Failures and Error-Floors of Iterative Decoders

Bane Vasić, *Fellow, IEEE*, Shashi Kiran Chilappagari, *Member, IEEE*
and Dung Viet Nguyen, *Student Member, IEEE*

Abstract

We present on overview of a family of methods developed recently for the analysis of the error floors of low-density parity-check (LDPC) codes.

I. Introduction

Low-density parity-check (LDPC) codes [1], [2], have been the focus of intense research over the past decade due to their ability to approach theoretical limits of reliable transmission over various channels even when decoded by sub-optimal low complexity algorithms. LDPC codes are a class of linear block codes which can be defined by sparse bipartite graphs. Traditional algorithms for decoding LDPC codes are based on belief propagation (BP) [3], and operate on a graphical representation of the code, known as *Tanner graph* [4]. These algorithms operate by iteratively passing messages along the edges of the Tanner graph and are generally referred to as iterative message passing algorithms. The BP, as an algorithm to compute marginals of functions on a graphical model, has its roots in the broad class of Bayesian inference problems [5]. While inference using BP is exact only on loop-free graphs (trees), it surprisingly provides close approximations to exact marginals on loopy graphs. However, an understanding of the behavior of BP in the latter case is far from complete. Exact computation of probabilistic inference in Bayesian belief networks is not only NP-hard in general, but is hard even under strong restrictions of graphical model topology [6].

The properties of LDPC codes under different decoding algorithms have been analyzed by studying ensembles of codes in the limit of code length approaching infinity, or equivalently under the assumption that the underlying Tanner graph is a tree. A technique known as density evolution has been developed to derive the capacity of LDPC codes under the message passing algorithms. The method analyzes the average behavior of code ensembles and shows that a code selected at random behaves very close to ensemble average. These results are in the spirit of Shannon's work in which random codes were shown to achieve capacity without exhibiting a explicit way to select such a code. Unfortunately, the methods to analyze ensembles of codes are of limited use for the performance analysis of a given finite length code.

From a practical perspective, however, it would be beneficial to characterize the performance of a particular code. Analytical characterization of the performance of a code requires an understanding of the conditions that lead to the failure of the decoder under question. A decoder is said to have failed when the output of the decoder is not a codeword. Note that it is possible that the decoder finds a codeword different from the transmitted codeword and this is generally referred to as a decoding error. The decoding failures and errors are a function of the code, the decoder and the channel. In the case of iterative message algorithms, the decoders operate on the Tanner graph of the code and consequently the failures and errors are also a function of the topological structure of the Tanner graph. They also depend on the message update rules, but also on the message passing schedule and the implementation aspects such as message precision.

The theoretically best possible performance for a code is achieved under the so-called maximum likelihood (ML) decoder. Given a received vector, the ML decoder finds the *nearest* codeword, where the metric to characterize the nearness is dependent on the channel. Note that the ML decoder by definition does not fail to find a codeword. For hard decision channels, the nearest codeword is the one with the least Hamming distance from the received vector. Hence the ML decoding of a binary code on a hard decision channel is a Voronoi binning of points in the Hamming space with codewords as centers of the regions. The ML decoder makes an error when the received vector falls into the Voronoi region of another codeword. Voronoi decomposition completely characterizes the performance of the ML decoding algorithm. Naturally, such an algorithm has exponential complexity and consequently most codes are decoded using low-complexity and practical decoding algorithms.

Algebraic codes such as the Bose-Chaudhuri-Hocquenghem (BCH) codes and Reed-Solomon (RS) codes are decoded using a class of hard decision decoders known as bounded distance decoders (BDDs). A BDD, as the name suggests, is capable of correcting a fixed number of errors. A BDD decodes a received vector to a codeword within a Hamming distance of t (if such a codeword exists). Otherwise, the decoder reports a failure. All the vectors within a distance of t from a codeword are said to fall in the decoding sphere of the codeword. Hence, a BDD with correction capability t fails whenever the received vector does not fall into the decoding sphere of any codeword.

Such a level of understanding of failures has not been achieved for iterative decoders. The understanding of failures of iterative decoders has assumed significance in the wake of a phenomenon known as *error floor*. Generally, the performance

B. Vasić is with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721 USA (e-mail: vasic@ece.arizona.edu).
S. K. Chilappagari is with Marvell Semiconductor Inc., Santa Clara, CA 95054 USA (email: shashickiran@gmail.com).
Manuscript received December 2, 2012; revised February 3, 2013. This work was funded in part by NSF under grant CCF-0963726

of a code under a particular decoding algorithm is characterized by the Bit-Error-Rate (BER) or the Frame-Error-Rate (FER) curve plotted as a function of the Signal-to-Noise Ratio (SNR). A typical BER/FER vs SNR curve consists of two distinct regions. At small SNR, the error probability decreases rapidly with SNR, with the curve looking like a *waterfall*. The decrease slows down at moderate values turning into the *error floor* asymptotic at very large SNR [7]. This transient behavior and the error floor asymptotic originate from the sub-optimality of decoder, i.e., the ideal maximum-likelihood (ML) curve would not show such a dramatic change in the BER/FER with the SNR increase. While the slope of the BER/FER curve in the waterfall region is the same for almost all the codes in the ensemble, there can be a huge variation in the slopes for different codes in the error floor region [8]. Since for sufficiently long codes the error floor phenomenon manifests itself in the domain unreachable by brute force Monte-Carlo (MC) simulations, analytical methods are necessary to characterize the FER performance. This is very crucial for practical applications such as data storage where the requirements on data reliability are often under 10^{-15}. To guarantee such reliability, one should be able to predict the performance of a given code that would be deployed in the system.

Significant research efforts have been directed toward answering questions related to the minimum distance and guaranteed error correction properties of code ensembles. Gallager showed that for column weight $\gamma \geq 3$ and row weight $\rho > \gamma$, the minimum distance of the (γ,ρ) ensemble of LDPC codes increases linearly with the code length. The correction of a linear number of errors was however not established. When we say that an algorithm will correct a linear number of worst case errors for a code $\mathcal{C}$ of length n, we mean that for some constant $0 < \alpha < 1$, the algorithm corrects any arbitrary error pattern with up to αn errors. Zyablov and Pinsker [9] showed that, asymptotically the parallel bit flipping algorithm can correct a linear number of errors for almost all the codes in the regular ensemble with $\gamma \geq 5$. Sipser and Spielman [10] used expander graph arguments to analyze two bit flipping algorithms, serial and parallel. Specifically, they showed that these algorithms can correct a linear number of errors if the underlying Tanner graph is a good expander. Burshtein and Miller [11] applied expander based arguments to show that message passing algorithms can also correct a linear number of errors when the degree of each variable node is more than five. Recently, Burshtein [12] showed that regular codes with variable nodes of degree four are capable of correcting a linear number of errors under the parallel bit flipping algorithm. Burshtein [12] was also the first to observe that it cannot be proved that almost all the codes in the ensemble of column-weight-three LDPC codes can correct a linear number of errors as a non-negligible fraction of codes have parallel edges in their Tanner graphs and such codes cannot correct a single worst case error. A stronger statement was proved in [13], where it was shown that at sufficiently large code length, no code in the ensemble of column-weight-three codes can correct a linear number of errors in the code length. The result was extended to the bit flipping algorithms also.

Tanner studied a class of codes constructed based on bipartite graphs and short error correcting codes [4]. Tanner's work is a generalization of the LDPC codes proposed by Gallager [1] and hence the name GLDPC codes. Tanner proposed code construction techniques, decoding algorithms and complexity and performance analysis for these codes and derived bounds on the rate and minimum distance. Sipser and Spielman [10] analyzed a special case of GLDPC codes (which they termed as expander codes) using expansion arguments and proposed explicit constructions of asymptotically good codes capable of correcting a linear number of errors. Zemor [14] improved the fraction of correctable errors under a modified decoding algorithm. Barg and Zemor in [15] analyzed the error exponents of expander codes and showed that expander codes achieve capacity over the BSC. Janwa and Lal [16] studied GLDPC codes in the most general setting by considering unbalanced bipartite graphs. Miladinovic and Fossorier [17] derived bounds on the guaranteed error correction capability of GLDPC codes for the special case of failures only decoding.

It is well known that a random graph is a good expander with high probability [10]. However, the fraction of nodes having the required expansion is very small and hence the code length to guarantee correction of a fixed number of errors must be large. Moreover, determining the expansion of a given graph is known to be NP hard [18], and spectral gap methods cannot guarantee an expansion factor of more than $1/2$ [10].

In spite of the aforementioned advances made in the analysis of ensembles of LDPC codes, there is clearly a lack of understanding in characterizing the failures of iterative decoders for a particular code. This can be attributed to the unique challenges posed by these decoders. Firstly, the decoders operate on a Tanner graph representation of the code which is not unique for a given code. What is the best Tanner graph representation for a code is a long standing open problem in coding theory. Secondly, there are many variants of the standard BP decoder and each decoder can have potentially different failures. Thirdly, the implementation of the decoder itself can play an important role in determining the failures. Moreover, the iterative nature of the decoders make it difficult to define a universal criterion for termination of the decoder and hence the notion of output of the decoder remains largely ambiguous. Remarkably, these difficulties were overcome in the case of iterative decoding on the binary erasure channel (BEC) for which the decoding failures are completely characterized in the work of Di *et al.* using combinatorial structures in the Tanner graph known as *stopping sets*. Later Orlitsky *et al.* [19] provided asymptotic results on the stopping set distribution for different code ensembles. The success in this case has motivated researchers to proceed along similar lines, wherein the effect of the topological structures in the Tanner graph on the failures are studied for a given decoding algorithm and a given channel.

One such approach to the study of error correction capability of LDPC codes is based on the girth of the underlying Tanner graph. While determining the expansion of a given graph is NP hard, the girth of a graph can be computed with a linear

complexity algorithm (in the number of vertices in the graph). This makes girth based analysis of codes attractive from a practical perspective. Tanner in [4] studied codes based on bipartite graphs and derived lower bounds on the minimum distance of graph based codes as a function of the left degree and girth of the Tanner graph. For $d_v \geq 3$, the minimum distance was shown to increase exponentially with the girth, which implies that for ML decoding the error correction capability increases exponentially with the girth. In [20], it was shown that the size of variable node sets that have the expansion required by the expansion based arguments also increases exponentially with girth for $d_v \geq 5$, leading to the result that the error correction capability under the bit flipping algorithms grows exponentially with the girth. In [21] it was shown that the error correction capability of column-weight-three codes under the Gallager A algorithm grows linearly with the girth of the Tanner graph and consequently logarithmically with the code length. It is worth noting here that the minimum stopping set [22] size also grows exponentially with the girth for $d_v \geq 3$ [23].

Failures of iterative decoders for graph based codes were first studied by Wiberg [24] who introduced the notions of *computation trees* and *pseudo-codewords*. Subsequent analysis of the computation trees was carried out by Frey *et al.* [25] and Forney *et al.* [26]. For iterative decoding on the AWGNC, MacKay and Postol [27] were the first to discover that certain "near codewords" are to be blamed for the high error floor in the Margulis code. A huge stride towards the understanding of the error floor problem in a general setting, was made through the pioneering work of Richardson [7], who showed that the problem is at least partially combinatorial in nature. Richardson introduced the notion of *trapping sets*, which are certain problematic loopy structures present in the graph of the code that cause the decoder to fail for certain low-noise configurations, and are a function of both the code and the decoding algorithm in operation. Using this notion, he proposed a semi-analytical method for estimating the error-rates of BP decoding on the AWGNC in the error floor by identifying the dominant trapping sets. Another approach based on statistical physics was also proposed for the AWGNC by Stepanov *et al.* [28] through the notion of *instantons*, which are certain low-noise configurations that swayed the decoding trajectory away from the correct codeword. Later, Chilappagari *et al.* [29] in line with Richardson's work, presented results on the error floor estimation for the BSC under Gallager B decoding. Another notion called *absorbing sets*, was proposed by Dolecek *et al.* [30], which are a special type of trapping sets that are purely combinatorial and stable under the bit-flipping algorithm, and this notion enabled them to perform a detailed analysis on array-based LDPC codes.

Although it was shown by McGregor and Milenkovic [31] that performing an exhaustive search of all trapping sets in the graph of a given code is NP-hard, several good practical approaches have been proposed. Milenkovic *et al.* [32] examined the asymptotic trapping set distributions for both regular and irregular LDPC code ensembles. Cole *et al.* [33] proposed a method to estimate the error floor based on importance sampling, which is similar to Richardson's approach in [7]. Wang *et al.* proposed an efficient algorithm to exhaustively enumerate both trapping sets and stopping sets for codes with relatively short to moderate block lengths ($N \approx 500$). Abu-Surra *et al.* [34] proposed an efficient improved impulse method that could enumerate trapping sets by augmenting the Tanner graph with auxiliary variable nodes, and treating the problem as if it were searching for low-weight codewords. Although the method does not guarantee enumeration of all trapping sets, it is reasonably reliable and is applicable to any arbitrary graph. A method that uses the *branch-and-bound* approach to enumerate stopping sets in a given code was proposed by Rosnes and Ytrehus [35]. Karimi and Bannihasemi [36] recently proposed an algorithm which could efficiently enumerate the dominant trapping sets present in any arbitrary graph by recursively expanding the cycles present in the graph. More recently, Zhang and Siegel [37] proposed an efficient technique that expanded on the branch-and-bound approach by transforming the bounding step to an LP formulation, and which allowed a complete enumeration of trapping sets up to a certain size with reasonable computation time.

In spite of the fact that all the above works are useful for enumerating trapping sets in a given graph which aid in estimating the error floor, none of them directly address the issue of how to utilize the knowledge of trapping sets for constructing better codes or improving the decoding algorithms. For instance, it is highly non-trivial to determine which particular subgraphs (that correspond to certain trapping sets) should be avoided during the construction of codes in order to improve the error floor performance. Vasić *et al.* [38] proposed the *trapping set onotology*, which is a hierarchy of trapping sets that exploits the topological relations, and can be utilized for code construction or decoder design. We discuss this in detail in this chapter.

It has been observed by numerous researchers that failures of various iterative decoders as well as linear programming (LP) decoder are due to noise configurations with similar topologies. However, no exact statement about such connection has been established nor proven. The failures of the LP decoder can be understood in terms of the vertices of the so-called *fundamental polytope* which are also known as pseudo-codewords [39]. Vontobel and Koetter [40] introduced a theoretical tool known as graph cover approach and used it to establish connections between the LP and the message passing decoders using the notion of the fundamental polytope. They showed that the pseudo-codewords arising from the Tanner graph covers are identical to the pseudo-codewords of the LP decoder. Vontobel and Koetter [41] also studied the relation between the LP and the min-sum decoders.

Kelley and Sridhara [42] studied pseudo-codewords arising from graph covers and derived bounds on the minimum pseudo-codeword weight in terms of the girth and the minimum left-degree of the underlying Tanner graph. The bounds were further investigated by Xia and Fu [43]. Smarandache and Vontobel [44] found pseudo-codeword distributions for the special cases of codes from Euclidean and projective planes. Pseudo-codeword analysis has also been extended to the convolutional LDPC codes by Smarandache *et al.* [45]. Milenkovic *et al.* [32] studied the asymptotic distribution of trapping sets in regular and

irregular ensembles. Wang *et al.* [46] proposed an algorithm to exhaustively enumerate certain trapping sets.

II. Preliminaries

A. LDPC Codes

LDPC codes belong to the class of linear block codes which can be defined by sparse bipartite graphs [4]. The Tanner graph [4] G of an LDPC code $\mathcal{C}$ is a bipartite graph with two sets of nodes: the set of variable nodes $V = \{1, 2, \ldots, n\}$ and the set of check nodes $C = \{1, 2, \ldots, m\}$. The check nodes (variable nodes resp.) connected to a variable node (check node resp.) are referred to as its neighbors. The set of neighbors of a node u is denoted by $\mathcal{N}(u)$. The degree d_u of a node u is the number of its neighbors. A vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ is a codeword if and only if for each check node, the modulo two sum of its neighbors is zero. An (n, γ, ρ) regular LDPC code has a Tanner graph with n variable nodes each of degree γ and $n\gamma/\rho$ check nodes each of degree ρ. This code has length n and rate $r \geq 1 - \gamma/\rho$ [4]. It should be noted that the Tanner graph is not uniquely defined by the code and when we say the Tanner graph of an LDPC code, we only mean one possible graphical representation.

B. Channel Assumptions

We assume that a binary codeword $\mathbf{y}$ is transmitted over a noisy channel and is received as $\hat{\mathbf{y}}$. The support of a vector $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, denoted by $\text{supp}(\mathbf{y})$, is defined as the set of all positions i such that $y_i \neq 0$. In this paper, we consider binary input memoryless channels with discrete or continuous output alphabet. Since, the channel is memoryless, we have

$$\Pr(\hat{\mathbf{y}}|\mathbf{y}) = \prod_{i \in V} \Pr(\hat{y}_i|y_i)$$

and hence can be characterized by $\Pr(\hat{y}_i|y_i)$, the probability that $\hat{y}_i$ is received given that y_i was sent. The negative log-likelihood ratio (LLR) corresponding to the variable node $i \in V$ is given by

$$\gamma_i = \log\left(\frac{\Pr(\hat{y}_i|y_i = 0)}{\Pr(\hat{y}_i|y_i = 1)}\right).$$

Two binary input memoryless channels of interest are the BSC with output alphabet $\{0, 1\}$ and the AWGNC with output alphabet $\mathbb{R}$. On the BSC with transition probability ϵ, every transmitted bit $y_i \in \{0, 1\}$ is flipped [1] with probability ϵ and is received as $\hat{y}_i \in \{0, 1\}$. Hence, we have

$$\gamma_i = \begin{cases} \log\left(\frac{1-\epsilon}{\epsilon}\right) & \text{if } \hat{y}_i = 0 \\ \log\left(\frac{\epsilon}{1-\epsilon}\right) & \text{if } \hat{y}_i = 1 \end{cases}$$

For the AWGN channel, we assume that each bit $y_i \in \{0, 1\}$ is modulated using binary phase shift keying (BPSK) and transmitted as $\overline{y}_i = 1 - 2y_i$ and is received as $\hat{y}_i = \overline{y}_i + n_i$, where $\{n_i\}$ are i.i.d. $N(0, \sigma^2)$ random variables. Hence, we have

$$\gamma_i = \frac{2\hat{y}_i}{\sigma^2}.$$

C. Message Passing Decoding Algorithms

Message passing decoders operate by passing messages along the edges of the Tanner graph representation of the code. Gallager in [1] proposed two simple binary message passing algorithms for decoding over the BSC; Gallager A and Gallager B. There exist a large number of message passing algorithms (sum-product algorithm, min-sum algorithm, quantized decoding algorithms, decoders with erasures to name a few [47]) in which the messages belong to a larger alphabet.

Let $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)$, an n-tuple be the input to the decoder. Let $\omega_{i\to\alpha}^{(k)}$ denote the message passed by a variable node $i \in V$ to its neighboring check node $\alpha \in C$ in the k^{th} iteration and $\varpi_{\alpha\to i}^{(k)}$ denote the message passed by a check node α to its neighboring variable node i. Additionally, let $\varpi_{*\to i}^{(k)}$ denote the set of all incoming messages to variable node i and $\varpi_{*\setminus\alpha\to i}^{(k)}$ denote the set of all incoming messages to variable node i except from check node α.

[1] The event of a bit changing from 0 to 1 and vice-versa is known as flipping.

Gallager A/B Algorithm:: The Gallager A/B algorithms are hard decision decoding algorithms in which all the messages are binary. With a slight abuse of the notation, let $|\varpi_{*\to i} = m|$ denote the number of incoming messages to i which are equal to $m \in \{0, 1\}$. Associated with every decoding round k and variable degree d_i is a threshold b_{k,d_i}. The Gallager B algorithm is defined as follows.

$$\begin{aligned}
\omega_{i\to\alpha}^{(0)} &= \hat{y}_i \\
\varpi_{\alpha\to i}^{(k)} &= \left(\sum_{j\in\mathcal{N}(\alpha)\setminus i} \omega_{j\to\alpha}^{(k-1)} \right) \bmod 2 \\
\omega_{i\to\alpha}^{(k)} &= \begin{cases} 1, & \text{if } |\varpi_{*\setminus\alpha\to i}^{(k)} = 1| \geq b_{k,d_i} \\ 0, & \text{if } |\varpi_{*\setminus\alpha\to i}^{(k)} = 0| \geq b_{k,d_i} \\ \hat{y}_i, & \text{otherwise} \end{cases}
\end{aligned}$$

The Gallager A algorithm is a special case of the Gallager B algorithm with $b_{k,d_i} = d_i - 1$ for all k. At the end of each iteration, a decision on the value of each variable node is made based on all the incoming messages and possibly the received value.

The Sum-Product Algorithm:: A soft decision decoding algorithm, which is the best possible one if the messages are calculated locally in the Tanner graph of the code, is called the sum-product algorithm. With a moderate abuse of notation, the messages passed in the sum-product algorithm are described below:

$$\begin{aligned}
\omega_{i\to\alpha}^{(0)} &= \gamma_i \\
\varpi_{\alpha\to i}^{(k)} &= 2\tanh^{-1}\left(\prod_{j\in\mathcal{N}(\alpha)\setminus i} \tanh\left(\frac{1}{2}\omega_{j\to\alpha}^{(k-1)} \right) \right) \\
\omega_{i\to\alpha}^{(k)} &= \gamma_i + \sum_{\delta\in\mathcal{N}(i)\setminus\alpha} \varpi_{\delta\to i}^{(k)}
\end{aligned}$$

The result of decoding after k iterations, denoted by $\mathbf{x}^{(k)}$, is determined by the sign of $m_i^{(k)} = \gamma_i + \sum_{\alpha\in\mathcal{N}(i)} \varpi_{\alpha\to i}^{(k)}$. If $m_i^{(k)} > 0$ then $x_i^{(k)} = 0$, otherwise $x_i^{(k)} = 1$.

In the limit of high SNR, when the absolute value of the messages is large, the sum-product becomes the min-sum algorithm, where the message from the check α to the bit i looks like:

$$\varpi_{\alpha\to i}^{(k)} = \min\left|\omega_{*\setminus i\to\alpha}^{(k-1)}\right| \cdot \prod_{j\in\mathcal{N}(\alpha)\setminus i} \operatorname{sign}\left(\omega_{j\to\alpha}^{(k-1)}\right)$$

The min-sum algorithm has a property that the Gallager A/B and LP decoders also possess — if we multiply all the likelihoods γ_i by a factor, all the decoding would proceed as before and would produce the same result. Note that we do not have this "scaling" in the sum-product algorithm.

To decode the message in complicated cases (when the message distortion is large) we may need a large number of iterations, although typically a few iterations would be sufficient. To speed up the decoding process one may check after each iterations whether the output of the decoder is a valid codeword, and if so halt decoding.

III. Overview of Decoding Failures

To characterize the performance of a coding/decoding scheme for linear codes over any output symmetric channel, one can assume, without loss of generality, the transmission of the all-zero-codeword, i.e. $\mathbf{x} = \mathbf{0}$, when the decoding algorithm satisfies certain symmetry conditions (see Definition 1 and Lemma 1 in [47]). The iterative decoding algorithms that we consider in this paper satisfy these symmetry conditions. Henceforth, we assume that that $\mathbf{x} = \mathbf{0}$.

If the output of the decoder is not equal to the transmitted codeword then we say that a decoding failure had occurred. Since the all-zero-codeword is transmitted, a decoding failure occurs if the output vector from the decoder contains at least a 1-bit. For channels with discrete-output, we express the probability of a decoding failure, i.e., the frame error rate as a function of the SNR s, as follows:

$$FER(s) = \sum_{\mathbf{y}} P_s(\mathbf{y})\theta(\mathbf{y}), \tag{1}$$

where the sum goes over all the possible outputs $\mathbf{y}$ of the channel for the all-zero-codeword input. If the symbols from the channel take values from a continuous range, we replace the sum with an integral: $\sum \to \int d\mathbf{y}$. As a result, the probability mass function $P_s(\mathbf{y})$ becomes a probability density function $f_s(\mathbf{y})$. The function $\theta(\mathbf{y})$ in Eq. (1) is an indicator function, i.e.,

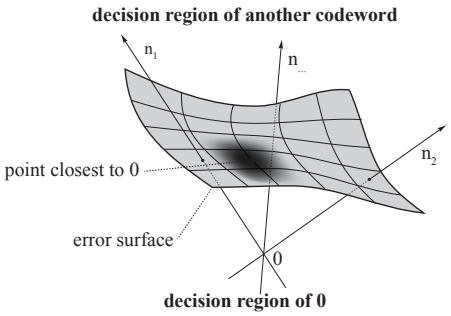

Fig. 1. Illustration of error surface.

it is defined to be zero in the case of successful decoding, and is unity in the case of failure. Also, $P_s(\mathbf{y})$ is the probability of observing $\mathbf{y}$ as the output of a discrete-output channel characterized by the SNR s.

The above sum/integral cannot be calculated exact and hence it is typically approximated by retaining only a finite number of terms corresponding to the most probable failures. Approximation with a finite number of terms becomes asymptotically exact in the limit of large SNR, while at smaller SNRs, the accuracy of the estimation depends on the number of terms being considered. The details of the approximate evaluations, however, are different for discrete-output and continuous-output channels, as we now discuss following the spirit of [48].

For discrete-output channels, there are finite number of terms and we account for k-most probable channel output vectors $\mathbf{y}$. We write $FER(s) \approx \sum_{\beta=1}^{k} \mathcal{N}_\beta P_s(\mathbf{y}_\beta)\theta(\mathbf{y})$, where the multiplicity factor $\mathcal{N}_\beta$ counts the number of vectors $\mathbf{y}$ that are equivalent under bit permutations.

For continuous-output channels, there are infinite number of terms. However, the dominant terms are those that correspond to channel output vectors $\mathbf{y}$ at which the noise probability density function has local maxima. We call an output vector $\mathbf{y}$ at which the noise probability density function achieves local maximum a stationary point. Only stationary points are included in the estimation. For example, on the AWGNC, a stationary point is a noise configuration with minimal (probably locally) value of the L^2 norm of $\mathbf{y}$. Note that the L^2 norm of a vector $\mathbf{y}$ is equal to $\sqrt{\sum_{i\in V} y_i^2}$ and that for the AWGNC, a noise configuration with smaller the L^2 norm is more probable. A stationary point that leads to a decoding failure is called an instanton.

The FER approximation for continuous-output channels should also include, in addition to the multiplicities, the curvature corrections around the stationary point as discussed in [49], [50]. In other words, $FER(s) \approx \sum_{\beta=1}^{k} \mathcal{N}_\beta \mathcal{C}(\mathbf{y}_\beta) P(\mathbf{y}_\beta)\theta(\mathbf{y})$, where $\mathcal{C}(\mathbf{y}_\beta)$ is the curvature factor. The multiplicities of the dominant noise configurations are determined by the automorphisms of the code, and if nothing is known about it, one may assume that all multiplicities are equal to 1. The FER is affected by the curvature of the error surface in the vicinity of the dominant noise configuration. We note that the most important information about a noise configuration that leads to a decoding failure (an instanton) is its weight which determines the slope of the FER *vs.* SNR curve in the asymptotic. As observed in [48], in the case of the AWGNC and $s \to \infty$, $\mathcal{C}(\hat{\boldsymbol{y}}_\beta) = O(1/\sqrt{s})$, the decay of the noise correlations is exponential along a direction orthogonal to the error surface and quadratic along the remaining $N-1$ components of the noise vector (see Fig. 1 for an illustration of the error surface).

For any typical finite-length LDPC code under iterative decoding, the FER *vs.* SNR curve consists of two distinct regions: the waterfall region, and the error floor region. In the waterfall region (which is the low SNR region), the error-rate drops significantly with increase in SNR making the curve look like a waterfall. On the other hand, in the error floor region (which is the high SNR region), the decrease in the error-rate dramatically slows down and the curve tends to flatten out turning into an error floor. Fig. 2 illustrates the two regions on a typical FER curve of an LDPC code plotted as a function of the cross-over probability α of the BSC.

To better understand the error floor phenomenon, we consider as an example, the estimation of the FER on the BSC. Let t be the guaranteed error correction capability of an (N, K) LDPC code $\mathcal{C}$. This means that if the vector output from the channel $\mathbf{y}$ differs from the transmitted codeword by at most t positions then the decoding is guaranteed to be successful. On the other hand, there exists a vector $\mathbf{y}$ with distance $t+1$ from the transmitted codeword that leads to a decoding failure. Let n_i be the number of received vectors of weight i, that lead to a decoding failure. Then the frame error rate $FER(\alpha)$ can be determined as

$$FER(\alpha) = \sum_{i=t+1}^{N} n_i(\alpha)^i (1-\alpha)^{(N-i)}.$$

Let α be the transition probability of BSC and c_k be number of configurations of received bits for which k channel errors lead to codeword (frame) error. The frame error rate (FER) is given by:

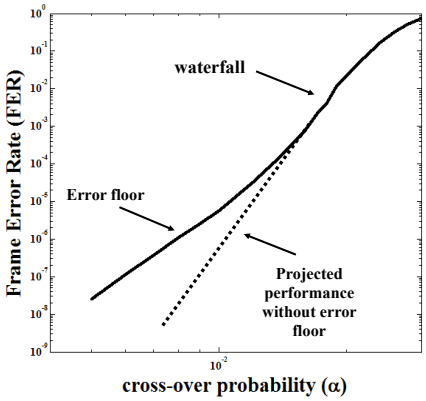

Fig. 2. Typical performance of BP decoding on an LDPC code over the BSC

$$FER(\alpha) = \sum_{k=i}^{n} c_k \alpha^k (1-\alpha)^{(n-k)}$$

where i is the minimal number of channel errors that can lead to a decoding error (size of instantons) and n is length of the code.

On a semilog scale, the FER is given by

$$\begin{aligned} \log(FER(\alpha)) &= \log\left(\sum_{i=t+1}^{N} n_i(\alpha)^i (1-\alpha)^{(N-i)}\right) \\ &= \log(n_{t+1}) + (t+1)\log(\alpha) + \log\left((1-\alpha)^{(N-(t+1))}\right) \\ &+ \log\left(1 + \frac{n_{t+2}}{n_{t+1}}\alpha(1-\alpha)^{-1} + \ldots + \frac{n_N}{n_{t+1}}\alpha^{(N-(t+1))}(1-\alpha)^{-(t+1)}\right) \end{aligned}$$

For a small α, the expression is dominated by the first two terms thereby reducing to

$$\log(FER(\alpha)) \approx \log(n_{t+1}) + (t+1)\log(\alpha)$$

From the above expression, we can see that on a $\log(FER)$ vs $\log(\alpha)$ scale, the slope of the error floor (which occurs at small α) is governed by the guaranteed error correction capability t.

The guaranteed error correction capability of a code has always been an important parameter especially for applications such as magnetic recording, and optical and solid-state storage. Recall that under maximum-likelihood decoding, a linear block code with minimum distance $d_{\min}$ can achieve a guaranteed error correction of $\lfloor (d_{\min}-1)/2 \rfloor$. However, for an LDPC code under iterative decoding, its guaranteed error correction is highly non-trivial to determine since iterative decoding is suboptimal and the guaranteed error correction cannot be simply derived from the minimum distance of the code. This highlights the main difference between classical error correcting codes (such as BCH and Reed-Solomon codes) and LDPC codes. While the former emphasizes solely on maximizing the minimum distance, the latter requires, in addition to having good minimum distances, analysis of failures of particular decoding algorithms on low-weight error configurations.

As previously mentioned, the error floor problem arises due to the suboptimality of iterative decoding on loopy graphs. It can be troublesome for applications requiring very low target error-rates, and especially when high-rate codes are employed, which have relatively dense graphs. Although asymptotic techniques such as density evolution [47] provide an accurate prediction of the FER performance in the early waterfall region, the point at which the error floor starts as well as its slope are greatly dependent on the particular structure of a code, and hence cannot be predicted by density evolution. Therefore, finite-length analysis techniques are required for the study of error floors. Besides, for codes with moderate to large lengths, the ability to predict error floors becomes even more critical as the error floor may be unreachable by Monte-Carlo simulations. In the next section, we present a finite length analysis methodology which relies on the notion of trapping sets.

IV. Combinatorial Characterization of Decoding Failures

It is evident from previous discussions that the main element in the analysis and estimation of error floor is the determination of smallest-weight error patterns that lead to decoding failures. In this section, we focus on the combinatorial characterization of decoding failures on various channels utilizing the notion of trapping sets.

In practice, we assume that the iterative decoder performs a finite number D of iterations. To define the notion of trapping sets, we shall assume that D is as large as neccessary. 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 l^{th} iteration. We say that a variable node v is *eventually correct* if there exists a positive integer l_{c} such that for all l with $l_{\text{c}} \leq l$, $\hat{x}_v^l = x_v$. Then, trapping sets are defined as follows.

Definition 1 ([7]): A trapping set for an iterative decoding algorithm is a non-empty set of variable nodes in a Tanner graph G that are *not* eventually correct. Such a set of variable nodes, say $\mathbf{T}$, 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.

One can readily see that the above definition of trapping sets does not specify which property of a set of variable nodes to form a trapping set. This is because such a property, if exists, would depend on a particular channel and decoding algorithm. In the subsequent subsections, we discuss the notion of trapping sets for a few common channels.

A. Trapping Sets on the BEC

Trapping sets on the BEC are well-characterized through the work of Di et al. [22]. They are purely combinatorial objects and are called stopping sets. The condition on a set of variable nodes to form a stopping set is stated as follows.

Definition 2: A stopping set $\mathbf{S}$ is a subset of V, the set of variable nodes, such that all neighbors of $\mathbf{S}$ are connected to $\mathbf{S}$ at least twice.

B. Trapping Sets on the BSC

For the BSC, since the input to the decoder is discrete, it is easier to define noise (error) configurations in terms of number of bits flipped in the input. The configurations with least number of flips will be the most dominant in the error floor region.

Although a complete combinatorial characterization of trapping sets on the BSC has not been found, 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 [13], [20], [29], [51]. 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.

Assume the transmission of the all-zero codeword over the BSC. The all-zero-codeword assumption can be applied if the channel is output symmetric and the decoding algorithm satisfies certain symmetry conditions (see Definition 1 and Lemma 1 in [47]). The Gallager A/B algorithm, the bit flipping algorithms and the SPA all satisfy these symmetry conditions. 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 $\mathbf{F}(\mathbf{y})$ denote the set of variable nodes that are not eventually correct.

Definition 3: For transmission over the BSC, $\mathbf{y}$ is a fixed point of the decoding algorithm if $\text{supp}(\mathbf{y}) = \text{supp}(\hat{\mathbf{x}}^l)$ for all l. If $\mathbf{F}(\mathbf{y}) \neq \emptyset$ and $\mathbf{y}$ is a fixed point, then $\mathbf{F}(\mathbf{y}) = \text{supp}(\mathbf{y})$ is 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 non-elementary fixed set.

Remark: The classification of fixed sets as elementary and non-elementary fixed sets is identical to the classification of trapping sets as elementary and non-elementary trapping sets in [32].

Theorem 1 ([20]): 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 : (a) Every variable node in $\mathbf{I}$ has at least $\lceil \frac{d_{\text{v}}}{2} \rceil$ neighbors in $\mathbf{E}$ and (b) No collection of $\lfloor \frac{d_{\text{v}}}{2} \rfloor + 1$ 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 [52].

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 4 ([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], [52], [53].

C. Trapping Sets on the AWGNC

Compared to the BSC, the understanding of trapping sets on the AWGNC is much less completed. Very often, the analysis of decoding failures of decoding algorithms on the AWGNC

Although it has been rigorously proven only for the Gallager A/B algorithm and the bit flipping algorithms that fixed sets are trapping sets, it has been widely recognized in the literature that the subgraphs of these combinatorial objects greatly

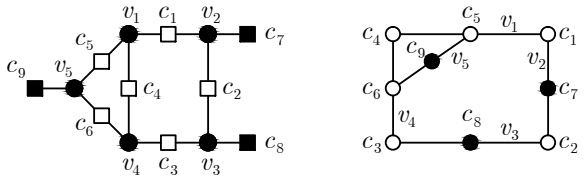

Fig. 3. Graphical representation of the $(5,3)\{2\}$ trapping set: (a) Tanner graph representation, (b) Line-point representation.

contribute to the error floor for various iterative decoding algorithms and channels. The instanton analysis performed in [48] 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 [30], the notion of absorbing sets was defined. These are sets of variable nodes which satisfy condition (a) of Theorem 1. These authors also defined fully absorbing sets, which are combinatorially identical to fixed sets. 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 [27], [54]) are also fixed sets.

V. Case Study: Column-Weight-Three Codes with the Gallager A/B Algorithm on the BSC

In the remaining of the paper we use the following definition of a trapping set.

Definition 5: Trapping sets are fixed sets of the Gallager A/B algorithm.

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

A. Graphical representation

The induced subgraph of a trapping set (or any set of variable nodes) is a bipartite graph. We use a graphical representation 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 $\mathscr{I}$ are called flags. If $(p, \ell) \in \mathscr{I}$, we say that point p "lies on" line 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 non-isomorphic 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 set of odd-degree check nodes is $\{c_7, c_8, c_9\}$. In the line-point representation of $\mathcal{T}_1$ which is shown in Fig. 3(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. 3(a). They correspond to the five lines in Fig. 3(b).

B. Topological relation

The following definition gives the topological relations among trapping sets.

Definition 6: 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 $\mathbf{T}_2$.

C. Evolution of trapping sets

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 $\circledast$ 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 $\circledast$ point then they become a single white shaded point. Similarly, if a white shaded point is merged with a $\circledast$ point then the result is a single black shaded point. Before merging points, one must decide on: (i) the number of the $\circledast$ points on the new line and (ii) which points on the line-point representation of the predecessor trapping set are to be merged with the $\circledast$ points.

Because the merging must ensure that every line passes through at least two white shaded points, there must be at least two $\circledast$ points on the line to be merged, i.e., there can be two or three $\circledast$ 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 $\circledast$ points then there are two distinct ways to select two black shaded points. The merging, which is demonstrated in Fig. 4(a) and (b), results in two different $(5, 3)$ trapping sets. On the other hand, if there are three $\circledast$ points then there is only one distinct way to select three black shaded points. The merging, which is demonstrated in Fig. 4(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. 4(d). Note that the $(4, 2)$ trapping set is neither a successor nor a predecessor of the $(4, 4)$ trapping set.

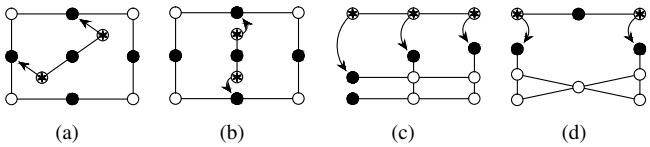

Fig. 4. $(5, b)$ trapping sets can be obtained by adding a line to the $(4, 4)$ trapping set: (a) the $(5, 3)\{1\}$ trapping set, (b) the $(5, 3)\{2\}$ trapping set and (c) the $(5, 1)$ trapping set; or by adding a line to the $(4, 2)$ trapping set: (d) the $(5, 1)$ trapping set.

With the evolution of the $(5, 3)\{2\}$ trapping set from the $(4, 4)$ trapping set presented above, 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. 5.

Fig. 6 illustrates a hierarchical relationship among trapping sets originating from the $(4, 4)$ trapping set. For simplicity, we assume that codes have girth $g = 8$ in all examples.

The web page [55] contains a complete description of trapping setts for column weight three codes up to eight variable nodes, including their topological structure and parent-child relationships. As an example, Table I shows the number of topologically different (a, b) trapping set for different girths, g.

D. FER Estimation

Chilappagari *et al.* [29] gave a method for FER estimation which consists of the following three main steps.

1) Identifying the relevant classes of trapping sets
2) Calculating the number of trapping sets of each class
3) Calculating the contribution of each class of trapping set

In the case of column weight three codes and Gallager A algorithm, the class of relevant trapping sets can be obtained by the analysis of messages passed inside and outside a graph. The number of different trapping sets can be enumerated using

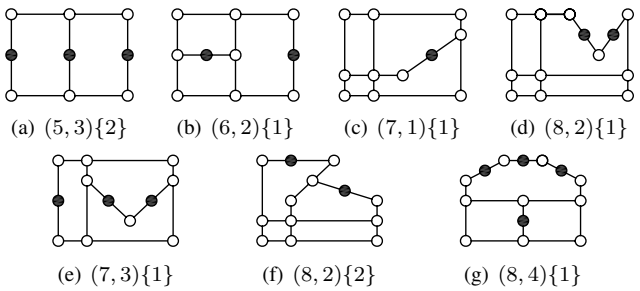

Fig. 5. The $(5, 3)\{1\}$ trapping set and its successors of size less than or equal to 8 in girth-8 LDPC codes.

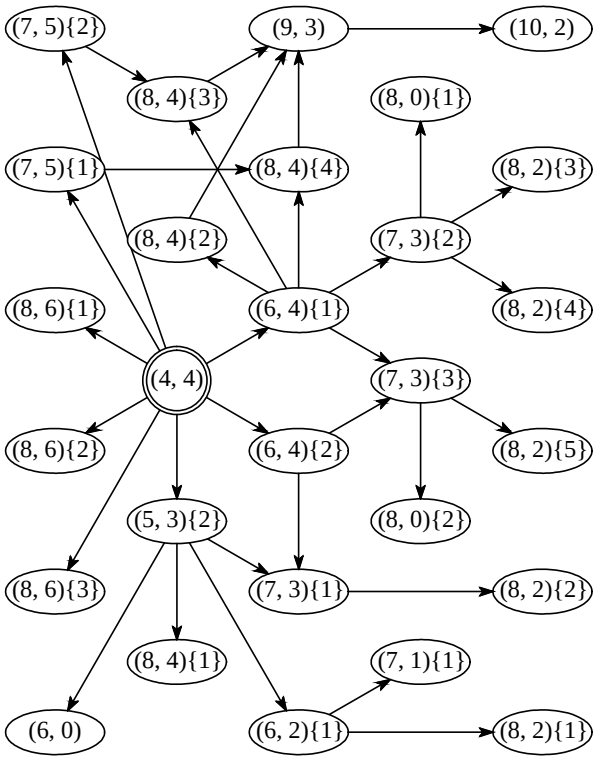

Fig. 6. Hierarchy of trapping sets of interest originating from the $(4, 4)$ trapping set for regular column-weight-three codes of girth 8.

TABLE I
NUMBER OF TRAPPING SETS

(a, b) TS	#TS	$g = 6$	$g = 8$	$g = 10$	$g = 12$	$g = 14$	$g = 16$
(3,3)	1	1					
(4,4)	1		1				
(4,2)	1	1					
(4,0)	1	1					
(5,5)	1			1			
(5,3)	2	1	1				
(5,1)	1	1					
(6,6)	1				1		
(6,4)	4	2	2				
(6,2)	4	3	1				
(6,0)	2	1	1				
(7,7)	1					1	
(7,5)	6	3	3	1			
(7,3)	10	7	3				
(7,1)	4	3	1				
(8,8)	1						1
(8,6)	10	4	3	2	1		
(8,4)	25	15	9	1			
(8,2)	19	14	5				
(8,0)	5	3	2				

their topological relations. The work by Nguyen et al. [56] gives a detailed procedure for this. The contribution of a class of trapping sets $\mathcal{X}$, $\Pr\{\mathcal{X}\}$, to the FER is calculated by:

$$\Pr\{\mathcal{X}\} = \sum_{r=m}^{M} \Pr\{\mathcal{X}|\ r\text{-errors}\}\Pr\{r\text{-errors}\} \tag{2}$$

$$\Pr\{\mathcal{X}|\ r\text{-errors}\} = \frac{|\mathcal{X}|\binom{r}{m}}{\binom{n}{m}} \tag{3}$$

$$\Pr\{r \text{ errors}\} = \binom{n}{r}\alpha^r(1-\alpha)^{n-r} \tag{4}$$

$\Pr\{\mathcal{X}|r\text{-errors}\}$ is the probability that the decoder ends in a trapping set of the class $\mathcal{X}$, given that the channel introduced r errors. We assume that if m of the r errors belong to a trapping set, the decoder ends in that trapping set (the justification of this assumption is discussed in detail in Section IV). $|\mathcal{X}| / \binom{n}{m}$ is the probability that a given set of m variable nodes is part of a trapping set of class $\mathcal{X}$. Hence, $|\mathcal{X}| \binom{r}{m} / \binom{n}{m}$ is the probability that m of the r errors belong to a trapping set. $\Pr\{r\text{-errors}\}$ is the probability that the channel introduced r errors and is calculated using binomial expansion as in (4).

M is the maximum number of errors which can end up in a trapping set. If more than M errors are introduced by the channel, the decoder still fails to correct these errors but in this case the decoder does not end in a trapping set. For a given α, the number of errors introduced by the channel will be binomially distributed with mean $n\alpha$. Hence most of the error vectors have weights centered around $n\alpha$. For the values of α for which $n\alpha$ is much less than M, most of the error events will be due to trapping sets. For values of α such that $n\alpha$ is comparable to M, the FER is worse but the contribution of trapping set errors is smaller. In other words, M determines the value of α at which trapping set errors start dominating. The calculated FER approaches the real value for lower and lower values of α. Determining M is non trivial, and is found by a combination of heuristics, simulations and code properties like minimum distance.

VI. Combating Error Floors

From the previous sections, it is clear that the error floor problem arises due to the suboptimality of iterative decoding on loopy graphs. For this reason, to alleviate the effect of error floor, it is natural to consider two approaches:

1) Find a Tanner graph on which a given decoding algorithm is least susceptible to (error floor) failures.
2) Based on error floor analysis on a given Tanner graph, adjust a given decoding algorithm/decoder to achieve better error floor performance.

In fact, for optimal performance, one should attempt to find a pair of code and decoding algorithm that yields the best performance. However, as we will discuss later, the adjustments of codes and decoding algorithms are subject to certain constraints such as code length, code rate and implementation complexity. Other directions have been explored in the literature such as using multiple Tanner graph representations of a code and postprocessing. Therefore, we divide the discussion in this section into three main parts. We first discuss methods to construct good codes, or equivalently good Tanner graphs, then we discuss methods to improve iterative decoding, and finally present some novel strategies to improve error floor performance by post processing.

A. Improving Error Floor Performance by Constructing Better Tanner Graphs

Constructing Better Tanner Graphs by Increasing Girth: Since belief propagation provides optimal decoding on cycle-free Tanner graphs, the most natural way to construct a good Tanner graph with cycles is to make it look more like a cycle-free one. This can be done by increasing the girth of such a graph. One can also justify increasing girth be the following 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 $d_{\mathrm{v}} \geq 3$ [4]. Second, trapping sets containing shortest cycles in the Tanner graph are eliminated when the girth is increased. In addition, the following results can also 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 $d_{\mathrm{v}} \geq 5$ [20]; and the lower bound on the minimum BSC pseudo-codeword weight for linear programming decoding was also shown to increase exponentially with the girth [42]. Notably, this lower bound on the minimum BSC pseudo-codeword weight of an LDPC code whose Tanner graph has girth greater than 4 was proven to be tight if and only if the minimum pseudo-codeword is a real multiple of a codeword [43]. 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 $d_{\mathrm{v}} \geq 3$ [23].

With numerous existing approaches to construct a Tanner graph with large girth, we can only discuss a few. The most well-known technique to construct large girth Tanner graphs is the progressive edge growth (PEG) construction, proposed by Hu *et al.* [57]. In the PEG construction, one starts with a set of n variable nodes and a set of m check nodes, initially unconnected. Then, edges are introduced progressively to connect variable nodes and check nodes. The edge-selection procedure is carried out in a manner such that the placement of a new edge on the graph has as small an impact on the girth as possible, i.e., it optimizes the local girths of variable nodes. The fundamental idea of the PEG construction is to find the most distant check node and then to place a new edge connecting the variable node and this most distant check node [57]. The PEG algorithm is simple and flexible in the sense that it can be used to construct codes of any given length, rate and degree sequence. Several improved versions of the PEG algorithms have been proposed [58], [59]. Most notably, a new metric, called extrinsic message degree (EMD) was introduced in [58] to measure cycle connectivity in bipartite graphs of LDPC codes. This metric enables treating short cycles differently based on their harmfulness. In particular, while the PEG construction only optimizes local girths, the PEG construction with EMD allows some short cycles with good graph connectivity, while eliminates longer cycles with poor graph connectivity. This technique lowers the error floors of irregular LDPC codes significantly, while only slightly degrading the waterfall-region performance [58].

One drawback of the PEG construction is that it only gives random LDPC codes, which are not suitable for practical applications due to implementation complexity. Nevertheless, the Tanner graph of a structured LDPC codes can also be

constructed progressively to archive high girth as shown in [60]. The difference between progressively building a Tanner graph of a structured code and a random code is that in the former, a set of edges are added at a time. In [60], a line-point represenation of a Tanner graph is used, the code construction involves progressively adding blocks of parallel lines while maintaining desired girt. Of course, this comes with certain limitations on the code length, the degree sequence and possibly the code rate.

It is necessary to mention another line of approaches to construct high girth structured LDPC codes [61]–[65]. The common goal of these approaches is to derive constraints on the algebraic description of the parity-check matrix so that the girth of the corresponding Tanner graph is greater than a certain minimum value.

Although increasing girth is a valid approach to construct better Tanner graph, this method is subject to strict constraints. 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 [51]. 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. In the next subsection, we discuss such an approach.

Constructing Better Tanner Graphs by Avoiding Trapping Sets: 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. We give details on how to select such subgraphs at a later point. Once the set of forbidding subgraphs is determined, the remaining task is to construct a Tanner graph free of these subgraphs. In general, there are two approaches to construct such a Tanner graph.

In the first approach, which is only applicable for structured codes, a subgraph is described by a system of linear equations. Recall that the parity-check matrix H of a structured code usually corresponds to a matrix over an algebraic structure, say $\mathcal{W}$. Elements of $\mathcal{W}$ are particular values of variables of the systems of equations that describe a subgrap. The Tanner graph corresponding to $\mathcal{W}$ 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 [61] and [30]. 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 [66], 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.

The second approach is similar to the PEG algorithm in the sense that a Tanner graph is constructed progressively. However, unlike the PEG algorithm which constructs a Tanner graph with a given set of parameters, in the progressive construction of Tanner graphs free of trapping sets, the length is usually not pre-specified. The construction is based on a check and select-or-disregard procedure. For example, in the progressive construction of structured regular LDPC codes described in [56], the Tanner graph of the code is built in ρ stages, where ρ is the row weight of $\mathcal{H}$. ρ is not pre-specified, and a code is constructed with the goal of making the rate as high as possible. At each stage, a set of 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. After a block of edges is tentatively added, the Tanner graph is checked to see if it contains any undesired trapping set. If it does, 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 introducing undesired trapping sets to the Tanner graph.

When constructing a Tanner graph by progressively adding variable nodes, the under-constructing Tanner graph is checked see if it contains certain trapping sets. This can only be done by exhaustively search for trapping sets. It is well-known that the problem of searching for trapping set is NP hard [31], [67]. Previous work on this problem includes exhaustive [46], [68] and non-exhaustive approaches [34], [69]. Exhaustive approaches usually come with high complexity. While non-exhaustive approaches require much lower complexity, they are not suitable for the purpose of constructing codes. In section V, we describe a database of trapping sets (the TSO) for column-weight-three LDPC codes under the Gallager A/B algorithm. We shall now briefly describe a method to search for these trapping sets. An efficient search of the Tanner graph for these trapping sets relies on the topological relations 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. It is easy to show that 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 $\mathcal{T}_1$. This techniques are sufficient to efficiently search for a large number of trapping sets in the TSO.

Let us now give a general rationale for deciding which trapping sets should be forbidden in the Tanner graph of a code. To facilitate the discussion, we assume that the forbidden trapping sets are drawn 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 trade-off between the rate and the choice of forbidden trapping sets is also a trade-off between the rate and the error floor performance. While an explicit formulation of this trade-off 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 [51]. 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 are determined in this manner.

Remarks: According to the above 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 3: 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 trapping sets as sets of variable nodes that are not eventually correct. To avoid confusion, we refer to them as erroneous sets. 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 non-eventually correct variable nodes are not the most harmful ones. Although avoiding subgraphs induced by sets of non-eventually 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 [29], [51], [70], [71]. 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 [27] 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 [7] 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.* [33] further developed the importance sampling based method to analyze error floors of moderate-length LDPC codes, while instantons were used to predict error floors [72]–[74].

We end the discussion on code construction techniques by giving the following examples.

Example 4: Consider the $(155, 64)$ Tanner code [75]. This code is a $(3,5)$-regular LDPC code. Its Tanner graph contains $(5,3)\{2\}$ trapping sets. Since the critical number of $(5,3)\{2\}$ trapping sets is three, the code cannot correct three errors under the Gallager A/B algorithm on the BSC. We used the construction technique described above to construct a different code with the same length, column weight and row weight. Let $\mathcal{C}_1$ denote this code. It is a $(155, 64)$ LDPC code with girth $g = 8$ and minimum distance $d_{\min} = 12$. The Tanner graph of $\mathcal{C}_1$ contains no $(5,3)\{2\}$ trapping sets. Therefore, $\mathcal{C}_1$ is capable of correcting any three-error pattern under the Gallager A/B algorithm on the BSC. The FER performance of $\mathcal{C}_1$ under the Gallager A/B algorithm for a maximum of 100 iterations is shown in Fig. 7. The FER performance of the Tanner code is also shown for comparison.

Example 5: Fig. 8 shows the FER performance of a $(3150, 2520)$ regular QC LDPC code constructed using array masking as proposed in [76] in comparison with another code, denoted as $\mathcal{C}_5$. The Tanner graph of $\mathcal{C}_5$ has girth $g = 8$ and does not contain $(5,3)\{2\}$ and $(8,2)$ trapping sets. It can be seen that by avoiding certain trapping sets in the Tanner graph of $\mathcal{C}_5$, we can achieve a significant lower error floor.

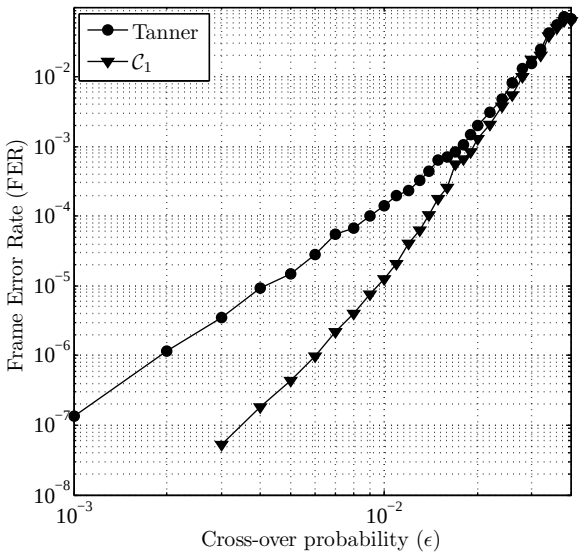

Fig. 7. Frame error rate performance of the Tanner code and code $\mathcal{C}_1$ under the Gallager A/B algorithm on the BSC with maximum of 100 iterations.

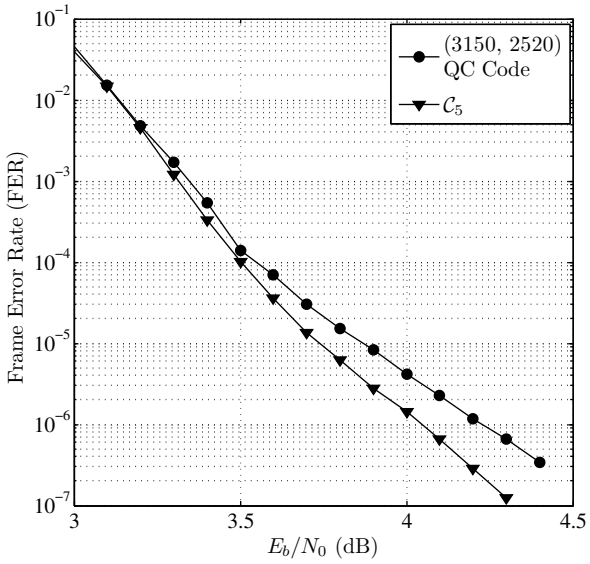

Fig. 8. Frame error rate performance of codes in Example **??** under the SPA on the AWGNC.

B. Improving Error Floor Performance by Designing Better Decoders

While the afore-mentioned code construction techniques have successfully produced codes with lowered error floors, their central idea of avoiding loopy structures during construction becomes very difficult to maintain when the code rate is relatively high as graphs become denser. A natural approach to further improve the error performance in the error floor region is to design a better decoder. Designing better decoder is especially important in applications for which constructing better codes are not possible. The problem of designing better decoders is even more relevant given the fact that negative effects of finite precision introduced when decoders are realized in hardware can further deteriorate the error-rate performance.

A glimpse at the iterative decoders developed so far reveals a wide range of decoders of varying complexity. The simple binary message passing algorithms such as the Gallager A/B algorithms occupy one end of the spectrum, while the BP lies at the other end of the spectrum. The gamut of decoders filling the intermediate space can simply be understood as the implementation of the BP (and variants) at different levels of precision.

Finite Precision of Messages, Quantized Belief Propagation and Low-Complexity Modifications: Early works on the design of quantized BP decoders include the Gallager-E and other quantized decoders proposed by Richardson and Urbanke [47], and reduced-complexity BP decoders such as the normalized min-sum and offset min-sum decoders proposed by Chen *et al.* [77] and by Fossorier *et al.* [78]. More recent works include the quantized BP decoders proposed by Lee and Thorpe [79], and also

by Kurkosi and Yagi [80]. In all of the aforementioned works, the quantization schemes are designed based on optimizing for the best decoding threshold on a given code using the asymptotic technique of density evolution (DE). Another important point to note in the context of this dissertation, is that these decoders were all designed with the primary goal of approaching the performance of the floating-point BP algorithm, as a performance loss is typically associated with quantization of messages. Since asymptotic methods are inapplicable to finite length codes, these decoders designed for the best DE thresholds do not guarantee a good performance on a finite length code especially in the high SNR region.

There have also been several works that have addressed the error problem from a decoding perspective by proposing modifications to the BP decoding algorithm. Some of the notable works include augmented BP by Varnica *et al.* [81], multilple-bases BP by Hehn *et al.* [82], informed dynamic scheduling by Casado *et al.* [83], multi-stage decoding by Wang *et al.* [84], averaged decoding by Laendner and Milenkovic [85], and use of post-processing to lower the error floors by Han and Ryan [86] and also by Zhang *et al.* [87]. While all these schemes certainly provided performance enhancements in the error floor, all of them require either a considerable increase in decoding complexity due to the modifications and post-processing or are restricted to a particular code whose structure is well-known. In addition, they do not take finite precision into account and this can drastically affect the performance gains when implemented in hardware due to possible numerical precision issues.

The fact that message quantization can further contribute to the error floor phenomenon was first observed by Richardson [7], and later Zhang *et al.* [88] also studied the effects of quantization on the error floor behavior of hardware-implemented BP over the AWGNC. More recently, Butler and Siegel [89] showed that the error floor on the AWGNC was greatly affected by the numerical saturations arising from the quantization of the messages, and the error floor performance improved when there was limited or no saturation in the messages.

Therefore, addressing the error floor problem while also taking finite-precision into account in the decoder design is critically important especially with emerging applications in communication and data storage systems now requiring very low error-rates, faster processing speeds, lower memory usage, and reduced chip area. In this regard, there are some relevant works worth mentioning. Zhao *et al.* in [90] proposed several modifications to the offset min-sum decoder while taking quantization into account. Their proposed quantization schemes enabled their offset-min sum decoder to approach performance of floating-point BP algorithm on the Additive White Gaussian Noise channel (AWGNC) with six bits of quantization with possible improvement in the error floor. Most notable work is a self-correcting min-sum by Savin [91] which modifies the variable node update rule to disregard unreliable messages. Zhang *et al.* in [88] also studied the effects of quantization on the error floors of BP over the AWGNC, and designed schemes that led to substantial error floor reductions when six bits of quantization were used. More recently, Zhang and Siegel [92], [93] proposed a quasi-uniform quantization scheme for various BP-based decoders over the BSC as well as the AWGNC. On the BSC, they were able match the performance of different types of floating-point min-sum decoders with 4 bits of quantization, while they required at least 6 bits of quantization to match the performance of floating-point BP.

Decoding Beyond Belief Propagation: Most of the above-mentioned approaches share a common goal, that is, approach the performance of BP decoding. A drawback of approaching BP performance lies in the fact that BP decoding itself is pruned to error floor failure. In [94], Planjery *et al.* proposed a radically different approach. Unlike previous approaches, Planjery's work is mainly concerned with the development of novel finite precision iterative decoding algorithms with performance "beyond belief propagation". The main goals of designing beyond belief propagation decoders are: 1) to surpass the error floor performance of conventional floating-point BP decoder at much lower complexity while taking finite precision into account, and 2) to enhance the guaranteed correction ability of a code and reduce the performance gap between iterative decoding and ML decoding in the error floor. The result of Planjery's *et al* work is a class of decoding algorithms named *finite alphabet iterative decoders* (FAIDs). The key features that clearly distinguish FAIDs from all other previously mentioned works on finite precision iterative decoders are: 1) the messages are not quantized values of log-likelihoods or probabilities, and 2) the variable node update functions are simple well-defined maps rather than approximations of the update functions used in BP. The maps for variable node update in FAIDs are designed with the goal of increasing the guaranteed error-correction capability by using the knowledge of potentially harmful subgraphs that could be present in any given code, thereby improving the slope of the error floor on the BSC [95]. Since the variable nodes in the proposed decoders are now equipped to deal with potentially harmful neighborhoods, which is in contrast to BP which treats the loopy Tanner graph as a tree, the proposed decoders are capable of surpassing the floating-point BP in the error floor. As impressively as it can be, there exist 3-bit precision FAIDs capable of surpassing BP in the error floor.

Planjery *et al.* [96] proposed a technique called *decimation* which is incorporated into the message update rule and involves fixing certain bits of the code to a particular value. By judiciously choosing the rule, decimation significantly reduces the number of iterations required to correct a fixed number of errors, while maintaining the good performance of the original decoder in the error floor region. The adaptive version of this technique is shown to further improve the guaranteed error correction [96]. Nguyen and Vasić [97] introduced a new class of bit flipping algorithms over the binary symmetric channel which employ one additional bit at a variable node to represent its *strength*. This additional bit increase in the guaranteed error correction capability. Remarkably, this algorithm permits enumeration of its trapping sets by a recursive procedure.

Approaching ML Performance via Iterative Decoding: A natural question that arises in the context of iterative message passing algorithms is the following: how far are these sub-optimal algorithms from ML decoding? A closely related question is that how close can they be made to the ML decoder? These questions have opened up a plethora of interesting problems and led to some very interesting results.

Not surprisingly, decoding on the BEC provides the starting point and the most optimistic results. The size of the smallest stopping set in the Tanner graph of a code $\mathcal{C}$ is generally referred to as the stopping distance of the code. The stopping distance for decoding on the BEC can be thought of as analogous to the minimum distance $d(\mathcal{C})$ for ML decoding. It is however worth noting that while the minimum distance is a property of the code, the stopping distance is a property of the specific Tanner graph or the parity-check matrix H chosen for decoding and is therefore denoted by $s(H)$. The parity-check matrix of a linear block code generally has $n-k$ rows, where k is the dimension of the code. In [98] and [99], it was observed that adding linearly dependant rows to H can increase the stopping distance of the code. Schwartz and Vardy [100] introduced and studied the notion of stopping redundancy of a code. The redundancy of a code is defined as the minimum number of rows in a parity-check matrix of the code. The stopping redundancy of a code is defined as the minimum number of rows in a parity-check matrix such that the stopping distance is equal to the minimum distance of the code. Schwartz and Vardy [100] showed that it is always possible to find a parity-check matrix H such that $s(H) = d(\mathcal{C})$ and presented bounds on the stopping redundancy for some families of codes. Han and Siegel improved the bounds on the stopping redundancy.

Decoding on redundant Tanner graphs has also been studied in the context of other channels, especially for classical linear block codes that are generally characterized by high density parity check (HDPC) matrices. Kothiyal *et al.* [101] studied a scheme known as adaptive belief propagation (ABP) which chooses new parity check sets based on soft information reliability. Jiang and Narayanan [102] introduced stochastic shifting based iterative decoding (SSID) algorithm for cyclic codes that makes use of the permutation groups. Halford and Chugg [103] proposed random redundant decoding (RRD) that is applicable to arbitrary linear block codes. Dimnik and Beery [104] further improved the RRD for HDPC codes. Knudsen *et al.* [105] described a simple graph operation known as edge local computation that improves the performance of the sum product algorithm. A different approach has been studied by Hehn, *et al.* [106] who investigated a method of decoding that operates in parallel on a collection of parity-check matrices. The approach, known as multiple-bases belief propagation (MBBP), performs joint output processing on a set of decoder representations to estimate the transmitted codeword. The multiple bases approach was also investigated by Jin and Fossorier [107].

Another avenue of improving performance of iterative decoders is using different decoders in parallel or in serial. It has been shown [108] that this is more efficient than using redundant rows of the parity check matrix or using redundant parity check matrices. In [108], Declercq *et al.* further increased the guaranteed error correction capability from what is achievable by a FAID. The proposed scheme uses a plurality of FAIDs on column-weight-three LDPC codes which collectively correct more errors than a single FAID. The collection of FAIDs is chosen to ensure that individual decoders correct different error patterns (called decoder diversity). The idea of using different decoders was proposed in the context of bit-fipping in [97]. It was shown in [97] that decoders that employ a properly selected group of the algorithms with different update rules, operating in parallel offer high speed and low error floor decoding.

Analogous to stopping distance for decoding on the erasure channel is the notion of minimum pseudoweight for the AWGNC and BSC (see [24] and [109]). Zumbragel, Skacheck and Flanagan [110] studied the pseudoredundancy of binary linear codes for the BSC and AWGNC building over the preliminary work of Kelley and Sridhara [42]. They proved that for most of the codes there is no parity-check matrix such that the minimum pseudoweight is equal to the minimum distance. A related notion of trapping redundancy along with several bounds were investigated by Laendner *et al.* in [111]. Kashyap [112] studied a class of codes known as geometrically perfect codes and investigated similar problems. Smarandache *et al.* [113] studied the pseduoweight spectrum gap for binary linear codes.

VII. Connections

The maximum likelihood (ML) decoding of the code $\mathcal{C}$ allows a convenient LP formulation in terms of the *codeword polytope* poly($\mathcal{C}$) [39] whose vertices correspond to the codewords in $\mathcal{C}$. The ML-LP decoder finds $\mathbf{f} = (f_1, \ldots, f_n)$ minimizing the cost function $\sum_{i=1}^{n} \gamma_i f_i$ subject to the $\mathbf{f} \in$ poly($\mathcal{C}$) constraint. The formulation is compact but impractical as the number of constraints is exponential in the code length.

Hence a *relaxed* polytope is defined as the intersection of all the polytopes associated with the local codes introduced for all the checks of the original code. Associating $(f_1, \ldots, f_n)$ with bits of the code we require

$$0 \leq f_i \leq 1, \quad \forall i \in V \tag{5}$$

For every check node α, let $N(\alpha)$ denote the set of variable nodes which are neighbors of α. Let $E_\alpha = \{T \subseteq N(\alpha) : |T| \text{ is even}\}$. The polytope Q_α associated with the check node α is defined as the set of points $(\mathbf{f}, \mathbf{w})$ for which the following

constraints hold

$$0 \leq w_{\alpha,T} \leq 1, \qquad \forall T \in E_\alpha \tag{6}$$

$$\textstyle\sum_{T \in E_\alpha} w_{\alpha,T} = 1 \tag{7}$$

$$f_i = \textstyle\sum_{T \in E_\alpha, T \ni i} w_{\alpha,T}, \quad \forall i \in N(\alpha) \tag{8}$$

Now, let $Q = \cap_\alpha Q_\alpha$ be the set of points $(\mathbf{f}, \mathbf{w})$ such that (5)-(8) hold for all $\alpha \in C$. (Note that Q, which is also referred to as the fundamental polytope [40], [114], is a function of the Tanner graph G and consequently the parity-check matrix H representing the code $\mathcal{C}$.) The Linear Code Linear Program (LCLP) can be stated as

$$\min_{(\mathbf{f},\mathbf{w})} \sum_{i \in V} \gamma_i f_i, \text{ s.t. } (\mathbf{f}, \mathbf{w}) \in Q.$$

For the sake of brevity, the decoder based on the LCLP is referred to in the following as the LP decoder. A solution $(\mathbf{f}, \mathbf{w})$ to the LCLP such that all f_is and $w_{\alpha,T}$s are integers is known as an integer solution. The integer solution represents a codeword [39]. It was also shown in [39] that the LP decoder has the ML certificate, i.e., if the output of the decoder is a codeword, then the ML decoder would decode into the same codeword. The LCLP can fail, generating an output which is not a codeword.

It is appropriate to mention here that the LCLP can be viewed as the zero temperature version of BP-decoder looking for the global minimum of the so-called Bethe free energy functional [115].

The assumption of the transmission of the all-zero-codeword also holds for the LP decoding of linear codes on output symmetric channels, as the polytope Q is highly symmetric and looks exactly the same from any codeword (see [39] for proof).

If an instanton of a channel/decoder is known, the respective pseudo-codeword can be easily found, and conversely if a pseudo-codeword is given (i.e. we know for sure that there exists a configuration of the noise which is sandwiched in between the pseudo-codeword and the zero-codeword) the respective instanton can be restored. In fact, this inversion is in the core of the pseudo-codeword/instanton search algorithms discussed in Section VI.

A. Pseudo-codewords for LP Decoders

In contrast to the iterative decoders, the output of the LP decoder is well defined in terms of pseudo-codewords.

Definition 7: [39] An *integer pseudo-codeword* is a vector $\mathbf{p} = (p_1, \ldots, p_n)$ of non-negative integers such that, for every parity check $\alpha \in C$, the neighborhood $\{p_i : i \in N(\alpha)\}$ is a sum of local codewords.

The interested reader is referred to Section V in [39] for more details and examples. Alternatively, one may choose to define a *re-scaled pseudo-codeword*, $\mathbf{p} = (p_1, \ldots, p_n)$ where $0 \leq p_i \leq 1, \forall i \in V$, simply equal to the output of the LCLP. In the following, we adopt the re-scaled definition. The cost associated with LP decoding of a vector $\hat{\mathbf{y}}$ to a pseudo-codeword $\mathbf{p}$ is given by

$$C(\hat{\mathbf{y}}, \mathbf{p}) = \sum_{i \in V} \gamma_i p_i.$$

For an input $\hat{\mathbf{y}}$, the LP decoder outputs the pseudo-codeword $\mathbf{p}$ with minimum $C(\hat{\mathbf{y}}, \mathbf{p})$. Since the cost associated with LP decoding of $\hat{\mathbf{y}}$ to the all-zero-codeword is zero, a decoder failure occurs on the input $\hat{\mathbf{y}}$ if and only if there exists a pseudo-codeword $\mathbf{p}$ with $C(\hat{\mathbf{y}}, \mathbf{p}) \leq 0$.

A given code $\mathcal{C}$ may have different Tanner graph representations and consequently potentially different fundamental polytopes. Hence, we refer to the pseudo-codewords as corresponding to a particular Tanner graph G of $\mathcal{C}$.

Definition 8: [26] Let $\mathbf{p} = (p_1, \ldots, p_n)$ be a pseudo-codeword distinct from the all-zero-codeword of the code $\mathcal{C}$ represented by Tanner graph G . Then, the *pseudo-codeword weight* of $\mathbf{p}$ is defined as follows:

- $w_{BSC}(\mathbf{p})$ for the BSC is

$$w_{BSC}(\mathbf{p}) = \begin{cases} 2e, & \text{if } \sum_e p_i = \left(\sum_{i \in V} p_i\right)/2; \\ 2e-1, & \text{if } \sum_e p_i > \left(\sum_{i \in V} p_i\right)/2. \end{cases}$$

  where e is the smallest number such that the sum of the e largest p_is is at least $\left(\sum_{i \in V} p_i\right)/2$.
- $w_{AWGN}(\mathbf{p})$ for the AWGNC is

$$w_{AWGN}(\mathbf{p}) = \frac{(p_1 + p_2 + \ldots + p_n)^2}{(p_1^2 + p_2^2 + \ldots + p_n^2)}$$

The minimum pseudo-codeword weight of G denoted by $w_{min}^{BSC/AWGN}$ is the minimum over all the non-zero pseudo-codewords of G.

We now give definitions specific to the BSC.

Definition 9: (Median for LP decoding over the BSC) The median noise vector (or simply the median) $M_(\mathbf{p})$ of a pseudo-codeword $\mathbf{p}$ distinct from the all-zero-codeword is a binary vector with support $S = \{i_1, i_2, \ldots, i_e\}$, such that $p_{i_1}, \ldots, p_{i_e}$ are the $e(= \lceil (w_{BSC}(\mathbf{p}) + 1)/2 \rceil)$ largest components of $\mathbf{p}$.

Note that for input $\hat{\mathbf{y}} = M(\mathbf{p})$ for some non-zero pseudo-codeword $\mathbf{p}$, we have $C(\hat{\mathbf{y}}, \mathbf{p}) \leq 0$ and hence leads to a decoding failure (the output of the decoder, however, need not be the pseudo-codeword we start with).

Definition 10: (Instanton for LP decoding over the BSC) The BSC *instanton* $\mathbf{i}$ is a binary vector with the following properties: (1) There exists a pseudo-codeword $\mathbf{p}$ such that $C(\mathbf{i}, \mathbf{p}) \leq C(\mathbf{i}, \mathbf{0}) = 0$; (2) For any binary vector $\mathbf{r}$ such that $\text{supp}(\mathbf{r}) \subset \text{supp}(\mathbf{i})$, there exists no pseudo-codeword with $C(\mathbf{r}, \mathbf{p}) \leq 0$. The size of an instanton is the cardinality of its support.

An attractive feature of LP decoding over the BSC is that any input whose support contains an instanton leads to a decoding failure (which is not the case for Gallager A decoding over the BSC) [116]. This important property is in fact used in searching for instantons.

Specifically, for LP decoding over the BSC and the Gallager algorithm, the slope of the FER curve in the error floor region is equal to the cardinality of the smallest size instanton (see [51] for a formal description).

a

VIII. Conclusion

The introduction of turbo codes in the mid 1990s and the subsequent rediscovery of LDPC codes shortly thereafter initiated what is now sometimes called the era of "modern coding theory. At the heart of this theory is the fact that these families of codes can be efficiently decoded by iterative message-passing and linear programming (LP) algorithms. Analytical tools such as density evolution show that suitably designed LDPC code ensembles asymptotically approach, and on some channels even achieve, the channel capacity under message passing and LP decoding.

However, the capacity-approaching property holds only in the limit of infinite codeword length, and it has been found empirically that, when applied to finite-length codes, message passing and LP decoding suffer an abrupt performance degradation, known as the error floor phenomenon, in the low-noise regime. In view of its direct implications for future deployment of LDPC codes in high-performance communication and storage systems, characterization of the error floor is arguably one of the most important open problems in coding theory today.

In contrast to the complicated dynamics of IMP decoding, LP decoding outcomes are precisely characterized in terms of pseudocodewords corresponding to the vertices of the LP constraint region, or fundamental polytope. Through the concept of graph cover decoding, which captures the local nature of iterative decoding, the fundamental polytope establishes an important connection between IMP and LP decoder performance. Recent investigations of message passing and LP decoder failure in the error floor region have demonstrated the role played by the local nature of the decoding algorithms and the presence of certain substructures, generically referred to as trapping sets, in the underlying graphical representation of the code. For the BEC, decoding failure can be completely characterized in terms of the subgraphs defined by stopping sets. For the BSC and AWGNC, however, the picture is far from complete, and their remains a need for general design methodologies for the construction of codes and IMP and LP decoders with guaranteed error floor performance. The proposed research addresses these needs.

Recent investigations of message passing and LP decoder failure in the error floor region have demonstrated the role played by the local nature of the decoding algorithms and the presence of certain substructures, generically referred to as trapping sets, in the underlying graphical representation of the code. In contrast to the complicated dynamics of IMP decoding, LP decoding outcomes are precisely characterized in terms of pseudocodewords corresponding to the vertices of the LP constraint region, or fundamental polytope. Through the concept of graph cover decoding, which captures the local nature of iterative decoding, the fundamental polytope establishes an important connection between IMP and LP decoder performance.

While decoding failures on the BEC can be completely characterized in terms of the subgraphs defined by stopping sets, for the BSC and AWGNC the picture is far from complete, and their remains a need for general design methodologies for the construction of codes and message passing and LP decoders with guaranteed error floor performance.

We presented a study of graph substructures and corresponding error patterns involved in decoding failures on various channel under different iterative message passing and LP decoding algorithms. The intriguing structural dependence among these subgraphs suggests a comprehensive framework for studying the error floor performance of LDPC codes, as well as for designing codes with guaranteed error floor performance, not only on the BSC, but also on the AWGN and other channels.

The problem treated in this chapter is one of the major challenges in modern coding theory. It require using novel topological perspective to advance our fundamental understanding of the relationship between graphical code representations and the performance of iterative decoding algorithms on several core channel models. In particular, it can serve as a foundation for elucidating the relationship between IMP and LP decoding, and lead to the design of LDPC codes with a superior performance in the error floor region.

Acknowledgment

This work was funded by NSF under grant CCF-0963726, and partially by IDEMA ASTC and DARPA-KECoM through contract N66001–10–1–4079.

REFERENCES

[1] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
[2] D. J. C. Mackay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
[3] J. Pearl, *Probablisitic Reasoning in Intelligent Systems*. San Francisco, CA: Kaufmann, 1988.
[4] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, May 1981.
[5] B. J. Frey, *Graphical models for machine learning and digital communication*. Cambridge, MA, USA: MIT Press, 1998.
[6] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks (research note)," *Artif. Intell.*, vol. 42, no. 2–3, pp. 393–405, 1990.
[7] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Annual Allerton Conf. on Commun., Control and Computing*, 2003, pp. 1426–1435.
[8] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, Mar. 2008.
[9] V. V. Zyablov and M. S. Pinsker, "Estimation of the error-correction complexity for Gallager low-density codes," *Problems of Information Transmission*, vol. 11, no. 1, pp. 18–28, 1976.
[10] M. Sipser and D. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1710–1722, Jun. 1996.
[11] D. Burshtein and G. Miller, "Expander graph arguments for message-passing algorithms," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 782–790, Feb. 2001.
[12] D. Burshtein, "On the error correction of regular LDPC codes using the flipping algorithm,," *IEEE Trans. Inf. Theory*, vol. 54, no. 2, pp. 517–530, Feb. 2008.
[13] S. K. Chilappagari and B. Vasić, "Error-correction capability of column-weight-three LDPC codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2055–2061, May 2009.
[14] G. Zemor, "On expander codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 835–837, 2001.
[15] A. Barg and G. Zemor, "Error exponents of expander codes," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1725–1729, 2002.
[16] H. Janwa and A. K. Lal, "On Tanner codes: minimum distance and decoding," *Appl. Algebra Eng. Commun. Comput.*, vol. 13, no. 5, pp. 335–347, 2003.
[17] N. Miladinovic and M. Fossorier, "Generalized LDPC codes with Reed-Solomon and BCH codes as component codes for binary channels," in *Proc. IEEE Global Telecommun. Conf.*, vol. 3, 2005, pp. 1239–1244.
[18] N. Alon, S. Hoory, and M. Linial, "The moore bound for irregular graphs," *Graphs and Combinatorics*, vol. 18, no. 1, pp. 53–57, 2002.
[19] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 929–953, Mar. 2005.
[20] S. K. Chilappagari, D. V. Nguyen, B. Vasić, 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.
[21] ——, "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.
[22] 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.
[23] 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.
[24] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Univ. Linköping, Sweden, Dept. Elec. Eng., 1996.
[25] B. Frey, R. Koetter, and A. Vardy, "Signal-space characterization of iterative decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 766–781, Feb. 2001.
[26] J. Forney, G.D., "Codes on graphs: normal realizations," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.
[27] D. J. C. MacKay and M. J. Postol, "Weaknesses of Margulis and Ramanujan–Margulis low-density parity-check codes," *Electronic Notes in Theoretical Computer Science*, vol. 74, 2003.
[28] M. Stepanov and M. Chertkov, "Instanton analysis of low-density-parity-check codes in the error-floor regime," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 9–14 2006, pp. 9–14.
[29] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasić, "Error floors of LDPC codes on the binary symmetric channel," in *Proc. IEEE Int. Conf. on Commun.*, vol. 3, Istanbul, Turkey, Jun. 2006, pp. 1089–1094.
[30] 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.
[31] 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.
[32] 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.
[33] 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 Annual Conference on Information Sciences and Systems*, Princeton, NJ, USA, Mar. 22–24 2006, pp. 823–828.
[34] S. Abu-Surra, D. Declercq, D. Divsalar, and W. E. Ryan, "Trapping set enumerators for specific LDPC codes," in *Proc. Inf. Theory and Appl. Workshop*, La Jolla, CA, USA, Jan. 31–Feb. 5 2010, pp. 1–5.
[35] E. Rosnes and O. Ytrehus, "An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices," *IEEE Trans. Inf. Theory*, vol. 55, no. 9, pp. 4167–4178, Sept 2009.
[36] M. Karimi and A. Banihashemi, "Efficient algorithm for finding dominant trapping sets of LDPC codes," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6942–6958, Nov. 2012.
[37] X. Zhang and P. Siegel, "Efficient algorithms to find all small error-prone substructures in LDPC codes," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2011, pp. 1–6.
[38] B. Vasić, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery, "Trapping set ontology," in *Proc. 47th Allerton Conf. on Communications, Control, and Computing*, Allerton House, Monticello, IL, USA, Sept. 30–Oct. 2 2009, pp. 1–7.
[39] J. Feldman, M. Wainwright, and D. Karger, "Using linear programming to decode binary linear codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 954–972, Mar. 2005.
[40] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite length analysis of message-passing iterative decoding of LDPC codes," 2005. [Online]. Available: http://arxiv.org/abs/cs.IT/0512078
[41] P. Vontobel and R. Koetter, "On the relationship between linear programming decoding and min-sum algorithm decoding," in *Proc. Int. Symp. Inf. Theory and its Appl.*, Parma, Italy, Oct. 10–13 2004, pp. 991–996.
[42] C. Kelley and D. Sridhara, "Pseudocodewords of Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 53, no. 11, pp. 4013–4038, Nov. 2007.
[43] 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.
[44] R. Smarandache and P. Vontobel, "Pseudo-codeword analysis of Tanner graphs from projective and Euclidean planes," *IEEE Trans. Inf. Theory*, vol. 53, no. 7, pp. 2376–2393, Jul. 2007.

[45] R. Smarandache, A. E. Pusane, P. O. Vontobel, and D. J. Costello, "Pseudo-codewords in LDPC convolutional codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2006, pp. 1364–1368.

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

[47] 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.

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

[49] V. Chernyak, M. Chertkov, M. Stepanov, and B. Vasić, "Instanton method of post-error-correction analytical evaluation," in *Proc. IEEE Inf. Theory Workshop*, 2004, pp. 220–224.

[50] M. Stepanov and M. Chertkov, "Instanton analysis of low-density-parity-check codes in the error-floor regime," in *Proc. Int. Symp. Inf. Theory*, Jul. 9–14 2006, pp. 9–14.

[51] M. Ivkovic, S. K. Chilappagari, and B. Vasić, "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.

[52] 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.

[53] ——, "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.

[54] 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.

[55] B. Vasić, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery, "Trapping set ontology," in *Error Correction Coding Laboratory Web Page*. [Online]. Available: http://www2.engr.arizona.edu/~vasiclab/project.php?id=9

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

[57] 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.

[58] T. Tian, C. Jones, J. Villasenor, and R. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun*, vol. 52, no. 8, pp. 1242–1247, Aug. 2004.

[59] H. Xiao and A. Banihashemi, "Improved progressive-edge-growth (PEG) construction of irregular LDPC codes," *IEEE Commun. Letters*, vol. 8, no. 12, pp. 715–717, Dec. 2004.

[60] B. Vasić, 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.

[61] 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.

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

[63] 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*, vol. 53, no. 8, Nice, France, Jun. 24–29 2007, pp. 2885–2891.

[64] 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, USA, Jun. 27–Jul. 2 2004, p. 236.

[65] 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 and Systems*, Seattle, WA, USA, May 18–21 2008, pp. 1846–1849.

[66] J. Wang, L. Dolecek, and R. Wesel, "Controlling LDPC absorbing sets via the null space of the cycle consistency matrix," in *Proc. Int. Conf. on Commun.*, Kyoto, Japan, Jun. 5–9 2011, pp. 1–6.

[67] 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.

[68] 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, Texas, USA, Jun. 13–18 2010, pp. 739–743.

[69] 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 and Its Applications*, Auckland, New Zealand, Dec. 7–10 2008, pp. 1–6.

[70] S. K. Chilappagari, A. R. Krishnan, and B. Vasić, "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.

[71] 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.

[72] M. G. Stepanov, V. Chernyak, M. Chertkov, and B. Vasić, "Diagnosis of weaknesses in modern error correction codes: A physics approach," *Physical Review Letters*, vol. 95, no. 22, pp. 228 701–228 704, Nov. 2005.

[73] V. Chernyak, M. Chertkov, M. G. Stepanov, and B. Vasić, "Error correction on a tree: an instanton approach," *Physics Review Letter*, vol. 93, no. 19, pp. 198 702–198 705, Nov. 2004.

[74] V. Chernyak, M. Chertkov, M. Stepanov, and B. Vasić, "Instanton method of post-error-correction analytical evaluation," in *Proc. IEEE Information Theory Workshop (ITW '04)*, San Antonio, TX, Oct. 2004, pp. 220–224.

[75] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," in *Proc. 5th Int. Symp. Commun. Theory and Applications*, Ambleside, UK, Jul. 15–20 2001.

[76] 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.

[77] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 7, p. 1232, Jul. 2005.

[78] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.

[79] J.-S. Lee and J. Thorpe, "Memory-efficient decoding of LDPC codes," in *Proc. Int. Symp. Inf. Theory*, Adelaide, Australia, Sep. 2005, pp. 459–463.

[80] B. M. Kurkoski and H. Yagi, "Quantization of binary-input discrete memoryless channels with applications to LDPC decoding," 2011. [Online]. Available: http://arxiv.org/abs/1107.5637

[81] N. Varnica, M. Fossorier, and A. Kavcic, "Augmented belief propagation decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 55, no. 7, pp. 1308–1317, Jul. 2007.

[82] T. Hehn, J. Huber, O. Milenkovic, and S. Laendner, "Multiple-bases belief-propagation decoding of high-density cyclic codes," *IEEE Trans. Commun.*, vol. 58, no. 1, pp. 1–8, Jan. 2010.

[83] A. I. V. Casado, M. Griot, and R. D. Wesel, "Ldpc decoders with informed dynamic scheduling," *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3470–3479, Dec. 2010.

[84] Y. Wang, J. S. Yedidia, and S. C. Draper, "Multi-stage decoding of ldpc codes," in *Proc. Int. Symp. Inf. Theory*, Austin, TX, USA, Jul. 2009, pp. 2151–2155.

[85] S. Laendner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. Int. Conf. Wireless Networks, Commun., and Mobile Commun.*, Maui, HI, USA, Jun. 2005, pp. 630–635.
[86] Y. Han and W. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1663–1673, Jun. 2009.
[87] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. Wainwright, "Lowering LDPC error floors by postprocessing," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2008, pp. 1–6.
[88] ——, "Low-floor decoders for LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 11, pp. 3258–3268, Nov. 2009.
[89] B. K. Butler and P. H. Siegel, "Error floor approximation for LDPC codes in the AWGN channel," 2012. [Online]. Available: http://arxiv.org/abs/1202.2826
[90] J. Zhao, F. Zarkeshvari, and A. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 549–554, Apr. 2005.
[91] V. Savin, "Self-corrected min-sum decoding of ldpc codes," in *ISIT*, 2008, pp. 146–150.
[92] X. Zhang and P. H. Siegel, "Quantized min-sum decoders with low error floor for LDPC codes," in *Proc. Int. Symp. Inf. Theory*, Boston, MA, USA, Jul. 2012, pp. 2871–2875.
[93] ——, "Will the real error floor please stand up?" in *Proc. IEEE Int.Conf. Signal Processing and Commun.,*, Bangalore, India, Jul. 2012, pp. 1–5.
[94] S. Planjery, D. Declercq, S. Chilappagari, and B. Vasić and, "Multilevel decoders surpassing belief propagation on the binary symmetric channel," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2010, pp. 769–773.
[95] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasić, "Finite alphabet iterative decoders, part i: Decoding beyond belief propagation on bsc," *IEEE Trans. Commun. (Submitted)*, 2012.
[96] S. K. Planjery, B. Vasić, and D. Declercq, "Enhancing the error correction of finite alphabet iterative decoders via adaptive decimation," in *Proc. IEEE Int. Symp. Theory*, Boston, MA, USA, Jul. 1–6 2012, pp. 2876–2880.
[97] D. V. Nguyen and B. Vasić, "Two-bit bit flipping algorithms for LDPC codes and collective error correction," *submitted to IEEE Trans. Inf. Theory*, Aug. 2012.
[98] N. Santhi and A. Vardy, "On the effect of parity-check weights in iterative decoding," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, june-2 july 2004, p. 101.
[99] J. S. Yedidia, J. Chen, and M. P. C. Fossorier, "Generating code representations suitable for belief propagation decoding," in *Proceedings of the 40th Annual Allerton Conference on Communications, Control and Computing*, September 2002.
[100] M. Schwartz and A. Vardy, "On the stopping distance and the stopping redundancy of codes," *Information Theory, IEEE Transactions on*, vol. 52, no. 3, pp. 922 –932, march 2006.
[101] A. Kothiyal, O. Takeshita, W. Jin, and M. Fossorier, "Iterative reliability-based decoding of linear block codes with adaptive belief propagation," *Communications Letters, IEEE*, vol. 9, no. 12, pp. 1067 – 1069, dec. 2005.
[102]
[103] T. Halford and K. Chugg, "Transactions letters - random redundant iterative soft-in soft-out decoding," *Communications, IEEE Transactions on*, vol. 56, no. 4, pp. 513 –517, april 2008.
[104] I. Dimnik and Y. Be'ery, "Improved random redundant iterative hdpc decoding," *Communications, IEEE Transactions on*, vol. 57, no. 7, pp. 1982 –1985, july 2009.
[105] J. G. Knudsen, C. Riera, L. E. Danielsen, M. G. Parker, and E. Rosnes, "Iterative decoding on multiple tanner graphs using random edge local complementation," in *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory - Volume 2*, ser. ISIT'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 899–903. [Online]. Available: http://dl.acm.org/citation.cfm?id=1701275.1701319
[106] T. Hehn, J. Huber, O. Milenkovic, and S. Laendner, "Multiple-bases belief-propagation decoding of high-density cyclic codes," *Communications, IEEE Transactions on*, vol. 58, no. 1, pp. 1 –8, january 2010.
[107] W. Jin and M. Fossorier, "Reliability-based soft-decision decoding for memory channel," in *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, june 2007, pp. 2811 –2815.
[108] D. Declercq, B. Vasić, S. K. Planjery, and E. Li, "Finite alphabet iterative decoders, part ii: Improved guaranteed error correction of ldpc codes via iterative decoder diversity," *IEEE Trans. Commun. (Submitted)*, 2012.
[109] G. D. Forney, R. Koetter, F. R. Kschischang, and A. Reznik, "On the effective weights of pseudocodewords for codes defined on graphs with cycles," in *Codes, systems and graphical models*. Springer, 2001, pp. 101–112.
[110] J. Zumbra andgel, V. Skachek, and M. Flanagan, "On the pseudocodeword redundancy of binary linear codes," *Information Theory, IEEE Transactions on*, vol. 58, no. 7, pp. 4848 –4861, july 2012.
[111] S. Laendner, T. Hehn, O. Milenkovic, and J. Huber, "The trapping redundancy of linear block codes," *IEEE Trans. Inform. Theory*, vol. 55, no. 1, pp. 53–63, Jan. 2009.
[112] N. Kashyap, "A decomposition theory for binary linear codes," *Information Theory, IEEE Transactions on*, vol. 54, no. 7, pp. 3035 –3058, july 2008.
[113] R. Smarandache, A. Pusane, P. Vontobel, and D. Costello, "Pseudocodeword performance analysis for ldpc convolutional codes," *Information Theory, IEEE Transactions on*, vol. 55, no. 6, pp. 2577 –2598, june 2009.
[114] R. Koetter and P. O. Vontobel, "Graph covers and iterative decoding of finite-length codes," in *Proc. 3rd Int. Conf. on Turbo Codes and Related Topics*, Sep. 1–5 2003, pp. 75–82.
[115] M. J. Wainwright and M. I. Jordan, "Variational inference in graphical models: the view from the marginal polytope," in *Proc. 40th Allerton Conf. on Commun., Control, and Computing*, 2003.
[116] S. K. Chilappagari, M. Chertkov, and B. Vasić, "An efficient instanton search algorithm for LP decoding of LDPC codes over the BSC," *IEEE Trans. Inf. Theory*, vol. 57, no. 7, pp. 4417–4426, Jul. 2011.

Bane Vasic received his B.Sc., M.Sc., and Ph.D. from the University of Nis, Serbia. He is currently a Professor of Electrical and Computer Engineering and Mathematics at the University of Arizona, Tucson. Prior to this appointment, he was at Bell Laboratories. He is a Member of the Editorial Board of the IEEE Transactions on Magnetics, and was a chair or technical program chair for several workshops and conferences including: IEEE CTW 2003 and 2007, DIMACS Workgroup and Workshop on Theoretical Advances in Information Recording, 2004, LANL Workshop on Applications of Statistical Physics to Coding Theory, 2004, Communication Theory Symposium within ICC 2006. He authored a number of journal and conference articles, book chapters and edited three books, and his patents are implemented in Bell Labs chips. His research interests include coding theory, communication theory, constrained systems, and digital communications and recording.

Shashi Kiran Chilappagari 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 January 2010. He is currently the manager of the flash architecture group at Marvell Semiconductor Inc, Santa Clara. 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.

Dung Viet Nguyen received the B.S. and Ph.D. degrees in electrical engineering from the University of Arizona, Tucson, in 2007 and 2012, respectively. He is joining the flash architecture group at Marvell Semiconductor Inc, Santa Clara. His research interests include, combinatorics, graph theory, coding theory and information theory.