Augmented Belief Propagation Decoding of Low-Density Parity Check Codes

Nedeljko Varnica, Marc P. C. Fossorier, *Fellow, IEEE*, and Aleksandar Kavčić, *Senior Member, IEEE*

***Abstract*—We propose an augmented belief propagation (BP) decoder for low-density parity check (LDPC) codes which can be utilized on memoryless or intersymbol interference channels. The proposed method is a heuristic algorithm that eliminates a large number of pseudocodewords that can cause nonconvergence in the BP decoder. The augmented decoder is a multistage iterative decoder, where, at each stage, the original channel messages on select symbol nodes are replaced by saturated messages. The key element of the proposed method is the symbol selection process, which is based on the appropriately defined subgraphs of the code graph and/or the reliability of the information received from the channel. We demonstrate by examples that this decoder can be implemented to achieve substantial gains (compared to the standard locally-operating BP decoder) for short LDPC codes decoded on both memoryless and intersymbol interference Gaussian channels. Using the Margulis code example, we also show that the augmented decoder reduces the error floors. Finally, we discuss types of BP decoding errors and relate them to the augmented BP decoder.**

***Index Terms*—belief propagation (BP) decoding, error floors, low-density parity check (LDPC) codes, pseudocodewords.**

I. Introduction

A BELIEF propagation (BP) decoder of turbo-like codes [1], [2] performs iterative message passing on a code graph and is derived under the message-independence assumption. This assumption is valid only while the iteration number is smaller than quarter of the girth (size of the smallest cycle) of the graph. Unfortunately, the graphs of finite-length low-density parity check (LDPC) codes [1], [3] with good distance properties contain a large number of cycles [4] and typically have a small girth. Nevertheless, for very large block lengths, LDPC codes decoded by BP decoders can be optimized to practically achieve the capacities of a variety of channels [5]–[8]. However, when the code block length is short, the presence of cycles in LDPC code graphs introduces a noticeable loss in performance if the standard BP decoding is performed instead of the optimal maximum-likelihood (ML) decoding.

Paper approved by A. H. Banihashemi, the Editor for Coding and Communication Theory of the IEEE Communications Society. Manuscript received May 2, 2004; revised June 24, 2005 and March 15, 2006. This work was supported by the National Science Foundation (NSF) under Grants CCR-0098029 and CCR-0118701. This paper was presented in part at the IEEE International Symposium on Information Theory, Chicago, IL June-July 2004.

N. Varnica was with Harvard University, Cambridge, MA 02138 USA. He is now with Marvell Semiconductor, Inc., Santa Clara, CA 95054 USA (e-mail: nvarnica@marvell.com).

M. Fossorier is with the Department of Electrical Engineering, University of Hawaii at Manoa, Honolulu, HI 96822 USA (e-mail: marc@aravis.eng.hawaii.edu).

A. Kavčić was with the Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA. He is now with the University of Hawaii, Honolulu, HI 96822 USA (e-mail: alek@ahi.eng.hawaii.edu).

Digital Object Identifier 10.1109/TCOMM.2007.900611

One approach that often improves the performance of short LDPC codes is the algebraic (rather than random) parity check matrix construction, which typically controls the minimum distance and/or code graph girth [9]–[14]. For example, Tanner *et al*. proposed an algebraic code construction method that yields a (155,64) code with girth, diameter, and minimum distance equaling 8, 6, and 20, respectively [9]. Although the performance of the BP decoder for this code on the additive white Gaussian noise (AWGN) channel is better than the performance of the BP decoder for the corresponding randomly constructed regular Gallager code [1], [3], it is still about 1dB away from the performance of the ML decoder for this code. Many other *short* LDPC codes have significant BP-to-ML performance gaps; examples of such codes include the (504,252) MacKay regular and irregular codes [3]. The BP-to-ML decoding gaps often exist for *longer* LDPC codes (with block lengths of several thousand symbols); large gaps are typically observed in the error floor region [15].

In this paper, we propose an augmented BP decoder. This algorithm relies on the original BP decoder in the initial iterations. However, when the original BP fails (which is detected using code parity check constraints), the augmented BP algorithm is employed in a multistage decoding fashion, where, at each stage, the channel message on a select variable node is replaced by a saturated message. Several node selection algorithms are proposed. These node selection algorithms follow the idea that the least reliable symbols are selected first.

The symbol nodes are selected based on their channel information and their degrees (and the degrees of their neighbors) in the appropriately defined subgraphs of the code graph. We note that an algorithm that selects a set of symbol nodes based on their degrees only and attempts all possible combinations of saturation values on the selected set was proposed in [16]. The major differences between our decoder and the decoder in [16] are that the node selection algorithm is improved and that the decoding is performed in stages. This leads to decoder performance improvements for all the codes that we tested.

We demonstrate the performance of the proposed augmented decoder on several short code examples. For the (155,64) Tanner code, we verify that the performance of the augmented BP decoder (with a large number of iterations) approaches the ML decoding performance to within 0.2 dB, while its complexity is far lower than the complexity of the ML decoder. Generally, the effectiveness of the proposed algorithm in the waterfall region reduces as the code block lengths increase. On the other hand, on two chosen code examples (one short code and one long code) with high error floors, we show that the algorithm is effective in the error floor region. For the long Margulis code (which is considered here because of its good minimum distance [17]),

we demonstrate that the augmented BP method can reduce error floors by at least two orders of magnitude. The complexity–performance tradeoffs can be achieved by choosing the number of iterations and/or the number of correction stages in the augmented BP algorithm. The method is universal in the sense that it does not depend on the specific parity check code or specific BP decoder implementation (for example, possible implementations include approximate and quantized BP algorithms [18], [19] as well as certain improved BP decoders, e.g., [20]).

The remainder of this paper is organized as follows. In Section II, we provide necessary definitions regarding LDPC code subgraphs. In Section III, we present augmented BP decoders. In Section IV, we discuss types of BP decoding errors and relate them to the proposed decoders. Implementations of the augmented BP decoders are addressed in Section V. We interpret our methods using the pseudocodewords perspective [21], [22] in Section VI. Section VII gives decoding performance results and Section VIII concludes the paper.

II. Definitions of Sets and Subgraphs

An LDPC code is a linear block code with a sparse parity check matrix [1]. Every parity check matrix can be represented by a bipartite graph with two sets of nodes: variable (symbol) nodes, which represent the encoded symbols, and check nodes, which represent the parity check constraints [23]. For a given LDPC code $\mathbf{C}_H$ with the parity check matrix H, denote the set of variable nodes by V and the set of check nodes by C. A node $v \in V$ is connected to a node $c \in C$ by a single undirected edge if the symbol represented by v is involved in the parity check equation represented by c. Denote the set of edges by E. The bipartite code graph G is defined as a triple of sets V, E, and C, i.e., $G = (V, E, C)$. An (N, k) LDPC code $\mathbf{C}_H$ has block length $N = |V|$ and code size[1] $|\mathbf{C}_H| = 2^k$. Denote the encoded vector of this code by $\underline{x} = [x_0, x_1, \ldots, x_{N-1}]^T$. Let the number of parity checks be $N_c = |C| \geq N - k$. The parity check vector $\underline{z} = [z_0, z_1, \ldots, z_{N_c-1}]^T$ is given by $\underline{z} = H \cdot \underline{x}$ and is typically a zero vector (except for LDPC coset codes [24]).

Let $\underline{y}$ be the received vector at the channel output. Denote by $\underline{\hat{x}}^{(L)} \stackrel{\text{def}}{=} \underline{\hat{x}}^{(L)}(\underline{y})$ the decoded vector obtained after L iterations of BP decoding, initialized with the a priori probabilities yielded from $\underline{y}$. A check node corresponding to the parity check z_i is said to be *unsatisfied* if the ith entry of the syndrome $\underline{s} = H \cdot \underline{\hat{x}}^{(L)}$ is not equal to z_i.

Definition 1 (*Node and edge sets*)

1) Set of unsatisfied checks (SUC) $C_S^{(L)} \subseteq C$ is the set of check nodes that are unsatisfied after L iterations of the standard or augmented (to be described promptly) BP decoding.
2) Set of suspicious variable nodes $V_S^{(L)} \subseteq V$ is the set of variable nodes connected to $C_S^{(L)}$ by at least one edge in the LDPC code graph G.
3) $E_S^{(L)} \subseteq E$ is the set of edges connecting the nodes in $V_S^{(L)}$ and the nodes in $C_S^{(L)}$.

Definition 2

The *SUC graph* $G_S^{(L)}$ is the graph induced by the SUC $C_S^{(L)}$, *i.e.*, $G_S^{(L)} = (V_S^{(L)}, E_S^{(L)}, C_S^{(L)}) \subseteq G$.

Definition 3 (*Variable node degrees with respect to a graph*)

1) Denote the degree of a variable node $v \in V$ with respect to the code graph G by $d_G(v)$.
2) Denote the degree of a variable node $v \in V$ with respect to the SUC graph $G_S^{(L)}$ by $d_{G_S}(v)$.
3) Let $d_G^{\max} = \max_{v \in V} d_G(v)$ and $d_{G_S}^{\max} = \max_{v \in V_S^{(L)}} d_{G_S}(v)$. Denote by $S_v^{\max}$ the set of all variable nodes whose degree d_{G_S} equals $d_{G_S}^{\max}$, i.e., $S_v^{\max} = \{v \in V_S^{(L)} : d_{G_S}(v) = d_{G_S}^{\max}\}$.

Note that $d_{G_S}(v) = 0$ if $v \notin V_S^{(L)}$ and $d_{G_S}(v) > 0$ if $v \in V_S^{(L)}$. For brevity, we will sometimes omit the argument of d_{G_S}, which will be clear from the context.

Definition 4 (*Neighborhood with respect to SUC graph*)

Consider a pair of nodes in $V_S^{(L)}$. We call these two nodes *neighbors with respect to the SUC graph* if they are incident to at least one (common) check node $c \in C_S^{(L)}$. For a node $v \in V_S^{(L)}$, we denote by $n_v^{(\ell)}$ the number of such neighbors whose degree in the SUC graph is $d_{G_S} = \ell$.

The following three observations are made based on intuition and empirically confirmed by experiments on the code examples discussed in Section VII. We believe that these observations hold for many other practical codes, but since we cannot offer proofs for the observations they would need to be tested for a given specific code that one wishes to implement.

Observation 1. The higher the degree $d_{G_S}(v)$ of a variable node $v \in V_S^{(L)}$, the more likely is the decoded symbol corresponding to v to be in error.

Next, we make an observation that refines Observation 1. Let ℓ_m be the maximum value of ℓ for which there exists $v \in S_v^{\max}$ such that $n_v^{(\ell)} > 0$. Consider two variable nodes $v_i, v_j \in S_v^{\max}$.

Observation 2. If $n_{v_i}^{(\ell_m)} < n_{v_j}^{(\ell_m)}$, the decoded symbol corresponding to the node $v_i \in S_v^{\max}$ is more likely to be incorrect than the decoded symbol corresponding to the node $v_j \in S_v^{\max}$.

The intuition behind this observation is as follows. From Observation 1, we have that if a symbol node has a high degree d_{G_S}, the likelihood that this symbol is in error is relatively high. For the symbols which have no high-degree neighbors (with respect to the SUC graph), this likelihood should be even higher as the probability that one of their neighbors is responsible for the "check node error" is low.

The final observation is made for memoryless channels only. For a node $v_i \in V$ denote by $\mathcal{O}(v_i) = log\frac{p(x_i=0|\underline{y})}{p(x_i=1|\underline{y})}$ its decoder input, i.e., its initial message obtained from the received vector $\underline{y}$. For example, for binary phase-shift keying (BPSK) signaling on the AWGN channel with the noise standard deviation σ, we have $\mathcal{O}(v_i) = \frac{2 \cdot y_i}{\sigma^2}$. The following observation is made based on the statistics of the absolute values of $\mathcal{O}(v)$.

[1]We consider binary LDPC codes in this paper. The methods presented here can be extended to decoding of nonbinary LDPC codes in a straightforward fashion.

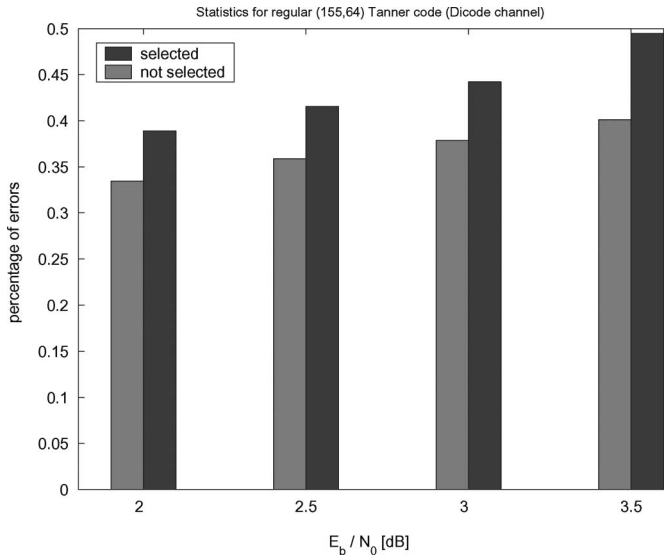

Fig 1. Statistics of the (155,64) Tanner code confirming Observations 1 and 2. The statistics were collected at $E_b/N_0 = 2-, 2.5-, 3-$, and 3.5 dB on the blocks for which BP decoder failed after 100 iterations. This figure shows the percentage of errors among nodes selected by node selection algorithm 2, compared to the percentage of errors for nodes with the same maximum degree ($d_{G_s}^{\max}$) but not selected because they had smaller number of neighbors with the highest degree in the SUC graph. Very similar statistics were obtained for all other regular and irregular LDPC codes considered in this paper.

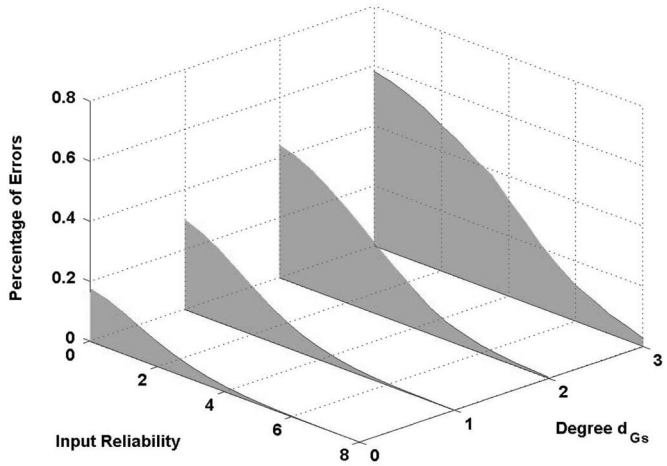

Fig 2. Statistics of the (155,64) Tanner code confirming Observations 1 and 3. The statistics were collected at $E_b/N_0 = 2.0$ dB on the blocks for which BP decoder failed after 100 iterations. Very similar statistics were obtained for all other regular and irregular LDPC codes considered in this paper.

Observation 3. The decoded symbol corresponding to a variable node $v \in S_v^{\max}$ is more likely to be incorrect if its decoder input is less reliable, i.e., if $|\mathcal{O}(v)|$ is lower.

The empirical confirmation of Observations 1 and 2 for the (155,64) Tanner code on a Gaussian channel with memory is given in Fig. 1. The empirical data for Observations 1 and 3 for the same code on the memoryless AWGN channel is shown in Fig. 2. (We obtained similar empirical data for many other codes, including a regular (504,252) MacKay code and an irregular (504,252) code [3]. Due to space limitations, we do not show this data here.)

TABLE I
STATISTICS FOR THE UNSUCCESSFULLY DECODED BLOCKS FOR THE (155, 64) TANNER LDPC CODE USED FOR BPSK TRANSMISSION OVER AN AWGN CHANNEL

Variable node statistics at $E_b/N_0 = 2.5$dB				
	$d_{G_S}=0$	$d_{G_S}=1$	$d_{G_S}=2$	$d_{G_S}=3$
Channel LLR (avg value)	2.8	2.3	1.5	1.1
Check node LLR (avg value)	3.6	1.6	0.2	-0.7
Number of nodes (avg value)	97.2	43.4	11.9	2.5
% of nodes in error	8.1	9.3	31.2	51.1

A large number of all-zero blocks was transmitted. The statistics for the variable nodes with different degrees with respect to SUC are shown. The top two rows show the statistics of the log-likelihood ratio (LLRs) $\log \frac{p(x_i=0)}{p(x_i=1)}$ obtained from the channel and from the check nodes (after L=100 iterations), respectively. The bottom two rows show the average numbers of variable nodes with degree d_{G_S} =i(i=0, 1, 2, 3, 4) and the probabilities that these nodes are in error.

III. AUGMENTED BP DECODER

The augmented decoding process is a two-step process. In the first step, we select a variable node that we wish to correct. This is described in Section III-A. In the second step, we perform the correction in stages. This is detailed in Section III-B.

A. Node Selection

Our node selection method is based on Observations 1 and 3 (or on Observations 1 and 2). As an illustration of the importance of Observation 1, consider the (155,64) Tanner code statistics in Table 1. The nodes with degree $d_{G_S} = d_{G_S}^{\max} = 3$ are most likely to be in error (roughly 50% of these nodes are in error, see the bottom row of Table 1). Furthermore, the average sum of the log-likelihood ratio (LLR) messages that these nodes receive from the check nodes ("Check node LLR" in Table 1) is negative, i.e., on average, the nodes with degree $d_{G_S} = d_{G_S}^{\max}$ receive incorrect information from the check nodes. Therefore, it is natural to attempt correction of one of the nodes in $S_v^{\max}$. Which one of the nodes in $S_v^{\max}$ is selected, is decided according to either Observation 2 or Observation 3. The following node selection algorithm, based on Observations 1 and 3, is proposed for memoryless channels with soft outputs.

Node Selection Algorithm 1 (*based on Observations 1 and 3*):

Step 1: Assume that the SUC $C_S^{(L)}$ is given. Find $V_S^{(L)}$ and the corresponding SUC graph $G_S^{(L)} \subseteq G$.

Step 2: Find degrees $d_{G_S}(v)$ for all $v \in V_S^{(L)}$ and form the set $S_v^{\max} = \{v \in V_S^{(L)} : d_{G_S}(v) = d_{G_S}^{\max}\} \subseteq V_S^{(L)}$.

Step 3: Select the node v_p as $v_p := \arg \min_{v \in S_v^{\max}} |\mathcal{O}(v)|$.

An analogous algorithm based on Observations 1 and 2 can be obtained by modifying Step 3 in algorithm 1. This algorithm is more successful on channels with memory.

Node Selection Algorithm 2 (*based on Observations 1 and 2*):

Replace *Step 3* in node selection algorithm 1 with the following three steps:

Step 3: Initialize $P := S_v^{\max}$, $\ell := d_{G_S}^{\max}$.

Step 4: Form the set $Q \subseteq P$ of variable nodes from P with the fewest neighbors of degree $d_{G_S} = \ell$, i.e., $Q = \{v \in P :$

$n_v^{(\ell)} = \min_{u \in P} n_u^{(\ell)}\}$. Set $P := Q$. If $|P| \neq 1$ and $\ell > 1$, decrement ℓ ($\ell := \ell - 1$) and repeat *Step 4*.

Step 5: Select node v_p from the set P randomly.

We next show how to use information correction on the selected node to potentially correct erroneous nodes.

B. Augmented Decoding Procedures

If nodes v_t and c_m are connected, we denote by $\mathcal{V}(v_t, c_m)$ the message transmitted from node v_t to the node c_m. Similarly, we denote by $\mathcal{C}(c_m, v_t)$ the message transmitted from node c_m to node v_t. The vector $\underline{\mathcal{C}} = \{\mathcal{C}(\cdot,\cdot)\}$ denotes the set of messages transmitted from check to variable nodes. The messages $\underline{\mathcal{O}} = [\mathcal{O}(v_0), \mathcal{O}(v_1), \cdots, \mathcal{O}(v_{N-1})]$ denote the soft values fed into the LDPC decoder from the channel (or the channel detector). The vector $\mathcal{M} = (\underline{O}, \underline{C})$ represents the set of messages received by the variable nodes in V.

After selecting node v_p using one of the node selection algorithms, the testing is performed by replacing the decoder input $\mathcal{O}(v_p)$ with the maximum value $+S$ or the minimum value $-S$, which, due to computer precision or quantization, are used to represent $+\infty$ and $-\infty$, respectively. The binary tree shown in Fig. 3 depicts the general decoding principle. In Fig. 3, the predefined maximum number of correction stages $j_{\max}$ is set to $j_{\max} = 3$. At each stage j (where $j \in \{(1, 2, \ldots, j_{\max}\})$, we perform node selections and corrections and run additional L_j decoding iterations. The symbol T, shown inside the circles in Fig. 3, denotes the counter of the number of tested combinations. Note that in the jth stage, we select a different node v_p for every T. To prevent selection of the already corrected nodes, during the iterations in which $\mathcal{O}(v) = \pm S$, we enforce $d_{G_S}(v) = 0$.

The decoding starts with L_0 iterations of the standard BP decoding. If the BP decoder does not reach a codeword after L_0 iterations, a symbol node is selected using one of the node selection algorithms presented in Section III-A. Denote the node selected for correction at this stage (stage $j = 1$) by $v_p^{(0)}$ and the vector of messages received by the variable nodes at the end of the L_0th decoding iteration by $\mathcal{M}^{(0)}$. We then attempt decoding with changed values of the message $\mathcal{O}(v_p^{(0)})$. Two tests are performed at this stage: one corresponding to the value $\mathcal{O}(v_p^{(0)}) = -S$ (the top branch, corresponding to T = 1, in the tree shown in Fig. 3), the other corresponding to the value $\mathcal{O}(v_p^{(0)}) = +S$ (the bottom branch, corresponding to T = 2, in the tree shown in Fig. 3). In both tests, additional L_1 decoding iterations are performed after each correction. If a valid codeword is reached at this stage, the decoder stores it and continues.

At the next stage (stage $j = 2$), new symbol nodes are selected and new messages are stored. We denote by $\mathcal{M}^{(1)}$ the stored vector of messages and by $v_p^{(1)}$ the node selected after the L_1 decoding iterations performed with $\mathcal{O}(v_p^{(0)}) = -S$. Similarly, by $\mathcal{M}^{(2)}$ and $v_p^{(2)}$ we denote the vector of messages and the selected node (respectively) after the L_1 iterations performed with $\mathcal{O}(v_p^{(0)}) = +S$. At this stage, we need to test $2^j = 4$ possibilities:

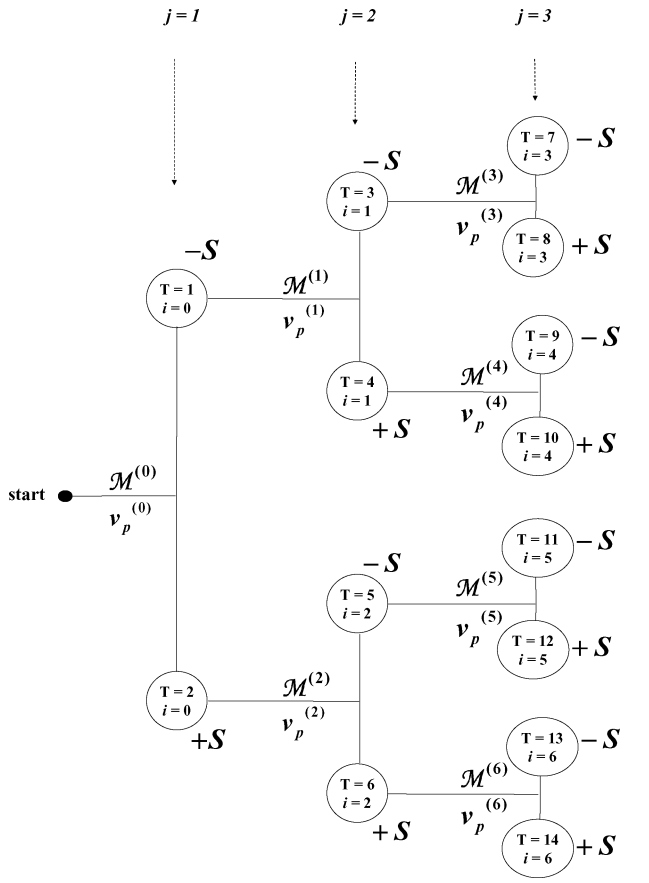

Fig 3. Branching tree of parallel testing schedule for $j_{\max} = 3$. The order of testing is indicated by T (shown inside the circles). The stored values are shown along the corresponding branches in the tree. If the reinitialization is used, messages $\mathcal{M}^{(T)}$ need not be saved.

1) [top two branches in the second stage of the tree in Fig. 3] When $\mathcal{O}(v_p^{(0)}) = -S$, we test $\mathcal{O}(v_p^{(1)}) = \pm S$ (denoted by $T = 3$ and $T = 4$).

2) [bottom two branches in the second stage of the tree in Fig. 3]. When $\mathcal{O}(v_p^{(0)}) = +S$, we test $\mathcal{O}(v_p^{(2)}) = \pm S$ (denoted by $T = 5$ and $T = 6$).

Each of these four tests is performed by running additional L_2 decoding iterations on the code graph with the corresponding corrected information. This procedure continues until the number of decoding stages $j_{\max}$ is reached. Generally, the number of tests performed at stage j is 2^j and, therefore, the total number of decoding tests is $\sum_{i=1}^{j_{\max}} 2^i = 2^{j_{\max}+1} - 2$.

We can either restart or continue the decoding process after the attempted correction of the channel information. If we continue the decoding without reinitialization, the messages and the node locations that the decoder needs to store are indicated along the corresponding branches of the tree in Fig. 3. For example, it can be seen from Fig. 3 that when this decoder finishes the tenth decoding test ($T = 10$ in Fig. 3), to attempt the eleventh decoding test ($T = 11$ in Fig. 3), the vector of messages $\mathcal{M}^{(5)}$ and the node location $v_p^{(5)}$ need to be restored. (Note that the messages $\mathcal{O}(v_p^{(0)})$ and $\mathcal{O}(v_p^{(2)})$ in the vector $\mathcal{M}^{(5)}$ are already set to $\mathcal{O}(v_p^{(0)}) = +S$ and $\mathcal{O}(v_p^{(2)}) = -S$.)

To restart the decoding after each node selection step in the augmented BP decoder, we simply need to reinitialize all LLR messages in $\underline{\mathcal{C}}$ to zero. In this scenario we only need to store and restore $v_p^{(i)}$ and $\mathcal{O}(v_p^{(i)})$, for $i \in \{0, \cdots, 2^{j_{\max}} - 2\}$. The messages $\mathcal{M}^{(\mathrm{T})}$ need not be saved in this decoding scenario.

In the earlier description of the decoding principle, it was assumed that all valid codewords are collected. In fact, this "list decoding" algorithm is only one decoding approach. The other approach is the "greedy decoding" algorithm, which stops decoding as soon as the first valid codeword is found. These algorithms are described in algorithms A and B as follows.

1) Algorithm A: "List Decoding" Algorithm: In the "list decoding" procedure, we test all combinations and record each valid codeword candidate. (A valid codeword candidate is a vector $\underline{w}$ that satisfies the parity check equation $\mathbf{H} \cdot \underline{w} = \underline{z}$.) Collected codeword candidates are stored in the set of codeword candidates W. In the end, after exhausting all $2^{j_{\max}+1} - 2$ possibilities, if $W \neq \emptyset$, we pick the most likely candidate in W.

We emphasize several differences with respect to the decoding approach in [16] here. In our approach, the node selection is based on the degrees d_{G_S} and the decoder input reliabilities (or the neighborhoods with respect to the SUC). This improves the performance compared to the method in [16], which is based on the degrees d_{G_S} only. Note that the selection of $m > 1$ nodes at a time can be achieved in the augmented BP algorithm by setting $L_j = 0$ for $j = 1, \cdots, m-1$ and $L_m > 0$, and similarly $L_j = 0$ for $j = m+1, \cdots, 2m-1$ and $L_{2m} > 0$, and so on. It is, however, advantageous to select one node at a time (e.g., set $L_j = \mathrm{const} > 0$ for all j) and perform decoding in stages. With this approach, if after additional $L_j > 0$ iterations, a codeword is not reached, the augmented BP method reveals several other suspicious v nodes in a different graph G_S induced by a newly obtained SUC.

2) Algorithm B: "Greedy Decoding" Algorithm: The "greedy decoding" algorithm terminates as soon as it reaches the first valid codeword and declares this codeword to be the decoded vector $\underline{\hat{x}}$. The order of testing in algorithm B influences its convergence speed (whereas the complexity of algorithm A is not influenced by the order of testing). Simulations in the waterfall region reveal that it is beneficial to first test the possibility of incorrect sign of the extrinsic information (received from the check nodes) [see Table 1]. Referring to Fig. 3, we can, therefore, write the tests performed on a given symbol node $v_p^{(i)}$ as

$$\mathcal{O}\left(v_p^{(i)}\right) := +S \cdot (-1)^T \cdot \operatorname{sign}\left(\sum_{k=1}^{d_G(v_p^{(i)})} \mathcal{C}\left(c_{v_p^{(i)}}^{(k)}, v_p^{(i)}\right)\right)$$

where $c_{v_p^{(i)}}^{(k)}$ denote the check nodes incident to the variable node $v_p^{(i)}$. This improves the probability that the decoder terminates in the early tests.

IV. Algorithm Improvements Based on Types of Errors in BP Decoding

In this section, we define types of BP decoding errors and relate them to performances of the augmented BP decoders.

A. Types of BP Decoding Errors

We distinguish three types of BP decoding errors: 1) stable errors on saturated SUC graphs, 2) stable errors on nonsaturated SUC graphs, and 3) unstable errors. These types of errors can be defined by considering whether the decoder converges to a steady (stable) state. To this end, denote by $V_e^{(L)} \subseteq V$ the set of variable nodes in error after L decoding iterations by a BP decoder D. That is, $V_e^{(L)} = \{v_i \in V : \hat{x}_i^{(L)} \neq x_i\}$.

Definition 5

The decoder D is said to have reached a steady state in the interval $[M_1, M_2]$ if $V_e^{(L)} = V_e^{(M_1)}$ for all $L \in [M_1, M_2]$. If D reaches a steady state with $V_e^{(M_1)} \neq \emptyset$ and all the messages in the SUC graph are saturated,[2] then we call this error stable on a saturated SUC graph. If D reaches a steady state with $V_e^{(M_1)} \neq \emptyset$ and the messages in the SUC graph are not saturated, then we call this error stable on a nonsaturated SUC graph. Otherwise, if D does not reach a steady state, the error is said to be *unstable*.

In theory, we should let $M_2 \to \infty$ in Definition 5. In practice, we can let $M_2 - M_1$ be as low as 20 and test the stability with some choice of maximal M_1, say $M_1 \leq 100$. Note that the sets $V_e^{(L)}$ are not available at the decoder, but that the decoded vectors $\underline{\hat{x}}^{(L)}$ (where $L \in [M_1, M_2]$) are sufficient for the stability test.

Observation 4. Stable errors on saturated SUC graphs are dominant in the error floor region [high signal-to-noise ratios (SNRs)], whereas *stable errors on nonsaturated SUC graphs* are dominant at low SNRs. At medium SNRs (in the waterfall region), the dominant errors are *unstable errors*, especially for large values of S. As we increase SNR in the waterfall region, observing an increase in the ratio of the number of *stable errors on saturated SUC graphs* over the number of *unstable errors* is a very good indicator of the proximity of the error floor.

Similar to [25], [26], we can use the stability test to obtain an efficient stopping criterion and reduce the average number of iterations for the standard BP decoders. The decoding process can be stopped as soon as the steady state on interval $[M_1, M_1 + M]$ is achieved, where the interval size M is predefined, say $M = 20$. Here, M_1 is defined as the minimal integer that satisfies $\underline{\hat{x}}^{(M_1)} = \underline{\hat{x}}^{(L)}$ for all $L \in [M_1, M_1 + M]$. This method is useful for very low and very high SNRs (in the error floor region) where almost every error is stable.

B. Error Floors—Stable Errors on Saturated SUC Graphs

The size of SUC in the error floor region is typically very small. Naturally, a small-size SUC is very suitable for correction

[2] In practice, this "saturation value" (denoted by α in the rest of this paper) is decoder-imposed and can be slightly smaller than the predefined maximum value S.

since it requires a small number of operations according to node selection algorithms 1 and 2. The downside of having small-size SUCs in the error floor region is discussed in the following theorem and its corollary. This theorem establishes an important property of the stable errors on saturated SUC graphs.

Theorem 1. Consider the following check-node update rule for saturated messages in the decoder: whenever all the messages that a check node receives are saturated, the updated messages (sent back to the adjacent variable nodes) are also saturated.[3] Assume that for a given block:

1) the BP decoder fails after L decoding iterations and the observed error is stable on a saturated SUC graph;

2) for any node $c \in C_S^{(L)}$ and any node $v \in V_S^{(L)}$ that is incident to c, the message $\mathcal{V}(v,c)$ that the node v sends to the node c remains the same in different iterations in steady state.[4]

Then, for any $v \in V$, we have that $d_{G_S}(v) \leq \lfloor \frac{d_G(v)}{2} \rfloor$. □

Proof. Proof is given in the Appendix. From Theorem 1, we have the following corollary.

Corollary 1. Under the assumptions 1) and 2) from Theorem 1, for a regular LDPC code with $d_G^{\max} = 3$, the maximal variable node degree with respect to the SUC graph is $d_{G_S}^{\max} = 1$.

Since $d_{G_S}^{\max} = 1$, for these codes, d_{G_S} plays no role in the node selection process in the error floor region. [5] Thereby, in node selection algorithm 1, the node is selected from $V_S^{(L)}$ based on its soft channel value only. While this method still works well for channels with soft outputs, we present, next, an alternative approach.

Definition 6

Extended set of unsatisfied checks (ESUC) $C_E^{(L)}$ is the set of check nodes connected to at least one variable node in $V_S^{(L)}$. The *ESUC graph* $G_E^{(L)} = (V_S^{(L)}, E_E^{(L)}, C_E^{(L)})$ is the graph induced by $V_S^{(L)}$.

For example, in the error floor region of the Margulis code first established in [17], there always exists exactly one check node $c \in (C_E^{(L)} \setminus C_S^{(L)})$ whose degree with respect to the ESUC graph is $d_{G_E}(c) = 2$. (All other check nodes in $(C_E^{(L)} \setminus C_S^{(L)})$ have degree $d_{G_E} = 1$.) Thereby, in the node selection algorithms, we can select the pair of variable nodes incident to c (with respect to the ESUC graph). We note that both symbols corresponding to these variable nodes have to be either correct (which holds for $(|V_e^{(L)}|, |C_S^{(L)}|) = (12,4)$ near-codewords [17]) or incorrect (which holds for $(|V_e^{(L)}|, |C_S^{(L)}|) = (14,4)$ near-codewords). This simultaneous selection approach combined with the attempted corrections on the selected nodes can be used as an alternative to the methods described in algorithms A and B. Unfortunately, the usage of this alternative approach is limited to near-codewords for which there exists $c \in C_E^{(L)}$ with degree $d_{G_E}(c) = 2$, such as in the Margulis code.

[3]This assumption is practical as the messages entering check-nodes were previously truncated to the saturation value, *i.e.*, the absolute values of these messages before truncation were greater than the saturation value.

[4]These assumptions are realistic for a vast majority of the erroneous blocks in the error floor region of every code that we tested.

[5]Note that the algorithm in [16] considers only the degrees d_{G_S}.

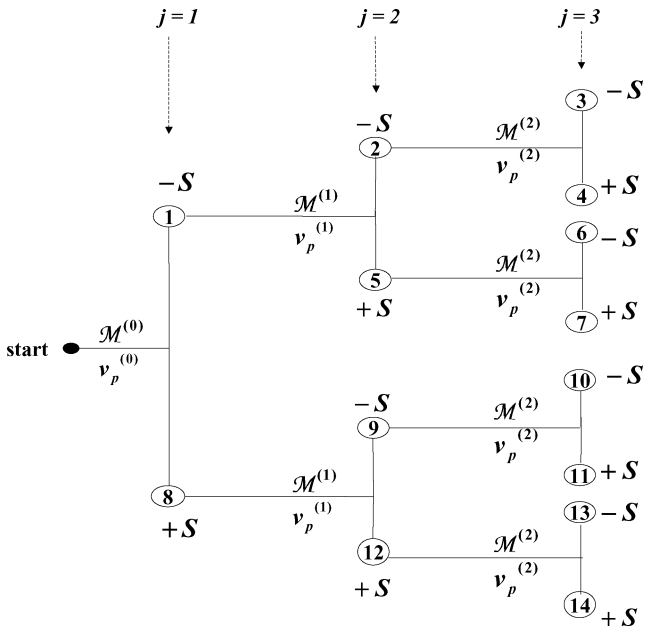

Fig 4. Branching tree of sequential testing schedule for $j_{\max} = 3$. The order of testing is indicated inside the circles. The necessary memory locations for storing the messages $\mathcal{M}$ and the node addresses v_p are shown along the corresponding branches in the tree. The additional storage is of the order $\Omega(|E| \cdot j_{\max})$ since the locations can be overwritten. This is indicated by usage of the same symbols for different branches. For example, after the additional L_1 iterations in step $T = 8$, the locations $\mathcal{M}^{(1)}$ and $v_p^{(1)}$ (used for $T = 1$) are overwritten.

V. Implementation

There exist several different approaches in implementation of the decoding algorithms proposed in Section III-B. For instance, as discussed in Section III-B, if the memory requirements are very stringent, we can simply restart the decoder every time we select a new variable node candidate by a node selection algorithm. In this way, we only need to store the locations $v_p^{(i)}$ and the decoder input values $\mathcal{O}(v_p^{(i)})$ of the selected nodes, which presents a reduction of the additional memory requirements. However, to achieve the same performance at low-to-medium SNRs, the restart of the decoding process typically requires more iterations than the continuation. In the error floor region, due to saturation of a large number of messages, it is sometimes more beneficial to reinitialize the decoder.

Another approach that softens the memory requirements is to run the sequential testing procedure shown in Fig. 4, rather than the parallel testing procedure shown in Fig. 3. The disadvantage of the sequential approach is that the convergence speed of algorithm B is reduced, whereas the advantage of this procedure is that it reduces the number of memory locations from $O(|E| \cdot 2^{j_{\max}})$ to $O(|E| \cdot j_{\max})$ [compare Fig. 3 and 4].

The choice of $j_{\max}$, L_0 and L_j ($j \in \{1, \cdots, j_{\max}\}$), and the option of employing algorithm A or B, provide complexity versus performance tradeoffs. For very small values of $j_{\max}$, we note that $|W|$ is typically 0 or 1 and algorithm B has almost identical performance as (the slower) algorithm A. For large values of $j_{\max}$, algorithm A has superior performance, while algorithm B is faster. Some examples are given in Section VII. Note that the number of iterations L_0 can be predetermined or we can choose L_0 based on the stability test, as described in

Section IV. For instances in which the decoder achieves a steady state, we can introduce the information correction as soon as the steady state is reached.

VI. Pseudocodewords Elimination Perspective

In [21] and [22] the properties of iterative decoders of LDPC codes were considered using the pseudocodeword perspective. If $\underline{\hat{c}}$ is a codeword on a computational tree [21] or on an m-cover [22] of the LDPC code graph G and $\omega_i(\underline{\hat{c}})$ denotes the fraction of times a variable node v_i assumes the value $x_i = 1$ in the vector $\underline{\hat{c}}$, then $\underline{\omega}(\underline{\hat{c}}) = (\omega_0(\underline{\hat{c}}), \cdots, \omega_{N-1}(\underline{\hat{c}}))$ is a pseudocodeword of the code defined by G. The pseudoweight of this pseudocodeword on an AWGN channel is defined as [21]

$$\frac{\left(\sum_{i=0}^{N-1} \omega_i\left(\underline{\hat{c}}\right)\right)^2}{\sum_{i=0}^{N-1} \omega_i\left(\underline{\hat{c}}\right)^2}.$$

For details we refer the reader to [21] and [22]. Clearly, by using the proposed methods, we enforce the correct values on all the nodes whose channel information is properly corrected. For example, if $\underline{x} = \underline{0}$ and the augmented BP algorithm selects a symbol x_i, for some $0 \le i \le N-1$, then (assuming that $\mathcal{O}(v_i)$ is set to $+S$) all the pseudocodewords with $\omega_i > 0$ are eliminated. Hence, the set of pseudocodewords is reduced and the pseudodistance spectrum is improved. In theory, we can increase the minimum pseudodistance to match the minimum distance by setting $j_{\max} = N$ and $L_j = 0$ for $j < N$ and $L_N = 1$. In practice $j_{\max} \ll N$, and the node selection process becomes very important. For a given decoder input $\underline{\mathcal{O}}$, proper selection of only few nodes by the augmented BP algorithms very often eliminates all the pseudocodewords that, for this particular input, cause the nonconvergence to a codeword. We next consider several code examples and show the gains obtained by the augmented decoders with small values of $j_{\max}$ and $L = L_1 = \cdots = L_{j_{\max}}$ in these examples.

VII. Simulation Results

We demonstrate the performance of the augmented BP methods on discrete-time binary-input Gaussian channels given by the following channel law

$$Y_t = \sum_{i=0}^{J} h_i X_{t-i} + W_t \tag{1}$$

where $X_t \in \{-1, +1\}$ and $Y_t \in \mathbb{R}$, respectively, denote the input and the output of the channel at the discrete-time instant t; W_t represents the noise at time t and is assumed to be i.i.d. and Gaussian with zero-mean and the variance σ^2. The channel memory is captured by the coefficients h_i, $i \in \{0, \cdots, J\}$, where J denotes the channel ISI memory length. For these channels, E_b/N_0 is defined as $\frac{E_b}{N_0} = 10\log_{10}\frac{\sum_j \|h_j\|^2}{\sigma^2} - 10\log_{10}(2k/N)$.

A. Memoryless AWGN Channel

1) Waterfall Region Improvement: For the memoryless AWGN channel (with $J=0$ and $h_0=1$), we first demonstrate the performance of the methods for the (3,6)-regular (504,252) MacKay code [27] and an irregular (504,252) LDPC code (privately communicated by T. Richardson). In Fig. 5 we show the performance of algorithm B, with $L_0=100$ and $L=L_1=\cdots=L_{11}=10$, and the original BP performance for these codes.[6] Also shown is the best known (BP-OSD) performance [28] for the regular (504,252) code; at SNR >2.0 dB, about 10% of BP-OSD decoding errors are maximum-likelihood (ML) errors.

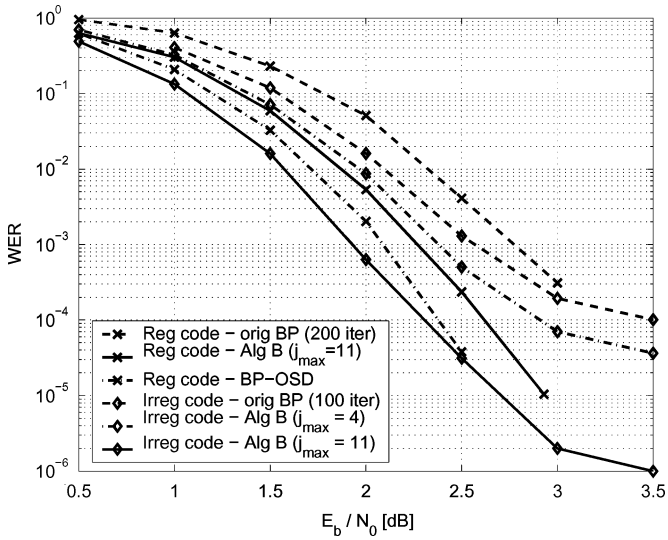

Fig 5. Word-error rate (WER) versus E_b/N_0 for the regular (504,252) MacKay code and for an irregular (504,252) code on the AWGN channel. Comparison between algorithm B (with $L_0 = 100$ and $L = 10$), the original BP, and the BP-OSD decoder. (The BP-OSD decoder has a significantly higher decoding complexity than algorithm B.)

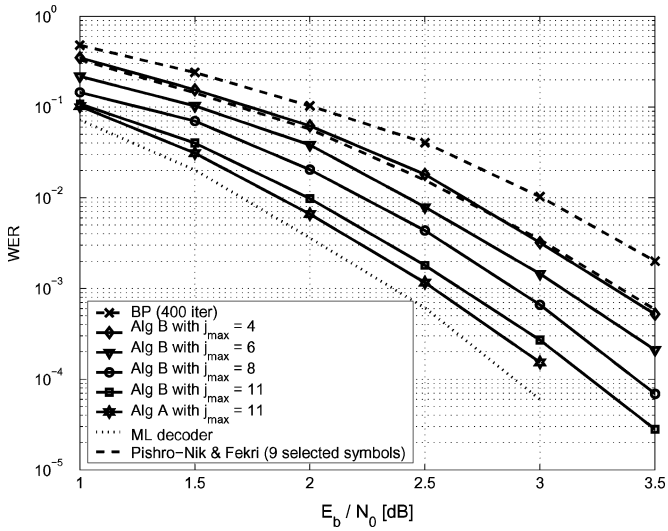

Fig 6. WER versus E_b/N_0 for the (155,64) Tanner code on the AWGN channel. Performances of the original BP decoder, algorithm B for $L_0 = 100$, $L = 10$ and $j_{\max} = 4, 6, 8, 11$, and algorithm A with $L_0 = 100$, $L = 10$ and $j_{\max} = 11$ are shown. Also shown is the performance of Pishro-Nik and Fekri decoder, which selects nine symbols and attempts correction on these symbols simultaneously.

In Fig. 6 the performances of algorithms A and B, with the parameters $L_0=100$, $L=L_1=L_2=\cdots=L_{j_{\max}}=10$ and

[6] We consider word-error rates in this paper. The bit-error rate curves follow similar trends as the word-error rate curves.

TABLE II
AVERAGE NUMBER OF ITERATIONS IN THE ORIGINAL BP DECODER (WITH $(L_0 = 400)$) AND THE IMPROVED BP DECODERS (WITH $L_0 = 100, j_{\max} = 4$, AND $L = 10$)

Average number of iterations			
	Standard BP max 400 iter.	Algorithm A max 400 iter.	Algorithm B max 400 iter.
at $E_b/N_0 = 1.0$dB	189.9	201.2	171.4
at $E_b/N_0 = 1.5$dB	113.3	122.7	96.6
at $E_b/N_0 = 2.0$dB	49.3	53.6	41.9
at $E_b/N_0 = 2.5$dB	20.4	22.4	16.7
at $E_b/N_0 = 3.0$dB	8.1	8.4	7.1
at $E_b/N_0 = 3.5$dB	4.0	4.2	3.9

In the blocks successfully decoded by the original BP and in blocks successfully decoded by algorithm B, the decoding was stopped as soon as valid codeword was reached. In algorithm A, the full 400 iteration were performed for the blocks that failed after $L_0 = 100$ iterations.

several values of $j_{\max}$ for the (155,64) Tanner code [9] are shown. Also shown are the BP decoding performance and the empirical lower bound for the ML decoding of the (155,64) code [29]. From Fig. 6, we see that algorithm A practically bridges the gap to the ML decoding performance and outperforms the standard BP decoding by about 1 dB when the number of decoding stages is $j_{\max} = 11$.

Note that any desired complexity can be achieved by adjusting the parameters $j_{\max}$, L_0, and L. Naturally, this affects the performance gains. In Table II, the average number of iterations for the original BP (with the maximum of 400 iterations) and for algorithms A and B with $L_0 = 100$, $j_{\max} = 4$, and $L = 10$ as a function of E_b/N_0 are given for the (155,64) code. From Fig. 6 and Table II, we see that algorithm B with $L_0 = 100, j_{\max} = 4$, and $L = 10$ outperforms the original BP with the same maximum number of 400 iterations (and almost identical average number of iterations) at WER $= 2 \times 10^{-3}$ by about 0.35 dB. Also shown in Fig. 6 is the performance of the algorithm in [16], which selects symbols based on the their degrees in the SUC graph. Selecting nine symbols using the algorithm in [16] and testing different saturation values on these symbols has a very similar complexity to the complexity of algorithm B with reinitialization and $j_{\max} = 8$. The gain of the algorithm B over the method in [16] is about 0.3 dB for the (155,64) Tanner code.

2) Error Floor Reduction: From Fig. 5, we see that the proposed approach reduces the error floor of the irregular (504,252) code. As another example, we tested algorithm B with the simultaneous symbol correction (based on the ESUC graph definition given in Section IV-B) on the Margulis code which suffers from $(|V_e^{(L)}|, |C_S^{(L)}|) = (12, 4)$ and $(|V_e^{(L)}|, |C_S^{(L)}|) = (14, 4)$ near-codewords [17]. The results for $L_0 = 200$, $j_{\max} = 5$, and $L = 20$ are shown in Fig. 7. We see that the performance in the error floor region is improved by at least two orders of magnitude.

It is worth noting, however, that error floors are highly dependent on the decoding algorithm applied (for some results, see [18] and [30]) and can vary significantly depending on the number of quantization levels in the real system applications. For instance, the error floors for the Margulis code can be significantly lower than the ones given in [17] and shown in Fig. 7 if the predefined saturation parameter S is set to a high value, say $S = 700.0$ (we set the saturation parameter to $S = 10.0$ in all simulations presented in this paper). Naturally, setting S to a very high value may not be practical for implementations.

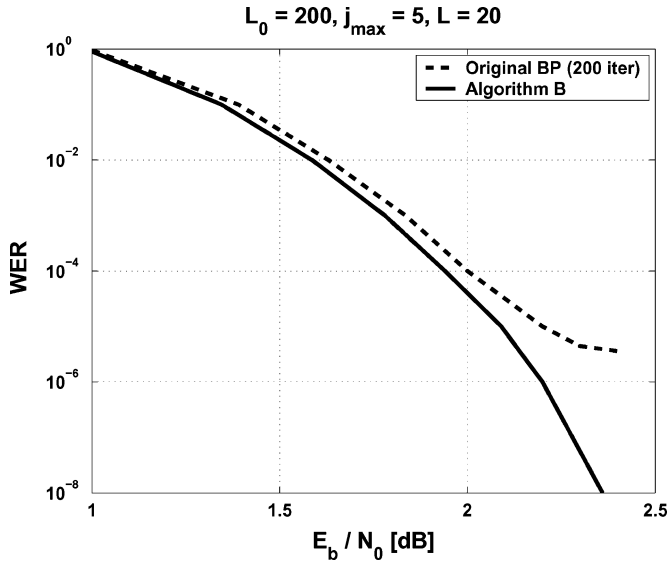

Fig 7. WER versus E_b/N_0 for the (2640,1320) Margulis code on the AWGN channel. Comparison between algorithm B (with $j_{\max} = 5$, $L_0 = 200$, and $L = 20$) and the original BP.

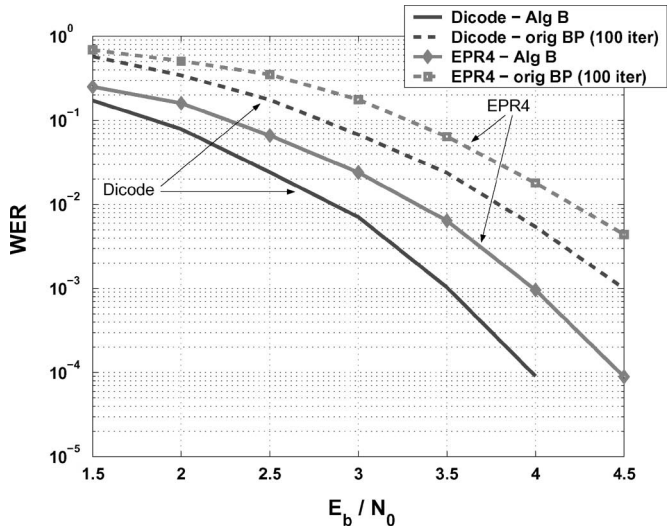

Fig 8. WER versus E_b/N_0 for the (155,64) Tanner code on the Dicode and EPR4 channels. Comparison between algorithm B (with node selection algorithm 2 and $j_{\max} = 11$ and $L = 20$) and the original BP. Five LDPC decoding iterations were performed in the LDPC portion of the joint code/channel graph before information was updated through the channel trellis portion of the joint graph.

B. Intersymbol Interference Channels

To decode a codeword transmitted over a channel with ISI, we combine the channel detector Bahl–Cocke–Jelinek–Raviv (BCJR) trellis with the LDPC decoder graph into a joint code/channel graph [24]. The only modification in the implementation of algorithms A and B for ISI channels is that the

correction is performed on both decoder inputs $\mathcal{O}(v_p^{(j)})$ and the corresponding channel detector (BCJR) inputs. Here, we demonstrate the performance of the proposed methods on the dicode channel (with $J=1, h_0=h_1=\frac{1}{\sqrt{2}}$ and $h_1=\frac{-1}{\sqrt{2}}$) and the EPR4 channel (with $J=3, h_0=h_1=\frac{1}{2}$, and $h_2=h_3=-\frac{1}{2}$). The performance results are given for node selection algorithm 2, which is based on Observations 1 and 2. The results for the (155,64) Tanner code on dicode and EPR4 channels are shown in Fig. 8 for $j_{\max}=11$ and $L=20$. We performed five iterations of LDPC decoding before feeding its extrinsic information back to the BCJR detector [8]. The number of detected ML errors is roughly 15% of the total number of recorded errors suggesting that the augmented BP decoders closely approach the ML performance for this code on ISI channels as well.

VIII. Conclusion

We presented augmented BP decoding algorithms for LDPC codes. The methods are based on assigning saturated (maximum-reliable) messages to the select variable (symbol) nodes. The key point of the algorithm is the node selection process. The nodes are selected based on the LDPC code subgraph obtained from the set of unsatisfied parity check nodes of the code graph and their corresponding channel output. We have shown in the (155,64) Tanner code example that the augmented BP algorithms outperform the standard BP decoding by more than 1 dB. We have verified that the method approaches the performance of the optimal ML decoder to within 0.2 dB if we allow a very large number of decoding iterations. More importantly, the parameters in the proposed scheme can easily be chosen to meet a wide variety of complexity versus performance tradeoffs. We also defined the types of errors and empirically observed that the dominant errors in the waterfall region are "unstable" and that the dominant errors in the error floor region are "stable on saturated subgraphs." Finally, it was shown that the augmented BP decoding methods reduce error floors for two code examples with high error floors. Although this paper demonstrated augmented BP decoding for the memoryless and ISI Gaussian channels, the approach can be extended in a straightforward manner to other channel models (such as binary symmetric channel and fading channel, for example).

Appendix

Lemma 1. Consider a decoded block for which the assumptions 1) and 2) from Theorem 1 hold. For a node $c \in C_S^{(L)}$ in this block, we have that

1) c sends the correct (and saturated) messages to the incident nodes which are in error; and
2) c sends the incorrect (and saturated) massages to the incident nodes which are not in error. □

Proof. We can assume without loss of generality that $+\alpha$ denotes the correct (and saturated) message and $-\alpha$ denotes the incorrect (and saturated) message. We now prove the first claim, i.e., we show that $c \in C_S^{(L)}$ cannot send $-\alpha$ to an incident node $v \in V_e^{(L)}$:

If $\mathcal{C}(c,v)=-\alpha$, then there exists $v_1 \notin V_e^{(L)}$ such that $\mathcal{V}(v_1,c)=-\alpha$ or there exists $v_2 \in (V_e^{(L)} \backslash \{v\})$ such that $\mathcal{V}(v_2,c)=+\alpha$. This, however, is not possible, since $v_1 \notin V_e^{(L)}$ means that

$$\mathcal{C}(c,v_1)+\mathcal{V}(v_1,c)>0$$

which together with $\mathcal{V}(v_1,c)=-\alpha$ implies that $\mathcal{C}(c,v_1)>\alpha$, i.e., that $\mathcal{C}(c,v_1)$ exceeds the saturation value. Similarly, $\mathcal{C}(c,v_2)+\mathcal{V}(v_2,c)<0$ together with $\mathcal{V}(v_2,c)=+\alpha$ implies that $\mathcal{C}(c,v_2)<-\alpha$.

Proof of claim 2) is analogous to the proof of claim 1). □

We now use this Lemma to prove Theorem 1.

Sketch of the Proof of Theorem 1. From Lemma 1, we see that the message sent from a node $c \in C_S^{(L)}$ to a node $v \in (V_S^{(L)} \cap V_e^{(L)})$ is saturated and correct. Therefore, since $v \in V_e^{(L)}$, we have

$$\mathcal{O}(v)+d_{G_S}(v)\cdot(+\alpha)+\sum_{k=1}^{d_G(v)-d_{G_S}(v)}\mathcal{C}(c_v^{(k)},v)<0,$$

where $c_v^{(k)}$ are the nodes from $(C \setminus C_S^{(L)})$ that are incident to v. Since $|\mathcal{O}(v)|<\alpha$ and $|\mathcal{C}(c_v^{(k)},v)|\le\alpha$ (for all $c_v^{(k)}$), it follows that

$$d_{G_S}(v)\cdot\alpha<(d_G(v)-d_{G_S}(v)+1)\cdot\alpha.$$

Thereby,

$$d_{G_S}(v)<\frac{d_G(v)+1}{2},$$

i.e.,

$$d_{G_S}(v)\le\left\lfloor\frac{d_G(v)}{2}\right\rfloor.$$

Similarly, for $v \in (V_S^{(L)} \setminus V_e^{(L)})$, we have

$$\mathcal{O}(v)+d_{G_S}(v)\cdot(-\alpha)+\sum_{k=1}^{d_G(v)-d_{G_S}(v)}\mathcal{C}(c_v^{(k)},v)>0$$

and, hence,

$$d_{G_S}(v)\cdot(-\alpha)>(d_G(v)-d_{G_S}(v)+1)\cdot(-\alpha).$$

Thereby, we obtain again

$$d_{G_S}(v)\le\left\lfloor\frac{d_G(v)}{2}\right\rfloor$$

which completes the proof. □

References

[1] R. G. Gallager, *Low-Density Parity Check Codes*. Cambridge, MA: MIT Press, 1962.
[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Commun.*, May 1993, pp. 1064–1070.
[3] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *Electron. Lett.*, vol. 32, pp. 1645–1646, 1996.
[4] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs?," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2173–2180, Sep. 1999.

[5] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.
[6] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.
[7] S.-Y. Chung, G. D. Forney, Jr., 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.
[8] N. Varnica and A. Kavčić, "Optimized low-density parity-check codes for partial response channels," *IEEE Commun. Lett.*, vol. 7, no. 4, pp. 168–170, Apr. 2003.
[9] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," in presented at the ICSTA 2001, Ambleside, U.K.
[10] Y. Kou, S. Lin, and M. P. C. Fossorier, "Low density parity check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.
[11] J. Rosenthal and P. O. Vontobel, "Constructions of LDPC codes using Ramanujan graphs and ideas from Margulis," in *Proc. Allerton Conf. Commun. Control*, 2000, pp. 248–257.
[12] A. Prabhakar and K. Narayanan, "Pseudo-random construction of low density parity check codes using linear congruential sequences," *IEEE Trans. Commun.*, vol. 50, no. 9, pp. 1389–1396, Sep. 2002.
[13] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.
[14] H. Xiao and A. H. Banihashemi, "Improved progressive-edge-growth (PEG) construction of irregular LDPC codes," *IEEE Commun. Lett.*, vol. 8, no. 12, pp. 715–717, Dec. 2004.
[15] T. Richardson, "Error floors of LDPC codes," presented at the Allerton Conf. Commun. Control, Monticello, IL, 2003.
[16] H. Pishro-Nik and F. Fekri, "Improved decoding algorithms for low-density parity-check codes," presented at the 3rd Int. Conf. Turbo Codes Relat. Top., Brest, France, Aug. 2003.
[17] D. J. MacKay and M. S. Postol, "Weaknesses of Margulis and Ramanujan–Margulis low-density parity-check codes," *Electron. Notes Theor. Comput. Sci.*, vol. 74, pp. 1–8, 2003.
[18] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.
[19] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 549–554, Apr. 2005.
[20] M. R. Yazdani, S. Hemati, and A. H. Banihashemi, "Improving belief propagation on graphs with cycles," *IEEE Commun. Lett.*, vol. 8, no. 1, pp. 57–59, Jan. 2004.
[21] N. Wiberg, "Codes and decoding on general graphs" Ph.D. dissertation, Linkoping Univ, Linkoping, Sweden, 1996.
[22] R. Koetter and P. O. Vontobel, "Graph-covers and iterative decoding of finite length codes," presented at the 3rd Int. Symp. Turbo Codes Relat. Top., Brest, France, 2003.
[23] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.
[24] A. Kavčić, X. Ma, and M. Mitzenmacher, "Binary intersymbol interference channels: Gallager codes, density evolution, and code performance bounds," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1636–1652, Jul. 2003.
[25] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.
[26] R. Shao, L. Shu, and M. P. C. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117–1120, Aug. 1999.
[27] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
[28] M. Isaka, M. P. C. Fossorier, and H. Imai, "On the suboptimality of iterative decoding for finite length codes," presented at the IEEE Int. Symp. Inf. Theory, Lausanne, Switzerland, Jul. 2002.
[29] M. P. C. Fossorier, "Iterative reliability-based decoding of low-density parity check codes," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 908–917, May 2001.
[30] D. Declercq and F. Verdier, "A general framework for parallel implementation of LDPC codes.," *IEEE Trans. Inf.*, to be published.

Nedeljko Varnica received the B.S. degree from the School of Electrical Engineering, University of Belgrade, Belgrade, Serbia, in 2000, and the M.S. and Ph.D. degrees from Harvard University, Cambridge, MA, in 2001 and 2005, respectively, all in electrical engineering.

Since 2005, he has been with Marvell Semiconductor, Inc., Santa Clara, CA. Prior to this, he was with Maxtor Corporation, Shrewsbury, MA, and Lucent Bell Laboratories, Murray Hill, NJ. He has also been a Visiting Researcher at the University of Hawaii at Manoa, Honolulu. His current research interests include communication theory, information theory, and channel and source coding and their applications to digital data storage and wireless communications.

Dr. Varnica was the corecipient, with Aleksandar Kavčić and Xiao Ma, of the 2005 IEEE Best Paper Award in Signal Processing and Coding for Data Storage.

Marc P. C. Fossorier (S'89–M'90–SM'00) was born in Annemasse, France, on March 8, 1964. He received the B.E. degree from the National Institute of Applied Sciences (I.N.S.A.), Lyon, France, in 1987, and the M.S. and Ph.D. degrees from the University of Hawaii at Manoa, Honolulu, in 1991 and 1994, respectively, all in electrical engineering.

He joined the faculty of the University of Hawaii at Manoa in 1996, where he is currently an Associate Professor in the Department of Electrical Engineering. In 2002, he was a Visiting Professor at Ecole Nationale Superieure des Telecommunications (ENST), Paris, France. His current research interests include decoding techniques for linear codes, communication algorithms, and statistics. He is the coauthor (with S. Lin, T. Kasami, and T. Fujiwara) of *Trellises and Trellis-Based Decoding Algorithms* (Kluwer Academic, 1998).

Prof. Fossorier is a member of the IEEE Information Theory and Communications Society. He was the recipient of the 1998 National Science Foundation (NSF) Career Development Award. He has served as an Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY, the IEEE COMMUNICATIONS LETTERS, and the IEEE TRANSACTIONS ON COMMUNICATIONS.

Aleksandar Kavčić (S'93–M'98–SM'04) received the Dipl.-Ing. degree in electrical engineering from Ruhr-University, Bochum, Germany, in 1993, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1998.

From 1998 to 2002, he was an Assistant Professor of electrical engineering in the Division of Engineering and Applied Sciences, Harvard University, where from 2002 to 2006, he was a John L. Loeb Associate Professor of Natural Sciences. In the Fall of 2005, he was a Visiting Associate Professor at the City University of Hong Kong, and in the Spring of 2006, a Visiting Scholar at the Chinese University of Hong Kong. Since 2007, he has been with the University of Hawaii, Honolulu, where he is currently an Associate Professor of electrical engineering.

Prof. Kavčić received the IBM Partnership Award in 1999 and the NSF CAREER Award in 2000. He is a co-recipient of the 2005 IEEE Best Paper Award in Signal Processing and Coding for Data Storage. From 2001 to 2004, he was an Associate Editor for Detection and Estimation of the IEEE TRANSACTIONS ON INFORMATION THEORY. He is currently the Chair of the Technical Committee for Data Storage of the IEEE Communications Society.