Low-Floor Decoders for LDPC Codes

Yang Han and William E. Ryan
University of Arizona
{yhan,ryan}@ece.arizona.edu

Abstract

One of the most significant impediments to the use of LDPC codes in many communication and storage systems is the error-rate floor phenomenon associated with their iterative decoders. The error floor has been attributed to certain sub-graphs of an LDPC code's Tanner graph induced by so-called trapping sets. We show in this paper that once we identify the trapping sets of an LDPC code of interest, a sum-product algorithm (SPA) decoder can be custom-designed to yield floors that are orders of magnitude lower than the conventional SPA decoder. We present two classes of such decoders: a *bi-mode* syndrome-erasure decoder and three *generalized-LDPC* decoders. We demonstrate the effectiveness of these decoders for two codes, the rate-1/2 (2640,1320) Margulis code which is notorious for its floors and a rate-0.3 (640,192) quasi-cyclic code which has been devised for this study.

1 Introduction

Low-density parity-check (LDPC) codes [1] have been intensely researched by coding theorists and practitioners over the past decade due to their near-capacity performance when decoded with low-complexity iterative decoders. Many of their properties have been well studied, yet the error-rate floor phenomenon has remained an open problem. Since many systems such as data storage devices and optical communication systems require extremely low error rates, solving the error floor problem has been a critical issue during the past decade.

Many attempts have been made to solve or mitigate the floor problem by designing LDPC codes with low floors ([2], [3], [4], [5], [6], [7], [8]), however, only a few papers focus on decoder-based strategies for lowering floors ([9], [10], [11]). In this paper, we propose decoder-based strategies which include: a bi-mode syndrome-erasure decoder and three generalized-LDPC (G-LDPC) decoders. The bi-mode decoder is essentially a post-processing technique and the G-LDPC decoders involve decoder modifications.

It is well known that the error floors of LDPC codes under message-passing decoding are usually due to low-weight *near-codewords* [12] or *trapping sets* [13], rather than low-weight codewords. A (w, v) trapping set is a set of w variable nodes (VNs) which induce a subgraph with v odd-degree check nodes and an arbitrary number of even-degree check nodes (CNs). Throughout the paper, to simplify the language, we shall often use "trapping set" when we are referring to the induced subgraph. When w and v are relatively small, errors in each of the w VNs create v parity check failures and tend to lead to situations from which the iterative decoder cannot escape. Iterative decoders are

This work was funded by a gift from INSIC and NASA grant NNX06AC17G.

susceptible to trapping set problems because they work locally in a distributed-processing fashion to (hopefully) arrive at a globally optimum decision. Of course, iterative decoders are vulnerable to cycles in the graph on which the decoder is based, for cycles often lead the iterative decoder away from the ML codeword. (Trapping sets are unions of several cycles.)

The most dominant trapping sets can be determined from computer simulations in the floor region. Moreover, for most practical codes, because the trapping set-induced subgraphs associated with the error floor are relatively small, and because their cardinalities are not too large, it is feasible to discover and enumerate all of the dominant trapping sets. Hence, once we obtain the trapping set information for an LDPC code by simulation and by some graph-search techniques, we explicitly target the known trapping sets with novel, custom-tailored iterative decoder designs. These low-floor decoders lower the floor by orders of magnitude and, although the specific trapping sets depend on decoder specifics such as algorithm type and message word size, the effectiveness of our low-floor techniques do not.

In this paper, we consider the binary-input additive white Gaussian noise (AWGN) channel and the sum-product algorithm (SPA) decoder as our baseline system. For illustration purposes, we chose two LDPC codes to demonstrate the effectiveness of these decoders: (1) the rate-0.5 (2640, 1320) Margulis code which is notorious for its trapping set-induced floors [12], [13], and (2) a short quasi-cyclic rate-0.3 (640, 192) code.

The rest of the paper is organized as follows. Section 2 briefly reviews the two LDPC codes under study. Section 3 describes the bi-mode decoder that recovers trapping sets by a post-processing erasure decoding algorithm. In Section 4, we explore the concept of *generalized LDPC* Tanner graphs and their G-LDPC decoders. The idea is to decode conventional LDPC codes as if they were G-LDPC codes, with the goal of eliminating trapping sets error events. The algorithm for finding the trapping sets of G-LDPC decoders is also presented in that section, after which the frame error rate (FER) performance can be accurately predicted using the method in [13].

2 Codes Under Study

Throughout the paper, we focus on two LDPC codes. Both of the codes have the following structure: the parity-check matrix $\mathbf{H}$ can be conveniently arranged into an $M \times N$ array of permutations of the $Q \times Q$ identity matrices and $Q \times Q$ zero matrices. This structure simplifies the analysis of the code ensemble because the Tanner graph possesses an automorphism of order Q, and thus simplifies the search for all of the trapping sets of a code. Obviously, if the $Q \times Q$ permutation matrices are circulant, the code is quasi-cyclic (QC), a characteristic that facilitates encoder and decoder implementations.

2.1 The Margulis Code

The error floor of the Margulis construction of a regular (3, 6) Gallager code has been well studied [12]. The rate-1/2 (2640,1320) Margulis code used in this paper is generated from the *special linear group* $SL_2(11)$. Its parity check matrix can be expressed as a 120×240 array of 11×11 permutation matrices. The "weakness" of this algebraically constructed code is a relatively high error floor due to trapping sets, as discovered by Mackay and Postol in [12] using computer simulations. The source of error floor is 1320 isomorphic (12,4) trapping sets and 1320 isomorphic (14,4) trapping sets, both types depicted in Fig. 1. The isomorphic trapping sets are equivalent in terms of their contributions to the FER floor performance.

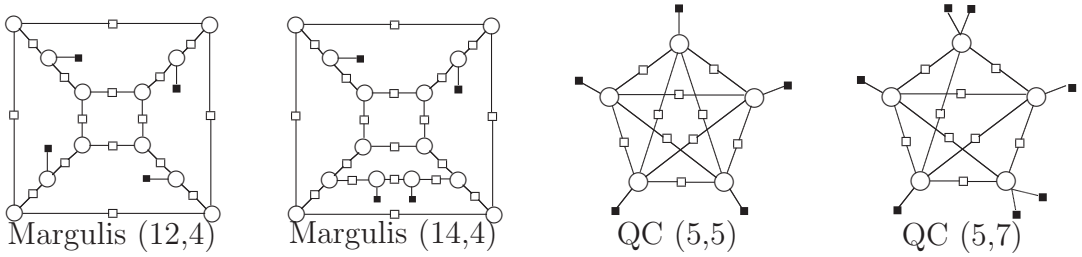

Figure 1: Trapping sets of the Margulis and QC codes decoded by the SPA decoder. ◯ = VNs. □ = mis-satisfied CNs. ■ = unsatisfied CNs

This code is extremely difficult to deal with in terms of decoder designs (as we will show later) because its trapping sets are highly entangled in a way that is more involved than what we just described. For instance, every VN of the code belongs to six different (12,4) trapping sets. It is for this reason we have studied the Margulis code: if we can devise decoder design techniques which lower the Margulis floor, it is likely that these techniques can lower the floor of any LDPC code.

In [13], Richardson proposed a semi-analytical method to predict the performance of LDPC codes in the floor region and he verified results for the Margulis code on the binary-input AWGN channel using FPGA simulations. He also showed that the 1320 (12,4) trapping sets account for about 75% of the error floor performance, and the (14,4) trapping sets account for about 23%. (Low-weight trapping sets are always more dominant than high-weight trapping sets.) In this paper, we use a software SPA decoder simulator which we have verified produces the same performance and prediction curves as those in [13].

2.2 The Short QC Code

The rate-0.3 (640,192) QC code is designed using a progressive edge growth-like (PEG-like) algorithm [3] and the approximate cycle extrinsic message degree (ACE) algorithm [4]. It has a circulant size $Q = 64$ and column weight $w_c = 5$. The $\mathbf{H}$ matrix is a 7×10 array of 64×64 circulant permutation matrices and zero matrices. We observed in our simulations two trapping set classes with 64 isomorphic trapping sets in each class, with a representative from each class shown in Fig. 1. The (5,5) trapping set is the dominant one, as it contributes to over 90% of the error floor level.

3 Bi-Mode Syndrome-Erasure Decoder

In [9], the authors proposed a post-processing technique to lower the error floor by using a look-up table of known trapping sets. After conventional SPA decoding, this table is used to process the residual error blocks much like a syndrome decoder. A drawback to this approach is, if the cardinality of trapping sets is large, the implementation complexity of table look-ups may be very costly. Inspired by this technique, we propose another post-processing syndrome decoder with two decoding modes, but which avoids look-up-table decoding. In our technique, post-processing involves simple graph-based erasure decoding.

In the error-floor region, there are three types of error events: (1) unstable error events, which dynamically change from iteration to iteration and for which w and v are typically large; (2) stable trapping sets, for which w and v are typically small and thus are the main cause of the error floor; and (3) oscillating trapping sets, which periodically vary with the number of decoder iterations and

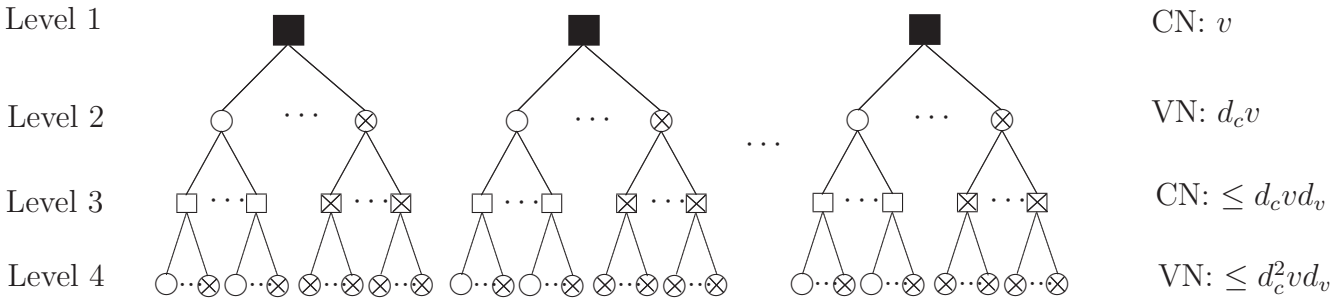

Figure 2: Erasure flagging trees. $\otimes$ = VN outside trapping set. $\boxtimes$ = CN outside trapping set.

which are sometimes subsets of stable trapping sets. The targets of the bi-mode decoder are the dominant stable trapping sets and some of the dominant oscillating trapping sets. In the first mode, SPA decoding is performed with a sufficient number of iterations for the decoder to reach one of the three error event situations just listed. The second mode is the post-processing mode, which is activated only when the syndrome weight of error event falls into the set of syndrome weights of the target trapping sets. The key role of the second mode is, using syndrome information, to produce an erasure set that contains all of the VNs of the trapping set reached by the decoder, thus resulting in a pure *binary erasure channel* (BEC) with only correct bits and erasures. Iterative erasure decoding based on the LDPC code's graph can then resolve all of the erasures, including the bits that were originally part of the trapping set error event.

It was observed that in the error-floor region, after running extensive Monte Carlo simulations, most of the error patterns correspond to the so-called *elementary trapping sets* [14], whose induced subgraphs have only degree-one and degree-two CNs. That is, for an elementary trapping set, the unsatisfied CNs are usually connected to the trapping set exactly once. We make this assumption during the discussion of the bi-mode decoding algorithm. Assume also that the decoder converges to a trapping set and suppose an unsatisfied CN has degree d_c. The goal of that CN is to find which one of the d_c neighbors is in error. Our experiments have shown that the LLR magnitude of the bit in error is not necessarily the smallest among the d_c VNs connected to the unsatisfied CN. Thus, we propose flagging as erasures all of the d_c neighbors of the unsatisfied CN and then performing iterative erasure decoding in the neighborhood of that unsatisfied CN.

The set of erasures is generated by erasing the VNs of the trees whose roots are the unsatisfied CNs, as in Fig. 2. The depth of the trees depends on the trapping set structures of the code being decoded. Consider a regular LDPC code with row weight d_c and column weight d_v. For a (w, v) trapping set, the number of nodes in each level of the trees is listed in Fig. 2. For a trapping set with $w \leq v$, *i.e.*, every VN is connected to at least one unsatisfied CN, v trees with two levels can cover the whole trapping set. Otherwise, when $w > v$, bigger trees are necessary. As the trees grow, the number of erasures grows exponentially and the whole trapping set will eventually be covered; however, the growth is not without limit, because too many erasures may form so-called *stopping sets*, and overwhelm the decoder. As will be explained below for the Margulis code, for some codes, additional steps are necessary to produce a smaller erasure set covering the whole trapping set.

3.1 Margulis Code Solution

We observed from simulations that the Margulis code with an SPA decoder is sometimes trapped in two oscillating (6,18) trapping sets or two oscillating (7,21) trapping sets. The union of the (6,18)

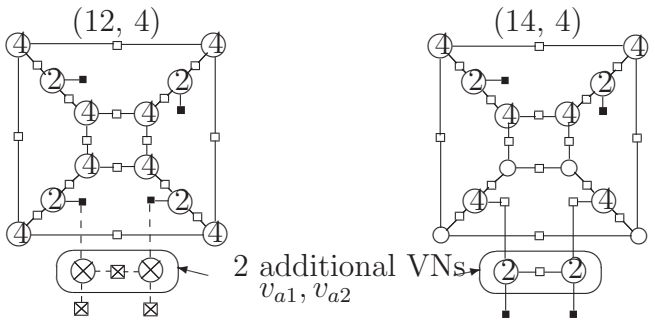

Figure 3: Margulis code erasure flagging (the numbers on the VNs are tree level numbers).

pair is a (12,4) trapping set, and that of the (7,21) pair is a (14,4) trapping set. In the decoder solution for this code, the target trapping sets include the stable trapping sets as well as the two oscillating configurations. It is obvious that the oscillating trapping sets and some moderate-length unstable error events satisfying $w \leq v$ can be flagged with two-level trees and recovered successfully. As depicted in Fig. 3, any (12,4) trapping set can be flagged and recovered using trees with four levels; however, four-level trees cover only 10 VNs of any (14,4) trapping set. If 6-level trees are used, the erasure decoder is overwhelmed by too many erasures.

To solve this problem, we explore the graphical structures of trapping sets. As shown in Fig. 1, a (14,4) trapping set has a similar structure as a (12,4) trapping set. In fact, any (14,4) trapping set contains a (12,4) subset plus two additional VNs. From the syndrome weight itself, the decoder cannot distinguish which of these two trapping set classes it is dealing with. The algorithm described below which resolves this issue is based on the following observation on the trees generated from the (12,4) and (14,4) trapping sets: As seen in Fig. 3 for either trapping set, there exists exactly two VNs in the second level which share a common level-3 CN. These two VNs are what distinguish the (12,4) and (14,4) trapping sets (examine Fig. 3). Toggling the values of these VNs will switch a (12,4) trapping set to a (14,4) trapping set, and vice versa.

Algorithm 1 Auxiliary Algorithm for (12,4) and (14,4) Trapping Sets

1. Take the four unsatisfied CNs and create four trees each of four levels with these CNs as roots.
2. Erase all the VNs in level-2 with flag "A", and all the VNs in level-4 with flag "B".
3. For each level-3 CN, count the number of its neighboring type-A VNs. Once a CN c_a, with two neighboring type-A VNs (v_{a_1} and v_{a_2}), is found, stop the search and go to Step 4.
4. Flip the bit values of v_{a_1} and v_{a_2}, and re-calculate the syndrome, producing a new set of four unsatisfied CNs.
5. Repeat Steps 1 and 2 with the new syndrome information and then go to Step 6.
6. Including all the VNs with both A and B type erasure flags, perform iterative erasure decoding in that neighborhood until all of the erasures are resolved.

Note that c_a, v_{a_1} and v_{a_2} are unique for any (12,4) or (14,4) trapping set. In the case when a (12,4) trapping set is reached in the first decoding mode, Steps 1 and 2 erase the whole trapping set,

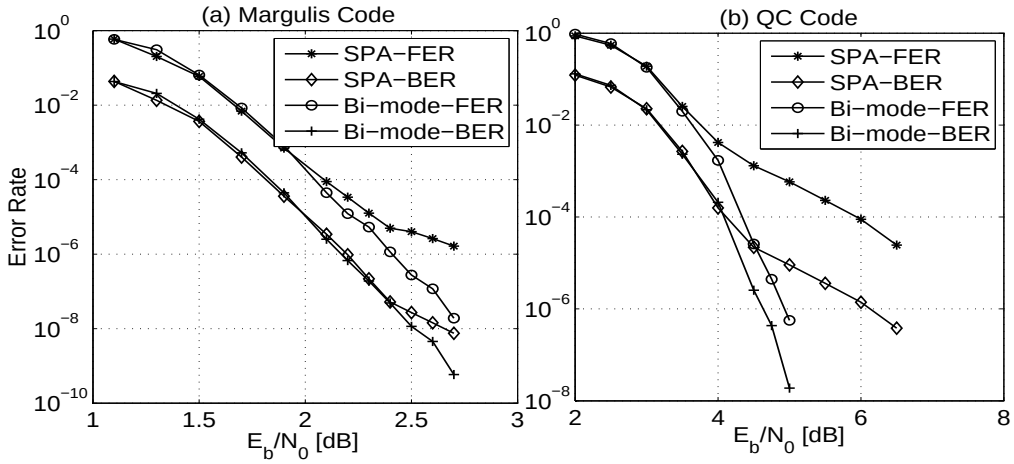

Figure 4: Performance of bi-mode decoders.

and Steps 3-5 are not necessary; however, Steps 3-5 are still performed because the decoder cannot distinguish (12,4) and (14,4) trapping sets. To a (12,4) trapping set, these extra steps simply flag as erasures more VNs that are not in error without overwhelming the erasure decoder. 352 erasures will be produced for any (12,4) or (14,4) trapping set, which can be recovered by the LDPC code successfully within 3 or 4 iterations. The performance results for this bi-mode decoder with the Margulis code is presented in Fig. 4(a). No floor is observed down to FER $\sim 10^{-8}$.

3.2 Short QC Code Solution

For both trapping set classes of the short QC code, every VN is associated with at least one unsatisfied CN ($w \leq v$). Hence, one level of VNs per tree is enough to include a whole trapping set in the erasure set. Whenever the SPA decoder gets trapped in an error event with syndrome weight 5 or 7, the decoding enters the second mode. 34 erasures are flagged for any (5,5) trapping set and 54 are flagged for any (5,7) trapping set, all of which can be recovered with one erasure decoding iteration. The performance of this code with bi-mode decoder is presented in Fig. 4(b). No floor is seen down below FER $\sim 10^{-6}$, so the floor is at least two orders of magnitude lower.

We remark that the bi-mode decoder has the following advantages relative to other floor-lowering techniques: (1) Beside stable trapping sets and oscillating trapping sets, this technique can also handle some unstable error events with $w \leq v$; (2) No outer codes are employed, so the gain is achieved without any code rate loss; and (3) The erasure decoding post-processing has very low computational complexity, which is equivalent to solving linear equations with binary unknowns, and thus involves only a series of binary XORs.

4 Generalized-LDPC Decoder

In this section, we propose novel SPA decoders which, loosely speaking, are designed by transforming the LDPC code into a generalized LDPC (G-LDPC) code. A G-LDPC code, like an LDPC code, is a code that can be described by a sparse bipartite graph with variable nodes and constraint nodes. However, for G-LDPC codes the constraints may be more general than single parity-check (SPC) constraints. For example, a constraint node can represent an arbitrary (n', k') binary linear code.

To see how the G-LDPC philosophy arises, recall that each constraint node decoder in an SPA decoder is locally optimum. It is possible in principle to group all of the SPC constraints into one combined global constraint and design a decoder for that graph. But that would be an ML decoder (for example), which has unacceptable complexity. Our strategy is to instead take one step toward that ideal and cleverly combine only a few SPC-CNs at a time. Specifically, we combine those CNs corresponding to unsatisfied checks in the problematic trapping sets and call the combination a *super-CN*. Observe that an advantage of doing so is that cycles and other deleterious graphical structures (from the perspective of an iterative decoder) may be removed.

The constituent decoder for the super-CN can be any soft-input/soft-output decoder. We use a (locally) optimal super-CN decoder which employs the BCJR algorithm designed to the "BCJR trellis" [15] for the linear code represented by the super-CN. We allow both standard CNs and super-CNs to co-exist in the generalized Tanner graph. The G-LDPC decoder passes soft information iteratively between VNs and (super-)CNs in the same manner as the SPA decoder.

Below we present three different G-LDPC decoders (equivalently, methods for grouping SPC-CNs into super-CNs). Due to space limitations, we only present the solution for the Margulis code. The approaches are based on the knowledge of the dominant trapping sets. The goal of the SPC-CN grouping methods is to eliminate the dominant trapping sets, thus lowering the error floor. In selected cases, we predict the FER performance in the floor region using the method in [13].

4.1 G-LDPC Decoder I

The idea behind the Type I G-LDPC decoder is to mutually fortify the unsatisfied CNs in the dominant trapping sets. Because the Margulis code trapping sets are highly overlapped (intersections are non-empty), we chose to require that no two super-CNs share any common SPC-CNs. We first grouped the four unsatisfied check nodes in the 198 non-overlapping (12,4) trapping sets into 198 super-CNs. We also grouped the four unsatisfied check nodes in the remaining 22 non-overlapped (14,4) trapping sets into another 22 super-CNs. In addition to these 220 super-CNs (66.7% of all CNs), each of which is a composition of four SPC-CNs, the generalized Tanner graph also has 440 standard SPC nodes. At each super-CN, the BCJR algorithm was applied to the BCJR trellis for the four associated check equations. The SPC-CNs used standard SPA processing.

We ran Monte Carlo simulations on the generalized Tanner graph and the performance in terms of frame error rate (FER) and bit error rate (BER) are shown in Fig. 6. We observe in the figure that the error floor is lowered by an order of magnitude compared to the standard SPA decoder. The simulator ran 220 iterations in order to obtain the stable trapping sets. No (12,4) or (14,4) trapping sets were observed by the G-LDPC I decoder. The new trapping sets of the G-LDPC I decoder, listed in Fig. 5, are transformed from the original ones by deleting/adding one VN from/to the (12,4) or (14,4) trapping sets. Given the algebraic construction of the Margulis code, we can find 1320 subgraphs of each configuration in Fig. 5 in the original graph. However, from the G-LDPC decoder's point of view, not all of the 1320 subgraphs are isomorphic. For example, a (13,5) subgraph in the original Tanner graph, is a dominant trapping set for the G-LDPC I decoder if and only if its five unsatisfied CNs are not involved in any super-CN and the additional VN to a (12,4) is attached to the rest of the subgraph through a super-CN. Similarly for the other configurations in Fig. 5. This is the reason that the multiplicities of the G-LDPC trapping sets are much smaller.

These observations allow us to find a complete list $\mathcal{T}$ of the new trapping sets that are related to the original ones by searching the generalized Tanner graph, or more efficiently by using a two-step importance sampling (IS) [16][17] method which we will describe in Section 4.4. Fig. 5 lists the multiplicities of the dominant G-LDPC I trapping sets. The contribution of each trapping set

G-LDPC I			G-LDPC II		
Trapping Set	no.	FER Floor (%)	Trapping Set	no.	FER Floor (%)
(11, 5)	22	17.4%	(14, 6)	13	54.3%
(13, 5)	77	48.7%	(14, 8)	4	1.0%
(14, 6)	77	4.1%	(15, 7)	11	8.4%
(15, 5)	99	27.3%	(16, 6)	11	26.2%
(15, 7)	22	0.3%	(16, 8)	18	5.4%
(16, 6)	33	1.1%	(17, 7)	10	1.4%
(17, 5)	22	1.2%	(18, 8)	11	3.2%

Figure 5: The dominant trapping set classes of Margulis code under G-LDPC I and II decoders.

class to the FER floor was evaluated by the technique in [13]. These results demonstrate that the G-LDPC I decoder eliminates the (12,4) and (14,4) trapping sets as problems. Although new dominant trapping sets arise, the new trapping set configurations are larger in weight and have much smaller multiplicities, and are thus less harmful. The floor prediction curves in Fig. 6(a) are obtained by weighted sums of the error rates of each type in the table.

4.2 G-LDPC Decoder II

The G-LDPC II decoder further targets the trapping sets emerged in G-LDPC I decoder. Note that only 66.7% of the CNs were combined in G-LDPC I. Therefore, we can group appropriately selected CNs into more super-CNs. According to Fig. 5, three trapping set classes amount to over 90% of the floor of G-LDPC I: (11,5), (13,5) and (15,5). We combined the five non-overlapping unsatisfied SPC-CNs of these trapping sets, arriving at a G-LDPC graph with 229 SPC-CNs and 258 super-CNs. 82.6% of the SPC-CNs are combined in this decoder. The trapping sets observed for the G-LDPC II decoder can also be enumerated using the method in Section 4.4 (results shown in Fig. 5). A prediction curve is drawn in Fig. 6(a), which is over two orders of magnitude lower than the standard decoder. This procedure of adding more super-CNs according to new trapping sets may continue until no SPC-CNs remain in the generalized graph or the system's floor performance requirement is achieved. Of course, the improvement is at the cost of increasing complexity. This realization led us to the G-LDPC III decoder.

4.3 G-LDPC Decoder III

It was observed through simulations of many LDPC codes that, in the floor region, a frame error event usually contains a single trapping set. Instead of combining unsatisfied CNs within a trapping set, the G-LDPC III decoder takes trapping sets in pairs and combines an unsatisfied CN of one trapping set to that of the other. When the decoder is trapped in one trapping set, reliable information from its companion trapping set will be passed along the super-CN, allowing recovery from the trapping set error event.

For the Margulis code, corresponding to the 1320 (12,4) trapping sets (660 pairs), 660 super-CNs are formed each consisting of two SPC-CNs. Due to the overlap of trapping sets (for example, every CN is one of the unsatisfied CNs in four different trapping sets), all the trapping sets form a linked overlapped network. The IS method in Section 4.4 was applied to this decoder and no trapping sets were observed. The effectiveness of this simple G-LDPC decoder is also confirmed by simulations, which shows no floor down to FER $\sim 10^{-8}$.

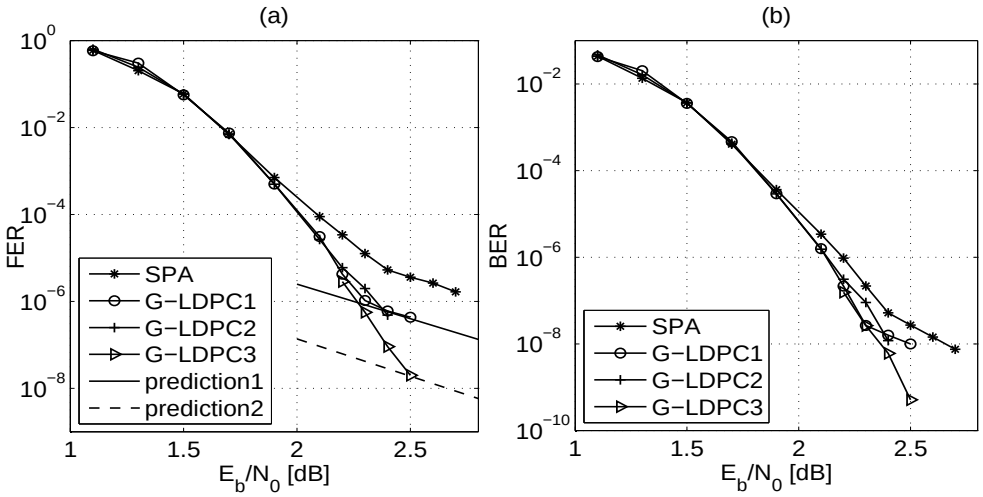

Figure 6: Performance of Margulis code with G-LDPC decoders.

4.4 Finding The Trapping Sets of G-LDPC Decoders

This section describes the two-step IS method for finding the dominant trapping sets of G-LDPC decoders ("G-LDPC trapping sets") starting with the trapping sets obtained from SPA decoder ("SPA trapping sets"):

(a) *Identify the SPA trapping sets that lead to G-LDPC trapping sets*: Without loss of generality, we assume all-zero codewords are transmitted (all +1 words on the channel). Apply a bias of -2 to each VN of every SPA trapping set. Then observe the G-LDPC decoder failures (no noise is added, only the biases). If the decoder clears all the errors or an unstable error event is obtained, then we assume that the SPA trapping set leads to no G-LDPC trapping sets. Otherwise, record the stable patterns.

(b) *Find all relevant trapping sets*: The recorded patterns from Step (a) are not in the exact forms of G-LDPC trapping sets on AWGN channel yet. They are used as initial conditions in this step. Based on these patterns, we bias each location with a Gaussian random variable (RV) with mean -2 and variance corresponding to an SNR value in the floor region. We then observe the G-LDPC decoder outputs. The stable low-weight error events are the trapping sets of interest. Their multiplicities are also obtained. Note that one pattern from Step (a) may lead to several stable trapping sets, thus multiple realizations of the RV need to be observed.

The advantage of applying the two-step IS instead of Step (b) alone is: When biasing with Gaussian noise, multiple G-LDPC decoding instances are performed. It could be very time-consuming because every SPA trapping set has to be tested and the cardinality may be large (for example, there are 2640 SPA trapping sets to be tested for the Margulis code).

References

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

[2] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, pp. 2711–2736, Nov. 2001.

[3] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth tanner graphs," in *Proc. of IEEE GLOBECOM*, vol. 2, pp. 995–1001, Nov. 2001.

[4] T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, "Construction of irregular LDPC codes with low error floors," in *Proc. IEEE Int. Conf. Comm.*, vol. 5, pp. 3125–3129, May 2003.

[5] J. Xu, L. Chen, I. Djurdjevic, S. Lin, and K. Abdel-Ghaffar, "Construction of regular and irregular LDPC codes: geometry decomposition and masking," *IEEE Trans. Inf. Theory*, vol. 53, pp. 121–134, Jan. 2007.

[6] W.-Y. Weng, A. Ramamoorthy, and R. D. Wesel, "Lowering the error floors of irregular high-rate LDPC codes by graph condition," in *Proc. 60th Vehic. Tech. Conf.*, vol. 4, pp. 2549–2553, 2004.

[7] D. Divsalar and C. Jones, "Protograph based low error floor LDPC coded modulation," in *IEEE Military Comm. Conf.*, vol. 1, pp. 378–385, Oct. 2005.

[8] S. K. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *IEEE Int. Conf. Comm.*, June 2006.

[9] E. Cavus and B. Daneshrad, "A performance improvement and error floor avoidance technique for belief propagation decoding of LDPC codes," in *16th IEEE Int. Symp. on Personal, Indoor and Mobile Radio Comm.*, pp. 2386–2390, Sept. 2005.

[10] S. Lander and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Int. Conf. on Wireless Networks, Communications and Mobile Computing*, vol. 1, pp. 630–635, June 2005.

[11] M. Chertkov, "Reducing the error floor," in *2007 ITW*, Sept. 2007.

[12] D. MacKay and M. Postol, "Weaknesses of margulis and ramanujan-margulis low-density parity-check codes," *Electronic Notes in Theoretical Computer Science*, vol. 74, 2003.

[13] T. Richardson, "Error floors of LDPC codes," in *Proc. 41st Allerton Conf. on Communications, Control, and Computing*, (Allerton House, Monticello, Illinois, USA), Oct. 2003.

[14] 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, pp. 39–55, Jan. 2007.

[15] R. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inf. Theory*, pp. 1072–1092, July 1996.

[16] E. Cavus, C. Haymes, and B. Baneshrad, "A highly efficient importance sampling method for LDPC codes using trapping sets." Submitted to *IEEE Trans. Comm.*

[17] C. A. Cole, S. G. Wilson, E. K. Hall, and T. R. Giallorenzi, "A general method for finding low error rates of LDPC codes." Submitted to *IEEE Trans. Comm.* (Available: http://arxiv.org/abs/cs/0605051), June 2006.