Breaking the Trapping Sets in LDPC Codes: Check Node Removal and Collaborative Decoding

Soonyoung Kang, *Student Member, IEEE*, Jaekyun Moon, *Fellow, IEEE*, Jeongseok Ha, *Member, IEEE*, and Jinwoo Shin

***Abstract*—Trapping sets strongly degrade performance of low-density parity check (LDPC) codes in the low-error-rate region. This creates significant difficulties for the deployment of LDPC codes to low-error-rate applications such as storage and wireless systems with no or limited retransmission options. We propose a novel technique for breaking trapping sets based on collaborative decoding that utilizes two different decoding modes. While the main decoding mode executes message passing based on the original parity check matrix of the corresponding LDPC code, the sub-decoding mode operates on a modified parity check matrix formed by removing a portion of check nodes in the factor graph representation of the given code. The modified parity check matrix is designed to promote a passing of correct information into erroneous variable nodes in the trapping set. Theoretical properties of the proposed trapping-set-breaking technique have been established based on the notion of the improved separation for the trapped variable nodes. Simulation results show that the proposed collaborative LDPC decoding scheme switching between the two decoding modes back and forth effectively breaks dominant trapping sets of various known types of regular and irregular LDPC codes.**

***Index Terms*—Low-density parity-check codes, message passing, error floor, trapping set, collaborative decoding.**

I. Introduction

THE LDPC codes [1], [2] decoded by message-passing algorithms have excellent error correcting capabilities [3]. A wide variety of communication systems including Wi-Fi, 10G Ethernet and digital video broadcasting have adopted LDPC codes as part of their standard specifications. However, the LDPC codes tend to exhibit error floors in their error rate curves as the signal-to-noise-ratio (SNR) increases. This characteristic behavior presents serious issues in applications that target very low error rates like wireless systems where data retransmission is highly undesirable due to low latency requirements or storage devices where retransmission is simply not an option. It has been observed that the error floor is mainly caused by special sets of hard-to-correct error events, called *near codewords*, *pseudo codewords* or *trapping sets* [4]–[6].

Manuscript received April 26, 2015; revised August 7, 2015 and October 21, 2015; accepted October 28, 2015. Date of publication November 5, 2015; date of current version January 14, 2016. This work was supported by NRF of Korea under Grant 2010-0029205 and SK hynix-KAIST Storage Media Solutions Center. The associate editor coordinating the review of this paper and approving it for publication was D. Declercq.

The authors are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, South Korea (e-mail: soonyoung@kaist.ac.kr; jmoon@kaist.edu; jsha@kaist.edu; jinwoos@kaist.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2015.2498161

Several methods for lowering error floor using decoder modifications have been introduced. In [8], the bi-mode, bit-pinning and generalized-LDPC (G-LDPC) decoding strategies are discussed. While the bit-pinning method requires additional outer coding based on the prior knowledge of the trapping sets, the other two methods focus on decoder modification and thus can be used to decode any LDPC coded signals in principle, provided the trapping sets are already characterized.

The two-step decoding strategy described in [20] has also been shown to lower the error floor of certain regular LDPC codes. In the first step, a conventional message-passing decoder runs through a fixed number of iterations. If decoding fails in the first step, some VNs near unsatisfied CNs are collected into a set and are allowed to receive fixed, magnified messages from unsatisfied CNs and fixed, attenuated messages from satisfied CNs. Again, this method requires prior knowledge of the trapping set in determining the bias levels. Note that characterizing trapping sets in itself is a very difficult task [15], [16]. McGregor and Milenkovic [15] have shown that even approximating the size of trapping sets in the Tanner graph, let alone evaluating them, is an NP-hard problem.

Planjery *et al*. introduced a finite alphabet iterative decoder (FAID) [27] wherein the VN-to-CN message is not updated as in the conventional message passing algorithms but instead assigned a value from a finite alphabet according to the given CN-to-VN messages and channel output value. Assigning the finite alphabet to a specific VN-to-CN message requires the knowledge of the trapping sets in the given code's factor graph. The results reported in [27] show that for a number of column-weight-three codes a 3-bit FAID yields better performance in the error floor region than the floating point belief propagation decoder over the binary symmetric channel (BSC). This scheme, however, requires knowledge of the trapping sets; in addition its applicability to more general additive white Gaussian noise channel (AWGNC) is not clear.

A two-bit bit-flipping (TBF) algorithm of [26] uses an extra bit for representing the VN-to-CN message strength. Another two-bit information is also employed at each CN for storing its previous and present states. Different bit-flipping functions have also been devised in [26] based on the knowledge of the trapping sets. The main idea is that the multiple functions run sequentially or in parallel with each function targeting a different set of errors. Again, the TBF scheme is applicable to BSC only, and it also depends on the trapping set knowledge.

The backtracking method of [9] is able to lower the error floors of various types of LDPC codes *without* prior knowledge of the trapping sets. In this approach, two backtracking steps are implemented if the message passing decoder fails. At the first backtracking step, for every adjacent (directly connected) VN of unsatisfied CNs, the sign of the channel log-likelihood ratio (LLR) is flipped one by one. Since each flipping is followed by an independent run of message passing, the total number of decoding iterations depends on the number of unsatisfied CNs, the node degree of each unsatisfied CN and the number of iterations in each independent run of message passing. The second backtracking, which ensues if the first backtracking step fails, is based on a more narrow selection of VNs for flipping, but each flipping still requires independent message passing. The backtracking is a very efficient technique for lowering error floors if it is applied to trapping sets with a small number of unsatisfied CNs. However, for some regular codes of which the dominant trapping sets have a large number of unsatisfied CNs, the backtracking technique requires a huge number of flipping trials and decoding iterations. In a subsequent work [10], the same authors introduce two techniques called early trap detection and variable node degree based flipping. While the former cuts down the number of decoding iterations, the latter reduces the number of flippings. However, the required number of iterations for the backtracking is still large. The experimental result in [10] shows that, for example, more than 300 decoding iterations are required to lower the error floor of the ($n = 1944$, $k = 972$) IEEE 802.11n LDPC code [13] by an order of magnitude.

In [22], Zhang and Siegel established the condition for breaking trapping sets[1] on the BSC and the AWGNC using message-passing decoders under some reasonable assumptions. Consider a computation tree [24] with an incorrect VN of a trapping set as root. If the root has at least one adjacent CN with degree-one (in the subgraph associated with a trapping set) that does not have any incorrect VN among all its own descendant nodes within k decoding iterations, then the root VN is said to be *k-separated*. Note that a degree-one CN adjacent to a trapped erroneous VN is an unsatisfied CN. It is shown in [22] that if all incorrect VNs in the trapping set are *k-separated* for sufficiently large k, then the errors in the trapping set are successfully corrected by a message-passing decoder under a mild assumption (in the low-error-rate regime) that all VNs outside the given trapping set receive uniformly reliable messages from the channel. This successful linking of the "separation" of the trapped VNs to the ability to escape from the trapping set provides a powerful key to developing effective trapping-set-breaking solutions, including the method presented in this paper. Unfortunately, however, the VNs in the dominant trapping sets of the LDPC codes in practice are often not sufficiently separated. For example, the VNs in the (12,4) trapping set of the Margulis code [6] are only 2- or 3-separated while all VNs in the (8,8) trapping set of the ($n = 2048$, $k = 1723$) Reed-Solomon (RS) based LDPC code [14] are just 1-separated.

A non-uniform quantization scheme referred to as *quasi-uniform* quantization is also introduced in [22]. In this scheme, $(q + 1)$-bit quantization utilizes 2^q uniform quantization intervals as well as 2^q exponentially-widening quantization intervals, effectively enlarging the dynamic range of the quantization process without increasing the bit resolution. While this scheme appears highly effective in lowering error floors in many different types of LDPC codes, in order for the method to work well, the internal bit resolution needed in the computation of the sums of the LLRs in the VN-to-CN message passing stage has to be maintained at a level much higher than required by the $(q + 1)$-bit quantization scheme. For this reason, performance advantages of the quasi-uniform quantization method over conventional uniform quantization are not clear when the same bit resolution is maintained in the computation of the intermediate soft-decision quantities within the message-passing operation.

The contributions and distinguishing features of our work are as follows. In relation to the work of [22], we introduce in this paper algorithms for *extending* the VN separation k by modifying the parity-check matrix. We show that by strategically removing some CNs near an incorrect VN, k can be extended. Since it is not possible to know the location of incorrect VNs *a priori*, we introduce two probabilistic approaches for generating the modified parity-check matrix. In applying the proposed schemes, an irregular modified factor graph may arise whether the original factor graph is regular or irregular. Since the *k-separation* condition of [22] can guarantee the correction of trapped VNs in the regular factor graphs only, we introduce a related notion of *k-strong-separation* for establishing sufficient conditions to break the trapping sets in *irregular* factor graphs as well. For practical application of the devised CN removal strategies, we suggest collaborative decoding using both the original parity-check matrix and the modified parity-check matrix. Message-passing on the modified parity-check matrix allows escape from the trapping set whereas the original parity-check matrix ensures reliable recovery across all codewords. Simulation results show that our method based on switching back and forth between the two decoding modes matched to two parity-check matrices effectively lowers the error floors of all regular and irregular LDPC codes tested. The proposed scheme can be implemented as two separate collaborating decoders or as a single decoder operating under two different modes corresponding to the original and the modified factor graph. Similar to the use of multiple FAIDs in [25] and the TBF of [26], our method can be viewed as utilizing multiple decoders for correcting trapping sets. However, again we stress that our method does not require prior knowledge of the trapping sets. In addition, while the FAID and TBF have been applied only to BSC, we confirm that our scheme works well on AWGNC. Compared to the scheme of [20], although our method requires more decoding iterations for the same improvement on the error floor, the scheme in [20] requires prior knowledge of the trapping set for selecting an appropriate value for biased message. Moreover, since the bias value is determined by the structure of a single trapping set, the scheme in [20] does not guarantee performance on the codes having various trapping sets such as irregular LDPC codes. In addition, our method has been validated against a wider variety of LDPC code types, which is important as not many known error floor reductions algorithms are robust enough to work well for different types of codes. Compared to the backtracking scheme

[1]Throughout this paper, the "breaking" of a trapping set is taken to mean correction of one or more trapped VN errors, which is a prerequisite for eventual correction of the entire VNs in the trapping set.

of [10], we achieve the same order of error floor reduction using a considerably smaller number of extra decoding iterations.

The paper is organized as follows. In Section II, we provide a quick overview on the message passing algorithm, the trapping set and the separation of the VNs in the trapping set. In Section III, we introduce the parity-check matrix modification techniques that can correct the trapped VNs. The proposed methods are presented in the form of theorems and propositions. In Section IV, we describe collaborative decoding utilizing both the original parity check matrix and the reduced modified parity check matrix. Experimental results are given in Section V. Conclusions are drawn in Section VI.

II. Preliminaries

A. Message Passing Algorithm

For the sake of maximum readability as well as establishing notations, we start with a quick overview of the message-passing decoding. The message passing algorithm for the LDPC codes is run iteratively in such a way that the CNs and the VNs exchange the messages at every iteration. The overall message passing algorithm is basically separated into two steps: the CN-to-VN message passing and the VN-to-CN message passing. In the VN-to-CN message passing, a VN sends information to a particular CN, as it collects information from all other CNs it is connected to. In this way, when a VN passes information to a CN, it is ensured that there is no information in that message that has come from the same CN.

Let $L_i^{(0)}$ denote the channel observation for VN i in the form of log-likelihood ratio (LLR) defined as

$$L_i^{(0)} = \log \frac{\Pr(x_i = 0|r_i)}{\Pr(x_i = 1|r_i)} \tag{1}$$

where x_i and r_i represent the coded bit and the channel output, respectively, for VN i.

Let $q_{i\to j}^{(l)}$ be the message that VN i passes onto CN j at lth iteration, and $s_{j\to i}^{(l)}$ the message that CN j sends to VN i at lth iteration. Then the message sent from VN i to CN j is given by

$$q_{i\to j}^{(l)} = L_i^{(0)} + \sum_{k\in M(i)\setminus j} s_{k\to i}^{(l-1)} \tag{2}$$

where the set $M(i)$ is the group of CNs tied to VN i and $M(i) \setminus j$ means the set $M(i)$ excluding CN j.

The message from CN j to VN i in the sum-product algorithm (SPA) version of the message-passing is described as

$$s_{j\to i}^{(l)} = \prod_{k\in N(j)\setminus i} \operatorname{sign}\left(q_{k\to j}^{(l)}\right) \cdot \phi\left(\sum_{k\in N(j)\setminus i} \phi\left(\left|q_{k\to j}^{(l)}\right|\right)\right) \tag{3}$$

where $\phi(x) = -\log[\tanh(x/2)]$, the set $N(j)$ is the group of VNs tied to CN j and $N(j) \setminus i$ means the set $N(j)$ excluding VN i.

Although the min-sum algorithm (MSA) based decoder requires a less computational power than the SPA-based decoder, its error correcting performance is worse than that of the latter. There is a simple technique for enhancing performance of the MSA called the scaled (or attenuated) min-sum algorithm (SMSA) based on a scaling-down of the magnitude of the CN-to-VN messages.

The message from CN j to VN i for the SMSA is described as

$$s_{j\to i}^{(l)} = \prod_{k\in N(j)\setminus i} \operatorname{sign}\left(q_{k\to j}^{(l)}\right) \cdot \alpha \cdot \min_{k\in N(j)\setminus i} \left|q_{k\to j}^{(l)}\right| \tag{4}$$

where $0 < \alpha < 1$ is the scaling factor. When $\alpha = 1$, (4) reduces to the CN-to-VN message passing for the MSA.

After computing $q_{i\to j}^{(l)}$ and $s_{j\to i}^{(l)}$ for all i and j, the overall message for VN i can be calculated as

$$L_i^{(l)} = L_i^{(0)} + \sum_{k\in M(i)} s_{k\to i}^{(l)} \tag{5}$$

where $L_i^{(l)}$ represents the LLR for VN i at the lth iteration.

B. Trapping Set and Its Separation Vector

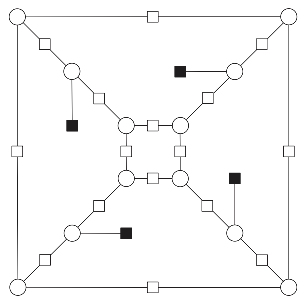
The trapping set was first defined in [4] as a set of VNs that do not get eventually corrected despite extensive decoding iterations. While a CN connected to an even number of incorrect VNs is always satisfied (or mis-satisfied, to be accurate), a CN connected to an odd-number of incorrect VNs is always unsatisfied.

An (a, b) trapping set is a set of a VNs which induces a subgraph that includes b unsatisfied CNs and any possible number of satisfied CNs. During iterative decoding, messages from the unsatisfied CNs try to change the states of their adjacent VNs to satisfy the parity constraints. On the other hand, the satisfied CNs tend to keep current hard decisions of the decoder associated with their adjacent VNs and reinforce the current states. The satisfied CNs in a trapping set, which are in fact mis-satisfied CNs, create a major issue and contribute to the error floor because they tend to maintain the current states of the nodes that reflect an incorrect codeword.

We use the term *elementary trapping set* which is known to be the main cause of the error floor [17]–[19]. Let $G = (V \cup C, E)$ denote the Tanner-graph of an LDPC code with the sets of VNs $V = \{v_1, v_2, \ldots, v_n\}$, of CNs $C = \{c_1, c_2, \ldots, c_m\}$ and of edges E. Let S be the subgraph induced by the trapping set and V_S be the set of VNs in S. Also, let C_S be the sets of CNs adjacent to the VNs in V_S. Define $C_1 \subset C_S$ to be the set of unsatisfied CNs with degree-one in S and $V_1 \subset V_S$ the set of adjacent VNs of the CNs in C_1.

Definition 1 (Elementary trapping set): A trapping set is an (a, b) elementary trapping set if there are a VNs inducing b unsatisfied CNs such that all CNs have either degree one or two in the corresponding subgraph S.

Fig. 1 shows two elementary trapping sets in actual LDPC codes. The circles are VNs, the filled squares are unsatisfied CNs and the empty squares are mis-satisfied CNs. The (12,4) and (8,8) trapping sets are the most dominant trapping sets of the (2643,1320) Margulis and the (2048,1723) RS-based LDPC codes, respectively [6], [21]. While four of the VNs in the (12,4)


(a) (12,4) trapping set and its subgraph

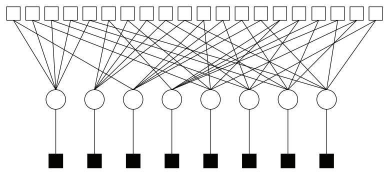

(b) (8,8) trapping set and its subgraph

Fig. 1. The most dominant trapping sets of the Margulis and (2048,1723) RS-based codes are the (12,4) and (8,8) trapping sets, respectively. Both are elementary trapping sets. The circles are erroneous VNs, the filled squares are unsatisfied CNs and the empty squares are mis-satisfied CNs.

trapping set are connected to degree-one CNs, all VNs in the (8,8) trapping set are connected to degree-one CNs. Since the elementary trapping sets are known to be the dominant cause of the error floors [17]; as in the developments of other known floor lowering techniques [18], [19], [27], we also focus on the elementary trapping sets. All trapping sets referred to in this paper are elementary trapping sets, unless noted otherwise.

A *computation tree* $T(v)$ is a tree with its root at VN v having as descendant nodes the participating CNs and VNs in the message-passing decoder [24]. The level of a computation tree is increased as the number of decoding iterations is increased. In particular, a subgraph of $T(v)$ having a set of nodes visited through k decoding iterations of a message passing decoder is called a *k-iteration computation tree* and denoted by $T_k(v)$ [22]. While the root of a computation tree is a VN in the original definition, it is useful for our purposes to make an obvious extension of the notion of the computation tree so that the root of a computation tree can also be a CN.

In [22], a useful notion of *separation* for a VN $v \in V_1$ in the trapping set V_S is introduced and a lower bound on the magnitude of the correct message sent from an unsatisfied CN to v has been established using this notion. The authors of [22] also show an upper bound on the incorrect message sent from a mis-satisfied CN to v. Given a VN $v \in V_1$, if there is at least one adjacent degree-one CN having no erroneous VN as a descendant node in $T_k(v)$, then v is said to be *k-separated*. The same authors show that if k is sufficiently large, the root VN can be corrected by the message-passing decoder over BSC or AWGNC.

However, the critical issue is that most of the dominant trapping sets in practice are composed of erroneous VNs with small separation. We define the *separation vector* $s(V_1)$ composed of the separation values k for all $v \in V_1$ in the trapping set V_S.

TABLE I
DOMINANT TRAPPING SETS AND THEIR SEPARATION VECTOR

Trapping Set	Code (n,k)	Separation Vector
(12,4)	Margulis (2640,1320)	(3, 3, 2, 2)
(14,4)	Margulis (2640,1320)	(3, 3, 2, 2)
(8,8)	IEEE 802.3an (2048,1723)	(1, 1, 1, 1, 1, 1, 1, 1)

The size of $s(V_1)$ is the same as $|C_1|$, the size of C_1, in subgraph S. Table I shows some representative LDPC codes and their dominant trapping sets along with the corresponding separation vectors. Note that, the separation vector is determined not only by the subgraph involving the trapping set but also by that of the graph structure outside the trapping set.

III. Parity-Check Matrix Modification

In this section, we introduce two algorithms for modifying the parity-check matrix for the purpose of enlarging the elements of the separation vector and hence promoting passing of correct information to erroneous VNs in the trapping set. A modified parity-check matrix in the proposed schemes is generated by removing some CNs in the original parity-check matrix. Note that certain mis-satisfied CNs determine the separation values of the trapping set and also cause the error floor.

In message-passing decoding, we can find the set of unsatisfied CNs, denoted by C_u, as well as its size $|C_u|$ at every decoding iteration. For the parity-check matrix modification, let $A(C_u)$ denote the set of CNs which are candidates for deselection in the original parity-check matrix. Let $A(C_u)$ be the set of all leaf CNs in $T_{t_i}(c_i)$, i.e., all CNs in level t_i, for all $c_i \in C_u$ for some $t_i \in \mathbf{t}$. We call $t_i \in \mathbf{t}$ the *modification length* for c_i and $\mathbf{t}$ the *modification vector*. If $\mathbf{t}$ is set equal to $s(V_1)$, then the particular mis-satisfied CNs that determine the separation vector would be included in $A(C_u)$, which is a desired scenario. For example, let V_1 be the (8,8) trapping set of the (2048,1723) RS-based code which has the separation vector $s(V_1) = (1, 1, 1, 1, 1, 1, 1, 1)$. In this case, every VN $v \in V_1$ is 1-separated. By setting $\mathbf{t} = (1, 1, 1, 1, 1, 1, 1, 1)$, $A(C_u)$ would include all CNs within 1 decoding iteration from an unsatisfied CN adjacent to each VN in V_1, and all mis-satisfied CNs causing 1-separation are automatically included in $A(C_u)$ (see Fig. 2). Now we suggest two strategies for modifying the parity-check matrix: one is referred to as fixed-number CN removal (FNCR) and the other is termed variable-number CN removal (VNCR).

Let H and H_m denote the original parity-check matrix and the modified parity-check matrix, respectively. Recall that when the LDPC decoder based on H is stuck in trapping set V_S, we can easily generate the computation trees $T(c_i)$, $c_i \in C_u$, based on the observed set of unsatisfied CNs C_u. The construction of H_m based on FNCR is described in Algorithm 1. We call d_f the *deselection degree* of FNCR.

Theorem 1: Let G be the Tanner graph of a regular LDPC code that contains a subgraph S induced by a trapping set. Let $k_r \in s(V_1)$ be the separation value for $v_r \in V_1 \subset S$. Assume that there is at least one VN in $T_{k_r}(v_r)$ which is not connected to the mis-satisfied CN that is the adjacent CN of incorrect VNs. If the modification vector $\mathbf{t}$ is equal to the separation vector

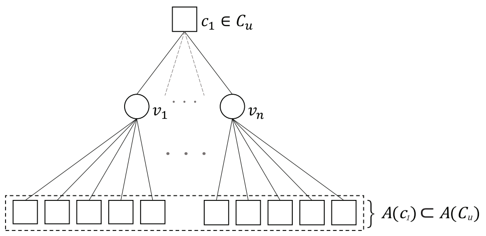


Fig. 2. A 1-iteration computation tree $T_1(c_1)$ with root at an unsatisfied CN c_1 in a (8,8) trapping set. Since c_1 is degree-one CN, one of the VN v_i, $i \in \{1, \ldots, n\}$ should be the VN of the trapping set. The set of CNs in $T_1(c_1)$ is denoted by $A(c_1) \subset A(C_u)$. Since all VNs in the (8,8) trapping set are 1-separated, $A(C_u)$ includes all mis-satisfied CNs causing 1-separation for the root VN. For FNCR, d_f child CNs are chosen for deselection for every v_i. For VNCR, every CN in $A(C_u)$ is subject to a Bernoulli trial for removal.

Algorithm 1. FNCR

```
initialization: Set $H_m$ identical to $H$. Set the modification vector $\mathbf{t}$.
for $i = 1$ to $|C_u|$ do
    Generate $T_{t_i}(c_i)$ based on $H$ where $c_i \in C_u$ and $t_i \in \mathbf{t}$, respectively;
    For every VN in the last level of $T_{t_i}(c_i)$, select $d_f$ child CNs randomly;
end for
Removed the selected CNs from $H_m$.
```

$s(V_1)$ and FNCR is run repetitively, there exists at least one $v_r \in V_1 \subset S$ for which separation will be increased within a finite number of runs as long as $1 \leq d_f < d_v - 1$, where d_v is the VN degree of the original parity-check matrix and d_f is the deselection degree of FNCR.

Proof: Consider a computation tree $T(c_r)$ with its root at CN c_r, where $c_r \in C_1$ is the adjacent unsatisfied CN of v_r. Let $n_{k_r}(c_r)$ be the number of mis-satisfied CNs in $T_{k_r}(c_r)$. From the definition of separation and the given assumption, we know that $n_{k_r-1}(c_r) = 0$, and less than or equal to $n_{k_r}(c_r)$ VNs are connected to one or more mis-satisfied CNs in $T_{k_r}(c_r)$.

For each VN that connected to several mis-satisfied CNs, the probability that the mis-satisfied CNs are removed by FNCR is given by

$$\binom{d_v - 1 - L}{d_f - L} \Big/ \binom{d_v - 1}{d_f} = \prod_{i=1}^{L} \left(\frac{d_f + 1 - i}{d_v - i} \right) \tag{6}$$

where L represents the number of mis-satisfied CNs connected to the certain VN. Then we can compute the probability that the separation of v_r is extended by FNCR:

$$p_s(v_r) = \prod_{i=1}^{q} \prod_{j=1}^{L_q} \left(\frac{d_f + 1 - j}{d_v - j} \right) \tag{7}$$

where q and L_q represent the number of VNs connected to several mis-satisfied CNs and that of the mis-satisfied CNs connected to the certain VNs. Note that, $L_1 + L_2 + \ldots + L_q = n_{k_r}(c_r)$.

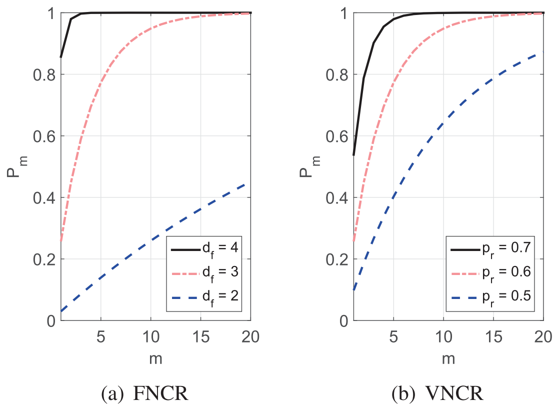


Fig. 3. Plots show probabilities of separation-extension for the (8,8) trapping set of the (2048,1723) RS-based code versus the number of trials for parity-check matrix modification based on FNCR and VNCR, respectively. m represents the number of trials for modification and P_m is the separation-extension probability given m trials.

We can further compute the probability that there exists at least one variable node $v_r \in V_1 \subset S$ for which separation is increased within m FNCR trials:

$$P_m = 1 - \left[\prod_{v_r \in V_1 \subset S} \{1 - p_s(v_r)\} \right]^m \tag{8}$$

Since $0 < p_s(v_r) < 1$, P_m goes to 1 as m increases, completing the proof. ■

We investigate how fast FNCR can extend the separation of erroneous VNs in the trapping set according to the specified d_f. Fig. 3(a) shows the number of FNCR trials for separation-extension of the VNs in the (8,8) trapping set of the (2048,1723) RS-based LDPC code. As shown in the figure, if d_f is set larger, the separation-extension probability gets higher and the speed of separation-extension becomes faster (see (8)). However, if d_f is too large, the general performance of a message-passing decoder will suffer. From (5), it is seen that the CN degree determines the magnitude of the overall LLR for a VN. Since a large d_f results in a small magnitude for the decoder output in a given iteration, setting d_f to a large value has a detrimental effect on the decoder performance given a fixed number of decoding iterations.

From the practical point of view, Algorithm 1 requires the modification vector which comes from the separation vector which in turn is from the knowledge of the trapping set. In the absence of the trapping set knowledge, however, we can simply start with an all-ones modification vector, the method which is justified by Corollary 1 below. We first establish a new definition on *weak CN*.

Definition 2 (Weak CN): Given a Tanner graph G and a subgraph S induced by a trapping set, consider a computation tree $T(c_r)$ with its root at CN c_r, where c_r is the adjacent unsatisfied CN of $v_r \in V_1 \subset S$. If a CN in the l^{th} level of $T(c_r)$ has the shortest path of all CNs in the same level to any $v \in V_S$ among its descendant nodes, then it is called "*weak CN* in the l^{th} level of $T(c_r)$".

It is obvious that there exists at least one weak CN in every level of the tree if separation is finite. Note that, since the nearest $v \in V_S$ from the unsatisfied CN c_r in $T(c_r)$ determines the separation value, removing weak CN results in separation extension. Corollary 1 shows that FNCR with all-ones modification vector removes the weak CN (i.e., the separation is extended) within a finite number of runs of FNCR.

Corollary 1: Let G be the Tanner graph of a regular LDPC code that contains a subgraph S induced by a trapping set. Assume that there is at least one VN in the first level of $T(c_r)$ that is not connected to the weak CN. If the modification vector **t** is set to all-ones vector and FNCR is run repetitively, there exists at least one $v_r \in V_1 \subset S$ for which separation will be increased within a finite number of runs as long as $1 \leq d_f < d_v - 1$, where d_v is the VN degree of the original parity-check matrix and d_f is the deselection degree of FNCR.

Proof: Consider a computation tree $T(c_r)$ with its root at c_r, where $c_r \in C_1$ is an adjacent unsatisfied CN of v_r. Let $w_1(c_r)$ be the number of weak CNs in the first level of $T(c_r)$. We can simply complete the proof using Theorem 1 by replacing $n_{k_r}(c_r)$ with $w_1(c_r)$ and (7) with

$$p_s(v_r) = \prod_{i=1}^{q} \prod_{j=1}^{M_q} \left(\frac{d_f + 1 - j}{d_v - j} \right) \tag{9}$$

where q and M_q represent the number of VNs connected to several weak CNs and that of the weak CNs connected to the certain VNs. Note that, $M_1 + M_2 + \ldots + M_q = w_1(c_r)$. ■

With a successful and sufficient separation, the resulting Tanner graph will allow a breaking of the trapping set. This is a rather direct consequence of Theorem 1 of [22] when applied to our modified factor graph, and we state it more formally as a proposition.

Proposition 1: Let G_m be the Tanner graph of an LDPC code which is based on a modified parity-check matrix generated by FNCR. Let S_m be the subgraph associated with the trapping set. Assume that the channel is either a BSC or an AWGNC, and that the messages from the channel to all VNs outside S_m are correct. Provided that a VN $v_r \in V_1 \subset S_m$ satisfies k-separation for a large k on G_m, the corresponding MSA (or SMSA) decoder will correct v_r.

Proof: The proof simply follows that of Theorem 1 in [22] after replacing d_v, the VN degree, with $d_v - d_f$, the reduced VN degree due to the application of FNCR in our case. ■

FNCR is simple, but unfortunately not suitable for some irregular LDPC codes such as the one based on the irregular-repeat-accumulate (IRA) [11] structure, which yield many degree-2 VNs. To this end, we introduce a more flexible algorithm called VNCR that can be easily applied to both regular and irregular codes. The construction of a modified parity-check matrix H_m based on VNCR is given in Algorithm 2. Recall that **t** is the modification vector which can be set once $s(V_1)$ of the trapping set is known or set to all-ones vector given no precise prior knowledge of the dominant trapping sets.

Theorem 2: Let G be the Tanner graph of an LDPC code that contains a subgraph S induced by a trapping set. Let $k_r \in s(V_1)$ be the separation value for $v_r \in V_1 \subset S$. Assume that there is at least one CN in $T_{k_r}(v_r)$ which is not connected to erroneous VN. If the modification vector **t** is set equal to the separation vector $s(V_1)$ and VNCR is run repetitively, there exists at least one $v_r \in V_1 \subset S$ for which separation is increased within a finite number of runs, provided $0 < p_r < 1$, where p_r is the probability of deselection.

Algorithm 2. VNCR

```
**initialization:** Set $H_m$ identical to $H$. Set the modification vector **t**.
**for** $i = 1$ to $|C_u|$ **do**
    Generate $T_{t_i}(c_i)$ based on $H$ where $c_i \in C_u$ and $t_i \in$ **t**, respectively;
    Let every leaf CN in $T_{t_i}(c_i)$ be subject to a Bernoulli trial that returns 1 with probability $p_r$; store its returned value;
**end for**
Remove the CNs with a returned value of 1 from $H_m$.
```

Proof: Consider a computation tree $T(c_r)$ with its root at CN c_r, where $c_r \in C_1$ be the adjacent unsatisfied CN of v_r. Let $n_{k_r}(c_r)$ be the number of mis-satisfied CNs in $T_{k_r}(c_r)$. We know that $n_{k_r-1}(c_r) = 0$ and $n_{k_r}(c_r) > 0$ from the given assumption and by the definition of separation. Since each mis-satisfied CN in $T_{k_r}(c_r)$ is subject to a Bernoulli trial with probability p_r, the probability that the separation for v_r is extended by VNCR is given by

$$p_s(v_r) = {p_r}^{n_{k_r}(c_r)}. \tag{10}$$

We can write the probability that there exists at least one variable node $v_r \in V_1 \subset S$ such that its separation is increased within m VNCR trials as

$$P_m = 1 - \left[\prod_{v_r \in V_1 \subset S} \{1 - p_s(v_r)\} \right]^m. \tag{11}$$

Since $0 < p_s(v_r) < 1$, P_m goes to 1 as m increases. This completes the proof. ■

Fig. 3(b) shows the number of VNCR trials for separation-extension of the VNs in the (8,8) trapping set of the (2048,1723) LDPC code. As p_r increases, the speed of separation-extension for VNCR gets higher. This behavior is consistent with one's expectation based on (10) and (11) above. However, a large p_r induces a small magnitude for the overall LLR for a VN in a given iteration. Thus, setting an appropriate p_r is important to maintain the decoder performance in a fixed number of message-passing iterations. While d_f for FNCR should be an integer in the range $1 \leq d_f < d_v - 1$, the parameter p_r for VNCR can be chosen more flexibly in the range $0 < p_r < 1$.

As with FNCR, we can apply VNCR with all-ones modification vector given the lack of the trapping set knowledge. Corollary 2 ensures that VNCR with all-ones modification vector extends the separation within a finite number of runs of VNCR.

Corollary 2: Let G be the Tanner graph of an LDPC code that contains a subgraph S induced by a trapping set. Assume that there is at least one CN in $T(c_r)$ which is not a weak CN. If the modification vector **t** is set equal to all-ones vector and VNCR is run repetitively, there exists at least one $v_r \in$

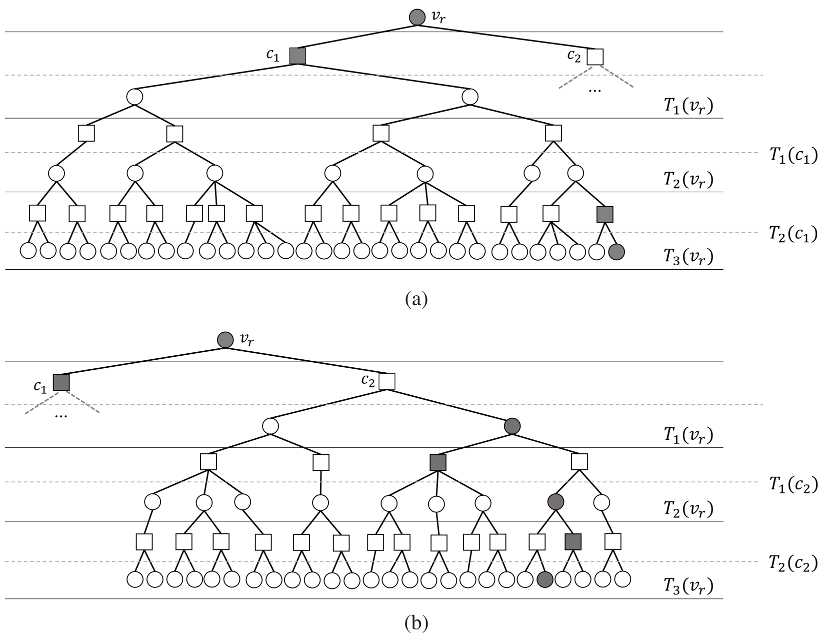


Fig. 4. Two sides of computation tree $T(v_r)$: (a) unsatisfied CN side and (b) mis-satisfied CN side.

$V_1 \subset S$ for which separation is increased within a finite number of runs, provided $0 < p_r < 1$, where p_r is the probability of deselection.

Proof: Consider a computation tree $T(c_r)$ with its root at CN c_r, where $c_r \in C_1$ be the adjacent unsatisfied CN of v_r. Let $w_1(c_r)$ be the number of weak CNs in the first level of $T(c_r)$. We can simply complete the proof using Theorem 2 by replacing (10) with

$$p_s(v_r) = {p_r}^{w_1(c_r)}. \quad (12)$$

■

Like in the case of FNCR, successful and sufficient separations of the trapped VNs using VNCR would allow escape from the trapping set. There is an important caveat here, however. Namely, unlike in the FNCR case, we cannot simply invoke Proposition 1 to assure correction of the trapped VN even under the successful and sufficient separation. This is because the application of VNCR inevitably generates an irregular modified factor graph, and a large separation alone would not be able to guarantee that the message arriving from the unsatisfied CN is larger than the overall messages from the mis-satisfied CNs for an irregular VN degree distribution. This is a consequence of the fact that Theorem 1 of [22], based on which Proposition 1 is constructed, cannot ensure trapped-error correction when VN degrees are not uniform. Nevertheless, while Proposition 1 cannot be utilized directly, imposing a somewhat narrower condition than the *k-separation* on the structure of the modified graph allows constructing a similar proposition. To this end, we first establish the following new definitions:

Definition 3 (k-strong): Given a Tanner graph G and a subgraph S induced by a trapping set, let $T_k(c_1)$ and $T_k(c_2)$ be subtrees of $T(v)$ where the unsatisfied CN $c_1 \in C_1$ and the mis-satisfied CN $c_2 \in (C_S \setminus C_1)$ are child nodes of VN $v \in V_1$. The VN v is said to be *k-strong*, if the minimum VN degree of $T_k(c_1)$ is greater than or equal to the maximum VN degree of $T_k(c_2)$.

Definition 4 (k-strongly-separated): Given a Tanner graph G and a subgraph S induced by a trapping set, a VN $v \in V_1$ is said to be *k-strongly-separated*, if the VN v is both *k-separated* and *k-strong*.

Fig. 4 illustrates these definitions. Consider an erroneous VN v_r and its computation tree $T(v_r)$ in Fig. 4. Both the unsatisfied CN side (i.e., the computation tree rooted at c_1) and the mis-satisfied CN side (i.e., the computation tree rooted at c_2, which is assumed to be the sole mis-satisfied CN in this example) are shown. On the unsatisfied side, $T_2(v_r)$ does not contain any other erroneous VN while $T_3(v_r)$ has one erroneous VN. Accordingly, v_r is *2-separated*. Also, notice that the minimum VN degree of $T_2(c_1)$ is 3, the same as the maximum VN degree of $T_2(c_2)$, meaning that v_r is *2-strong*. Hence, the erroneous VN v_r in Fig. 4 is *2-strongly-separated*.

Clearly, any VN in a regular LDPC code is *k-strong* for any k, because all VN degrees are the same. Therefore, if a trapped VN of any regular LDPC code is *k-separated*, it is also *k-strongly-separated*.

For a trapped VN that is *k-strongly-separated* for sufficiently large k, the message-passing algorithm can correct it, as formally stated below.

Proposition 2: Let G_m be the Tanner graph of an LDPC code which is based on a modified parity-check matrix generated by VNCR. Let S_m be the subgraph associated with the trapping set. Assume that the channel is either a BSC or an AWGNC, and that the messages from the channel to all VNs outside S_m are correct. Provided that a VN $v_r \in V_1 \subset S_m$ is *k-strongly-separated* for a large k on G_m, the corresponding MSA (or SMSA) decoder will correct v_r.

Proof: Let $c_1 \in C_1$ and $c_2 \in (C_S \setminus C_1)$ denote the adjacent unsatisfied and mis-satisfied CNs of v_r, respectively. The corresponding trees are denoted by $T(c_1)$ and $T(c_2)$, respectively. Let d_1 be the smallest VN degree in $T(c_1)$ and d_2 the greatest VN degree in $T(c_2)$.

According to the proof of Theorem 1 in [22], for a regular LDPC code under the SMSA decoder, the magnitude of the incoming correct message to v_r can be written as

$$\begin{aligned}|L_l| &= |L_0| + \alpha(d_v - 1)|L_{l-1}| \\ &> [\alpha(d_v - 1)]^l |L_0| \end{aligned} \tag{13}$$

where $|L_l|$ and $|L_0|$ denote the magnitudes of the incoming correct message at lth iteration and of the channel observation, respectively, and α is the scaling factor for the SMSA.

Now, for the modified irregular factor graph at hand, it is clear that the lower bound should instead be

$$|L_l| > [\alpha(d_1 - 1)]^l |L_0|. \tag{14}$$

As for bounding the amount of the incorrect message coming into v_r for a regular LDPC code, the authors of [22] use the fact that since each mis-satisfied CN in a (a, b) trapping set is connected to all other erroneous VNs of the trapping set within $j \geq a$ decoding iterations, there is at least one CN passing correct message having a sign opposite to that of the message from the mis-satisfied CN. Considering the computation tree rooted at v_r as a super-node with $(d_v - 1)^j$ child VNs, the upper bound on the magnitude of the incoming incorrect message to v_r has been shown to be

$$|L'_l| < |L_0|[\alpha(d_v - 1)^j - 1]^{\lceil l/j \rceil} \tag{15}$$

where $|L'_l|$ represents the magnitude of the incoming incorrect message at lth iteration and $\lceil l/j \rceil$ is the smallest integer greater than or equal to l/j.

We make a modification similar to the one leading to the lower bound in (14) to derive an upper bound on the incorrect incoming message to v_r for our modified trapping set structure. Specifically, replace d_v in (15) with d_2 to get

$$|L'_l| < |L_0|[\alpha(d_2 - 1)^j - 1]^{\lceil l/j \rceil}. \tag{16}$$

Since $d_1 \geq d_2$ by assumption, $|L_l|$ of (14) is greater than $|L'_l|$ of (16) with a large l, i.e., the incoming correct message overwhelms the incorrect one, in which case v_r will be corrected. A similar proof can be constructed for the SPA. ■

IV. Collaborative Decoding

In the previous section, we have shown that decoding based on a modified parity-check matrix that results from the proposed CN removal strategies effectively changes the structure of the trapping set. In particular, the established theorems guarantee that the separations of one or more trapped VNs will be enhanced using the suggested probabilistic CN removal methods. This result, combined with the main theoretical results of [22] linking large separation with ability to break trapping sets, suggests that decoding on the modified parity check matrix with some CNs removed will tend to lower error floors, as stated in the propositions of the last section. As with the work of [22], however, our work still could not tell precisely how large the separation k should be for a given code and trapping structure

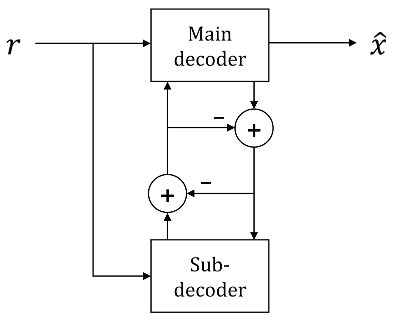


Fig. 5. System model: the main decoder runs through its own internal iterations until decoding succeeds, runs through a predetermined number of iterations or gets stuck in a trapping set. After the main decoder run, the extrinsic information is passed to the sub-decoder. The sub-decoder uses the extrinsic information of the main decoder as well as the LLR values from the channel. The sub-decoder stops when the iteration number exceeds some predetermined level, and passes its extrinsic information back to the main decoder. The whole process is repeated until all errors are corrected or the total number of iterations reaches a set level.

and how to achieve a particular separation value using the proposed CN removal strategies. Furthermore, for irregular codes and/or application of VNCR, it has been shown that being *k-strong* is sufficient for a trapped but k-separated VN to ensure its eventual correction, but it is not clear whether being *k-strong* is critical in providing a high probability of success in breaking trapping sets in practice.

Finding precise solutions analytically to these questions is difficult, and a resort is made to computer simulation to see whether the error floor behaviors improve using the proposed methods. Extensive simulations in fact reveal that for many different types of LDPC codes running a message-passing algorithm on a reduced parity check matrix after removing certain CNs as proposed does in most cases results in a breaking of the trapping sets, including the ones that are known to be tough to break (without prior knowledge of the trapping set structure), like the dominant trapping sets of the Margulis code. We do, however, also recognize that the modified parity-check matrix admits some codewords that are not valid according to the original parity-check matrix and this results in poor overall performance in error rate simulation studies. To get around this practical issue, we suggest taking advantage of both parity check matrices, the original one with reliable overall error rate performance and the modified one that is highly effective in breaking the trapping set when stuck. We specifically introduce the notion of collaborative decoding using two decoders based on two parity-check matrices.

We simply call the decoder based on the original parity-check matrix the main decoder and that based on the modified parity-check matrix the sub-decoder. Fig. 5 describes the collaborative LDPC decoding. The system is a concatenation of the main decoder and the sub-decoder. The vector $\mathbf{r}$ refers to the LLR values from the channel and $\hat{\mathbf{x}}$ is the decisions made by the decoding system.

In the first step, the main decoder runs through its own internal iterations until decoding succeeds, runs through a predetermined number of iterations, N_1, or gets stuck in a trapping set. When the number of unsatisfied CNs remains unchanged over some predetermined number of iterations, the main decoder is assumed stuck in a trapping set. In this case, the extrinsic information is passed to the sub-decoder. The sub-decoder uses the extrinsic information of the main decoder as the *a priori* information. The LLR data from the channel form another input to the sub-decoder.

The sub-decoder stops when the iteration number exceeds some predetermined level, N_2, and passes its extrinsic information back to the main decoder. This whole process is repeated until all errors are corrected or the total number of iterations reaches the maximum set iteration number, N_t. The system does not require additional parity bits for the sub-decoder. An estimated number of sub-decoder trials for separation extension of the trapping set would be available through (7), (8), (10) and (11) if the trapping set were known.

We note that while it is conceptually easier to view the decoding process as collaboration between two separate decoders, implementation can equally be based on a single decoder switching between two different modes, one based on the original factor graph and the other based on a reduced factor graph with some of the CNs removed. In the switching mode implementation, the edges of the removed CNs can simply be blocked off or the CN processing can be disabled on the specified nodes in the message-passing operations while in the reduced factor graph mode. Clearly, there is no additional burden in hardware implementation.

In the next section, simulation results will be presented which indicate that the suggested decoding methods indeed provide powerful and efficient ways of reducing error floors of all different types of LDPC codes tested.

V. Experimental Results

In this section, we set the modification vector to an all-ones vector for all our experiments to reflect practical scenarios where the underlying trapping sets are not known a priori. In addition, we simply fixed p_r to 0.7 regardless of the code type in running the VNCR algorithm. Fixing $p_r = 0.7$ means 70% of the candidate CNs attached to each VN under question (adjacent to unsatisfied CN) are removed. In the language of the FNCR algorithm, this is roughly equivalent to $d_f = 3$ for the (2048,1723) RS-based LDPC code in hand (i.e., removing 3 or so candidate CNs), which was the value of the deselection degree used in running FNCR in the presented simulation results. Note also that d_f must be smaller than $d_v - 1$, meaning that in the Margulis code tested d_f can only be 1. For the collaborative decoding, the iteration number for the main decoder is capped at 50, while it is always fixed at 5 for the sub-decoder. However, for the main decoder, if the number of unsatisfied CNs does not change over 5 iterations, the decoder stops; at that point the sub-decoder gets activated. The total iteration number for collaborative decoding is the sum of the iteration numbers for the main decoder and the sub-decoder. The decoding schemes and the corresponding iteration numbers are indicated for each curve. We use a 5-bit uniform quantization scheme denoted by Q4.1. The symbol Q$m.f$ represents an $(m+f)$-bit uniform quantization with an m-bit signed integer and an f-bit fraction. For example, Q4.1 means a uniform step

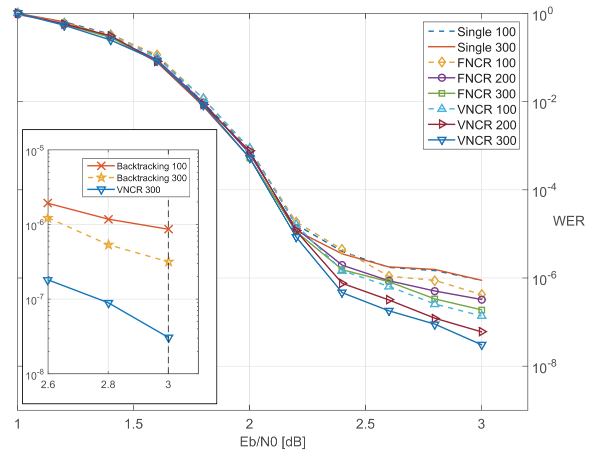


Fig. 6. Error rate performance of collaborative decoding for the (2640,1320) Margulis code.

size of 0.5 and a dynamic range of $[-7.5\ 7.5]$. This particular quantization strategy gives the best results among all 5-bit precision uniform quantization methods.

Fig. 6 shows the error rate performance of the proposed decoding compare to that of the conventional scheme based on a single decoder for the (2640,1320) Margulis code [6]. The plot presents the word error rate (WER) or the codeword failure rate versus E_b/N_0 over AWGNC. We set $d_f = 1$ for FNCR and $p_r = 0.7$ for VNCR. In addition, we use the SMSA with $\alpha = 0.75$ for single-decoder-based decoding and collaborating decoding using either FNCR or VNCR. We observe in the figure that FNCR and VNCR lower the error floor by one order and one and a half orders of magnitude, respectively, with 300 iterations. VNCR seems somewhat more effective than FNCR. Since d_v of the Margulis code is 3, d_f should be 1 and no other choices make sense. However, choosing p_r of VNCR is more flexible. The strategies of [8] also effectively lower the error floor of the Margulis code, but we emphasize that our method does not require any prior knowledge of the dominant trapping sets. The inset figure includes for additional comparison the WERs of the backtracking scheme of [10] in the floor region. Note that the backtracking scheme is the only method in the existing literature that can reduce error floors without explicit prior knowledge of the trapping sets. The backtracking tries LLR flipping for every adjacent VN of the unsatisfied CNs one by one. In addition, each trial requires independent decoding iterations. For our simulation of the backtracking method, we set 20 for the iteration number of the independent decoding, which is the same parameter used in [10]. Backtracking is performed until either all errors are corrected or the total number of iterations reaches the predetermined number. The results indicate that the backtracking with 100 iterations shows no improvement on the error floor but that with 300 iterations lowers the floor by about one half order of magnitude at 3 dB. At the same SNR, the collaborative decoder based on VNCR shows better WER performance by about one order of magnitude compared to backtracking at 300 decoding iterations.

The dominant trapping sets of the Margulis code include a relatively small number of degree-one CNs (Fig. 1(a)). In the (12,4) or the (14,4) trapping set of the Margulis code, only

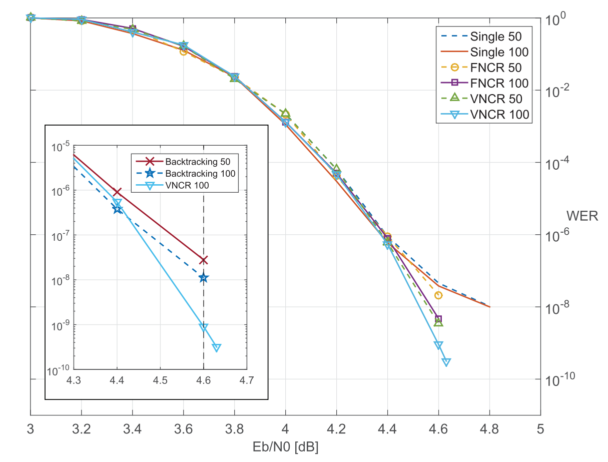


Fig. 7. Error rate performance of collaborative decoding for the (2048,1723) RS-based code.

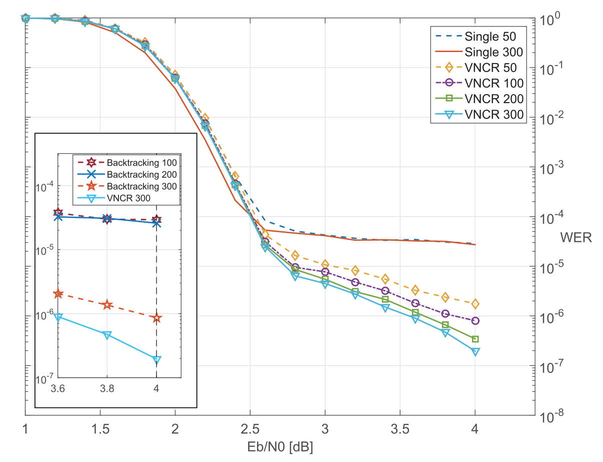


Fig. 8. Error rate performance of collaborative decoding for the (1944,972) irregular code.

4 VNs out of 12 or 14 are connected to degree-one CNs. Therefore, only a small number of VNs have opportunity to extend separation via FNCR or VNCR. However, for the (8,8) trapping set of the (2048,1723) RS-based code (Fig. 1(b)), all VNs are connected to degree-one CNs and our scheme works particularly well within a small number of decoding iterations.

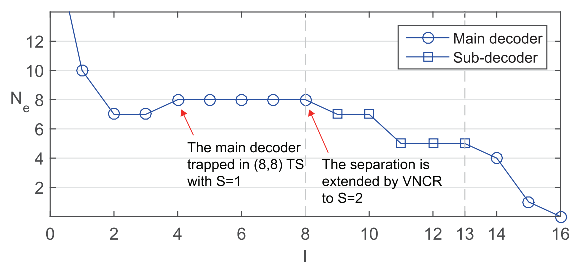
Fig. 7 compares the error rate results for the (2048,1723) RS-based LDPC code [14]. We set $d_f = 3$ for FNCR and $p_r = 0.7$ for VNCR. Again, we use the SMSA with $\alpha = 0.75$. We observe in the figure that the proposed decoding schemes lower the error floor significantly. FNCR lowers the error floor by about an order of magnitude with 100 decoding iterations. VNCR achieves a floor reduction by more than one order of magnitude with only 50 decoding iterations and more than two orders of magnitude with 100 decoding iterations. We observe that the VNCR curves are much steeper than the FNCR curves at high SNRs. VNCR with 100 iterations exhibits no error floor even at a 3.2×10^{-10} WER. Again, the WER curves in the inset provides comparison between the collaborative decoder and the backtracking scheme of [10]. The backtracking with 50 iterations hardly improves the error floor while with 100 iterations lowers the floor slightly. We observe that between the collaborative decoder and the backtracking scheme, both under 100 iterations, the former gives an error floor improvement by about one order of magnitude.

We also evaluated performance for an irregular LDPC code. Fig. 8 shows the performance of the (1944,972) IEEE 802.11n code [13], which is an irregular LDPC code based on the IRA structure. In this experiment, the MSA is used for decoding because we observe that the performance of MSA is definitely better than that of the SMSA for this code. We observe in the figure that the collaborative decoding lowers the error floor by more than an order of magnitude with 50 iterations. In addition, the performance improvement continues as the decoding iteration increases. The inset figure also shows that compared to backtracking, the proposed collaborative decoder based on VNCR improves the WER by a factor 8 in the floor region, when the decoding runs are limited to 300.

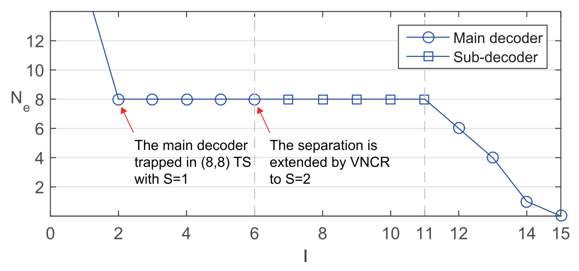
Overall, the experimental results show that the proposed scheme efficiently lowers error floor without prior knowledge of the trapping sets across different types of LDPC codes. The backtracking method of [10] also lowers error floor without the knowledge of the trapping sets and can potentially achieve improved floor reduction relative to our scheme, provided a larger number of decoding iterations is allowed. We, however, emphasize that our scheme works well with moderate numbers of decoding iterations. Our experiments in fact confirm that the collaborative decoder provides better floor reduction capability than the backtracking scheme for a fixed number of decoding runs.

While it is difficult to analytically show how large a separation-extension guarantees breaking trapping sets, additional insights into the required separation are obtained through extensive observations of how the bad VNs and separation vectors evolve in the course of collaborative decoding. For this, we collected over hundreds of trapping sets at error floor regions and observed the number of erroneous VNs and separation vectors at each iteration of the collaborative decoder. We confirmed that for the (2048,1723) RS-based LDPC and Margulis code that even a one-level separation extension drives the sub-decoder to correct several errors in a trapping set or pass good information to the main decoder. Representative results are depicted in Fig. 9 for the (8,8) trapping set corresponds to the (2048,1723) RS-based LDPC code. In the simulation, sub-decoders are constructed by VNCR with $p_r = 0.7$ and message passing is performed based on SMSA with Q4.1 quantization.

In Fig. 9, the y-axis labeled by N_e shows the number of erroneous VNs at the end of I^{th} decoding iteration where the squares and the circles represent the main and sub-decoder turn, respectively. The vertical-dashed lines indicate switching times to the sub-decoder and then back to the main decoder. The success of separation-extension can be judged by the maximum value of the separation vector denoted by S. In a typical error-correcting behavior depicted in Fig. 9(a), the main decoder cannot escape the (8,8) trapping set and terminates the first pass at the end of 8th iteration because the number of unsatisfied CN does not change for 5 iterations. Then the sub-decoder takes over using the extrinsic information from the main decoder and corrects three errors in 5 iterations. We observe in the experiment that all the VNs in the (8,8) trapping set are 1-separated



(a) Typical error-correcting behavior 1: the sub-decoder corrects several errors in the trapping set and thereby extends the separation of one or more VNs in the trapping set.



(b) Typical error-correcting behavior 2: the sub-decoder cannot correct any error in the trapping set but extends the separation of VNs in the trapping set.

Fig. 9. Dynamic behavior of the (8,8) trapping set during collaborative decoding.

until the end of the first pass of the main decoder (iteration number 4 to 8) and the separation of one or more VNs in the trapping set are extended to 2 by VNCR as soon as the sub-decoder takes over. In this case, a good message generated by the sub-decoder with extended separation effectively corrects several errors. Finally the main decoder corrects all remained errors in its second pass (iteration number 14 to 16).

Fig. 9(b) shows another representative error-correcting behavior pattern by the collaborative decoder. The main decoder trapped by the (8,8) trapping set and passes its turn to the sub-decoder after 6 iterations of decoding run. However, the sub-decoder cannot correct any error for 5 iterations (iteration number 7 to 11) and transfers its extrinsic information back to the main decoder (after iteration number 11). Then the main decoder corrects all errors in 4 iterations. We observe in the experiment that the one or more VNs in (8,8) trapping set successfully extended their separations from 1 to 2 at the beginning of the sub-decoder pass. Although no erroneous VNs are corrected in the sub-decoder pass, improved messages are transferred to the main decoder as the extrinsic information. We surmise that the sub-decoder generates good enough information for correcting errors in the trapping set with extended separation and helps the main decoder for eliminating the error floor of the (2048,1723) RS-based LDPC code. We also ran extensive simulations for the Margulis code and obtained consistent results.

VI. Conclusion

In this paper, we introduced a collaborative LDPC decoding scheme that could break trapping sets and lower the error floor significantly via message-passing that switched between two decoding modes. The factor graph on which the sub-decoding mode operates is derived from the original graph based on removal of certain CNs, and allows correct messages to be enhanced, which helps break the trapping sets. The simulation results show that the collaborative LDPC decoding works well for both regular and irregular codes. We stress that our approach does not need prior knowledge of the trapping sets. We provided formal discussions in the form of mathematical theorems, corollaries and propositions on the processes by which the proposed decoder escapes from trapping sets. Although the proposed scheme requires additional decoding iterations, its performance is considerably better than that of traditional decoding using a single decoder while requiring a much smaller number of iterations to achieve a significant error floor reduction than existing backtracking methods.

References

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. IT-8, pp. 21–28, Jan. 1962.
[2] D. J. C. Mackay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *IEE Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996.
[3] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.
[4] T. Richardson, "Error floors of LDPC codes," in *Proc. Allerton Conf. Commun. Control Comput.*, Monticello, IL, USA, Oct. 2003, pp. 1426–1435.
[5] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," *CoRR*, Dec. 2005 [Online]. Available: http://www.arxiv.org/abs/cs.IT/0512078
[6] D. MacKay and M. Postol, "Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes," *Electron. Notes Theor. Comput. Sci.*, vol. 74, pp. 97–104, 2003.
[7] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.
[8] Y. Han and W. E. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1663–1673, Jun. 2009.
[9] J. Kang, Q. Huang, S. Lin, and K. Abdel-Ghaffar, "An iterative decoding algorithm with backtracking to lower the error-floors of LDPC codes," *IEEE Trans. Commun.*, vol. 59, no. 1, pp. 64–73, Jan. 2011.
[10] X. Chen, J. Kang, S. Lin, and V. Akella, "Hardware implementation of a backtracking-based reconfigurable decoder for lowering the error floor of quasi-cyclic LDPC codes," *IEEE Trans. Circuit Syst. I*, vol. 58, no. 12, pp. 2931–2943, Dec. 2011.
[11] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat accumulate codes," in *Proc. 2nd Int. Symp. Turbo Codes Relat. Topics*, Brest, France, Sep. 2000, pp. 1–8.
[12] M. 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. 673–6–680, May 1999.
[13] *IEEE Draft Standard for Information Technology-Telecommunications and information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment: Enhancements for Higher Throughput*, IEEE Std. 802.11n/D2.00, Feb. 2007.
[14] *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements—Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Std. 802.3an, Sep. 2006.
[15] 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.
[16] M. Karimi and A. H. Banihashemi, "Efficient algorithm for finding dominant trapping sets of LDPC codes," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6942–6958, Nov. 2012.

[17] S. Landner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. Int. Conf. Wireless Netw. Commun. Mobile Comput.*, Maui, HI, USA, Jun. 2005, pp. 630–635.

[18] 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.

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

[20] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. J. Wainwright, "Lowering LDPC error floors by postprocessing," in *Proc. IEEE Global Commun. Conf.*, New Orleans, LA, USA, Nov. 2008, pp. 1–6.

[21] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam, and M. J. Wainwright, "Design of LDPC decoders for improved low error rate performance: Quantization and algorithm choices," *IEEE Trans. Wireless Commun.*, vol. 57, no. 11, pp. 3258–3268, Nov. 2009.

[22] X. Zhang and P. Siegel, "Quantized iterative message passing decoders with low error floor for LDPC codes," *IEEE Trans. Commun.*, vol. 62, no. 1, pp. 1–14, Jan. 2014.

[23] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[24] 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.

[25] 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.*, vol. 61, no. 10, pp. 4046–4057, Nov. 2013.

[26] D. V. Nguyen and B. Vasić, "Two-bit bit flipping algorithms for LDPC codes and collective error correction," *IEEE Trans. Commun.*, vol. 62, no. 4, pp. 1153–1163, Apr. 2014.

[27] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasić, "Finite alphabet iterative decoders—Part I: Decoding beyond belief propagation on the BSC," *IEEE Trans. Commun.*, vol. 61, no. 10, pp. 4033–4045, Oct. 2013.

Soonyoung Kang (S'12) received the B.S. degree in electrical engineering from Korea University, Seoul, South Korea, and the M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2010 and 2012, respectively. He is currently pursuing the Ph.D. degree in electrical engineering at KAIST. Since 2010, he has been a Member of the Communications and Storage Laboratory, KAIST. His research interests include coding theory and signal processing for data storage systems.

Jaekyun Moon (F'05) received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA. He is a Professor of Electrical Engineering with Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. From 1990 to 2009, he was with the faculty of the Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities, Minneapolis, MN, USA. He consulted as Chief Scientist for DSPG, Inc., from 2004 to 2007. He also worked as Chief Technology Officer with Link-A-Media Devices Corporation in 2008. His research interests include channel characterization, signal processing and coding for data storage and digital communication. He received the McKnight Land-Grant Professorship from the University of Minnesota. He served as Program Chair for the 1997 IEEE Magnetic Recording Conference (TMRC). He is also Past Chair of the Signal Processing for Storage Technical Committee of the IEEE Communications Society. He served as a Guest Editor for the 2001 IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS issue on Signal Processing for High Density Recording. He also served as an Editor for IEEE TRANSACTIONS ON MAGNETICS in the area of signal processing and coding for 2001–2006. He was the recipient of the IBM Faculty Development Awards as well as the IBM Partnership Awards, the National Storage Industry Consortium (NSIC) Technical Achievement Award for the invention of the maximum transition run (MTR) code.

Jeongseok Ha (M'06) received the B.E. degree in electronics from Kyungpook National University, Daegu, South Korea, the M.S. degree in electronic and electrical engineering from Pohang University of Science and Technology, Pohang, South Korea, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 1992, 1994, and 2003. He is now with the Korea Advanced Institute of Science and Technology (KAIST) Daejeon, South Korea, as an Associate Professor. His research interests include theories and applications of error-control codes and physical layer security.

Jinwoo Shin received the B.S. degrees in mathematics and computer science from Seoul National University, Seoul, South Korea, and the Ph.D. degree from Massachusetts Institute of Technology, Cambridge, MA, USA, in 2001 and 2010, respectively. He is currently an Assistant Professor with the Department of Electrical Engineering, KAIST, Daejeon, South Korea. After spending two years (2010–2012) at Algorithms and Randomness Center, Georgia Institute of Technology, Atlanta, GA, USA, and one year (2012–2013) at Business Analytics and Mathematical Sciences Department, IBM T. J. Watson Research, he started teaching at KAIST in the fall of 2013. He was a Programmer for computer-security softwares at Penta Security Systems in 2001–2005. He was the recipient of the Kennneth C. Sevcik (Best Student Paper) Award at SIGMETRICS 2009, the George M. Sprowls (Best MIT CS PhD Thesis) Award 2010, Best Paper Award at MOBIHOC 2013, and Best Publication Award from Applied Probability Society 2013.