notion of *instantons* to predict the error floors as well as to study the phenomenon from a statistical mechanics perspective. In [19], we showed how the family of instanton based techniques can be used to estimate and reduce error floors for different decoders operating on a variety of channels. Unfortunately, the instanton search is computationally prohibitive for the construction of moderate length codes, and in this paper, we propose another, simpler, method.

In the absence of a complete understanding of trapping sets for the SPA, the choice of forbidden subgraphs may be derived based on the understanding of trapping sets for simpler decoding algorithms as well as on the intuition gained from experimental results. This is the approach we take in this paper. A basis for removing harmful trapping sets for the SPA is the observation made in [19] that the decoding failures for various decoding algorithms and channels are closely related and that subgraphs responsible for these failures share some common underlying topological structures. These structures are either trapping sets for iterative decoding algorithms on the BSC or larger subgraphs containing these trapping sets.

The method consists of three main steps. First, we develop a database of trapping sets for the Gallager A/B algorithm on the BSC. This database, which is called the trapping set ontology (TSO),[1] contains subgraphs that are responsible for failures of the Gallager A/B decoder and also specifies the topological relations among them. Second, based on the TSO, we determine the *relative harmfulness* of different subgraphs for the SPA on the BSC by analyzing failures of the decoder on one subgraph in the presence or absence of other topologically related subgraphs. This analysis is performed repeatedly on a number of "test" Tanner graphs, which are intentionally constructed to either contain or be free of specific subgraphs. The relative harmfulness of a subgraph is evaluated based on its effect on the guaranteed correction capability of a code. Finally, a code is constructed so that its Tanner graph is free of the most harmful subgraphs.

We remark that our construction attempts to optimize a code for the SPA on the BSC. Due to much higher complexity, similar analysis on the AWGNC is difficult. However, experimental results show that codes constructed for the BSC also perform very well on the AWGNC. It should be noted that in [22], extensive computer simulation and hardware emulation suggest that *absorbing sets* mainly contribute to error floors of codes under the SPA on the AWGNC. Since absorbing sets are combinatorially similar to trapping sets for the Gallager A/B decoder, our newly constructed codes are also free of some (and probably the most harmful) absorbing sets and hence understandably possess good error performance on the AWGNC. Although absorbing sets were defined in the context of research that dealt with the AWGNC, the failure mechanism of the SPA due to these objects is not understood well enough to suggest an explicit strategy to construct good codes. As a result, optimizing codes for the BSC in order to obtain good performance on the AWGNC remains a reasonable approach.

As the title of this paper indicates, we focus on constructing structured LDPC codes. This is motivated by the fact that these codes are attractive for a number of applications. For example, encoding of quasi-cyclic (QC) LDPC codes can be efficiently implemented using shift registers with linear complexity [23], while decoding can be parallelized by exploiting the block structure of the parity-check matrices [24], [25]. Furthermore, as we show in this paper, symmetry and structure in a Tanner graph can greatly accelerate the trapping set search and enumeration as well as code construction.

To facilitate the discussion on removing trapping sets in structured codes, we give a new description of structured LDPC codes whose parity-check matrices are arrays of permutation matrices. In this description, Latin squares are used to define a set of permutation matrices that have disjoint support and to derive a simple necessary and sufficient condition for the Tanner graph of a code to be free of four cycles. As this description is concise and simple, it facilitates our discussion on the construction of codes free of small trapping sets. Besides, the class of codes to be described is general as it includes many existing structured LDPC codes. The new description also results in certain advantages. For example, the class of codes to be described contains array LDPC codes [26] but also includes higher rate codes than shortened array LDPC codes [4], [27], when the Tanner graphs are required to satisfy certain constraints.

The rest of this paper is organized as follows. Section II presents our TSO for the Gallager A/B decoder. The analytical construction of a code free of trapping sets is difficult, and hence, we resort to an efficient search of the Tanner graph for certain subgraphs. We briefly discuss these search techniques in Section III, with more details given in Appendix A. In Section IV, we describe structured LDPC codes whose parity-check matrices are arrays of permutation matrices obtained from Latin squares. In Section V, we describe, in general, the construction of a code free of certain trapping sets. We present the construction of codes for the Gallager A/B algorithm in Section VI and the construction of codes for the SPA on the BSC in Section VII. In Section VIII, we discuss the performance of a constructed code on the AWGNC and then conclude the paper.

Before proceeding to Section II, we provide some background related to LDPC codes. Let $\mathcal{C}$ denote an (n,k) LDPC code over the binary field $\mathrm{GF}(2)$. $\mathcal{C}$ is defined by the null space of H, an $m \times n$ *parity-check matrix* of $\mathcal{C}$. H is the biadja-

[1]This database of trapping sets was partially presented in [20] and is available online at [21].

cency matrix of G, a Tanner graph representation of $\mathcal{C}$. G is a bipartite graph with two sets of nodes: n variable (bit) nodes $V = \{1, 2, \ldots, n\}$ and m check nodes $C = \{1, 2, \ldots, m\}$. A vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a codeword if and only if $\mathbf{x}H^T = \mathbf{0}$, where H^T is the transpose of H. The support of $\mathbf{x}$, denoted as $\mathrm{supp}(\mathbf{x})$, is defined as the set of all variable nodes (bits) $v \in V$ such that $x_v \neq 0$. A d_{v}-left-regular LDPC code has a Tanner graph G in which all variable nodes have degree d_{v}. Similarly, a d_{c}-right-regular LDPC code has a Tanner graph G in which all check nodes have degree d_{c}. A $(d_{\mathrm{v}}, d_{\mathrm{c}})$-regular LDPC code is d_{v}-left-regular and d_{c}-right-regular. Such a code has rate $R \geq 1 - d_{\mathrm{v}}/d_{\mathrm{c}}$ [28]. The degree of a variable node (check node, respectively) is also referred to as the left degree (right degree, respectively) or the column weight (row weight, respectively). The length of the shortest cycle in the Tanner graph G is called the girth g of G.

II. TSO

In this section, we describe our trapping set database known as the TSO. The "ontology" indicates that the database is augmented by the topological relations among the trapping sets. We start with a brief discussion of trapping sets and related objects.

A. Trapping Sets

Denote by $\mathbf{x}$ the transmitted codeword. Consider an iterative decoder and let $\hat{\mathbf{x}}^l = (\hat{x}_1^l, \hat{x}_2^l, \ldots, \hat{x}_n^l)$ be the decision vector after the lth iteration. A variable node v is said to be *eventually correct* if there exists a positive integer l_{c} such that for all l with $l \geq l_{\mathrm{c}}$, $\hat{x}_v^l = x_v$.

Definition 1 [1]: A trapping set for an iterative decoding algorithm is a nonempty set of variable nodes in a Tanner graph G that are *not* eventually correct. A set of variable nodes $\mathbf{T}$ is called an (a, b) trapping set if it contains a variable nodes and the subgraph induced by these variable nodes has b odd-degree check nodes.

On the BSC, when decoding with the Gallager A/B algorithm, or the bit flipping (serial or parallel) algorithms, trapping sets are partially characterized under the notion of fixed sets. By partially, we mean that these combinatorial objects form a subclass of trapping sets, but not all trapping sets are fixed sets. Fixed sets have been studied extensively in a series of papers [2], [10], [14], [29]. They have been proven to be the cause of the error floor in the decoding of LDPC codes under the Gallager A/B algorithm and the bit flipping algorithms. For the sake of completeness, we give the definition of a fixed set as well as the necessary and sufficient conditions for a set of variable nodes to form a fixed set.

Assume the transmission of the all-zero codeword[2] over the BSC. With this assumption, a variable node is correct if it is 0 and corrupt if it is 1. Let $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ be the channel output vector and let $\mathbf{F}(y)$ denote the set of variable nodes that are not eventually correct.

[2]The all-zero-codeword assumption can be applied if the channel is output symmetric and the decoding algorithm satisfies certain symmetry conditions (see [30, Def. 1 and Lemma 1]). The Gallager A/B algorithm, the bit flipping algorithms and the SPA all satisfy these symmetry conditions.

Definition 2: For transmission over the BSC, $\mathbf{y}'$ is a fixed point of the decoding algorithm if and only if there exists a positive integer l_{f} such that $\mathrm{supp}(\mathbf{y}') = \mathrm{supp}(\hat{\mathbf{x}}^l)$ for all $l \geq l_{\mathrm{f}}$. If $\mathbf{F}(\mathbf{y}) \neq \emptyset$ and $\mathbf{y}'$ is a fixed point, then $\mathbf{F}(\mathbf{y}) = \mathrm{supp}(\mathbf{y}')$ is called a fixed set. A fixed set is an elementary fixed set if all check nodes in its induced subgraph have degree one or two. Otherwise, it is a nonelementary fixed set.

Remark: The classification of fixed sets as elementary and nonelementary fixed sets is identical to the classification of trapping sets as elementary and nonelementary trapping sets in [31].

Theorem 1 [10]: Let $\mathcal{C}$ be an LDPC code with d_{v}-left-regular Tanner graph G. Let $\mathbf{T}$ be a set consisting of variable nodes with induced subgraph $\mathbf{I}$. Let the check nodes in $\mathbf{I}$ be partitioned into two disjoint subsets; $\mathbf{O}$ consisting of check nodes with odd degree and $\mathbf{E}$ consisting of check nodes with even degree. Then, $\mathbf{T}$ is a fixed set for the bit flipping algorithms (serial or parallel) iff: 1) every variable node in $\mathbf{I}$ has at least $\lceil \frac{d_{\mathrm{v}}}{2} \rceil$ neighbors in $\mathbf{E}$, and 2) no collection of $\lfloor \frac{d_{\mathrm{v}}}{2} \rfloor + 1$ check nodes of $\mathbf{O}$ share a neighbor outside $\mathbf{I}$.

Note that Theorem 1 only states the conditions for the bit flipping algorithms. However, it is not difficult to show that these conditions also apply for the Gallager A/B algorithm. A similar characterization of fixed sets for the Gallager A/B algorithm is also given in [32].

The harmfulness of a fixed set is determined by its critical number. A fixed set is more harmful if it has a smaller critical number. The critical number of a fixed set is defined as follows.

Definition 3 [29]: The critical number of a fixed set is the minimal number of variable nodes that have to be initially in error for the decoder to end up in the fixed set.

Determining the smallest critical number of fixed sets present in a code or in general determining the weight of the smallest uncorrectable error patterns is a key step for estimating the FER performance of the code in the error-floor region. The problem of estimating the error floor of LDPC codes under hard-decision decoding on the BSC was considered in [29], [32], [33].

Although it has been rigorously proven only that fixed sets are trapping sets for the Gallager A/B algorithm and the bit flipping algorithms on the BSC, it has been widely recognized in the literature that the subgraphs of these combinatorial objects greatly contribute to the error floor for various iterative decoding algorithms and channels. The instanton analysis performed in [19] suggests that the decoding failures for various decoding algorithms and channels are closely related and subgraphs responsible for these failures share some common underlying topological structures. These structures are either trapping sets for iterative decoding algorithms on the BSC, of which fixed sets form a subset, or larger subgraphs containing these trapping sets. In [34], the notion of absorbing sets was defined. In odd-column-weight codes, these are sets of variable nodes which satisfy condition 1 of Theorem 1. These authors also defined fully absorbing sets, which are combinatorially identical to fixed sets (in odd-column-weight codes). By hardware emulation, they found that absorbing sets are the main cause of error floors for the SPA on the AWGNC. Various trapping sets identified by simulation (for example, those in [35] and [36]) are also fixed sets.

From these observations, it is expected that an LDPC code will have good performance in the error-floor region if the corresponding Tanner graph does not contain subgraphs induced by fixed sets. However, it is impossible to construct an LDPC code whose Tanner graph is free of all fixed sets when the length of the code is finite. It is also well known that imposing constraints on a Tanner graph reduces the rate of a code. Clearly, only subgraphs of some fixed sets can be avoided in the code construction. These need to be chosen carefully in order to obtain the best possible error-floor performance while maximizing the code rate.

Before one can attempt to determine the fixed sets that shall be forbidden in the Tanner graph of a code, there are two important issues that need to be addressed. First, a complete list of nonisomorphic fixed sets (up to a proper size) for a given set of code parameters (e.g., column weight and row weight) is needed. This is because the notion of an (a, b) fixed set (trapping set) is not sufficient. Given a pair of positive integers (a, b), there are possibly many fixed sets which induce nonisomorphic subgraphs containing a variable nodes and b odd-degree check nodes. Second, the topological relations among subgraphs induced by fixed sets need to be explored. The importance of these relations is threefold. First, the subgraph induced by a fixed set may be contained in the subgraph induced by another fixed set. In such a case, the absence of one subgraph yields to the absence of the other. Second, these relations help reduce the complexity of the search for subgraphs in a Tanner graph. Finally, these relations reduce the complexity of the analysis to determine the harmfulness of subgraphs.

In Section II-B, we present our database of fixed sets for regular column-weight-three LDPC codes with the special emphasis on their topological relationships. For the sake of simplicity, and because the term "trapping set" is widely used in the literature, we drop the term fixed sets and refer to these objects by the general term trapping sets. Therefore, in the remainder of this paper, a trapping set should not be understood as a set of noneventually correct variable nodes. Instead, it should be understood as a set of variable nodes in a given code with a specified induced subgraph or as a specific subgraph independent of a code (a precise description is given in Section II-B). In this context, we use the term *erroneous set* to refer to a set of noneventually correct variable nodes. We also remark that in this paper, if a code is said to be free of some trapping sets, then these trapping sets should be understood as trapping sets of the Gallager A/B algorithms.

B. TSO of Column-Weight-Three Codes for the Gallager A/B Algorithm on the BSC

In the remainder of this paper, we use the following definition of a trapping set.

Definition 4: Trapping sets are fixed sets of the Gallager A/B algorithm, i.e., trapping sets are sets of variable nodes whose induced subgraphs satisfy conditions 1 and 2 of Theorem 1.

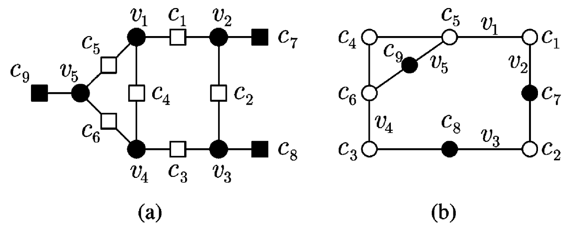

Fig. 1. Graphical representation of the $(5, 3)\{2\}$ trapping set: (a) Tanner graph representation. (b) Line–point representation.

This slight abuse of terminology is motivated by the desire to use the terminology that is common in the literature.

1) Graphical Representation: The induced subgraph of a trapping set (or any set of variable nodes) is a bipartite graph. In the Tanner graph (bipartite graph) representation of a trapping set, we use "●" to represent variable nodes, "■" to represent odd-degree check nodes, and "□" to represent even-degree check nodes. There exists an alternate graphical representation of trapping sets which allows their topological relations to be established more conveniently. This graphical representation is based on the incidence structure of lines and points. In combinatorial mathematics, an incidence structure is a triple $(\mathscr{P}, \mathscr{L}, \mathscr{I})$ where $\mathscr{P}$ is a set of "points", $\mathscr{L}$ is a set of "lines," and $\mathscr{I} \subseteq \mathscr{P} \times \mathscr{L}$ is the incidence relation. The elements of $\mathscr{P}$ are called flags. If $(\mathfrak{p}, \mathfrak{l}) \in \mathscr{I}$, we say that point $\mathfrak{p}$ "lies on" line $\mathfrak{l}$. In this line–point representation of trapping sets, variable nodes correspond to lines and check nodes correspond to points. A point, represented as a circle, is shaded black if it has an odd number of lines passing through it; otherwise, it is shaded white. An (a, b) trapping set is thus an incidence structure with a lines and b black-shaded points. The girth of the line–point representation equals the girth of the associated Tanner graph representation. To differentiate among (a, b) trapping sets that have nonisomorphic induced subgraphs when necessary, we index (a, b) trapping sets in an arbitrary order and assign the notation $(a, b)\{i\}$ to the (a, b) trapping set with index i.

Depending on the context, a trapping set can be understood as a set of variable nodes in a given code with a specified induced subgraph or it can be understood as a specific subgraph independent of a code. To differentiate between these two cases, we use the letter $\mathbf{T}$ to denote a set of variable nodes in a code and use the letter $\mathcal{T}$ to denote a type of trapping set which corresponds to a specific subgraph. If the induced subgraph of a set of variable nodes $\mathbf{T}$ in the Tanner graph of a code $\mathcal{C}$ is isomorphic to the subgraph of $\mathcal{T}$ then we say that $\mathbf{T}$ is a $\mathcal{T}$ trapping set or that $\mathbf{T}$ is a trapping set of type $\mathcal{T}$. $\mathcal{C}$ is said to contain type $\mathcal{T}$ trapping set(s).

Example 1: The $(5, 3)\{1\}$ trapping set $\mathcal{T}_1$ is a union of a six-cycle and an eight-cycle, sharing two variable nodes. The Tanner graph representation of $\mathcal{T}_1$ is shown in Fig. 1(a). The set of odd-degree check nodes is $\{c_7, c_8, c_9\}$. These check nodes are represented by black-shaded squares. In the line–point representation of $\mathcal{T}_1$ which is shown in Fig. 1(b), c_7, c_8, and c_9 are represented by black-shaded points. These points are the only points that lie on a single line. The five variable nodes $v_1, v_2, \ldots, v_5$ are represented by black-shaded circles in Fig. 1(a). They correspond to the five lines in Fig. 1(b). As an example, the column-weight-three MacKay random code of length 4095 from [37] has 19 617 sets of variable nodes whose induced subgraphs are isomorphic to the subgraph of $\mathcal{T}_1$. These sets of variable nodes are $(5,3)\{1\}$ trapping sets.

Remark: To avoid confusion between the graphical representations of trapping sets, we note that the Tanner graph representation of a trapping set always contains □ or ■. The line–point representation never contains □ or ■. We also note that circles represent variable nodes in a Tanner graph representation but they represent check nodes in a line–point representation. In the remainder of this paper, we only use the line–point representation.

2) Topological Relation: The following definition gives the topological relations among trapping sets.

Definition 5: A trapping set $\mathcal{T}_2$ is a successor of a trapping set $\mathcal{T}_1$ if there exists a proper subset of variable nodes of $\mathcal{T}_2$ that induce a subgraph isomorphic to the induced subgraph of $\mathcal{T}_1$. If $\mathcal{T}_2$ is a successor of $\mathcal{T}_1$ then $\mathcal{T}_1$ is a predecessor of $\mathcal{T}_2$. Furthermore, $\mathcal{T}_2$ is a direct successor of $\mathcal{T}_1$ if it does not have a predecessor $\mathcal{T}_3$ which is a successor of $\mathcal{T}_1$.

Remark: A trapping set $\mathcal{T}$ can have multiple, incomparable predecessors.

If $\mathcal{T}_2$ is a successor of $\mathcal{T}_1$, then the topological relation between $\mathcal{T}_1$ and $\mathcal{T}_2$ is solely dictated by the topological properties of their subgraphs. In the Tanner graph of a code $\mathcal{C}$, the presence of a trapping set $\mathbf{T}_1$ does not indicate the presence of a trapping set $\mathbf{T}_2$. If $\mathbf{T}_1$ is indeed a subset of a trapping set $\mathbf{T}_2$ in the Tanner graph of $\mathcal{C}$, then we say that $\mathbf{T}_1$ generates $\mathbf{T}_2$; otherwise, we say that $\mathbf{T}_1$ does not generate $\mathbf{T}_2$.

3) Family Tree of Trapping Sets: The following proposition follows from Theorem 1.

Proposition 1: Every trapping set $\mathcal{T}$ contains at least a cycle.

Proof: Assume that $\mathcal{T}$ is a trapping set that does not contain a cycle, i.e., the induced subgraph of $\mathcal{T}$ is a tree. Take any variable node as the root of the tree then the variable nodes which are neighboring to the leaf nodes with largest depth have only one adjacent check node with degree greater than 1. Therefore, these variable nodes have at least as many odd-degree check nodes as even-degree check nodes. This indicates that $\mathcal{T}$ is not a trapping set, which is a contradiction. Consequently, all trapping sets can be obtained by adjoining variable nodes to cycles. ■

Remark: Any cycle is a trapping set for regular column-weight-three codes.

We now explain how larger trapping sets can be obtained by adjoining variable nodes to smaller trapping sets. We begin with the simplest example: the evolution of $(5,b)$ trapping sets from the $(4,4)$ trapping set.

Example 2: The line–point representation of a $(5,b)$ trapping set may be obtained by adding one additional line to the line–point representation of the $(4,4)$ trapping set. The new line must pass through exactly three points to conform with the given variable node degree. The process of adding a new line can be considered as the merging of at least one point on the new line with certain points in the line–point representation of the $(4,4)$ trapping set. We use ⊛ to denote the points on the line that are to be merged with points in the line–point representation of the predecessor trapping set. If a black-shaded point is merged with a ⊛ point, then they become a single white-shaded point. Similarly, if a white-shaded point is merged with a ⊛ point, then the result is a single black-shaded point. Before merging points, one must decide on: 1) the number of the ⊛ points on the new line; and 2) which points on the line–point representation of the predecessor trapping set are to be merged with the ⊛ points.

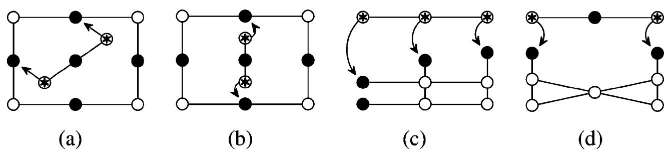

Fig. 2. $(5,b)$ trapping sets can be obtained by adding a line to the $(4,4)$ trapping set: (a) $(5,3)\{1\}$ trapping set, (b) $(5,3)\{2\}$ trapping set, and (c) $(5,1)$ trapping set; or by adding a line to the $(4,2)$ trapping set: (d) $(5,1)$ trapping set.

Because the merging must ensure that every line passes through at least two white-shaded points, there must be at least two ⊛ points on the line to be merged, i.e., there can be two or three ⊛ points. It is easy to see that if the girth is at least six, then white points cannot be selected. Consequently, if there are two ⊛ points, then there are two distinct ways to select two black-shaded points. The merging, which is demonstrated in Fig. 2(a) and (b), results in two different $(5,3)$ trapping sets. On the other hand, if there are three ⊛ points, then there is only one distinct way to select three black-shaded points. The merging, which is demonstrated in Fig. 2(c), results in the $(5,1)$ trapping set. We remark that the line–point representation of the $(5,1)$ trapping set may also be obtained by adding a line to the line–point representation of the $(4,2)$ trapping set, as demonstrated in Fig. 2(d). Note that the $(4,2)$ trapping set is neither a successor nor a predecessor of the $(4,4)$ trapping set.

The evolution of a trapping set from one of its predecessors in a regular-column-weight three code can now be described in a more general setting. Since every trapping set is a direct successor of some trapping set, it is sufficient to only consider the evolution of direct successors from their predecessors. Consider an (a,b) trapping set $\mathcal{T}_1$. Since $\mathcal{T}_1$ has a variable nodes, its line–point representation contains a lines. Each line has three points lying on it, with at most one point shaded black. There are b black-shaded points, each with an odd number of lines passing through it. An $(a+u, b+z)$ trapping set $\mathcal{T}_2$ may be obtained by adding u lines. These u new lines (and the points on them) form an incidence structure and since $\mathcal{T}_2$ is a direct successor of $\mathcal{T}_1$, this incidence structure is connected.[3] A successor trapping set $\mathcal{T}_2$ is obtained by pairwisely merging the ⊛ points with certain points of $\mathcal{T}_1$. The evolution of elementary trapping sets from their predecessors is further simplified by the following lemma.

Lemma 1: Let $\mathcal{T}$ be an elementary trapping set. Then, any elementary direct successor of $\mathcal{T}$ can be obtained by pairwisely merging the ⊛ points of one of those incidence structures listed in Fig. 3 with certain black-shaded points of $\mathcal{T}$.

Proof: See Appendix B. ■

[3]Each incidence structure corresponds to a bipartite graph. An incidence structure is connected if the corresponding bipartite graph is connected.

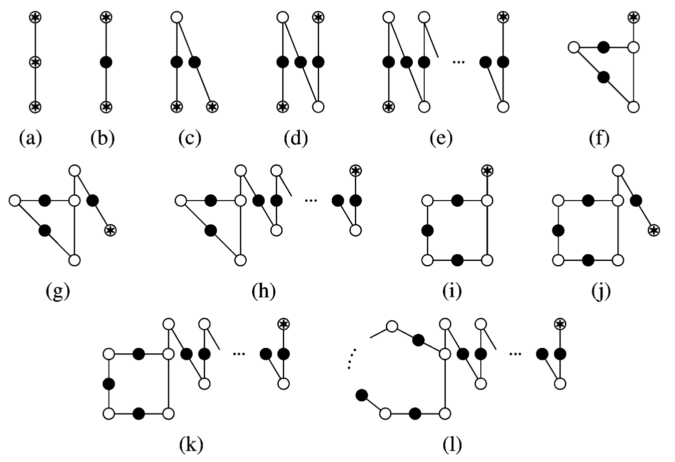

Fig. 3. Possible incidence structures formed by u new lines for elementary trapping sets.

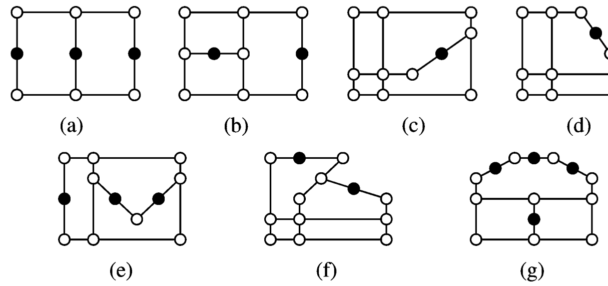

Fig. 4. $(5,3)\{1\}$ trapping set and its successors of size less than or equal to 8 in girth-8 LDPC codes. (a) $(5,3)\{2\}$. (b) $(6,2)\{1\}$. (c) $(7,1)\{1\}$. (d) $(8,2)\{1\}$. (e) $(7,3)\{1\}$. (f) $(8,2)\{2\}$. (g) $(8,4)\{1\}$.

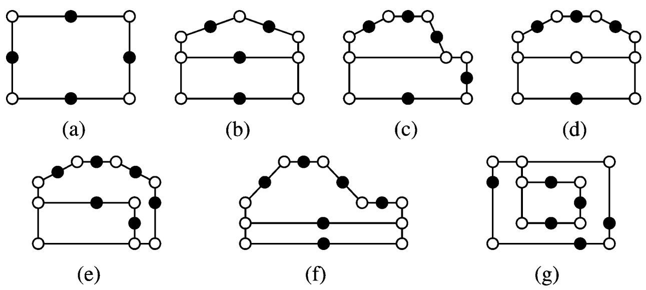

Fig. 5. $(4,4)$ trapping set and its direct successors of size less than or equal to 8 in girth-8 LDPC codes (excluding the $(5,3)\{1\}$ and $(6,4)\{1\}$ trapping sets shown in Figs. 1(b) and 6(a), respectively). (a) $(4,4)$. (b) $(6,4)\{2\}$. (c) $(7,5)\{1\}$. (d) $(7,5)\{2\}$. (e) $(8,6)\{1\}$. (f) $(8,6)\{2\}$. (g) $(8,6)\{3\}$.

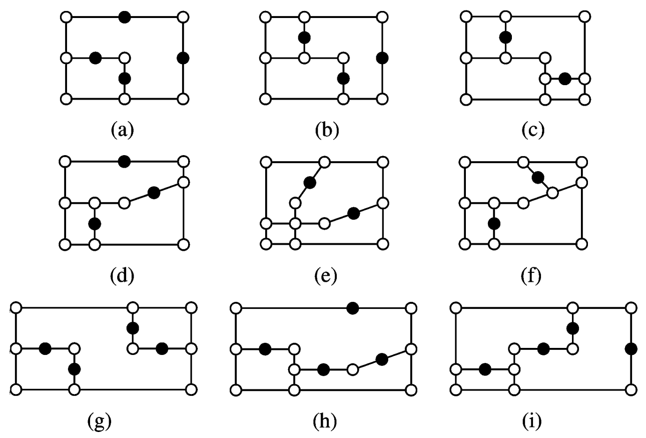

Fig. 6. $(6,4)\{1\}$ trapping set and its successors of size less than or equal to 8 in girth-8 LDPC codes. (a) $(6,4)\{1\}$. (b) $(7,3)\{2\}$. (c) $(8,2)\{3\}$. (d) $(7,3)\{3\}$. (e) $(8,2)\{4\}$. (f) $(8,2)\{5\}$. (g) $(8,4)\{2\}$. (h) $(8,4)\{3\}$. (i) $(8,4)\{4\}$.

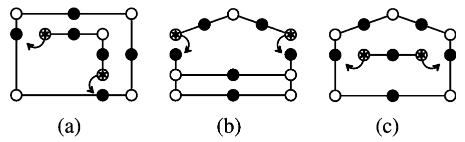

Fig. 7. Obtaining a larger trapping set by adding lines to a smaller one.

4) Examples: For the following examples, let us consider regular column-weight-three LDPC codes of girth $g = 8$ and elementary trapping sets.

Example 3: With the evolution of the $(5,3)\{2\}$ trapping set from the $(4,4)$ trapping set presented previously, we show the family tree of (a,b) trapping sets originating from the $(5,3)\{2\}$ trapping set with $a \leq 8$ and $b > 0$ in Fig. 4. The derivation of this family tree is explained as follows.

1) Since $a \leq 8$ and $b > 0$, only the structures shown in Fig. 3(b)–(d) can be chosen to merge with the $(5,3)\{2\}$ trapping set. Each of these structures have two $\circledast$ points. Due to symmetry, there is only one way of choosing two black-shaded points in the line–point representation of the $(5,3)\{2\}$ trapping set. Merging the line–point representation of the $(5,3)\{2\}$ trapping set with the structures shown in Fig. 3(b)–(d) results in the $(6,2)\{1\}$ trapping set, the $(7,3)\{1\}$ trapping set, and the $(8,4)\{1\}$ trapping set, respectively.
2) Similarly, only the structures shown in Fig. 3(b) and (c) can be chosen to merge with the $(6,2)\{1\}$ trapping set to result in the $(7,1)\{1\}$ trapping set and the $(8,2)\{1\}$ trapping set, respectively.
3) Finally, only the structure shown in Fig. 3(b) can be chosen to merge with the $(7,3)\{1\}$ trapping set. There are two distinct ways to select two black-shaded points in the line–point representation of the $(7,3)\{1\}$ trapping set. However, one of them leads to a girth violation. The other results in the $(8,2)\{2\}$ trapping set.

Example 4: By selecting two black-shaded nodes in Fig. 5(a) and merging them with two $\circledast$ nodes in Fig. 3(c), a $(6,4)$ trapping set can be obtained. Two distinct ways to select black-shaded nodes result in two different trapping sets: the $(6,4)\{1\}$ trapping set shown in Fig. 6(a) and the $(6,4)\{2\}$ trapping set shown in Fig. 5(b). The merging is demonstrated in Fig. 7(a) and (b). The family tree of (a,b) trapping sets originating from the $(6,4)\{1\}$ trapping set with $a \leq 8$ and $b > 0$ is illustrated in Fig. 6.

Example 5: In the same manner, other direct successors of the $(4,4)$ trapping set can be generated. Those (a,b) trapping sets with $a \leq 8$ and $b > 0$ are shown in Fig. 5. For a more complete list of trapping sets from the TSO, interested readers are referred to [21].

Remarks: A trapping set may originate from different predecessors. For example, the $(6,4)\{2\}$ trapping set is not only a

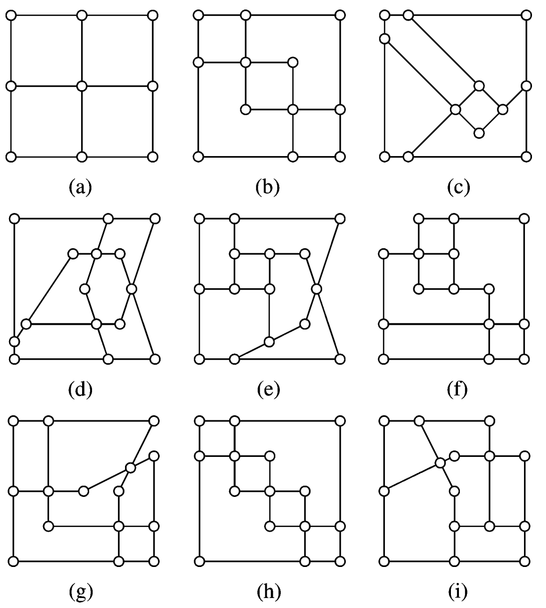

Fig. 8. All $(a, 0)$ trapping sets where $a \leq 10$ in girth-8 LDPC codes. (a) $(6,0)\{1\}$. (b) $(8,0)\{1\}$. (c) $(8,0)\{2\}$. (d) $(10,0)\{1\}$. (e) $(10,0)\{2\}$. (f) $(10,0)\{3\}$. (g) $(10,0)\{4\}$. (h) $(10,0)\{5\}$. (i) $(10,0)\{6\}$.

direct successor of the $(4,4)$ trapping set, but also a direct successor of the $(5,5)$ trapping set. The evolution of the $(6,4)\{2\}$ trapping set from the $(5,5)$ trapping set is demonstrated in Fig. 7(c).

5) Codewords: Let $\mathbf{y}$ be a codeword of $\mathcal{C}$ and let $\mathbf{T} = \mathrm{supp}(\mathbf{y})$. Then, $\mathbf{T}$ is an $(a,0)$ trapping set where $a = |\mathrm{supp}(\mathbf{y})|$. Conversely, $\mathcal{C}$ contains codewords of Hamming weight a if the Tanner graph of $\mathcal{C}$ contains $(a,0)$ trapping sets. Consequently, $\mathcal{C}$ has $d_{\min}$ as its minimum distance if and only if 1) the Tanner graph of $\mathcal{C}$ contains no $(a,0)$ trapping set where $a < d_{\min}$ and 2) the Tanner graph of $\mathcal{C}$ contains at least one $(d_{\min}, 0)$ trapping set. For regular column-weight-three codes, an $(a,0)$ trapping set is a direct successor of an $(a-1,3)$ trapping set. Therefore, the line–point representation of an $(a,0)$ trapping set may be obtained by pairwisely merging three black-shaded nodes in the line–point representation of an $(a-1,3)$trapping set with three $\circledast$ nodes in Fig. 3(a). In other words, from a list of all possible $(a-1,3)$ trapping sets, all possible $(a,0)$ trapping sets can be derived. The line–point representations of all possible $(a,0)$ trapping sets where $a \leq 10$ of girth-8 codes are shown in Fig. 8. It can be proven easily that all of these trapping sets are elementary.

III. Searching for Subgraphs in a Tanner Graph

In this section, we briefly describe the main idea behind our techniques of searching for elementary trapping sets from the TSO in the Tanner graph of a regular column-weight-three LDPC code. An efficient search of the Tanner graph for trapping sets relies on the topological relations among trapping sets defined in the TSO and/or carefully analyzing their induced subgraphs. Trapping sets are searched for in a way similar to how they have evolved in the TSO. Recall that by Proposition 1, the induced subgraph of every trapping set contains at least a cycle. Therefore, the search for trapping sets begins with enumerating cycles. Also recall that a cycle with a variable nodes is an (a,a) trapping set. After the cycles have been enumerated, they will be used in the search for larger trapping sets. A larger trapping set can be found in a Tanner graph by expanding a smaller trapping set. More precisely, given a trapping set $\mathbf{T}_1$ of type $\mathcal{T}_1$ in the Tanner graph of a code $\mathcal{C}$, our techniques search for a set of variable nodes such that the union of this set with $\mathbf{T}_1$ forms a trapping set $\mathbf{T}_2$ of type $\mathcal{T}_2$, where $\mathcal{T}_2$ is a successor of $\mathcal{T}_1$. Our techniques are sufficient to efficiently search for a large number of trapping sets in the TSO, especially for those to be avoided in the code constructions that we will present in subsequent sections. They can be easily expanded to search for other trapping sets as well. Notably, the complexity of the search for trapping sets in the Tanner graph of a structured code can be greatly reduced by utilizing the structural property of its parity-check matrix. Details on the implementation of these techniques are given in Appendix A. An implementation of our search algorithms can be downloaded from [20].

Let us mention other methods of searching for trapping sets in the Tanner graph of a code. It is well known that this problem is NP hard [38], [39]. Previous work on this problem includes exhaustive [40], [41] and nonexhaustive approaches [42], [43]. The main drawback of existing exhaustive approaches is their high complexity. Consequently, constraints must be imposed on trapping sets and on the Tanner graph in which trapping sets are searched for. For example, the method in [40] or [41] can only search for (a,b) trapping sets with $a \leq 11$, $b \leq 2$ in a Tanner graph with less than 1000 variable nodes. The complexity is much lower for nonexhaustive approaches. However, these approaches cannot guarantee that all trapping sets are enumerated, and hence are not suitable for the purpose of this paper. In [32], an algorithm was proposed for enumerating the smallest weight error patterns uncorrectable by a hard-decision decoder on the BSC instead of enumerating trapping sets. This approach is also not suitable for the purpose of this paper. We remark that our approach of searching for larger trapping sets by expanding smaller ones is also the approach used in a recent work [44].

IV. Structured LDPC Codes With Permutation Matrices Obtained From Latin Squares

In this section, we give a description of structured LDPC codes whose parity-check matrices are arrays of permutation matrices obtained from Latin squares. The class of codes described in this section is suitable to show our general method of constructing structured LDPC codes free of small trapping sets.

A. Permutation Matrices From Latin Squares

A permutation matrix is a square binary matrix that has exactly one entry 1 in each row and each column and 0's elsewhere. Our codes make use of permutation matrices that have disjoint support. These sets of permutation matrices can be obtained conveniently from Latin squares.

A *Latin square* of *size* q (or *order* q) is a $q \times q$ array in which each cell contains a single symbol from a q-set S, such that each symbol occurs exactly once in each row and exactly once in each column. A Latin square of size q is equivalent to the *Cayley table* (or *multiplication table*) of a quasi-group $\mathcal{Q}$ on q elements (see [45, pp. 135–152] for details).

For mathematical convenience, we use elements of $\mathcal{Q}$ to index the rows and columns of Latin squares and permutation matrices. Let $\mathcal{L} = [l_{i,j}]_{i,j\in\mathcal{Q}}$ denote a Latin square defined on the Cayley table of a quasi-group $(\mathcal{Q}, \otimes)$ of order q. We define f, an injective map from $\mathcal{Q}$ to $\mathrm{Mat}(q, q, \mathrm{GF}(2))$, where $\mathrm{Mat}(q, q, \mathrm{GF}(2))$ is the set of matrices of size $q \times q$ over $\mathrm{GF}(2)$, as follows:

$$\begin{aligned} f : \mathcal{Q} &\to \mathrm{Mat}(q, q, \mathrm{GF}(2)) \\ \alpha &\mapsto f(\alpha) = [m_{i,j}]_{i,j\in\mathcal{Q}} \end{aligned}$$

such that

$$m_{i,j} = \begin{cases} 1, & \text{if } l_{i,j} = \alpha \\ 0, & \text{if } l_{i,j} \neq \alpha \end{cases}.$$

According to this definition, a permutation matrix corresponding to the element $\alpha \in \mathcal{Q}$ is obtained by replacing the entries of $\mathcal{L}$ which are equal to α by 1 and all other entries of $\mathcal{L}$ by 0. It follows from the aforementioned definition that the images of elements of $\mathcal{Q}$ under f give a set of q permutation matrices that do not have 1's in common positions. This definition naturally associates a permutation matrix to an element $\alpha \in \mathcal{Q}$ and simplifies the derivation of parity-check matrices that satisfy the *row–column (RC)* constraint [46], as will be later demonstrated in this section.

Example 6: Let $\mathcal{Q}$ be a quasi-group of order 4 with Cayley table

$\otimes$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	1	0
3	3	2	0	1

The Latin square obtained from the Cayley table of $\mathcal{Q}$ is

$$\mathcal{L} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 1 & 0 \\ 3 & 2 & 0 & 1 \end{bmatrix}.$$

The injective map f sends elements of $\mathcal{Q}$ to four permutation matrices:

$$f(0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad f(1) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$f(2) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad f(3) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

B. LDPC Codes as Arrays of Permutation Matrices

It is now straightforward to describe an LDPC code whose parity-check matrix is an array of permutation matrices. Let $\mathcal{W} = [w_{i,j}]_{1\leq i\leq\mu, 1\leq j\leq\eta}$ be a $\mu \times \eta$ matrix over a quasi-group $\mathcal{Q}$, i.e.,

$$\mathcal{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,\eta} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,\eta} \\ \vdots & \vdots & \ddots & \vdots \\ w_{\mu,1} & w_{\mu,2} & \cdots & w_{\mu,\eta} \end{bmatrix}. \tag{1}$$

With some abuse of notation, let $\mathcal{H} = f(\mathcal{W}) = [f(w_{i,j})]$ be an array of permutation matrices obtained by replacing elements of $\mathcal{W}$ with their images under f, i.e.,

$$\mathcal{H} = \begin{bmatrix} f(w_{1,1}) & f(w_{1,2}) & \cdots & f(w_{1,\eta}) \\ f(w_{2,1}) & f(w_{2,2}) & \cdots & f(w_{2,\eta}) \\ \vdots & \vdots & \ddots & \vdots \\ f(w_{\mu,1}) & f(w_{\mu,2}) & \cdots & f(w_{\mu,\eta}) \end{bmatrix}. \tag{2}$$

Then, $\mathcal{H}$ is a binary matrix of size $\mu q \times \eta q$. The null space of $\mathcal{H}$ gives an LDPC code $\mathcal{C}$ of length ηq. The column weight and row weight of $\mathcal{C}$ are $d_v = \mu$ and $d_c = \eta$, respectively.

Remark: Different permutations of rows and columns of the Latin square $\mathcal{L}$ result in different sets of permutation matrices. These sets of permutation matrices result in different permutations of $\mathcal{H}$ in (2). Since permuting rows and columns of $\mathcal{H}$ only leads to the relabeling of the variable nodes and check nodes of the corresponding Tanner graph, different permutations of rows and columns of the Latin square $\mathcal{L}$ result in equivalent codes. Therefore, a code is completely specified by a quasi-group $\mathcal{Q}$ along with a matrix over $\mathcal{Q}$.

C. Deriving Parity-Check Matrices That Satisfy the RC Constraint

The Tanner graph of a code $\mathcal{C}$ constructed as above is free of four cycles if and only if its parity-check matrix satisfies the RC constraint. The RC constraint requires that any two rows (columns) of a parity-check matrix have 1's in at most one common position. The following lemma gives the necessary and sufficient conditions on $\mathcal{W}$ such that the matrix $\mathcal{H}$ (in (2)) satisfies the RC constraint.

Lemma 2: Let w_{i_1,j_1}, w_{i_1,j_2}, w_{i_2,j_1}, and w_{i_2,j_2} be entries of $\mathcal{W}$. Then, $\mathcal{H}$ satisfies the RC constraint iff the equations

$$\begin{aligned} \alpha_1 \otimes \beta_1 &= w_{i_1,j_1}, \quad \alpha_1 \otimes \beta_2 = w_{i_1,j_2} \\ \alpha_2 \otimes \beta_1 &= w_{i_2,j_1}, \quad \alpha_2 \otimes \beta_2 = w_{i_2,j_2} \end{aligned}$$

do not hold simultaneously for any $1 \leq i_1, i_2 \leq \mu$; $1 \leq j_1, j_2 \leq \eta$; $i_1 \neq i_2$; $j_1 \neq j_2$ and any $\alpha_1 \neq \alpha_2$, $\beta_1 \neq \beta_2 \in \mathcal{Q}$.

Proof: We will prove that $\mathcal{H}$ does not satisfy the RC constraint iff the aforementioned equations hold simultaneously for some entries w_{i_1,j_1}, w_{i_1,j_2}, w_{i_2,j_1}, and w_{i_2,j_2} of $\mathcal{W}$ with $i_1 \neq i_2$, $j_1 \neq j_2$ and some $\alpha_1 \neq \alpha_2$, $\beta_1 \neq \beta_2 \in \mathcal{Q}$.

$\mathcal{H}$ does not satisfy the RC constraint if and only if there exist w_{i_1,j_1}, w_{i_1,j_2}, w_{i_2,j_1}, and w_{i_2,j_2} which are entries of $\mathcal{W}$ with $i_1 \neq i_2$, $j_1 \neq j_2$ such that the matrix

$$\mathcal{H}_e = \begin{bmatrix} f(w_{i_1,j_1}) & f(w_{i_1,j_2}) \\ f(w_{i_2,j_1}) & f(w_{i_2,j_2}) \end{bmatrix}$$

does not satisfy the RC constraint.

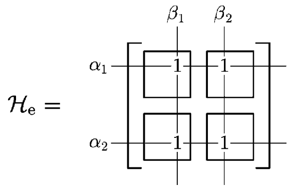

Fig. 9. Row and column indices of the 1 entries in the common positions.

Assume that $\mathcal{H}_e$ does not satisfy the RC constraint and let $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathcal{Q}$ be the row indices and the column indices, respectively, of the 1 entries of $\mathcal{H}_e$ in the common positions as demonstrated in Fig. 9. From the definition of the map f in Section IV-A, one can see that the equations listed in Lemma 2 hold simultaneously.

Conversely, assume that the aforementioned equations hold simultaneously for some w_{i_1,j_1}, w_{i_1,j_2}, w_{i_2,j_1}, and w_{i_2,j_2} with $i_1 \neq i_2$, $j_1 \neq j_2$, and some $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathcal{Q}$. Then, the 1 entries of $\mathcal{H}_e$ include those with positions shown in Fig. 9. Therefore, $\mathcal{H}_e$ does not satisfy the RC constraint. ■

For the following two corollaries of Lemma 2, $\langle \mathrm{GF}(q), +, \cdot \rangle$ is a Galois field, where $q = p^\vartheta$, $\vartheta \in \mathbb{N}\backslash\{0\}$ and p is prime. Let α be a primitive element of $\mathrm{GF}(q)$. The powers of α, $\alpha^{-\infty} \triangleq 0$, $\alpha^0 = 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}$, give all q elements of $\mathrm{GF}(q)$ and $\alpha^{q-1} = 1$.

Corollary 1: Let $\mathcal{Q} = \{1, \alpha, \ldots, \alpha^{q-2}\}$, and let $\otimes$ be the multiplicative operation of $\mathrm{GF}(q)$. The Tanner graph corresponding to $\mathcal{H}$ contains no cycle of length four iff $w_{i_1,j_1} \cdot w_{i_2,j_2} \neq w_{i_1,j_2} \cdot w_{i_2,j_1}$ for any $1 \leq i_1, i_2 \leq \mu$; $1 \leq j_1, j_2 \leq \eta$; $i_1 \neq i_2$; $j_1 \neq j_2$.

Remark: The LDPC codes obtained when $\mathcal{Q}$ is a multiplicative group of a finite field were proposed under a different formulation in [46].

Corollary 2: Let $\mathcal{Q} = \{0, 1, \alpha, \ldots, \alpha^{q-2}\}$ and let $\otimes$ be the subtractive operation of $\mathrm{GF}(q)$. The Tanner graph corresponding to $\mathcal{H}$ contains no cycle of length four iff $w_{i_1,j_1} + w_{i_2,j_2} \neq w_{i_1,j_2} + w_{i_2,j_1}$ for any $1 \leq i_1, i_2 \leq \mu$; $1 \leq j_1, j_2 \leq \eta$; $i_1 \neq i_2$; $j_1 \neq j_2$.

Remark: The LDPC codes obtained when $\mathcal{Q}$ is an additive group of a finite field include array LDPC codes [26].

D. Remarks

For simplicity, all the codes constructed in this paper are derived from the additive group of a finite field, i.e., the codes for which Corollary 2 holds. For any parity-check matrix $\mathcal{H}'$ which is an array of permutation matrices, we can permute the rows and columns to obtain $\mathcal{H}$ such that the topmost and leftmost permutation matrices of $\mathcal{H}$ are identity matrices. The matrix $\mathcal{H}$ is, then, the image of a matrix $\mathcal{W}$ under f, where entries on the first row and first column of $\mathcal{W}$ are $0 \in \mathrm{GF}(q)$. Therefore, in the rest of this paper, we only consider matrices $\mathcal{W}$ of which elements on the first row and on the first column are zeros. We denote $\mathcal{U}$ as the submatrix of $\mathcal{W}$ such that

$$\mathcal{W} = \begin{bmatrix} 0 & 0 \\ 0 & \mathcal{U} \end{bmatrix} \tag{3}$$

and then write $\mathcal{H} = f(\mathcal{W}) = \bar{f}(\mathcal{U})$.

V. Construction of Codes Free of Small Trapping Sets

In this section, we give a general method to construct regular LDPC codes free of a given collection of trapping sets. More precisely, codes are constructed so that their Tanner graphs are free of a given collection of subgraphs from the TSO. It is important to note that our method of constructing codes free of small trapping sets can also be applied to construct random codes, although the complexity required to search for trapping sets would be much higher. We organize our discussion by considering two separate problems: 1) determining a collection of forbidden subgraphs, i.e., which subgraphs that should be avoided in the Tanner graph; and 2) constructing a Tanner graph which is free of a given collection of subgraphs.

A. Determining the Collection of Forbidden Subgraphs

Let us give a general rationale for deciding which trapping sets should be forbidden in the Tanner graph of a code. As previously mentioned, these trapping sets are chosen from the TSO. It is clear that if a predecessor trapping set is not present in a Tanner graph, then neither are its successors. Since the size of a predecessor trapping set is always smaller than the size of its successors, a code should be constructed so that it contains as few small predecessor trapping sets as possible. However, forbidding smaller trapping sets usually imposes stricter constraints on the Tanner graph, resulting in a large rate penalty. This tradeoff between the rate and the choice of forbidden trapping sets is also a tradeoff between the rate and the error-floor performance. While an explicit formulation of this tradeoff is difficult, a good choice of forbidden trapping sets requires the analysis of decoder failures to reveal the *relative harmfulness* of trapping sets. It has been pointed out that for the BSC, the slope of the FER curve in the error-floor region depends on the size of the smallest error patterns uncorrectable by the decoder [2]. We, therefore, introduce the notion of the relative harmfulness of trapping sets in a general setting as follows.

Relative Harmfulness: Assume that under a given decoding algorithm, a code is capable of correcting any error pattern of weight ϑ but fails to correct some error patterns of weight $\vartheta + 1$. If the failures of the decoders on error patterns of weight $\vartheta + 1$ are due to the presence of (a_1, b_1) trapping sets of type $\mathcal{T}_1$, then $\mathcal{T}_1$ is the most harmful trapping set. Let us now assume that a code is constructed so that it does not contain $\mathcal{T}_1$ trapping sets and is capable of correcting any error pattern of weight $\vartheta + 1$. If the presence of (a_2, b_2) trapping sets of type $\mathcal{T}_2$ leads to decoding failure on some error patterns of weight $\vartheta + 2$, then $\mathcal{T}_2$ is the second most harmful trapping set. The relative harmfulness of other trapping sets is determined in this manner.

Remarks: According to the previous discussion, a smaller trapping set might not necessarily be more harmful than a larger one. Besides, for two trapping sets with the same number of variable nodes but with different number of odd-degree check nodes, the one with the smaller number of odd-degree check nodes might not necessarily be more harmful.

Example 7: Let us consider a regular column-weight-three LDPC code of girth 8 on the BSC and assume the Gallager A/B decoding algorithm. Since such a code can correct any error pattern of weight two, we want to find subgraphs whose presence leads to decoding failure on some error pattern of weight three. Since a code cannot correct all weight-three errors if its Tanner graph either contains $(5,3)\{2\}$ trapping sets or contains $(8,0)\{1\}$ trapping sets, the most harmful trapping sets are the $(5,3)\{2\}$ trapping set and the $(8,0)\{1\}$ trapping set.

To further explain the importance of the notion of relative harmfulness, let us slightly detour from our discussion and revisit the notion of erroneous sets. These are sets of variable nodes that are not eventually correct. It is indeed possible, in some cases, to identify some small erroneous sets in a code by simulation, assuming the availability of a fast software/hardware emulator. Unfortunately, erroneous sets identified in this manner generally have little significance for code construction. This is because the dynamics of an iterative decoder (except the Gallager A/B decoder on the BSC) is usually very complex and the mechanism by which the decoder fails into an erroneous sets is difficult to analyze and is not well understood. Usually, the subgraphs induced by the sets of noneventually correct variable nodes are not the most harmful ones. Although avoiding subgraphs induced by sets of noneventually correct variable nodes might lead to a lower error floor, the code rate may be excessively reduced. A better solution is to increase the slope of the FER curve with the fewest possible constraints on the Tanner graph. This can only be done by avoiding the most harmful trapping sets.

For the Gallager A/B algorithm on the BSC, the relative harmfulness of a trapping set is determined by its critical number. Hence, there have been several works in the literature in which the critical numbers of trapping sets are determined and codes are constructed so that the most harmful trapping sets are eliminated. Examples of these works include [2], [29], [47], [48]. Nevertheless, determining the relative harmfulness of trapping sets for other algorithms in general is a difficult problem. The original concept of harmfulness of a trapping set can be found in early works on LDPC codes as well as importance sampling methods to analyze error floors. MacKay and Postol [36] were the first to discover that certain "near codewords" are to be blamed for the high error floor in the Margulis code on the AWGNC. Richardson [1] reproduced their results and developed a computation technique to predict the performance of a given LDPC code in the error-floor domain. He characterized the troublesome noise configurations leading to the error floor using trapping sets and described a technique (of Monte Carlo importance sampling type) to evaluate the error rate associated with a particular class of trapping sets. Cole *et al.* [49] further developed the importance sampling-based method to analyze error floors of moderate-length LDPC codes, while we used instantons to predict error floors [16]–[18].

The main idea of our method is to determine the relative harmfulness of trapping sets from the TSO for the SPA on the BSC. It relies on the topological relationship among these trapping sets and will be presented in Section VII. Before presenting this method, we describe the construction of codes for the Gallager A/B algorithm on the BSC in Section VI.

B. Construction of a Code by Progressively Building the Tanner Graph

We now give an algorithm to progressively construct a (d_v, d_c)-regular LDPC code whose parity-check matrix is an array of permutation matrices. Our construction algorithm is inspired by the PEG algorithm [50] and the method in [27]. Let $\mathcal{C}$ be a (γ,ρ)-regular LDPC code whose parity-check matrix $\mathcal{H} = f(\mathcal{W})$ is an array of permutation matrices. The condition that a Tanner graph is free of a given collection of subgraphs can be understood as a set of constraints imposed on such a Tanner graph. Assume that the Tanner graph G corresponding to $\mathcal{H}$ is required to satisfy a set of constraints. Let τ denote this set of constraints.

The construction is based on a check and select-or-disregard procedure. The Tanner graph of the code is built in ρ stages, where ρ is the row weight of $\mathcal{H}$ (ρ is the number of columns of $\mathcal{W}$). Usually, ρ is not prespecified, and a code is constructed with the goal of making the rate as high as possible. Although for the class of codes under consideration, one can easily obtain some good lower bounds on the rate, determining the maximum possible rate is beyond the scope of this paper.

At each stage, a set of $|\mathcal{Q}|$ new variable nodes are introduced that are initially not connected to the check nodes of the Tanner graph. Blocks of edges are then added to connect the new variable nodes and the check nodes. Each block of edges corresponds to a permutation matrix and hence corresponds to an element of $\mathcal{Q}$. An element of $\mathcal{Q}$ may be chosen randomly, or it may be chosen in a predetermined order. After a block of edges is tentatively added, the Tanner graph is checked for condition τ. If the condition τ is violated, then that block of edges is removed and replaced by a different block. The algorithm proceeds until no block of edges can be added without violating condition τ. Details of the construction are given in Algorithm 1. For mathematical convenience, we append a symbol ψ to the quasi-group $\mathcal{Q}$ and define $f(\psi) = Z$, the all-zero matrix of dimension $|\mathcal{Q}| \times |\mathcal{Q}|$. Also, let Ψ be a matrix of size $\gamma \times 1$, where γ is the column weight of the code to be constructed. All entries of Ψ are set to ψ.

Algorithm 1 Progressively Building the Tanner Graph

```
$\mathcal{W} \leftarrow \gamma \times 1$ all-zero matrix; $\rho \leftarrow 1$
while $w_{\gamma,\rho} \neq \psi$ do
  $\mathcal{W} \leftarrow [\mathcal{W} \quad \Psi]; S \leftarrow \mathcal{Q}\backslash\{0\}; i \leftarrow 1; \rho \leftarrow \rho + 1;$
  while $S \neq \emptyset$ & $i \leq \gamma$ do
    if $i = 1$ then
      $w_{i,\rho} = 0; i \leftarrow i + 1;$
    else
      $w_{i,\rho} \leftarrow \xi \in S; \{\xi$ can be chosen randomly or it can be chosen in a predetermined order.$\}$
      if $f(\mathcal{W})$ satisfies $\tau$ then
        $i \leftarrow i + 1;$
      end if
      $S \leftarrow S\backslash\{\xi\};$
    end if
  end while
end while
$\rho \leftarrow \rho - 1$; Delete the last column of $\mathcal{W}$
```

The complexity of the algorithm grows exponentially with the column weight. The speed of a practical implementation of the algorithm also depends strongly on how the condition τ is checked on a Tanner graph. However, for small column weights, say 3 or 4, and small to moderate code lengths, the algorithm is well handled by state-of-the-art computers. For example, with the searching techniques described in Section III, the construction of a (790, 555) code which has girth 8, minimum distance at least 12 and which does not contain either $(6, 2)$ or $(8, 2)$ trapping sets takes less than 20 seconds on a 2.3 GHz computer.

Remarks:

1) It is worth mentioning that an alternative approach to the previous construction based on a check and select-or-disregard procedure is one in which a subgraph is described by a system of linear equations. Elements of a given matrix $\mathcal{W}$ are particular values of variables of these systems of equations. The Tanner graph corresponding to $f(\mathcal{W})$ contains the given subgraph if and only if elements of $\mathcal{W}$ form a proper solution of at least one of these linear systems of equations. For array LDPC codes, equations governing cycles and several small subgraphs have been derived in [4] and [34]. However, the problem of finding $\mathcal{W}$ such that its elements do not form a proper solution of any of these systems of equations is notoriously difficult. In a recent work [51], LDPC codes free of some absorbing sets were analytically constructed. However, that work only considers a class of regular LDPC codes known as separable, circulant-based codes and a limited number of small absorbing sets.
2) Algorithm 1 can be alternatively described as a process of progressively constructing an incidence structure. The construction begins with an incidence structure consisting of points with no lines. Blocks of parallel lines are then added based on a check and select-or-disregard procedure, similar as in [27] and [52].
3) One can easily come up with several variations of Algorithm 1. An interesting problem is to find more sophisticated algorithms which can result in codes with higher rate.

VI. LDPC Codes for the Gallager A/B Algorithm on the BSC

The error correction capability of regular column-weight-three LDPC codes on the BSC decoded with the Gallager A/B algorithm has been studied in [14], [15], and [47] and can be summarized as follows.

1) A column-weight-three LDPC code with Tanner graph of girth g cannot correct all $g/2$ errors.
2) A column-weight-three LDPC code with Tanner graph of girth $g \geq 10$ corrects all $g/2 - 1$ errors.
3) A column-weight-three LDPC code with Tanner graph of girth $g = 6$ can correct any two errors if and only if the Tanner graph does not contain a codeword of weight four.
4) A column-weight-three LDPC code with Tanner graph of girth $g = 8$ can correct any three errors if and only if a) the Tanner graph does not contain $(5, 3)\{2\}$ trapping sets, and b) the Tanner graph does not contain $(8, 0)\{1\}$ trapping sets.

The aforementioned conditions completely determine the set of constraints τ to be imposed on the Tanner graph of a code

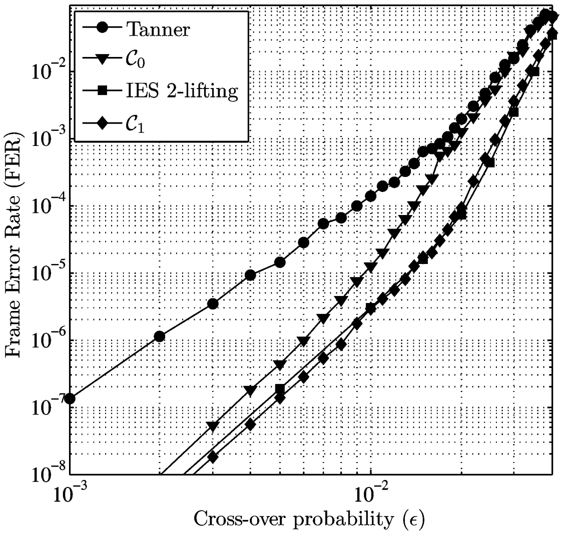

Fig. 10. FER performance of the Tanner code and code $\mathcal{C}_1$ under the Gallager A/B algorithm on the BSC with maximum of 100 iterations.

to achieve a given error-floor performance. The necessary and sufficient conditions to correct three errors were derived in [47]. These conditions require that the Tanner graph of the code has girth $g = 8$ and does not contain $(5, 3)$ and $(8, 0)$ trapping set. It is obvious that the $(5, 3)$ trapping set is indeed the $(5, 3)\{2\}$ trapping set. The $(8, 0)$ trapping set should be understood as the $(8, 0)\{1\}$ trapping set since it can be shown easily that the critical number of the $(8, 0)\{2\}$ trapping set is 4. In the following example, we present the construction of a code which can correct three errors.

Example 8 (Correct All Weight-Three Errors): Let us consider the (155, 64) Tanner code [53]. This code is a (3, 5)-regular LDPC code. Its Tanner graph contains $(5, 3)\{2\}$ trapping sets and hence cannot correct three errors under the Gallager A/B algorithm on the BSC. Let $q = 31$ and α be a primitive element of $\mathrm{GF}(q)$. Let $\mathcal{C}_0$ be an LDPC code defined by the parity-check matrix $\mathcal{H} = \bar{f}(\mathcal{U}_0)$ where

$$\mathcal{U}_0 = \begin{bmatrix} 1 & \alpha^5 & \alpha^{15} & \alpha^{23} \\ \alpha^{16} & \alpha^4 & \alpha^{24} & \alpha^{12} \end{bmatrix}.$$

$\mathcal{C}_0$ is a (155, 64) LDPC code with girth $g = 8$ and minimum distance $d_{\min} = 12$. The Tanner graph of $\mathcal{C}_0$ contains no $(5, 3)\{2\}$ trapping sets. Therefore, $\mathcal{C}_0$ is capable of correcting any three-error pattern under the Gallager A/B algorithm on the BSC. The FER performance of $\mathcal{C}_0$ under the Gallager A/B algorithm for a maximum of 100 iterations is shown in Fig. 10. The FER performance of the Tanner code is also shown for comparison.

In a similar manner, codes that can be guaranteed to correct more errors can be constructed.

We end this section with a discussion on the harmfulness of two trapping sets, say $\mathcal{T}_1$ and $\mathcal{T}_2$, which have the same critical number ϑ. To compare the harmfulness of $\mathcal{T}_1$ and $\mathcal{T}_2$, we use the notion of an inducing set of a trapping set defined as follows.

Definition 6: An inducing set of size ϑ of a trapping set $\mathcal{T}$ is a set of ϑ variable nodes such that if these variable nodes are initially in error then the Gallager A/B decoder will fail on a trapping set $\mathbf{T}$ of type $\mathcal{T}$.

With this definition, the harmfulness of $\mathcal{T}_1$ and $\mathcal{T}_2$ having common critical number ϑ is compared as follows: $\mathcal{T}_1$ is more harmful than $\mathcal{T}_2$ if the number of inducing sets of size ϑ of $\mathcal{T}_1$ is larger than the number of inducing sets of size ϑ of $\mathcal{T}_2$ (see [20] for a more detailed discussion). This concept is demonstrated by the following example.

Example 9: The $(4,4)$, $(6,4)\{1\}$, and $(6,4)\{2\}$ trapping sets all have critical number $\vartheta = 4$. The number of inducing sets of size 4 of the $(4,4)$ and $(6,4)\{2\}$ trapping sets is 1, while for the $(6,4)\{1\}$ trapping set, it is 2. Consequently, the $(6,4)\{1\}$ trapping set is more harmful than the $(4,4)$ trapping set and the $(6,4)\{2\}$ trapping set. We now construct a code which can correct three errors and which is also free of the $(6,4)\{1\}$ trapping set.

Let $q = 53$ and α be a primitive element of $\mathrm{GF}(q)$. Let $\mathcal{C}_1$ be an LDPC code defined by the parity-check matrix $\mathcal{H} = \bar{f}(\mathcal{U}_1)$ where

$$\mathcal{U}_1 = \begin{bmatrix} \alpha^2 & \alpha^{25} & \alpha^{40} & \alpha^{43} \\ \alpha^5 & \alpha^{35} & \alpha^{47} & \alpha^{36} \end{bmatrix}.$$

$\mathcal{C}_1$ is a (265, 108) LDPC code with girth $g = 8$ and minimum distance $d_{\min} \geq 12$. The Tanner graph of $\mathcal{C}_1$ is free of $(5,3)$ and $(6,4)$ trapping sets. The FER performance of $\mathcal{C}_1$ under the Gallager A/B algorithm for a maximum of 100 iterations is shown in Fig. 10. For comparison, we also show the FER performance of a code which is labeled in the plot as "IES 2-lifting." It is a (310, 126) LDPC code which is constructed from two copies of the Tanner code by the internal edge swapping (IES) algorithm [48]. It can be seen that although the length of $\mathcal{C}_1$ is about 15% shorter than the length of the IES 2-lifting code, the slope of the FER curves of both codes is 4 and $\mathcal{C}_1$ has a slightly better performance in the error-floor region. This indicates that the number of weight-four error patterns that the IES 2-lifting code fails to correct is larger than the number of weight-four error patterns that $\mathcal{C}_1$ fails to correct.

VII. LDPC Codes for the SPA on the BSC

In this section, we present the construction of regular column-weight-three codes for the SPA on the BSC. The main element of the construction is the determination of the set of most harmful trapping sets. Following the discussion of the notion of relative harmfulness in Section V-A, we approach this problem as follows.

Let us consider an LDPC code $\mathcal{C}$ and assume that $\mathcal{C}$ can correct any error pattern of weight ϑ under the SPA on the BSC. We are interested in determining the trapping sets whose presence leads to decoding failure on error patterns of weight $\vartheta + 1$. To simplify this problem, we only focus on initial error patterns of weight $\vartheta+1$ that surely lead to decoding failures of the Gallager A/B algorithms on the BSC. The basis for this simplification is as follows. Since it is well known that the SPA algorithm has a superior performance in both the waterfall and the error-floor regions compared to that of the Gallager A/B algorithm, we surmise that an error pattern correctable by the Gallager A/B algorithm is correctable with high probability by the SPA algorithm, although this fact remains unproven. The initial error patterns of weight $\vartheta + 1$ that are surely uncorrectable by the Gallager A/B algorithm can be easily derived from the TSO.

Assume the transmission of the all-zero codeword and let $\mathbf{y}$ be the received vector input to the decoder. Also, assume that $\mathrm{supp}(\mathbf{y}) = \mathbf{T}_1$, a trapping set of type $\mathcal{T}_1$ from the TSO with $\vartheta+1$ variable nodes. In other words, all the $\vartheta+1$ initially corrupt variable nodes belong to the trapping set $\mathbf{T}_1$. This error pattern results in a decoding failure of the Gallager A/B algorithm and hence is an initial error pattern of interest. As the decoder operates by passing messages along the edges of the Tanner graph, the decoding outcome depends heavily on the immediate neighborhood of the subgraph induced by variable nodes in $\mathbf{T}_1$. In many cases, a decoding failure of the SPA will only occur if $\mathbf{T}_1$ generates a trapping set $\mathbf{T}_2$ of type $\mathcal{T}_2$, where $\mathcal{T}_2$ is a successor of $\mathcal{T}_1$. In such cases, the presence of $\mathbf{T}_2$ in a code makes it incapable of correcting any error pattern of size $\vartheta + 1$, and hence, $\mathbf{T}_2$ is a harmful trapping set.

To evaluate the harmfulness of the $\mathcal{T}_2$ trapping set, all initial error patterns that consist of variable nodes of a $\mathcal{T}_1$ trapping set must be considered. Let $\mathscr{T}$ be the set of all trapping sets $\mathbf{T}$ of type $\mathcal{T}_1$. Partition $\mathscr{T}$ into two disjoint sets $\mathscr{T}_\mathrm{a}$ and $\mathscr{T}_\mathrm{b}$ such that a trapping set $\mathbf{T}$ in $\mathscr{T}_\mathrm{a}$ generates at least one $\mathcal{T}_2$ trapping set while a trapping set $\mathbf{T}$ in $\mathscr{T}_\mathrm{b}$ does not generate any $\mathcal{T}_2$ trapping set. For each trapping set $\mathbf{T}_i \in \mathscr{T}$, perform decoding on the input vector $\mathbf{y}_i$ where $\mathrm{supp}(\mathbf{y}_i) = \mathbf{T}_i$, at a cross-over probability ϵ of the channel. Let $\mathscr{T}_\mathrm{s}$ be the set of trapping sets $\mathbf{T}_i \in \mathscr{T}$ such that decoding is successful upon error pattern $\mathbf{y}_i$. Define $\chi_\mathrm{a}(\epsilon)$ and $\chi_\mathrm{b}(\epsilon)$ to be the rate of successful decoding for trapping sets in $\mathscr{T}_\mathrm{a}$ and $\mathscr{T}_\mathrm{b}$ at the cross-over probability ϵ of the channel as follows:

$$\chi_\mathrm{a}(\epsilon) = \frac{|\mathscr{T}_\mathrm{a} \cap \mathscr{T}_\mathrm{s}|}{|\mathscr{T}_\mathrm{a}|} \tag{4}$$

$$\chi_\mathrm{b}(\epsilon) = \frac{|\mathscr{T}_\mathrm{b} \cap \mathscr{T}_\mathrm{s}|}{|\mathscr{T}_\mathrm{b}|} \tag{5}$$

The harmfulness of $\mathcal{T}_2$ trapping sets of $\mathcal{C}$ is evaluated by comparing $\chi_\mathrm{a}(\epsilon)$ and $\chi_\mathrm{b}(\epsilon)$ for a wide range of ϵ. The larger the difference $\chi_\mathrm{b}(\epsilon) - \chi_\mathrm{a}(\epsilon)$, the more harmful $\mathcal{T}_2$ trapping sets are. The harmfulness of $\mathcal{T}_2$ trapping sets is also compared with the harmfulness of other successor trapping sets of $\mathcal{T}_1$, which is determined in the same fashion.

We note that this characterization of relative harmfulness, although heuristic, plays a critical role in the construction of good high-rate codes as no explicit quantification of harmfulness of trapping sets is known. This characterization of harmfulness also helps a code designer to determine more or less the exact subgraphs that are responsible for a certain type of decoding failure. It is, therefore, superior to searching for trapping sets by simulation.

We continue our discussion with three case studies in which we evaluate 1) the relative harmfulness of the $(6,2)\{1\}$ and $(8,2)$ trapping sets, 2) the relative harmfulness of the $(5,3)\{2\}$ trapping set, and 3) the relative harmfulness of the $(7,3)$, $(9,3)$, and $(10,2)$ trapping sets. For a better illustration of the relationship among these trapping sets, a hierarchy of trapping sets of interest originating from the $(4,4)$ trapping set is shown in

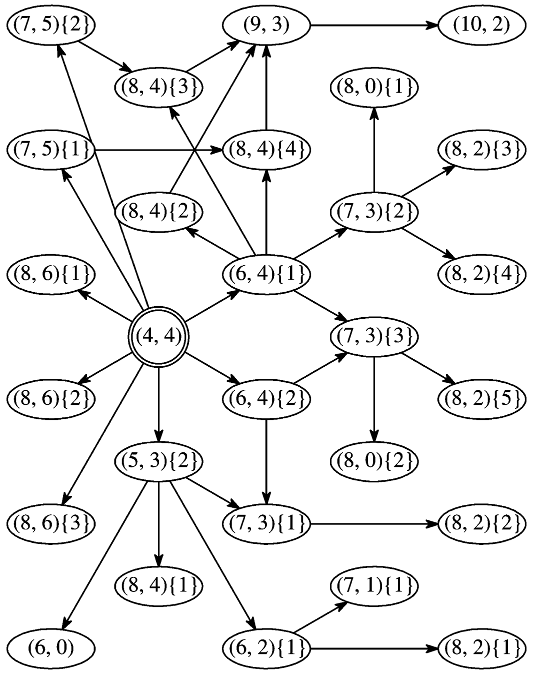

Fig. 11. Hierarchy of trapping sets of interest originating from the $(4,4)$ trapping set for regular column-weight-three codes of girth 8.

Fig. 11. For the first case, we present a detailed analysis. For the other two cases, we only give the results of the analysis. The analysis to be presented is a step toward the guaranteed correction of four, five, and six errors under the SPA on the BSC. For simplicity, we assume that codes have girth $g = 8$ in all examples, although the method of construction can be applied to girth 6 codes which would likely result in higher rate codes.

A. Harmfulness of the $(6,2)\{1\}$ and $(8,2)$ Trapping Sets

Since we consider codes with girth $g = 8$, let us mention an existing code of such girth. Consider the (530, 373) integer lattice code (or shortened array code [4]) given in [27]. This code has minimum distance $d_{\min} = 8$ and, hence, is unable to correct all weight-four error patterns. Clearly, the first step toward the guaranteed correction of four errors is to eliminate the $(6,0)\{1\}$, $(8,0)\{1\}$, and $(8,0)\{2\}$ trapping sets, which are the low-weight codewords. We, therefore, construct a code with minimum distance $d_{\min} \geq 10$. Let $q = 53$ and let α be a primitive element of $\mathrm{GF}(q)$ and let τ specify that the Tanner graph of a code has girth $g = 8$ and contains no $(6,0)\{1\}$, $(8,0)\{1\}$ and $(8,0)\{2\}$ trapping sets. Using the method of construction described in Section V-B, we obtain a regular column-weight-three code $\mathcal{C}_2$ with parity-check matrix $\mathcal{H}_2 = \bar{f}(\mathcal{U}_2)$ where

$$\mathcal{U}_2 = \begin{bmatrix} 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^7 & \alpha^{11} & \alpha^{12} & \alpha^{14} & \alpha^{27} \\ \alpha & \alpha^3 & \alpha^5 & \alpha^8 & \alpha^{10} & \alpha^{13} & \alpha^9 & \alpha^{38} & \alpha^{51} \end{bmatrix}.$$

$\mathcal{C}_2$ is a (530, 373) code. Similar to the aforementioned integer lattice code, $\mathcal{C}_2$ has column weight 3, row weight 10 and rate $R = 0.7$.

The Tanner graph of $\mathcal{C}_2$ contains 17 066 $(4,4)$ trapping sets. We partition the collection of $(4,4)$ trapping sets into nine disjoint sets $\mathscr{T}_{\mathrm{a}}, \mathscr{T}_{\mathrm{b}}, \ldots, \mathscr{T}_{\mathrm{i}}$ based on whether a $(4,4)$ trapping set generates $(5,3)\{2\}$, $(6,2)\{1\}$, $(8,2)$ or $(10,0)$ trapping sets. Note that, for simplicity, we do not differentiate among different $(8,2)$ and $(10,0)$ trapping sets in this analysis, although a more detailed treatment may reveal some differences in the harmfulness of those trapping sets. The classification and sizes of different sets of $(4,4)$ trapping sets are shown in Table I.

TABLE I
DISJOINT SETS OF $(4,4)$ TRAPPING SETS IN THE LDPC CODE $\mathcal{C}_2$. A ✓ INDICATES THAT THE $(4,4)$ TRAPPING SETS IN $\mathscr{T}_*$ GENERATE AT LEAST ONE CORRESPONDING TRAPPING SET

Sets $\mathscr{T}_*$	Trapping Sets Generated by $\mathscr{T}_*$				Total
	$(5,3)\{2\}$	$(6,2)\{1\}$	$(8,2)$	$(10,0)$	
$\mathscr{T}_{\mathrm{a}}$					4982
$\mathscr{T}_{\mathrm{b}}$				✓	53
$\mathscr{T}_{\mathrm{c}}$	✓				424
$\mathscr{T}_{\mathrm{d}}$			✓		7314
$\mathscr{T}_{\mathrm{e}}$			✓	✓	371
$\mathscr{T}_{\mathrm{f}}$	✓		✓		1855
$\mathscr{T}_{\mathrm{g}}$	✓		✓	✓	106
$\mathscr{T}_{\mathrm{h}}$	✓	✓			1007
$\mathscr{T}_{\mathrm{i}}$	✓	✓	✓		954
Total					**17066**

To evaluate the harmfulness of the $(5,3)\{2\}$, $(6,2)\{1\}$, $(8,2)$, and $(10,0)$ trapping sets, we perform decoding on all input vectors $\mathbf{y}_i$ where $\mathrm{supp}(\mathbf{y}_i) = \mathbf{T}_i$, a $(4,4)$ trapping set of $\mathcal{C}_2$. The result is as follows. For the trapping sets in $\mathscr{T}_{\mathrm{a}}$, $\mathscr{T}_{\mathrm{b}}$, $\mathscr{T}_{\mathrm{c}}$, and $\mathscr{T}_{\mathrm{g}}$, the decoder successfully decodes all input vectors $\mathbf{y}_i$ at all 250 values of ϵ that have been considered, i.e., $\chi_{\mathrm{a}}(\epsilon) = \chi_{\mathrm{b}}(\epsilon) = \chi_{\mathrm{c}}(\epsilon) = \chi_{\mathrm{g}}(\epsilon) = 100\%\ \forall\epsilon$. For the trapping sets in $\mathscr{T}_{\mathrm{d}}$, $\mathscr{T}_{\mathrm{e}}$, $\mathscr{T}_{\mathrm{f}}$, $\mathscr{T}_{\mathrm{h}}$, and $\mathscr{T}_{\mathrm{i}}$, the rate of successful decoding is shown in the form of a histogram in Fig. 12. As an example of how to interpret the result, consider the trapping sets in $\mathscr{T}_{\mathrm{f}}$. It can be seen that there are about 160 values (65% of all tested values) of ϵ at which decoding is successful for all input vectors $\mathbf{y}_i$. For about 90 values (30% of all tested values) of ϵ, decoding is successful for approximately nine out of ten input vectors $\mathbf{y}_i$.

The following facts can be observed.

1) The $(4,4)$ trapping sets in $\mathscr{T}_a$, $\mathscr{T}_{\mathrm{b}}$, and $\mathscr{T}_{\mathrm{c}}$ do not generate either $(6,2)\{1\}$ or $(8,2)$ trapping sets. The rate of successful decoding is 100% for all tested values of ϵ.
2) The $(4,4)$ trapping sets in $\mathscr{T}_{\mathrm{d}}$, $\mathscr{T}_{\mathrm{e}}$, and $\mathscr{T}_{\mathrm{f}}$ generate at least one $(8,2)$trapping set. Decoding is not always successful, but the rate of successful decoding is more than 90% for all tested values of ϵ.
3) The $(4,4)$ trapping sets in $\mathscr{T}_{\mathrm{h}}$ generate at least one $(6,2)\{1\}$ trapping set. The rate of successful decoding $\chi_{\mathrm{h}}(\epsilon)$ is significantly lower in general compared to $\chi_{\mathrm{d}}(\epsilon)$, $\chi_{\mathrm{e}}(\epsilon)$, and $\chi_{\mathrm{f}}(\epsilon)$.
4) The $(4,4)$ trapping sets in $\mathscr{T}_{\mathrm{i}}$ generate at least one $(6,2)\{1\}$ and one $(8,2)$ trapping set. The rate of successful decoding $\chi_{\mathrm{i}}(\epsilon)$ is lowest in general.
5) The $(4,4)$ trapping sets in $\mathscr{T}_{\mathrm{f}}$ generate at least one $(5,3)\{2\}$ trapping set, while the ones in $\mathscr{T}_d$ do not. In general, $\chi_{\mathrm{f}}(\epsilon) < \chi_{\mathrm{d}}(\epsilon)$.
6) The $(4,4)$ trapping sets in $\mathscr{T}_{\mathrm{e}}$ generate at least one $(10,0)$ trapping set while the ones in $\mathscr{T}_{\mathrm{d}}$ do not. In general, $\chi_{\mathrm{e}}(\epsilon) > \chi_{\mathrm{d}}(\epsilon)$.

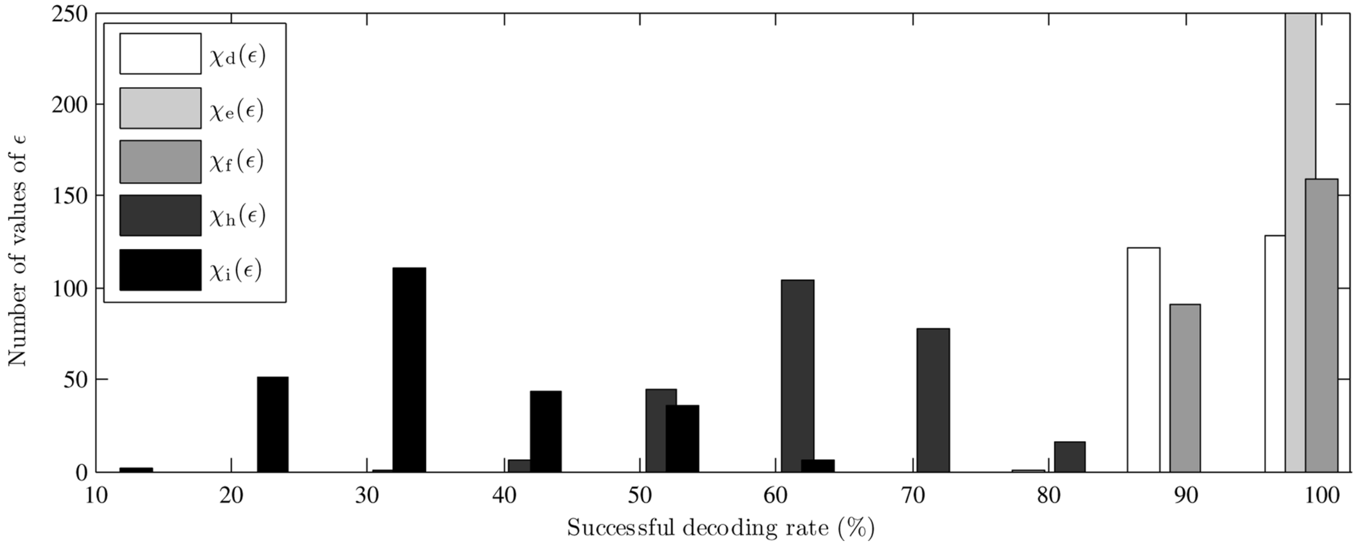

Fig. 12. Rate of successful decoding for different sets of $(4,4)$ trapping sets in code $\mathcal{C}_2$.

7) The $(4,4)$ trapping sets in $\mathscr{T}_g$ generate at least one $(10,0)$ trapping set, while the ones in $\mathscr{T}_f$ do not. $\chi_g(\epsilon) = 100\%$ for all tested values of ϵ.

The aforementioned observations strongly suggest that both $(6,2)\{1\}$ and $(8,2)$ trapping sets are harmful. However, the harmfulness of the $(6,2)\{1\}$ trapping set is much more evident than the harmfulness of the $(8,2)$ trapping set. Besides, it is interesting to notice that $\chi_g(\epsilon) = 100\%$ for all tested values of ϵ. All $(4,4)$ trapping sets in $\mathscr{T}_g$ generate at least one $(8,2)$ trapping set, one $(5,3)\{2\}$ trapping set, and one $(10,0)$ trapping set. In this case, the presence of $(10,0)$ trapping sets seem to increase the rate of successful decoding. This "positive" effect of $(10,0)$ trapping sets can also be seen when comparing $\chi_e(\epsilon)$ and $\chi_d(\epsilon)$. Finally, by comparing $\chi_f(\epsilon)$ and $\chi_d(\epsilon)$, it is suggestive that the $(5,3)\{2\}$ trapping sets have some negative effect on decoding if the $(4,4)$ trapping sets generate $(8,2)$ and $(10,0)$ trapping sets.

To further verify our prediction on the harmfulness of the $(6,2)\{1\}$ and $(8,2)$ trapping sets, we construct another code with the same parameters as those of $\mathcal{C}_2$. We denote this code by $\mathcal{C}_3$. The Tanner graph of $\mathcal{C}_3$ has stronger constraints than the Tanner graph of $\mathcal{C}_2$ as we impose that it has neither $(6,2)\{1\}$ nor $(10,0)$ trapping sets. Since $(10,0)$ trapping sets are not present, $\mathcal{C}_3$ has minimum distance of at least 12.

Let $\mathcal{C}_3$ be defined by the parity-check matrix $\mathcal{H}_3 = \bar{f}(\mathcal{U}_3)$ where

$$\mathcal{U}_3 = \begin{bmatrix} 1 & \alpha^2 & \alpha^4 & \alpha^{15} & \alpha^{17} & \alpha^{26} & \alpha^{31} & \alpha^{33} & \alpha^{36} \\ \alpha^{30} & \alpha^{16} & \alpha & \alpha^{19} & \alpha^7 & \alpha^{34} & \alpha^3 & \alpha^8 & \alpha^{22} \end{bmatrix}.$$

The Tanner graph of $\mathcal{C}_3$ contains 16483 $(4,4)$ trapping sets, which can be partitioned into four disjoint sets as shown in Table II.

We again perform decoding on all input vectors $\mathbf{y}_i$ where $\text{supp}(\mathbf{y}_i) = \mathbf{T}_i$, a $(4,4)$ trapping set of $\mathcal{C}_3$. The rate of successful decoding for trapping sets in $\mathscr{T}_c$ and $\mathscr{T}_d$ is shown in the form of a histogram in Fig. 13. For trapping sets in $\mathscr{T}_a$ and $\mathscr{T}_b$, decoding is always successful.

It can be seen that the results are consistent with the previously obtained results. Decoding is always successful for the $(4,4)$ trapping sets which generate neither $(6,2)\{1\}$ nor $(8,2)$ trapping sets. Besides, $\chi_c(\epsilon) > \chi_d(\epsilon)$, in general, since the $(4,4)$ trapping sets in $\mathscr{T}_c$ do not generate $(5,3)\{2\}$ trapping sets. These results validate our prediction on the harmfulness of successors of the $(4,4)$ trapping set. We have repeated the experiment for a collection of codes whose Tanner graphs do not contain either $(6,2)\{1\}$ or $(8,2)$ trapping sets. The consistency of the results led us to the following conjecture.

TABLE II
TYPES OF $(4,4)$ TRAPPING SETS IN THE (530, 373) LDPC CODE $\mathcal{C}_3$

Sets $\mathscr{T}_*$	Trapping Sets Generated by $\mathscr{T}_*$		Total
	$(5,3)\{2\}$	$(8,2)$	
$\mathscr{T}_a$			6890
$\mathscr{T}_b$	✓		795
$\mathscr{T}_c$		✓	6890
$\mathscr{T}_d$	✓	✓	1908
Total			**16483**

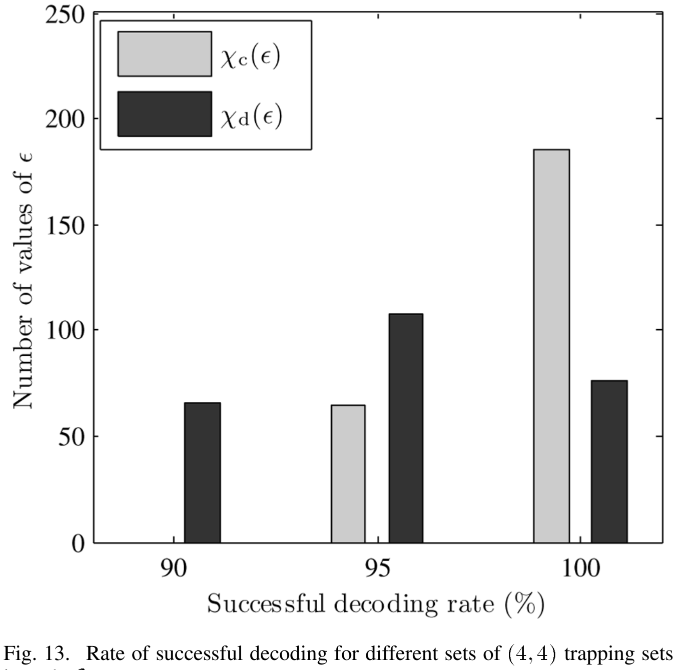

Fig. 13. Rate of successful decoding for different sets of $(4,4)$ trapping sets in code $\mathcal{C}_3$.

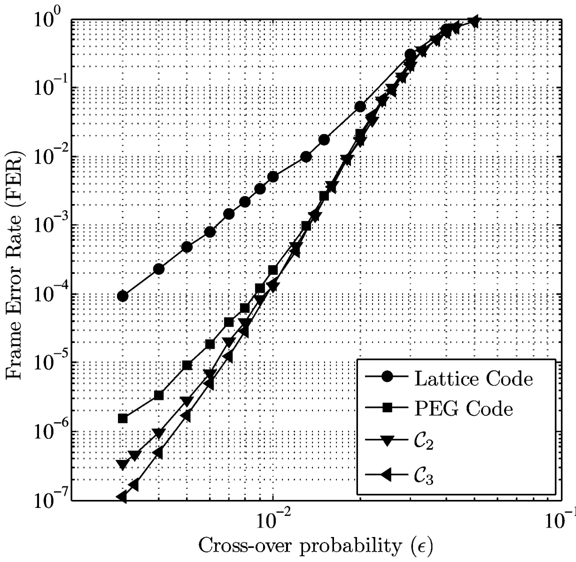

Fig. 14. FER performance of codes in Example 10 under the SPA on the BSC.

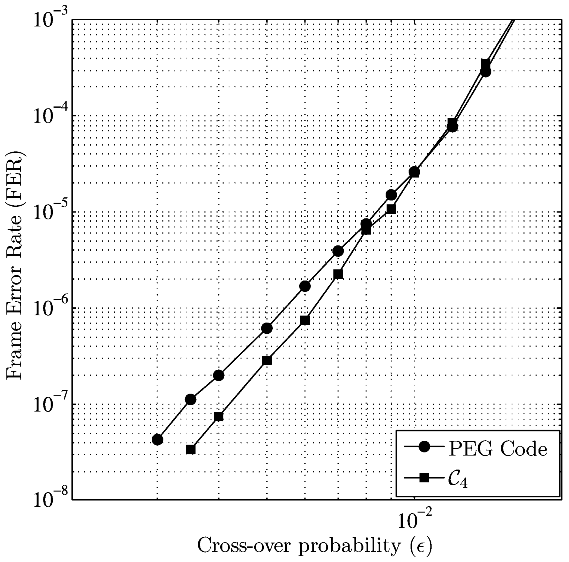

Fig. 15. FER performance of codes in Example 11 under the SPA on the BSC.

Conjecture 1: A regular column-weight-three code of girth $g = 8$ can correct any error pattern of weight 4 consisting of variable nodes of an eight-cycle under the SPA on the BSC if its Tanner graph contains neither $(6, 2)$ nor $(8, 2)$ trapping sets.

We remark that this conjecture only gives a sufficient condition. A code may correct any error pattern of weight 4 even if its Tanner graph contains $(8, 2)\{2\}$ trapping sets. For example, consider the Tanner code of length 155. The Tanner graph of this code does not contain a $(6, 2)\{1\}$ trapping set, but it contains $(8, 2)\{2\}$ trapping sets. However, decoding is always successful for all the $(4, 4)$ trapping sets at any value of ϵ. It might be possible to find a better sufficient condition by taking into account larger trapping sets, but such an analysis appears to be difficult.

Example 10: The FER performance of $\mathcal{C}_2$, $\mathcal{C}_3$, and the (530, 373) integer lattice code under the SPA with 100 iterations on the BSC is shown in Fig. 14. For comparison, Fig. 14 also shows the FER performance of a (530, 373) LDPC code constructed using the PEG algorithm [50]. This PEG code has girth $g = 6$ and minimum distance $d_{\min} = 6$. Clearly, $\mathcal{C}_3$, whose Tanner graph is free of $(6, 2)$ trapping sets, has the best performance. Although the Tanner graph of $\mathcal{C}_2$ contains some $(8, 2)$ trapping sets, it still outperforms the PEG code. The integer lattice code has the worst performance although it has girth $g = 8$.

Example 11: Let $q = 3^4$ and let $\mathcal{C}_4$ be defined by the parity-check matrix $\mathcal{H}_4 = \bar{f}(\mathcal{U}_5)$ where

$$\mathcal{U}_4 = \begin{bmatrix} \alpha^{2} & \alpha^{6} & \alpha^{9} & \alpha^{31} & \alpha^{33} & \alpha^{39} & \alpha^{57} & \alpha^{60} & \alpha^{67} \\ \alpha^{55} & \alpha^{12} & \alpha^{28} & \alpha^{46} & \alpha^{78} & \alpha^{37} & \alpha^{61} & \alpha^{76} & \alpha^{44} \end{bmatrix}.$$

$\mathcal{C}_4$ is a (810, 569) code with column weight 3, row weight 10, and rate $R = 0.7$. The Tanner graph of $\mathcal{C}_4$ has girth $g = 8$ and does not contain either $(6, 2)$ or $(8, 2)$ trapping sets. The FER performance of $\mathcal{C}_4$ under the SPA with 100 iterations on the BSC is shown in Fig. 15. For comparison, Fig. 15 also shows the FER performance of a (810, 567) PEG constructed code. This code has girth $g = 8$. It can be seen that $\mathcal{C}_4$ has a lower floor than the PEG code.

B. Harmfulness of the $(5, 3)\{2\}$ Trapping Set

Assuming the guaranteed correction of four errors, we are now interested in finding trapping sets whose presence leads to decoding failure on some error patterns of weight five. There are two trapping sets with five variable nodes from the TSO that can be present in the Tanner graph of a regular column-weight-three LDPC code with girth $g = 8$: the $(5, 3)\{2\}$ trapping set and the $(5, 5)$ trapping set. The result of our analysis indicates that $(5, 3)\{2\}$ trapping sets are the most harmful and should be forbidden in the Tanner graph of a code.

Example 12: Let $q = 211$ and let $\mathcal{C}_5$ be defined by the parity-check matrix $\mathcal{H}_5 = \bar{f}(\mathcal{U}_5)$ where $\mathcal{U}_5$ is given in (6), as shown at the bottom of the page. $\mathcal{C}_5$ is a (3165, 2554) code with column weight 3, row weight 15, and rate $R = 0.8$. The Tanner graph of $\mathcal{C}_5$ has girth $g = 8$ and does not contain either $(5, 3)\{2\}$ or $(8, 2)$ trapping sets. The FER performance of $\mathcal{C}_5$ under the SPA with 100 iterations on the BSC is shown in

$$\mathcal{U}_5 = \begin{bmatrix} \alpha^{8} & \alpha^{23} & \alpha^{47} & \alpha^{54} & \alpha^{55} & \alpha^{67} & \alpha^{78} & \alpha^{90} & \alpha^{108} & \alpha^{169} & \alpha^{177} & \alpha^{187} & \alpha^{192} & \alpha^{193} \\ \alpha^{61} & \alpha^{189} & \alpha^{190} & \alpha^{171} & \alpha^{170} & \alpha^{132} & \alpha^{182} & \alpha^{128} & \alpha^{71} & \alpha^{117} & \alpha^{129} & \alpha^{10} & \alpha^{160} & \alpha^{64} \end{bmatrix} \quad (6)$$

$$\mathcal{U}_6 = \begin{bmatrix} \alpha^{12} & \alpha^{15} & \alpha^{28} & \alpha^{34} & \alpha^{37} & \alpha^{57} & \alpha^{75} & \alpha^{85} & \alpha^{111} & \alpha^{157} & \alpha^{166} \\ \alpha^{163} & \alpha^{175} & \alpha^{60} & \alpha^{25} & \alpha^{118} & \alpha^{167} & \alpha^{156} & \alpha^{142} & \alpha^{30} & \alpha^{155} & \alpha^{31} \end{bmatrix} \quad (7)$$

$$\mathcal{U}_7 = \begin{bmatrix} \alpha^{18} & \alpha^{88} & \alpha^{112} & \alpha^{142} & \alpha^{157} & \alpha^{186} & \alpha^{196} & \alpha^{197} & \alpha^{228} & \alpha^{246} & \alpha^{288} & \alpha^{316} \\ \alpha^{25} & \alpha^{73} & \alpha^{155} & \alpha^{287} & \alpha^{328} & \alpha^{151} & \alpha^{75} & \alpha^{324} & \alpha^{148} & \alpha^{248} & \alpha^{62} & \alpha^{70} \end{bmatrix} \quad (8)$$

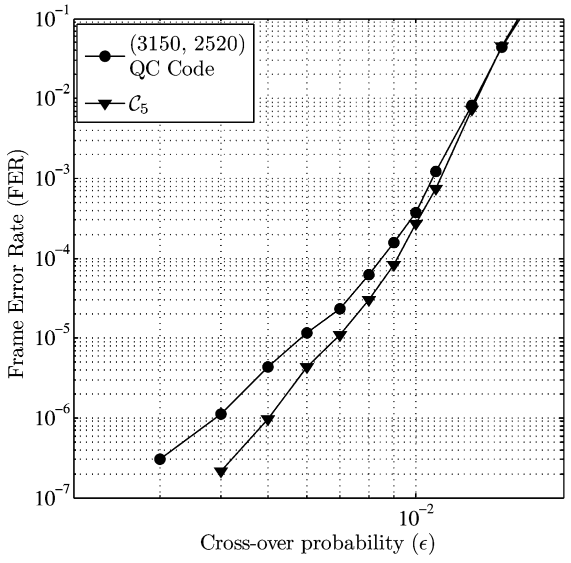

Fig. 16. FER performance of codes in Example 12 under the SPA on the BSC.

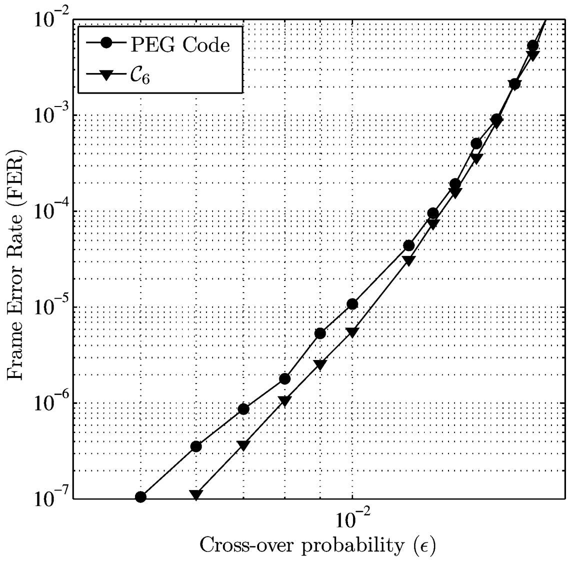

Fig. 17. FER performance of codes in Example 13 under the SPA on the BSC.

Fig. 16. For comparison, Fig. 16 also shows the FER performance of a (3150, 2520) regular QC LDPC code[4] constructed using array masking proposed in [46]. The parity-check matrix of this code is a 10×50 array of 63×63 circulants or zero matrices, which has column weight 3 and row weight 15. This code has girth $g = 8$. It can be seen that $\mathcal{C}_4$ has a lower error floor than the code constructed using array masking.

C. Harmfulness of $(7,3)$, $(9,3)$, *and* $(10,2)$ *Trapping Sets*

With the previous results, we now consider codes free of $(5,3)\{2\}$ and $(8,2)$ trapping sets and aim for the guaranteed correction of six errors. To guarantee the correction of six errors, codes must have minimum distances $d_{\min} \geq 14$. In other words, their Tanner graphs should be free of $(a,0)$ trapping sets $\forall a \leq 12$. Similar to the previous discussions, we analyze error patterns of weight six, focusing on those consisting of variable nodes of a trapping set. There are three trapping sets of size 6 from the TSO that can be present in the Tanner graph of a regular column-weight-three LDPC code with girth $g = 8$: the $(6,6)$ trapping set, $(6,4)\{1\}$ trapping set and the $(6,4)\{2\}$ trapping set. The results of our analysis and experiments suggest that the following trapping sets are harmful (in decreasing order of harmfulness):

1) The $(7,3)\{2\}$ trapping set and the $(7,3)\{3\}$ trapping set.
2) The $(10,2)$ trapping sets which are successors of the $(9,3)$ trapping sets below.
3) The $(9,3)$ trapping sets which are successors of the $(8,4)\{2\}$, $(8,4)\{3\}$, and $(8,4)\{4\}$ trapping sets (see Fig. 11 for an illustration of the relationship among these trapping sets).

Example 13: Let $q = 199$ and let $\mathcal{C}_6$ be defined by the parity-check matrix $\mathcal{H}_6 = \bar{f}(\mathcal{U}_6)$ where $\mathcal{U}_6$ is given in (7). $\mathcal{C}_6$ is a (2388, 1793) code with column weight 3, row weight 12, and rate $R = 0.75$. The Tanner graph of $\mathcal{C}_6$ has girth $g = 8$ and contains neither $(5,3)\{2\}$ trapping sets nor $(7,3)$ trapping sets, nor

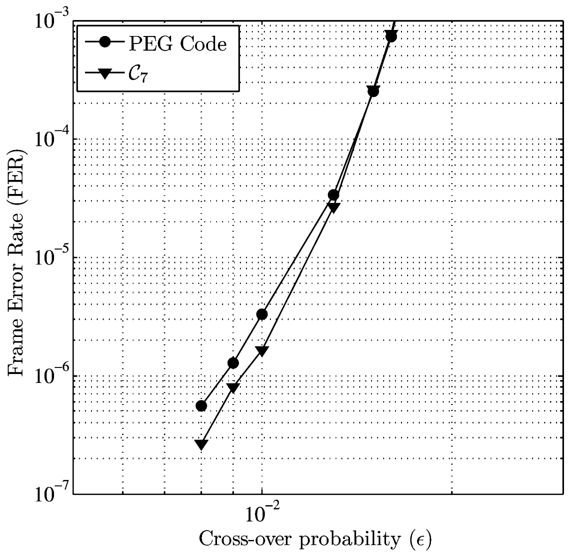

Fig. 18. FER performance of codes in Example 14 under the SPA on the BSC.

the $(10,2)$ trapping sets that are generated by either $(8,4)\{2\}$, $(8,4)\{3\}$, or $(8,4)\{4\}$ trapping sets. The FER performance of $\mathcal{C}_6$ under the SPA with 100 iterations on the BSC is shown in Fig. 17. For comparison, Fig. 17 also shows the FER performance of a (3150, 2518) PEG code. This code has girth $g = 8$. It can be seen that $\mathcal{C}_6$ has a lower error floor than the PEG code.

Example 14: Let $q = 337$ and let $\mathcal{C}_7$ be defined by a parity-check matrix $\mathcal{H}_7 = \bar{f}(\mathcal{U}_7)$ where $\mathcal{U}_7$ is shown in (8). $\mathcal{C}_7$ is a (4381, 3372) code with column weight 3, row weight 13, and rate $R = 0.77$. The Tanner graph of $\mathcal{C}_7$ has girth $g = 8$ and does contain either $(5,3)\{2\}$ or $(7,3)$ trapping sets and neither does it contain $(9,3)$ trapping sets that are generated by either $(8,4)\{2\}$, $(8,4)\{3\}$ or $(8,4)\{4\}$ trapping sets. The FER performance of $\mathcal{C}_7$ under the SPA with 100 iterations on the BSC is shown in Fig. 18. For comparison, Fig. 18 also shows the FER performance of a (4381, 3370) PEG code. This code has girth $g = 8$. It can be seen that $\mathcal{C}_7$ has a lower error floor than that of the PEG code.

[4]A QC code is a structured code whose parity-check matrix can be represented as an array of circulants.

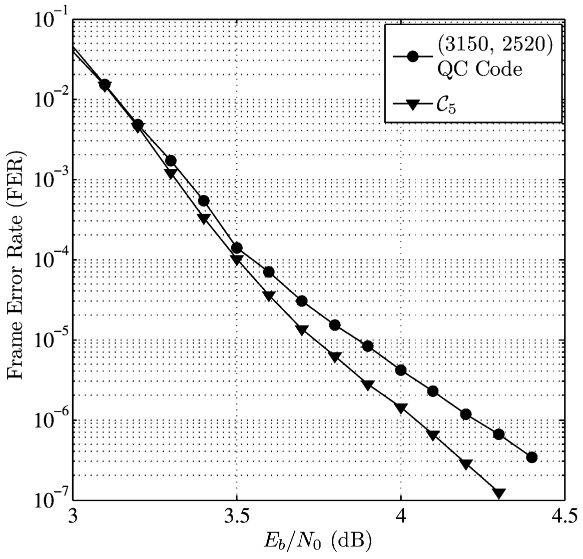

Fig. 19. FER performance of codes in Example 12 under the SPA on the AWGNC.

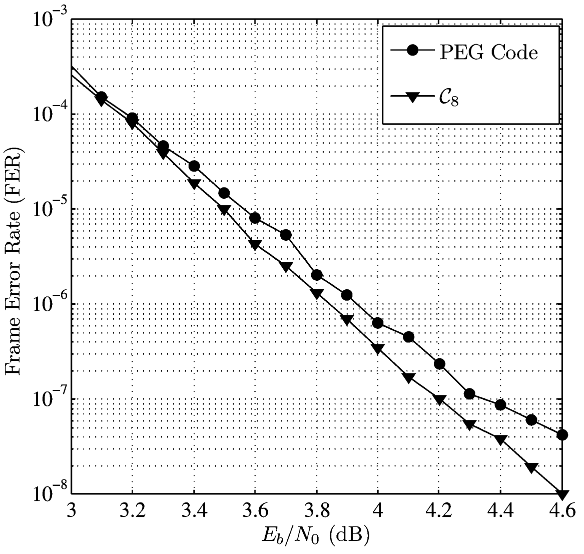

Fig. 20. FER performance of a rate-0.5 code under the SPA on the AWGNC.

VIII. Discussion

Although the codes presented in this paper are optimized for the BSC, they also have excellent performance on the AWGNC. As a demonstration, we show the FER performance of the code $\mathcal{C}_5$ from Example 12 under the SPA on the AWGNC in Fig. 19. Recall that the Tanner graph of $\mathcal{C}_5$ has girth $g = 8$ and does not contain $(5,3)\{2\}$ and $(8,2)$ trapping sets and that the (3150, 2520) regular QC LDPC code was constructed using array masking as proposed in [46]. It can be seen that although $\mathcal{C}_5$ was constructed for the BSC, it outperforms the other code, which was constructed for the AWGNC.

It can be noticed that in most of the examples in this paper, we compare the performance of the newly constructed codes with PEG codes. It is worth mentioning that many improved versions of the PEG construction exist in the literature. However, many of these works deal with irregular codes. In other works, codes with rate $R = 0.5$ and block length $n = 504$ (or $n = 1008$) were usually constructed and their simulated performance was shown in comparison with that of the PEG codes or the MacKay random codes of the same length. We remark that codes of this rate and length obtained from our construction can be made free of many harmful trapping sets. The following code serves as an example.

Let $q = 83$ and let α be a primitive element of $\mathrm{GF}(q)$. Also, let τ specify that the Tanner graph of a code has girth $g = 8$ and contains no $(5,3)\{2\}$, $(6,4)\{1\}$, $(7,3)$, and $(8,4)$ trapping sets. Using the method of construction described in Section V-B, we obtain a regular column-weight-three code $\mathcal{C}_8$ with parity-check matrix $\mathcal{H}_8 = \bar{f}(\mathcal{U}_8)$ where

$$\mathcal{U}_8 = \begin{bmatrix} \alpha^{13} & \alpha^{20} & \alpha^{33} & \alpha^{34} & \alpha^{66} \\ \alpha^{46} & \alpha^{27} & \alpha^{48} & \alpha^{59} & \alpha^{63} \end{bmatrix}.$$

$\mathcal{C}_8$ is a (498, 247) code with column weight 3, row weight 6, and rate $R = 0.504$. The Tanner graph of $\mathcal{C}_8$ contains 332 $(6,4)\{2\}$ trapping sets but contains no other (a,b) trapping sets with $a < 10$ and $b \leq 4$. Fig. 20 shows the FER performance of $\mathcal{C}_8$ under the SPA on the AWGNC in comparison with the performance of the PEG code of length $n = 504$ [37]. It can be seen that $\mathcal{C}_8$ has a lower error floor than the PEG code.

It should also be noted that while the method of removing trapping sets in this paper is applied to codes whose parity-check matrices are based on permutation matrices, there are other approaches to define a parity-check matrix of a structured LDPC code. They include finite geometries [54], balanced incomplete block designs [52], [55], protographs [56], Cayley graphs [57], [58], shortened Reed Solomon codes [59], and algebraic structures [26], [46], [53], [60], [61], to name the most popular ones. Some codes obtained by these constructions are equivalent (more on relationships among structured codes can be found in [62], [63], and the recent work [64]). Most of the aforementioned constructions only ensure that the Tanner graphs of codes are free of four cycles. Although a necessary and sufficient condition for a QC code to have a certain girth was derived in [53] and [65], the search for a code that satisfies this condition is computationally challenging. Some constructions also give codes with other desirable properties. Examples include codes constructed using Cayley graphs and protograph-based codes. If a code is constructed using Cayley graphs, then its Tanner graph has good local-girth distribution. On the other hand, the local neighborhood of a variable node in the Tanner graph of a protograph-based code is completely determined by the template protograph. Protograph-based codes can be made to have desirable symmetries [66] and can be encoded very efficiently [67].

To summarize, in this paper, we propose a construction of structured regular column-weight-three LDPC codes free of certain small trapping sets. The construction relies on the TSO, the method of searching for trapping sets and the evaluation of the relative harmfulness of different trapping sets. The feasibility of our construction is provided by the fact that the column weight is small. Generalizing our method to construct higher column-weight codes and irregular codes is possible but is complicated due to the exponential increase in the number of trapping sets as well as much higher complexity of the search algorithms. A large database of trapping sets with complicated topological relationships will also make it more complex to analyze failures of the decoder on one subgraph in the presence or absence of other topologically related subgraphs. Consequently, determining the relative harmfulness of different subgraphs might also be difficult. Nevertheless, our method might still be applicable to construct structured regular column-weight-four codes. Future research problems include the construction of such codes. This problem requires the derivation of a TSO for regular column-weight-four codes. Its feasibility also depends greatly on whether trapping sets can be enumerated and stored. As the search for new trapping sets after a block of columns is added only involves one new variable node (due to the use of the structural property of the code), its complexity might be well handled by state-of-the-art computers.

APPENDIX A
IMPLEMENTATION OF TECHNIQUES OF SEARCHING FOR TRAPPING SETS

In this appendix, we provide details on the techniques of searching for trapping sets. We begin by defining some simple subroutines that will be used in the search. Then, we discuss the technique to enumerate cycles. We continue by presenting the techniques to search for $(a+1, b-1)$ trapping sets and $(a+1, b)$ trapping sets that are generated by (a, b) trapping sets. Then, we discuss how the structural properties of a code can be used to reduce the complexity of the search. Finally, we give some remarks.

1) Subroutines: We assume that the following simple subroutines are used in our search algorithms.

a) $\mathbf{Y} = \mathbf{RowIntersectIndex}(\mathbf{X}_1, \mathbf{X}_2, \vartheta)$.
Let $\mathbf{X}_1$ and $\mathbf{X}_2$ be matrices with integer elements. $\mathbf{Y}$ is a matrix with two columns. If (i_1, i_2) is a row of $\mathbf{Y}$ then the i_1th row of $\mathbf{X}_1$ and the i_2th row of $\mathbf{X}_2$ share ϑ common entries.

b) $\mathbf{Y} = \mathbf{OddDegreeChecks}(H, \mathbf{X})$.
Let H be the parity-check matrix corresponding to a Tanner graph G of an LDPC code. Let $\mathbf{X}$ be a matrix with each row of $\mathbf{X}$ giving a set of variable nodes, i.e., elements in each row of $\mathbf{X}$ form a subset of V. Assume that all the subgraphs induced by variable nodes in rows of $\mathbf{X}$ have the same number of odd-degree check nodes. $\mathbf{Y}$ is a matrix with the same number of rows as $\mathbf{X}$. Elements of the ith row of $\mathbf{Y}$ are odd-degree check nodes in the subgraph induced by the variable nodes in the ith row of $\mathbf{X}$.

c) $\mathbf{Y} = \mathbf{TotalChecksOfDegreeK}(H, \mathbf{X}, \vartheta)$.
Let H be the parity-check matrix corresponding to a Tanner graph G. Let $\mathbf{X}$ be a matrix whose elements are variable nodes in G, i.e., elements of $\mathbf{X}$ are members of V. $\mathbf{Y}$ is a one-column matrix with the same number of rows as $\mathbf{X}$. The element in the ith row of $\mathbf{Y}$ is the number of check nodes with degree ϑ in the subgraph induced by the variable nodes in the ith row of $\mathbf{X}$.

d) $\mathbf{Y} = \mathbf{IsTrappingSet}(H, \mathbf{X})$.
Let H be the parity-check matrix corresponding to a Tanner graph G. Let $\mathbf{X}$ be a matrix whose elements are variable nodes in G. $\mathbf{Y}$ is a one-column matrix with the same number of rows as $\mathbf{X}$. The element in the ith row of $\mathbf{Y}$ is 1 if the variable nodes in the ith row of $\mathbf{X}$ form a trapping set and is 0 otherwise.

TABLE III
NUMBER OF CYCLES OF SEVERAL LDPC CODES AND RUN-TIME OF THE CYCLE SEARCHING ALGORITHM ON A 2.3-GHZ COMPUTER

Codes	Tanner [53]	Margulis [57]	MacKay [37]
n	155	2168	4095
$(d_{\mathrm{v}}, d_{\mathrm{c}})$	(3,6)	(3,6)	(3,17)
6-cycles	0	0	5183
8-cycles	465	1320	120930
10-cycles	3720	11088	2999961
Run-time (Seconds)	0.024	0.444	63.692

The aforementioned subroutines can be implemented using simple sparse matrix operations and, hence, are of low complexity.

2) Searching for Cycles of Length ϑ: Since every trapping set contains at least one cycle, the search for trapping sets always starts with finding cycles in the Tanner graph. All cycles of length ϑ that contain variable node v can be found by performing the following steps.

1) Find all the paths of length $\vartheta/2 - 1$ from each neighbor of v such that these paths do not contain v. The set of such paths that originate from the ith neighbor of v is denoted by N_i. It can be shown that $|N_i| \leq (d_{\mathrm{v}} - 1)^{t_1}(d_{\mathrm{c}} - 1)^{t_2}$ where
$$t_1 = \frac{\vartheta}{4} - 1,\ t_2 = t_1 + 1 \text{ if } \vartheta/2 \text{ is even}$$
$$t_1 = t_2 = \frac{\vartheta}{4} - \frac{1}{2} \text{ if } \vartheta/2 \text{ is odd}$$

2) For every pair of paths $o_i \in N_i$, $o_j \in N_j$, and $i \neq j$, determine if they have only one common node and if they end with the same node. If so, then a cycle of length ϑ has been found. This cycle is induced by the union of the sets of variable nodes in o_i, in o_j, and $\{v\}$. The maximum number of possible pairs o_i, o_j is $\sum_{i \neq j} |N_i||N_j|$.

The two steps described previously are executed for every variable node. To further simplify the search, after all the cycles containing v are found, v can be marked so that all the paths containing v are eliminated in Step 1 of the search at another variable node v'. The complexity of searching for cycles is proportional to the maximum number of pairs o_i, o_j and, hence, is polynomial in the degree of the variable nodes and check nodes for a given ϑ. However, as Step 1 and Step 2 are performed repeatedly for every variable nodes, the complexity increases only linearly in the code length. Note that our search algorithm not only counts the number of cycles but also records the variable nodes that each cycle contains. For this reason, existing efficient algorithms to count the number of cycles in a bipartite graph (for example, those proposed in [68] and [69]) cannot be applied directly.

Example 15: To illustrate the search algorithm, we list the number of cycles in some popular codes, as well as the run-times of the algorithm on a 2.6-GHz computer in Table III.

3) Searching for $(a+1, b-1)$ Trapping Sets Generated by (a, b) Trapping Sets: Let $\mathcal{T}_1$ be an (a, b) trapping set, $\mathcal{T}_2$ be an $(a+1, b-1)$ trapping set and let $\mathcal{T}_1$ be a predecessor of $\mathcal{T}_2$. Further, let $\mathbf{T}_1$ be a trapping set of type $\mathcal{T}_1$ in the Tanner graph of a code $\mathcal{C}$ and assume that $\mathbf{T}_1$ generates a trapping set $\mathbf{T}_2$ of type $\mathcal{T}_2$. As discussed in Section II, $\mathcal{T}_2$ is obtained by adjoining one variable node to $\mathcal{T}_1$. The line–point representation of $\mathcal{T}_2$ is obtained by merging two black-shaded nodes in the line–point representation of $\mathcal{T}_1$ with two $\circledast$ nodes in Fig. 3(b). Therefore, to search for $\mathbf{T}_2$, it is sufficient to search for a variable node that is connected to two odd-degree check nodes in the subgraph induced by variable nodes in $\mathbf{T}_1$.

Let $\mathbf{X}$ be a matrix such that each row of $\mathbf{X}$ contains variable nodes of a $\mathcal{T}_1$ trapping set in the Tanner graph G. H is the parity-check matrix which defines $\mathcal{C}$. All $\mathcal{T}_2$ trapping sets can be found by performing the following steps.

1) Find all odd-degree check nodes of all $\mathcal{T}_1$ trapping sets: $\mathbf{Y}_1 = \mathbf{OddDegreeChecks}(H, \mathbf{X})$.
2) Form $\mathbf{X}_1$, a one-column matrix with n rows where the element in the ith row is variable node i.
3) Form a matrix $\mathbf{Y}_2$ whose ith row gives all check nodes neighboring to the variable node i: $\mathbf{Y}_2 = \mathbf{OddDegreeChecks}(H, \mathbf{X}_1)$.
4) Find all pairs (i, j) such that the ith row of $\mathbf{Y}_1$ and the jth row of $\mathbf{Y}_2$ share 2 common entries: $\mathbf{Y}_3 = \mathbf{RowIntersectIndex}(\mathbf{Y}_1, \mathbf{Y}_2, 2)$.
5) If (i, j) is the lth row of $\mathbf{Y}_3$, adjoin variable node j to the ith row of $\mathbf{X}$ to form the lth row of $\mathbf{Y}_4$.
6) Determine the number of degree one check nodes in the subgraph induced by variable nodes in each row of $\mathbf{Y}_4$ and eliminate the rows of $\mathbf{Y}_4$ that do not have $b-1$ degree one check nodes, to obtain a matrix $\mathbf{Y}$. Each row of $\mathbf{Y}$ now contains variable nodes that induce a $\mathcal{T}_2$ trapping set in the Tanner graph of the code. $\mathbf{Y} = \mathbf{Y}_4(\mathbf{TotalChecksOfDegreeK}(H, \mathbf{Y}_4, 1) == b-1)$.

4) Searching for $(a+2, b)$ Trapping Sets Generated by (a, b) Trapping Sets: Let $\mathcal{T}_1$ be an (a, b) trapping set, $\mathcal{T}_2$ be an $(a+2, b)$ trapping set and let $\mathcal{T}_1$ be a predecessor of $\mathcal{T}_2$. Further, let $\mathbf{T}_1$ be a trapping set of type $\mathcal{T}_1$ in the Tanner graph of a code $\mathcal{C}$ and assume that $\mathbf{T}_1$ generates a trapping set $\mathbf{T}_2$ of type $\mathcal{T}_2$. Consider two variable nodes that share a check node. As discussed in Section II, $\mathcal{T}_2$ is obtained by adjoining these two variable nodes to $\mathcal{T}_1$. The line–point representation of $\mathcal{T}_2$ is obtained by merging two black-shaded nodes in the line–point representation of $\mathcal{T}_1$ with two $\circledast$ nodes in Fig. 3(c). Therefore, to search for $\mathbf{T}_2$, it is sufficient to search for a pair of variable nodes that share a common neighboring check node and each node is connected to one odd-degree check node in the subgraph induced by variable nodes in $\mathbf{T}_1$.

The search for $(a+2, b)$ trapping sets is very similar to the search for $(a+1, b-1)$ trapping sets described in the previous subsection. In particular, the following modifications should be made.

1) In Step 2, $\mathbf{X}_1$ is a two column matrix, each row contains a pair of variable nodes that share a common neighboring check node.
2) In Step 3, the ith row of $\mathbf{Y}_2$ gives all degree one check nodes in the subgraph induced by variable nodes in the ith row of $\mathbf{X}_1$.
3) In Step 5, variable nodes in the jth row of $\mathbf{X}_1$ are adjoined to the ith row of $\mathbf{X}$.
4) In Step 6, $b-1$ is replaced by b.

Since Step 4 does not take into account the case in which two check nodes of a new variable node are merged with two odd-degree check nodes of $\mathbf{T}_1$, the subroutine **IsTrappingSet** is used afterward to eliminate rows of $\mathbf{Y}$ that do not contain variable nodes that form a trapping set.

5) Using the Structural Property to Reduce the Complexity of the Search Algorithms: We discuss how to use the property of a structured LDPC code to reduce the complexity of the search algorithms. Let $\mathcal{C}$ be a structured LDPC code defined in Section IV with a Tanner graph representation G and a corresponding parity-check matrix H. To facilitate the discussion, we assume that the quasi-group $\mathcal{Q}$ is a cyclic group. Then, $\mathcal{C}$ is a QC LDPC code and H can be represented as an array of circulant permutation matrices, i.e., H takes the following form:

$$H = \begin{bmatrix} I & I & I & \cdots & I \\ I & P_{22} & P_{23} & \cdots & P_{2d_c} \\ I & P_{32} & P_{33} & \cdots & P_{3d_c} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & P_{d_v 2} & P_{d_v 2} & \cdots & P_{d_v d_c} \end{bmatrix}$$

where P_{ij} is a circulant permutation matrix of size $p \times p$ for $2 \le i \le d_v$, $2 \le j \le d_c$.

Let π be a function from $V \times \{0, 1, \ldots, p-1\}$ to V defined as follows:

$$\pi(v, s) = \mathfrak{q}(v) \times p + (\mathfrak{r}(v) + s) \bmod p$$

where $\mathfrak{q}(v)$ and $\mathfrak{r}(v)$ are the quotient and remainder after the division of v by p, respectively.

From properties of the parity-check matrix H, one can show the following.

Quasi-cyclicity property: Let $\mathbf{S} = \{v_1, v_2, \ldots, v_{|\mathbf{S}|}\} \subset V$ be a subset of variable nodes, then the subgraph induced by all the variable nodes in $\mathbf{S}$ is isormorphic to the subgraph induced by all the variable nodes in $\mathbf{S}_s = \{\pi(v_1, s), \pi(v_2, s), \ldots, \pi(v_{|\mathbf{S}|}, s)\}, \forall\, 0 \le s \le p-1$.

Now we can use this property to reduce the complexity of the search algorithms. Let $\mathbf{T}$ be a trapping set of type $\mathcal{T}$ which is present in the Tanner graph of a code $\mathcal{C}$. Then, one can obtain $p-1$ other type $\mathcal{T}$ trapping sets, namely $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{p-1}$, by shifting variable nodes of $\mathbf{T}$ using the map π. Assume that we want to find all trapping sets of type $\mathcal{T}'$ by expanding trapping sets $\mathbf{T}, \mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{p-1}$. Instead of expanding every single trapping set, one can choose to expand only one trapping set in the set $\{\mathbf{T}, \mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{p-1}\}$ to obtain $\mathcal{T}'$ trapping sets. The map π can then be used on the variable nodes of the obtained type $\mathcal{T}'$ trapping sets to obtain the other type $\mathcal{T}'$ trapping sets. This significantly reduces the complexity of the search algorithms since the complexity of expanding a trapping set is much greater than the complexity of shifting variable nodes.

6) Remarks: The aforementioned search procedures may not differentiate among different (a, b) trapping sets. For example, all $(5,3)\{1\}$ and $(5,3)\{2\}$ trapping sets are found if they are searched for as trapping sets generated by the $(4,4)$ trapping sets. Similarly, all $(7,3)\{2\}$ and $(7,3)\{3\}$ trapping sets are found as being generated by the $(6,4)\{1\}$ trapping sets. If searching for a specific type of trapping set is required, then it is necessary to further analyze the induced subgraph. For example,

TABLE IV
NUMBER OF CYCLES AND TRAPPING SETS OF THE TANNER CODE AND RUN-TIME OF THE SEARCHING ALGORITHMS ON A 2.3-GHz COMPUTER

Trapping Sets	Total	Run-time (Seconds)	
		(i)	(ii)
6-cycles	0	0.024	0.004
8-cycles	465		
10-cycles	3720		
$(5,3)\{2\}$	155	0.004	0.001
$(6,4)\{2\}$	930	0.023	0.001
$(7,3)\{1\}$	930	0.008	0.002
$(8,2)\{1\}$ and $\{2\}$	465	0.008	0.001

TABLE V
NUMBER OF CYCLES AND TRAPPING SETS OF THE CODE $\mathcal{C}_2$ AND RUN-TIME OF THE SEARCHING ALGORITHMS ON A 2.3-GHz COMPUTER

Trapping Sets	Total	Run-time (Seconds)	
		(i)	(ii)
6-cycles	0	1.362	0.109
8-cycles	17066		
10-cycles	183433		
$(5,3)\{2\}$	1590	0.130	0.004
$(6,2)\{1\}$	424	0.009	$< 10^{-8}$
$(7,3)\{1\}$	6254	0.260	0.007
$(8,2)\{1\}$	1166	0.160	0.002
$(8,2)\{2\}$	901	0.033	0.002
$(6,4)\{1\}$	85065	85.437	1.037
$(6,4)\{1\}$ and $\{2\}$	148983	273.854	0.232
$(7,3)\{2\}$ and $\{3\}$	23850	5.750	0.045
$(8,2)\{3\}$, $\{4\}$ and $\{5\}$	5936	0.409	0.015

notice that the $(5,3)\{1\}$ trapping set is a union of a six-cycle and an eight-cycle, sharing two variable nodes while the $(5,3)\{2\}$ trapping set is a union of two eight-cycles, sharing three variable nodes. Therefore, to search for all $(5,3)\{2\}$ trapping sets from a list of $(4,4)$ trapping sets, one would find all pairs of $(4,4)$ trapping sets that share three variable nodes, using the **RowIntersectIndex** subroutine. The union of each pair of $(4,4)$ trapping sets is then a set of five variable nodes. Each set of variable nodes forms a $(5,3)\{2\}$ trapping set if its induced subgraph contains three degree one check nodes. Similarly, all $(7,3)\{2\}$ trapping sets can be found by noticing that they are unions of two $(6,4)\{1\}$ trapping sets, sharing five variable nodes.

Example 16: We end this section by giving the statistics of small trapping sets present in the popular Tanner code (see Table IV) as well as in the code $\mathcal{C}_2$ (see Table V), which was given in Section VII. We also give the running times of the algorithms for two different cases: 1) the structural property of the code is not utilized; and 2) the structural property of the code is utilized. All the searches were performed on a 2.3-GHz computer.

APPENDIX B
PROOF OF LEMMA 1

We prove that the incidence structures formed by the u new lines must be one of those listed in Fig. 3. For $u = 1$ or $u = 2$, the proof is trivial and all possible incidence structures are listed in Fig. 3(a)–(c). For $u \geq 3$, we divide the proof into two cases.

Case 1: The incidence structure formed by the u new lines does not contain any cycle: We first argue that no line can pass through three white-shaded points. To see this, assume that at least one line passes through three white-shaded points, as demonstrated in Fig. 21(a). Then, the merging of the incidence structure formed by the u new lines with the line–point representation of $\mathcal{T}$ can be divided into two steps, as demonstrated in Fig. 21(b) and (c). Consequently, the resulting trapping set is not a direct successor of $\mathcal{T}$.

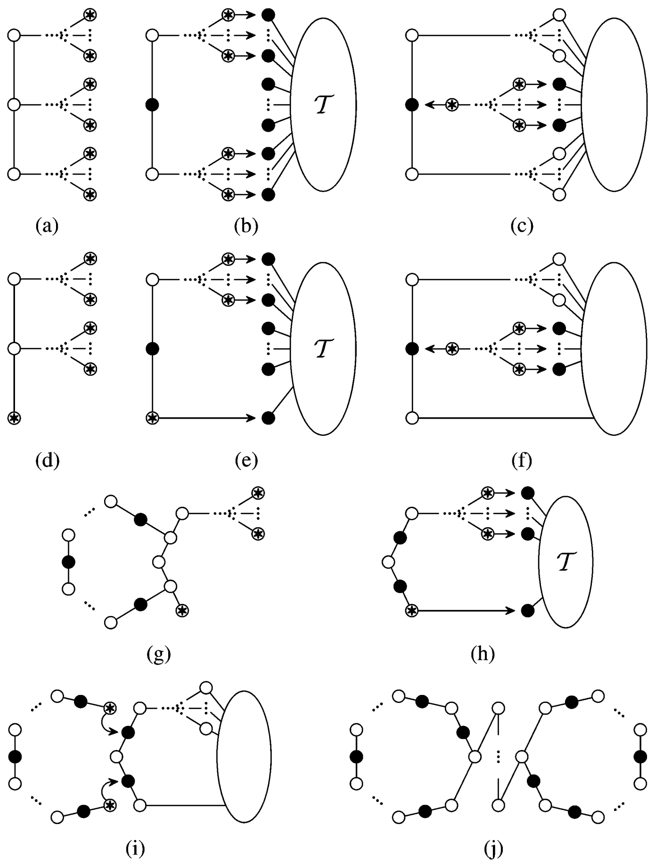

Fig. 21. Figures used in the proof of Lemma 1.

We also argue that if a line passes through two white-shaded points, then the third point cannot be a ⊛ point. This is because the merging of the incidence structure listed in Fig. 21(d) with $\mathcal{T}$ can be divided into two steps, as demonstrated in Fig. 21(e) and (f). Besides, a line can pass through at most one black-shaded point. As a result, the incidence structure formed by the u new lines can only be the one listed in Fig. 3(e) (or in Fig. 3(d) if $u = 3$).

Case 2: The incidence structure formed by the u new lines contains at least one cycle: We first argue that only one of those lines that form a cycle can pass through three white-shaded points or pass through two white-shaded points and a ⊛ point. The other lines must pass through exactly two white-shaded points and one black-shaded point. Otherwise, the merging of the incidence structure formed by the u new lines with the line–point representation of $\mathcal{T}$ can be divided into two steps. For example, the merging of the incidence structure listed in Fig. 21(g) with $\mathcal{T}$ can be divided into two steps, as demonstrated in Fig. 21(h) and (i).

Second, the lines that are not part of a cycle can neither pass through three white-shaded points nor pass through two white-shaded points and a ⊛ point. The explanation for this is similar to the explanation found in case 1.

Finally, we argue that there can be at most one cycle. To see this, assume that there are two cycles. Since only one of the lines that form a cycle can pass through three white-shaded points, the incidence structure formed by the u new lines must have the form as listed in Fig. 21(j). However, since the lines that are not part of any cycle can neither pass through three white-shaded points nor pass through two white-shaded points and a $\circledast$ point, there cannot be any $\circledast$ point in the incidence structure formed by the u new lines, which is a contradiction.

Consequently, all possible structures formed by the u new lines are listed in Fig. 3(l).

Acknowledgment

The authors highly appreciate the constructive comments and suggestions made by the anonymous reviewers and especially by the associate editor P. Vontobel. The work of S. K. Chilappagari was performed when he was with the Department of Electrical and Computer Engineering, University of Arizona, Tucson.

References

[1] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, Oct. 1–3, 2003, pp. 1426–1435.

[2] M. Ivkovic, S. K. Chilappagari, and B. Vasic, "Trapping sets in low-density parity-check codes by using Tanner graph covers," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3763–3768, Aug. 2008.

[3] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.

[4] O. Milenkovic, N. Kashyap, and D. Leyba, "Shortened array codes of large girth," *IEEE Trans. Inf. Theory*, vol. 52, no. 8, pp. 3707–3722, Aug. 2006.

[5] Y. Wang, J. S. Yedidia, and S. C. Draper, "Construction of high-girth QC-LDPC codes," in *Proc. 5th Int. Symp. Turbo Codes Related Top.*, Lausanne, Switzerland, Sep. 1–5, 2008, pp. 180–185.

[6] S. Kim, J.-S. No, H. Chung, and D.-J. Shin, "Quasi-cyclic low-density parity-check codes with girth larger than 12," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, Jun. 2007, vol. 53, pp. 2885–2891.

[7] J. Lu, J. M. F. Moura, and U. Niesen, "Grouping-and-shifting designs for structured LDPC codes with large girth," in *Proc. IEEE Int. Symp. Inf. Theory*, Chicago, IL, Jun. 27–Jul. 2 2004, pp. 236–236.

[8] Y.-K. Lin, C.-L. Chen, Y.-C. Liao, and H.-C. Chang, "Structured LDPC codes with low error floor based on PEG Tanner graphs," in *Proc. IEEE Int. Symp. Circuits Syst.*, Seattle, WA, May 18–21, 2008, pp. 1846–1849.

[9] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[10] 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.

[11] C. A. Kelley and D. Sridhara, "Pseudocodewords of Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 53, no. 11, pp. 4013–4038, Nov. 2007.

[12] S.-T. Xia and F.-W. Fu, "Minimum pseudoweight and minimum pseudocodewords of LDPC codes," *IEEE Trans. Inf. Theory*, vol. 54, no. 1, pp. 480–485, Jan. 2008.

[13] A. Orlitsky, R. Urbanke, K. Viswanathan, and J. Zhang, "Stopping sets and the girth of Tanner graphs," in *Proc. IEEE Int. Symp. Inf. Theory*, Lausanne, Switzerland, Jun. 30–Jul. 5 2002, p. 2.

[14] S. K. Chilappagari and B. Vasic, "Error-correction capability of column-weight-three LDPC codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2055–2061, May 2009.

[15] S. K. Chilappagari, D. V. Nguyen, B. V. Vasic, and M. W. Marcellin, "Error correction capability of column-weight-three LDPC codes under the Gallager A algorithm—Part II," *IEEE Trans. Inf. Theory*, vol. 56, no. 6, pp. 2626–2639, Jun. 2010.

[16] M. G. Stepanov, V. Chernyak, M. Chertkov, and B. Vasic, "Diagnosis of weaknesses in modern error correction codes: A physics approach," *Phys. Rev. Lett.*, vol. 95, no. 22, pp. 228701–228704, Nov. 2005.

[17] V. Chernyak, M. Chertkov, M. G. Stepanov, and B. Vasic, "Error correction on a tree: An instanton approach," *Phys. Rev. Lett.*, vol. 93, no. 19, pp. 198702–198705, Nov. 2004.

[18] V. Chernyak, M. Chertkov, M. G. Stepanov, and B. Vasic, "Instanton method of post-error-correction analytical evaluation," in *Proc. IEEE Inf. Theory Workshop*, San Antonio, TX, Oct. 24–29, 2004, pp. 220–224.

[19] S. K. Chilappagari, M. Chertkov, M. G. Stepanov, and B. Vasic, "Instanton-based techniques for analysis and reduction of error floors of LDPC codes," *IEEE JSAC Capacity Approach. Codes*, vol. 27, no. 6, pp. 855–865, Aug. 2009.

[20] B. Vasic, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery, "Trapping set ontology," in *Proc. 47th Allerton Conf. Commun., Control, Comput.*, Monticello, IL, Sep. 30–Oct. 2 2009, pp. 1–7.

[21] Trapping Set Ontology [Online]. Available: http://www.ece.arizona.edu/~vasiclab/Projects/CodingTheory/TrappingSetOntology.html 2009

[22] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "Analysis of absorbing sets for array-based LDPC codes," in *Proc. Int. Conf. Commun.*, Glasgow, Scotland, Jun. 24–29, 2007, pp. 6261–6268.

[23] Z. Li, L. Chen, L. Zeng, S. Lin, and W. H. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 54, no. 1, pp. 71–81, Jan. 2006.

[24] C.-H. Liu, C.-C. Lin, S.-W. Yen, C.-L. Chen, H.-C. Chang, C.-Y. Lee, Y.-S. Hsu, and S.-J. Jou, "Design of a multimode QC-LDPC decoder based on shift-routing network," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 56, no. 9, pp. 734–738, Sep. 2009.

[25] Z. Cui, Z. Wang, and Y. Liu, "High-throughput layered LDPC decoding architecture," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 4, pp. 582–587, Apr. 2009.

[26] J. L. Fan, "Array codes as low-density parity-check codes," in *Proc. 2nd Int. Symp. Turbo Codes Related Topics*, Brest, France, Sep. 4–7, 2000, pp. 543–546.

[27] B. Vasic, K. Pedagani, and M. Ivkovic, "High-rate girth-eight low-density parity-check codes on rectangular integer lattices," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1248–1252, Aug. 2004.

[28] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: MIT Press, 1963.

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

[30] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[31] O. Milenkovic, E. Soljanin, and P. Whiting, "Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 39–55, Jan. 2007.

[32] H. Xiao and A. H. Banihashemi, "Estimation of bit and frame error rates of finite-length low-density parity-check codes on binary symmetric channels," *IEEE Trans. Commun.*, vol. 55, no. 12, pp. 2234–2239, Dec. 2007.

[33] H. Xiao and A. H. Banihashemi, "Error rate estimation of low-density parity-check codes on binary symmetric channels using cycle enumeration," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1550–1555, Jun. 2009.

[34] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.

[35] Y. Zhang and W. Ryan, "Toward low LDPC-code floors: A case study," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1566–1573, Jun. 2009.

[36] D. J. C. MacKay and M. J. Postol, "Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes," *Electron. Notes Theor. Comput. Sci.*, vol. 74, pp. 97–104, 2003.

[37] D. J. C. MacKay, Encyclopedia of Sparse Graph Codes [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html [Online]. Available

[38] K. M. Krishnan and P. Shankar, "Computing the stopping distance of a Tanner graph is NP-hard," *IEEE Trans. Inf. Theory*, vol. 53, no. 6, pp. 2278–2280, Jun. 2007.

[39] A. McGregor and O. Milenkovic, "On the hardness of approximating stopping and trapping sets," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1640–1650, Apr. 2010.

[40] C. C. Wang, S. R. Kulkarni, and H. V. Poor, "Finding all error-prone substructures in LDPC codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 1976–1999, May 2009.

[41] G. B. Kyung and C.-C. Wang, "Exhaustive search for small fully absorbing sets and the corresponding low error-floor decoder," in *Proc. IEEE Int. Symp. Inf. Theory*, Austin, TX, Jun. 13–18, 2010, pp. 739–743.
[42] M. Hirotomo, Y. Konishi, and M. Morii, "Approximate examination of trapping sets of LDPC codes using the probabilistic algorithm," in *Proc. Int. Symp. Inf. Theory Its Appl.*, Auckland, New Zealand, Dec. 7–10, 2008, pp. 1–6.
[43] S. Abu-Surra, D. Declercq, D. Divsalar, and W. E. Ryan, "Trapping set enumerators for specific LDPC codes," in *Proc. Inf. Theory Appl. Workshop*, La Jolla, CA, Jan. 31–Feb. 5 2010, pp. 1–5.
[44] M. K. Dehkordi and A. H. Banihashemi, "An efficient algorithm for finding dominant trapping sets of LDPC codes," in *Proc. 6th Int. Symp. Turbo Codes Iterat. Inf. Process.*, Brest, France, Sep. 6–10, 2010, pp. 444–448.
[45] C. J. Colbourn and J. H. Dinitz, *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*. London, U.K.: Chapman & Hall/CRC Press, 2006.
[46] L. Lan, L. Zeng, Y. Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, "Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: A finite field approach," *IEEE Trans. Inf. Theory*, vol. 53, no. 7, pp. 2429–2458, Jul. 2007.
[47] S. K. Chilappagari, A. R. Krishnan, and B. Vasic, "LDPC codes which can correct three errors under iterative decoding," in *Proc. IEEE Inf. Theory Workshop*, Porto, Portugal, May 5–9, 2008, pp. 406–410.
[48] R. Asvadi, A. H. Banihashemi, and M. Ahmadian-Attari, "Lowering the error floor of LDPC codes using cyclic liftings," *IEEE Trans. Inf. Theory*, vol. 57, no. 4, pp. 2213–2224, Apr. 2011.
[49] C. A. Cole, S. G. Wilson, E. K. Hall, and T. R. Giallorenzi, "Analysis and design of moderate length regular LDPC codes with low error floors," in *Proc. 40th Annu. Conf. Inf. Sci. Syst.*, Princeton, NJ, Mar. 22–24, 2006, pp. 823–828.
[50] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.
[51] J. Wang, L. Dolecek, and R. Wesel, "Controlling LDPC absorbing sets via the null space of the cycle consistency matrix," in *Proc. Int. Conf. Commun.*, Kyoto, Japan, Jun. 5–9, 2011, pp. 1–6.
[52] B. Vasic and O. Milenkovic, "Combinatorial constructions of low-density parity-check codes for iterative decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1156–1176, Jun. 2004.
[53] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," presented at the presented at the 5th Int. Symp. Commun. Theory Appl., Ambleside, U.K., Jul. 15–20, 2001.
[54] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.
[55] B. Ammar, B. Honary, Y. Kou, J. Xu, and S. Lin, "Construction of low-density parity-check codes based on balanced incomplete block designs," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1257–1269, Jun. 2004.
[56] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," IPN Progr. Rep. 42-154. Pasadena, CA, Aug. 2003, Jet Propulsion Lab..
[57] G. A. Margulis, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, no. 1, pp. 71–78, 1982.
[58] J. Rosenthal and P. O. Vontobel, "Constructions of regular and irregular LDPC codes using Ramanujan graphs and ideas from Margulis," in *Proc. IEEE Int. Symp. Inf. Theory*, Washington, DC, Jun. 24–29, 2001, pp. 4–4.
[59] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols," *IEEE Commun. Lett.*, vol. 7, no. 7, pp. 317–319, Jul. 2003.
[60] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello Jr., "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.
[61] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-shannon-limit quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 52, no. 7, pp. 1038–1042, Jul. 2004.
[62] I. Djurdjevic, A. V. Kuznetsov, and E. M. Kurtas, "Relationships among classes of structured LDPC codes and their application to data storage," in *Proc. 4th Int. Symp. Turbo Codes Related Top.; 6th Int. ITG-Conf. Source Channel Coding*, Munich, Germany, Apr. 3–7, 2006, pp. 1–6.
[63] A. Mahadevan and J. M. Morris, "On RCD SPC codes as LDPC codes based on arrays and their equivalence to some codes constructed from Euclidean geometries and partial BIBDs," Commun. Signal Process. Lab., Comput. Sci. Electr. Eng. Dept., Univ. Maryland Baltimore County, Tech. Rep. CSPL TR: 2002-1, 2002, vol. 42–154.
[64] S. Kovalev and V. Y. Krachkovsky, "A simple method to construct LDPC codes based on projective planes," in *Proc. IEEE Int. Symp. Inf. Theory*, St. Petersburg, Russia, Jul. 31–Aug. 5 2011, pp. 742–746.
[65] 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.
[66] R. M. Tanner, "On quasi-cyclic repeat-accumulate codes," in *Proc. 41st Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, Sep. 22–24, 1999, pp. 249–259.
[67] H. Jin, T. J. Richardson, and V. Novichkov, "Methods and Apparatus for Encoding LDPC Codes," US Patent #6 961 888, 2005.
[68] T. R. Halford and K. M. Chugg, "An algorithm for counting short cycles in bipartite graphs," *IEEE Trans. Inf. Theory*, vol. 52, no. 1, pp. 287–292, Jan. 2006.
[69] M. Karimi and A. H. Banihashemi, "A message-passing algorithm for counting short cycles in a graph," in *Proc. IEEE Inf. Theory Workshop*, 2010, pp. 1–5.

Dung Viet Nguyen (S'07) received the B.S. degree in electrical engineering from the University of Arizona, Tucson, AZ, in 2007, where he is currently pursuing the Ph.D. degree. His research interests include digital communications and information theory.

Shashi Kiran Chilappagari (S'05–M'09) received the B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology, Madras, India in 2004 and Ph.D. in electrical engineering from the University of Arizona in 2008. He was a Research Engineer in the Department of Electrical and Computer Engineering at the University of Arizona, Tucson from January 2009 to December 2009. He is currently with Marvell Semiconductor Inc, Santa Clara, California. His research interests include error control coding and information theory with focus on the analysis of failures of various sub-optimal decoding algorithms for LDPC codes.

Michael W. Marcellin (S'81–M'87–SM'93–F'02) was born in Bishop, California, on July 1, 1959. He graduated summa cum laude with the B.S. degree in Electrical Engineering from San Diego State University in 1983, where he was named the most outstanding student in the College of Engineering. He received the M.S. and Ph.D. degrees in Electrical Engineering from Texas A&M University in 1985 and 1987, respectively.

Since 1988, Dr. Marcellin has been with the University of Arizona, where he holds the title of Regents' Professor of Electrical and Computer Engineering, and of Optical Sciences. He is currently on sabbatical at Universitat Autonoma Barcelona where he is a visiting professor and Marie Curie Fellow. His research interests include digital communication and data storage systems, data compression, and signal processing. He has authored or coauthored more than two hundred publications in these areas.

Dr. Marcellin is a major contributor to JPEG2000, the second-generation ISO standard for image compression. Throughout the standardization process, he chaired the JPEG2000 Verification Model Ad Hoc Group, which was responsible for the software implementation and documentation of the JPEG2000 algorithm. He is coauthor of the book, JPEG2000: Image compression fundamentals, standards and practice, Kluwer Academic Publishers, 2002. This book serves as a graduate level textbook on image compression fundamentals, as well as the definitive reference on JPEG2000. Dr. Marcellin served as a consultant to Digital Cinema Initiatives (DCI), a consortium of Hollywood studios, on the development of the JPEG2000 profiles for digital cinema.

Professor Marcellin is a Fellow of the IEEE, and is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi. He is a 1992 recipient of the National Science Foundation Young Investigator Award, and a corecipient of the 1993 IEEE Signal Processing Society Senior (Best Paper) Award. He has received teaching awards from NTU (1990, 2001), IEEE/Eta Kappa Nu student sections (1997), and the University of Arizona College of Engineering (2000, 2010). In 2003, he was named the San Diego State University Distinguished Engineering Alumnus. Professor Marcellin is the recipient of the 2006 University of Arizona Technology Innovation Award. From 2001 to 2006, Dr. Marcellin was the Litton Industries John M. Leonis Professor of Engineering. He is currently the International Foundation for Telemetering Professor of Electrical and Computer Engineering at the University of Arizona.

Bane Vasić (S'92–M'93–SM'02–F'10) received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the University of Nis, Nis, Yugoslavia (now Serbia), in 1989, 1991, and 1994, respectively.

From 1996 to 1997, he was a Visiting Scientist at the Rochester Institute of Technology and Kodak Research, Rochester, NY, where he was involved in research in optical storage channels. From 1998 to 2000, he was with Lucent Technologies, Bell Laboratories (Bell-Labs). He was involved in research in iterative decoding and low-density parity-check codes, as well as development of codes and detectors implemented in Bell-Labs chips. Presently, he is a Professor in the Electrical and Computer Engineering Department, University of Arizona, Tucson. His research interests include coding theory, information theory, communication theory, and digital communications and recording.

Dr. Vasić is a Member of the Editorial Board for the IEEE TRANSACTIONS ON MAGNETICS. He served as Technical Program Chair of the IEEE Communication TheoryWorkshop in 2003 and as Co-organizer of the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS)Workshops on Optical/Magnetic Recording and Optical Transmission, and Theoretical Advances in Information Recording in 2004. He was Co-organizer of the Los Alamos Workshop on Applications of Statistical Physics to Coding Theory in 2004, the Communication Theory Symposium within the IEEE International Conference on Communications (ICC 2006), and IEEE Communication Theory Workshop (CTW 2007).