# Soft-Information Post-Processing for Chase-Pyndiah Decoding Based on Generalized Mutual Information

Andreas Straßhofer<sup>\*</sup><sup>(0)</sup>, Diego Lentner<sup>\*</sup><sup>(0)</sup>, Gianluigi Liva<sup>†</sup><sup>(0)</sup>, and Alexandre Graell i Amat<sup>‡</sup><sup>(0)</sup> \*Institute for Communications Engineering, Technical University of Munich, Munich, Germany

<sup>†</sup>Institute of Communications and Navigation German Aerospace Center (DLR), Wessling, Germany

<sup>‡</sup>Chalmers University of Technology, Gothenburg, Sweden

*Abstract*—Chase-Pyndiah decoding is widely used for decoding product codes. However, this method is suboptimal and requires scaling the soft information exchanged during the iterative processing. In this paper, we propose a framework for obtaining the scaling coefficients based on maximizing the generalized mutual information. Our approach yields gains up to 0.11 dB for product codes with two-error correcting extended BCH component codes over the binary-input additive white Gaussian noise channel compared to the original Chase-Pyndiah decoder with heuristically obtained coefficients. We also introduce an extrinsic version of the Chase-Pyndiah decoder and associate product codes with a turbo-like code ensemble to derive a Monte Carlo-based density evolution analysis. The resulting iterative decoding thresholds accurately predict the onset of the waterfall region.

Index Terms—Density evolution, forward error correction, generalized mutual information, mismatched decoding.

# I. INTRODUCTION

Product codes [1] when combined with Chase-Pyndiah decoding [2] feature an attractive performance-complexity tradeoff and are therefore widely used for applications with very high throughput requirements (e.g., [3]). Chase-Pyndiah decoding is an efficient iterative soft-decision decoding algorithm that employs suboptimal Chase [4] decoding of the component codes followed by an information combining step to generate the soft messages exchanged. To improve the finite-length performance, Pyndiah further proposed to scale the soft outputs by heuristically-determined parameters [2]. To the best of the authors' knowledge, no information-theoretically motivated method for optimizing these scaling coefficients has been reported to date.

In this paper, we demonstrate that the generalized mutual information (GMI) is a suitable metric to determine the parameters to be used in the post-processing of the component code soft-output. The underlying idea is closely related to reconstructing soft information in min-sum decoding [5] and coarsely quantized message passing [6], [7] for (generalized) low-density parity-check (LDPC) codes. The authors of [8] derive a post-processing for a bit-interleaved coded modulation (BICM) system using a cost function based on the GMI. Another related work [9] proposes to use the Kullback-Leibler (KL) divergence to assess the difference between *accurate* and *mismatched* reliability values. The use of the KL divergence as

optimization criterion showed promising results in the context of max-log a posteriori probability (APP) decoding of turbo codes.

The remainder of this paper is structured as follows. In Section II, we review product codes and their decoding using the Chase-Pyndiah decoder. Section III develops a method for analyzing a modified Chase-Pyndiah decoder in the asymptotic limit of large block lengths. In Section IV we propose softinformation post-processing based on maximizing GMI. Finally, we provide simulation results in Section V and conclude the paper with Section VI.

# II. PRELIMINARIES

# A. Channel Model

We assume transmission over the binary-input additive white Gaussian noise (biAWGN) channel Y = X + Z, where  $X \in \{+1, -1\}$  and  $Z \sim \mathcal{N}(0, \sigma^2)$ . The channel quality is given as the signal-to-noise ratio (SNR)  $E_b/N_0 = (2R\sigma^2)^{-1}$ , where R is the code rate. Given a channel output y, we compute the channel log-likelihood ratio (LLR) as

$$l^{\rm ch} = \ln\left(\frac{p_{\mathsf{Y}|\mathsf{X}}(y|+1)}{p_{\mathsf{Y}|\mathsf{X}}(y|-1)}\right) = \frac{2}{\sigma^2}y.$$
 (1)

#### B. Product Codes

Product codes are serially concatenated codes, whose encoding is given by the following procedure. First, the message bits are arranged into a  $k \times k$  matrix. Then, each row is encoded using a row component code. Finally, each column of the resulting matrix is encoded by a column component code. In this paper, we focus on the practical case of identical row and column component codes. In particular, we consider component codes with parameters (n, k, d), where n is the block length, k the dimension, and d the minimum Hamming distance of the code. Then, the parameters of the product code are  $(n^2, k^2, d^2)$  and its rate  $k^2/n^2$ .

## C. Chase-Pyndiah Decoding

In this subsection we review Chase-Pyndiah decoding as in [2]. The decoder iterates between decoding the row and column component codes, where rows first or columns first can be chosen arbitrarily. In each half iteration  $\ell = 1, \ldots, \ell_{\text{max}}$ , i.e., either row or column decoding, each component decoder processes an incoming message vector  $l^{\text{in}}$  of length n as follows.

First, apply Chase decoding to  $l^{\text{in}}$ . Herefore, obtain the harddecision vector  $\mathbf{r} = (r_1, \ldots, r_n)$ , where

$$r_i = \begin{cases} 0 & \text{if } l_i^{\text{in}} \ge 0\\ 1 & \text{otherwise.} \end{cases}$$
(2)

From  $l^{\text{in}}$ , identify the p positions of lowest reliability  $|l_i^{\text{in}}|$ . From r generate a list of test words by the following bit flip procedure. Flip any single bit among the p least reliable bits (LRBs), giving p test words. Then, flip any two bits to get  $\binom{p}{2}$ more test words. Continue until the last test word, which is found by flipping all p LRBs. Next, decode the  $2^p$  test words (including r itself) using bounded distance decoding (BDD) to form the candidate list  $\mathcal{L}$  consisting of all unique codewords resulting from successful BDD attempts.

Second, for i = 1, ..., n, search for the following two codewords in  $\mathcal{L}$ : One with bit value zero at position i and highest likelihood given  $l^{\text{in}}$ , and another one with bit value one at position i and highest likelihood given  $l^{\text{in}}$ . Clearly, one of them is the maximum likelihood (ML) decision among the codewords of the candidate list. The other codeword is referred to as the *alternative codeword*. Generate the soft output as

$$w_{i} = \begin{cases} \frac{1}{2}d_{i}\sum_{k\neq i}(d_{k}-\bar{x}_{k}^{(i)})l_{k}^{\text{in}} & \text{if } \bar{x}^{(i)} \text{ exists} \\ d_{i} & \text{otherwise} \end{cases}$$
(3)

where d and  $\bar{x}^{(i)}$  are the modulated ML codeword and the modulated alternative codeword, respectively. In this work, we only consider values for p that allow omitting the case of an empty candidate list.

Third, collect the soft outputs of all component codes in the matrix W. Let  $\mathcal{I}$  be a set of index pairs (i, j) for which a component decoder found an alternative codeword. Then, Chase-Pyndiah decoding post-processes the soft output as

$$v_{i,j} = \begin{cases} \alpha \cdot (\operatorname{avg}(|\boldsymbol{W}_{\mathcal{I}}|))^{-1} \cdot w_{i,j} & \text{if } \bar{\boldsymbol{x}}^{(i,j)} \text{ exists} \\ \alpha \cdot (\operatorname{avg}(|\boldsymbol{W}_{\mathcal{I}}|))^{-1} \cdot \beta w_{i,j} & \text{otherwise} \end{cases}$$
(4)

where  $|\cdot|$  denotes the element-wise absolute value and  $\operatorname{avg}(\cdot)$  is the arithmetic average over all entries of a matrix.  $\alpha$  and  $\beta$  are the half iteration-dependent, heuristic scaling factors provided in [2]. The values of  $\alpha$  and  $\beta$  in the first eight half iterations are 0.1, 0.3, 0.5, 0.7, 0.9, 1, 1, 1 and 0.2, 0.4, 0.6, 0.8, 1, 1, 1, 1, respectively. In the subsequent half iterations, both  $\alpha$ and  $\beta$  are typically set to one as well.

Finally, let  $L^{ch}$  denote the channel LLR matrix whose entries are computed according to (1). The matrix of outgoing messages is then computed as (see [2])

$$\boldsymbol{L}^{\text{out}} = \left( \text{avg}\left( |\boldsymbol{L}^{\text{ch}}| \right) \right)^{-1} \cdot \boldsymbol{L}^{\text{ch}} + \boldsymbol{V}$$
 (5)

where the entries of the matrix V are the post-processed outputs obtained by (4). The output  $L^{\text{out}}$  of half iteration  $\ell$ forms then the input  $L^{\text{in}}$  for the following half iteration  $\ell + 1$ . Note that in the first half iteration  $L^{\text{in}}$  is initialized with the first term in (5), i.e., the normalized channel LLR matrix. Decoding terminates after  $\ell_{\text{max}}$  half iterations. Then, use the modulated ML codeword d of each component decoder obtained in the last half iteration to estimate the transmitted code matrix.

# D. Generalized Mutual Information

Let  $p_{Y|X}$  be the transition probability function of a channel from X to Y. Sometimes, e.g., due to complexity limitations, the receiver relies on a mismatched channel model  $q_{Y|X}$  and outputs the codeword that maximizes  $q_{Y|X}(x|y)$  for a given channel output y. Using this decoder, an achievable rate is the s-GMI [10]

$$I_{s}(\mathsf{X};\mathsf{Y}) = \mathbb{E}_{p_{\mathsf{X}\mathsf{Y}}} \left\{ \log_{2} \left( \frac{q_{\mathsf{Y}|\mathsf{X}}(\mathsf{Y}|\mathsf{X})^{s}}{\sum\limits_{x' \in \mathcal{X}} p_{\mathsf{X}}(x')q_{\mathsf{Y}|\mathsf{X}}(\mathsf{Y}|x')^{s}} \right) \right\}$$
(6)

with  $s \ge 0$ . We may simplify our considerations by setting s = 1 since the *s*-GMI is an achievable rate for any parameter *s*. An important upper bound to  $I_1(X; Y)$  is the mutual information (MI) I(X; Y), which is achieved if  $p_{Y|X} = q_{Y|X}$ . In the following, we refer to the 1-GMI simply as GMI.

#### **III. ASYMPTOTIC ANALYSIS**

In this section, we provide a Monte-Carlo density evolution (DE) of product codes under Chase-Pyndiah decoding. In the following, we consider a product code as an instance of a turbo-like code ensemble [11] and apply changes to the original Chase-Pyndiah decoder as in Section II-C such that the asymptotic analysis becomes applicable.

# A. Turbo-like Code Ensemble

A product code with (n, k, d) component codes can be represented by a bipartite graph. Each of the 2n constraint nodes (CNs) corresponds to the code constraints imposed by one component code. The edges are associated with the  $n^2$ code bits. Two CNs are connected by an edge if the associated code bit is part of the code constraints of the corresponding component codes. Finally, partitioning CNs into row CNs and column CNs leads to a bipartite graph. The deterministic product code structure defines a specific connection of row and column CNs. The set of codes defined by all possibilities to connect the row and column CNs gives the turbo-like code ensemble of length  $n^2$ . In Fig. 1, we use the edge interleaver  $\pi$  to visualize the turbo-like code ensemble that contains the (9,4,4) product code built upon (3,2,2) single parity check (SPC) component codes as an instance. The design rate of the code ensemble is given by

$$R_{\rm d} = \frac{n^2 - 2n(n-k)}{n^2} = 2\sqrt{R} - 1 \tag{7}$$

where R is the rate of the product code. We can now generate a sequence of code ensembles with increasing length by adding further CNs and edges. We remark that Chase-Pyndiah decoding can be seen as a message passing over the edges of the bipartite graph of the code. Then, each CN processes incoming messages  $l^{in}$  into outgoing messages  $l^{out}$ .



Fig. 1. Turbo-like code ensemble for SPC component codes of length n = 3.

#### B. Extrinsic Chase-Pyndiah Decoding

DE analysis requires *extrinsic message passing* over the edges of the bipartite graph [12], which is not the case for Chase-Pyndiah decoding. We therefore apply the following changes to the decoding process as described in Section II-C to make it extrinsic [13].

- 1) Every component decoder computes each  $w_i$ ,  $i = 1, \ldots, n$ , by replacing the *i*-th input message with the channel LLR, i.e., we set  $l_i^{\text{in}} = l_i^{\text{ch}}$ . Note that we may obtain a different candidate list for each *i*.
- 2) We set  $v_i = w_i$ , i.e., we omit the post-processing in (4).
- 3) In (5), the matrix  $L^{ch}$  is not normalized.
- 4) We compute the a posteriori LLR as  $l_i^{APP} = v_i + l_i^{ch} + l_i^{in}$ and obtain the final decision as

$$\hat{c}_i = \begin{cases} 0 & \text{if } l_i^{\text{APP}} \ge 0\\ 1 & \text{otherwise.} \end{cases}$$
(8)

We refer to this modified algorithm as *extrinsic Chase-Pyndiah* decoding. Note that due to the modification in Step 1, computing the soft output w is n times more complex than for original Chase-Pyndiah decoding. However, the extrinsic Chase-Pyndiah decoder is only used as a proxy to enable the DE analysis of the turbo-like code ensemble defined in Section III-A under Chase-Pyndiah decoding.

#### C. Monte-Carlo Density Evolution

Assume the all-zero codeword is transmitted over the biAWGN channel with SNR  $E_{\rm b}/N_0$ . Then, the channel LLRs are Gaussian distributed with mean  $\mu_{\rm ch} = 4R_{\rm d}(E_{\rm b}/N_0)$  and variance  $\sigma_{\rm ch}^2 = 2\mu_{\rm ch}$ . We want to track the distribution of the post-processed output  $p_{\rm V}^{(\ell)}$  and the distribution of the incoming messages  $p_{\rm L^{in}}^{(\ell)}$  throughout the half iterations  $\ell = 1, 2, \ldots$ . We do so using a Monte-Carlo simulation. The ensemble iterative decoding threshold  $(E_{\rm b}/N_0)^*$  is defined as the lowest SNR for which the probability  $\Pr[V^{(\ell)} < 0]$  tends to zero as the number of half iterations grows large. Note that a Monte-Carlo simulation can only approximate the iterative decoding threshold by processing a finite number of samples with a finite number of half iterations  $\ell_{\rm max}$ . We permute the incoming messages before each half iterations [14].



Fig. 2. Chase decoding and soft output generation followed by postprocessing and addition of the channel LLR.

# IV. EXTRINSIC CHASE-PYNDIAH DECODING WITH GMI-BASED SOFT-INFORMATION POST-PROCESSING

Belief propagation (BP) decoding of product codes employs optimal APP decoding of the component codes at the CNs and iteratively exchanges extrinsic LLRs between row and column decoders. The outputs  $w_i$  of (3), however, can only be considered as *approximate* LLR values since they violate the *consistency condition* [15]. Pyndiah therefore proposed to post-process w as in (4) and (5) to improve the performance. The post-processed output  $v_i$  is then in general considered to be *closer* to the true LLR values. Clearly, the same argument applies also to extrinsic Chase-Pyndiah decoding as introduced in Section III-B.

We propose to substitute this heuristic post-processing by a post-processing function  $v_i = f_{pp}(w_i; \theta)$  based on the GMI and parametrized by  $\theta = (\gamma, \delta)$ . Fig. 2 depicts the model under investigation, consisting of Chase decoding and soft output generation followed by the post-processing function and addition of the channel LLR.

For simplicity, we focus on functions of the form

$$f_{\rm pp}(w_i; \boldsymbol{\theta}) = \begin{cases} \gamma w_i & \text{if } \bar{\boldsymbol{x}}^{(i)} \text{ exists} \\ \delta w_i & \text{otherwise.} \end{cases}$$
(9)

Similar to  $\alpha$  and  $\beta$  in (4),  $\gamma$  and  $\delta$  are  $\ell$ -dependent parameters that can be precomputed offline.

Under a mismatched-decoding perspective, we can rephrase the problem as follows. Consider an arbitrary outgoing message  $l^{\text{out}}$  after  $\ell$  half iterations. The decoder in the  $\ell$ +1-th half iteration wrongly *interprets* this continuous value as an actual LLR, i.e., it imposes a mismatched model  $q_{\text{Lout}|X}$  that satisfies

$$l^{\text{out}} = \ln\left(\frac{q_{\text{L}^{\text{out}}|\mathsf{X}}(l^{\text{out}}|+1)}{q_{\text{L}^{\text{out}}|\mathsf{X}}(l^{\text{out}}|-1)}\right)$$
(10)

for any realization  $l^{\text{out}}$ . An achievable rate for this mismatched setup is then the GMI (see Section II-D)

$$I_1(\mathsf{X};\mathsf{L}^{\operatorname{out}}) = \mathbb{E}_{p_{\mathsf{L}^{\operatorname{out}}\mathsf{X}}} \left\{ \log_2 \left( \frac{q_{\mathsf{L}^{\operatorname{out}}|\mathsf{X}}(\mathsf{L}^{\operatorname{out}}|\mathsf{X})}{\sum\limits_{x'\in\mathcal{X}} p_{\mathsf{X}}(x')q_{\mathsf{L}^{\operatorname{out}}|\mathsf{X}}(\mathsf{L}^{\operatorname{out}}|x')} \right) \right\}.$$
(11)

The following proposition shows that  $I_1(X; L^{out})$  remarkably does not depend on the actual choice of  $q_{L^{out}|X}$  as long as it satisfies (10).



Fig. 3. BER performance of the  $(64^2, 51^2, 6^2)$  (--),  $(128^2, 113^2, 6^2)$  (--) and  $(256^2, 239^2, 6^2)$  (--) product codes under MCPD-GMI with p = 5 after 20 half iterations. Results for original Chase-Pyndiah decoding (--, --) are provided for comparison.

*Proposition.* The GMI  $I_1(X; L^{out})$  is given by

$$I_1(\mathsf{X};\mathsf{L}^{\mathsf{out}}) = 1 - \mathbb{E}\left\{\log_2\left[1 + \exp\left(-f_{\mathsf{pp}}(\mathsf{W};\boldsymbol{\theta}) - \mathsf{L}^{\mathsf{ch}}\right)\right]\right\},\tag{12}$$

where the expectation is with respect to  $p_{WL^{ch}|X=+1}$ .

Proof. We have

$$I_{1}(\mathsf{X};\mathsf{L}^{\mathrm{out}}) \stackrel{(a)}{=} 1 + \mathbb{E}_{p_{\mathsf{L}^{\mathrm{out}}\mathsf{X}}} \left\{ \log_{2} \left( \frac{q_{\mathsf{L}^{\mathrm{out}}|\mathsf{X}}(\mathsf{L}^{\mathrm{out}}|\mathsf{X})}{\sum\limits_{x'\in\mathcal{X}} q_{\mathsf{L}^{\mathrm{out}}|\mathsf{X}}(\mathsf{L}^{\mathrm{out}}|x')} \right) \right\}$$

$$\stackrel{(b)}{=} 1 - \mathbb{E}_{p_{\mathsf{L}^{\mathrm{out}}\mathsf{X}}} \left\{ \log_{2} \left( 1 + \left( \frac{q_{\mathsf{L}^{\mathrm{out}}|\mathsf{X}}(\mathsf{L}^{\mathrm{out}}|+1)}{q_{\mathsf{L}^{\mathrm{out}}|\mathsf{X}}(\mathsf{L}^{\mathrm{out}}|-1)} \right)^{-\mathsf{X}} \right) \right\}$$

$$\stackrel{(c)}{=} 1 - \mathbb{E}_{p_{\mathsf{L}^{\mathrm{out}}\mathsf{X}}} \left\{ \log_{2} \left( 1 + \exp(-\mathsf{L}^{\mathrm{out}}\mathsf{X}) \right) \right\}$$

$$\stackrel{(d)}{=} 1 - \mathbb{E}_{p_{\mathsf{L}^{\mathrm{out}}|\mathsf{X}=+1}} \left\{ \log_{2} \left( 1 + \exp(-\mathsf{L}^{\mathrm{out}}) \right) \right\}$$

$$(13)$$

where (a) and (b) follow from  $X \in \{-1, +1\}$  being uniformly distributed; (c) exploits that  $q_{L^{out}|X}$  must satisfy (10) and (d) uses the symmetry condition  $p_{L^{out}|X}(l^{out}|-x) = p_{L^{out}|X}(-l^{out}|x)$ . Resolving the output LLR as given by our model as  $L^{out} = f_{pp}(W; \theta) + L^{ch}$  completes the proof.

Our key idea is to optimize the parameters  $\theta^* = (\gamma^*, \delta^*)$  of the post-processing function (9) so that the GMI (12) is maximized, i.e.,

$$\boldsymbol{\theta}^{\star} = \operatorname*{arg\,max}_{\boldsymbol{\theta}} I_1(\mathsf{X};\mathsf{L}^{\mathrm{out}}) \,. \tag{14}$$

We refer to the extrinsic Chase-Pyndiah decoding where the output w is post-processed by a function with parameters defined by (14) as extrinsic Chase-Pyndiah decoding with GMI-based soft-information post-processing (ECPD-GMI).

## V. NUMERICAL RESULTS

Recall that ECPD-GMI is n times more complex than the original Chase-Pyndiah decoding due to the decoding rule in



Fig. 4. BER performance of the  $(256^2, 239^2, 6^2)$  product code under MCPD-GMI with p = 5 (-O-), p = 4 (-D-) and p = 3 (-O-). Results for original Chase-Pyndiah decoding (-O-, -D-, -A-) are provided for comparison. Vertical lines indicate iterative decoding thresholds computed using ECPD-GMI.

Step 1 in Section III-B. To allow fair comparisons with Chase-Pyndiah decoding, we adapt ECPD-GMI as follows. We define ECPD-GMI without the decoding rule in Step 1 as modified Chase-Pyndiah decoding with GMI-based soft-information post-processing (MCPD-GMI). We will use MCPD-GMI to obtain finite-length performance curves in a two-step process. First, for a given SNR, we decode a sufficiently large number of product code frames in parallel. In each half-iteration we use (14) to obtain the optimal parameters  $\theta^*$  for the postprocessing function. Second, we estimate the bit error rate (BER) by applying those optimal parameters. Despite their differences, we hope that iterative decoding thresholds under ECPD-GMI accurately predict the finite-length performance under MCPD-GMI.

First, we simulate the performance of product codes based on (64, 51, 6), (128, 113, 6), and (256, 239, 6) extended Bose-Chaudhuri-Hocquenghem (eBCH) component codes. The rates of the product codes are 0.635, 0.779 and 0.872, respectively. We choose the number of LRB positions for the Chase decoder to be p = 5 for all codes. Fig. 3 shows the BER after 20 half iterations, i.e., 10 row and column decoding runs. Solid curves correspond to MCPD-GMI, while dashed curves correspond to the original Chase-Pyndiah decoder. For the longest code we observe an increased coding gain of approximately 0.11 dB at a BER of  $10^{-4}$ .

In Fig. 4, we show BER results a product code with (256, 239, 6) eBCH component codes for different numbers of LRB positions p. Recall that the complexity of original Chase-Pyndiah decoding as well as MCPD-GMI scale exponentially in p, since the Chase decoder applies BDD to  $2^p$  words. Vertical lines show estimated iterative decoding thresholds of corresponding infinite length turbo-like code ensembles under the ECPD-GMI obtained via Monte-Carlo DE. We use  $10^6$  samples and 50 half iterations. The onset of the waterfall region is in good agreement with the results from the Monte-

TABLE I Optimal parameters  $(\gamma^*, \delta^*)$  for the  $(256^2, 239^2, 6^2)$  product code decoded with p = 5 at an SNR of 3.7 dB.

l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\gamma^{\star}$	0.16	0.20	0.21	0.22	0.24	0.25	0.27	0.28	0.30	0.33	0.35	0.37	0.39	0.42	0.43	0.43	0.43	0.42	0.41	0.39
$\delta^{\star}$	0.15	0.10	0.21	0.20	0.28	0.30	0.37	0.42	0.51	0.60	0.73	0.90	1.10	1.34	1.62	1.91	2.18	2.46	2.67	2.91

#### TABLE II

Iterative decoding thresholds for the turbo-like code ensemble with (32, 26, 4) Hamming component codes. All algorithms employ GMI-based post-processing.

Algorithm	BP	Iterative max-log APP	p = 8	p = 7	p = 6	p = 5	p = 4	p = 3	p = 2
$(E_{\rm b}/N_0)^{\star}$	1.47	1.54	1.54	1.55	1.58	1.66	1.72	1.87	2.14

Carlo DE. The performance gap between p = 3 and p = 5 at a BER of  $10^{-4}$  is around 0.32 dB, which is within 0.02 dB to the 0.34 dB gap predicted by the corresponding thresholds. Remarkably, the performance of MCPD-GMI with p = 4approaches original Chase-Pyndiah decoding with p = 5for low BERs. Thus, the new GMI-based post-processing can cut complexity of Chase-Pyndiah decoding in half while maintaining performance.

Recall that  $(\gamma^*, \delta^*)$  are derived for each SNR. Table I shows the optimal parameters for the  $(256^2, 239^2, 6^2)$  product code with p = 5 at an SNR of 3.7 dB. The values of  $\gamma^*$  are between zero and one, which suggests that we overestimate reliability if an alternative codeword exists. This observation is in agreement with the fact that the corresponding formula in (3) stems from max-log APP decoding, a decoder known for overestimating reliability [9]. Contrary, we have  $w_i = d_i$ if there is no alternative codeword for some position *i*. Since  $d_i \in \{-1, +1\}, \delta^*$  acts itself as reliability value.

Finally, we provide in Table II iterative decoding thresholds for various decoders with GMI-based post-processing. BP corresponds to a decoder, which decodes the (32, 26, 4) Hamming component codes with an optimal APP decoder. Recall that in the asymptotic setting BP computes optimal marginals. Experimental results show that  $\gamma^* \approx 1$  for GMI-based postprocessing of BP. When employing a suboptimal max-log APP component decoder instead, the threshold degrades by only 0.07 dB to 1.54 dB. We remark that the threshold of the same algorithm without GMI-based post-processing increases by approximately 0.5 dB. The thresholds for ECPD-GMI increase as we decrease the parameter p. However, p = 8 and the iterative max-log APP algorithm yield identical values for the threshold, up to the second decimal digit. The obtained thresholds are in good agreement with finite-length simulations.

#### VI. CONCLUSION

We proposed a framework for optimizing the scaling coefficients for Chase-Pyndiah decoding based on maximizing GMI. The parameters found improve upon original Chase-Pyndiah decoding with comparable decoding complexity. Iterative decoding thresholds for an extrinsic version of the algorithm accurately predict the BER of the modified decoder.

#### ACKNOWLEDGMENT

The authors thank Gerhard Kramer for helpful comments.

#### REFERENCES

- P. Elias, "Error-free coding," *Trans. IRE Prof. Group Inf. Theory*, vol. 4, no. 4, pp. 29–37, Sep. 1954.
- [2] R. M. Pyndiah, "Near-optimum decoding of product codes: block turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1003–1010, Aug. 1998.
- [3] A. Graell i Amat and L. Schmalen, "Forward error correction for optical transponders," in *Springer Handbook of Optical Networks*, B. Mukherjee, I. Tomkos, M. Tornatore, P. Winzer, and Y. Zhao, Eds. Cham, Switzerland: Springer International Publishing, 2020, pp. 177–257.
- [4] D. Chase, "Class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inf. Theory*, vol. 18, no. 1, pp. 170–182, Jan. 1972.
- [5] G. Lechner and J. Sayir, "Improved sum-min decoding for irregular ldpc codes," in *Int. Symp. Turbo Codes*, Apr. 2006, pp. 1–6.
- [6] G. Lechner, T. Pedersen, and G. Kramer, "Analysis and design of binary message passing decoders," *IEEE Trans. Commun.*, vol. 60, no. 3, pp. 601–607, Mar. 2012.
- [7] E. B. Yacoub and G. Liva, "Analysis of binary and ternary message passing decoding for generalized LDPC codes," in *Proc. Int. Symp. Probl. Redundancy Inf. Contr. Syst. (REDUNDANCY)*, Moscow, Russian Federation, Oct. 2021, pp. 137–142.
- [8] A. Alvarado, L. Szczecinski, T. Fehenberger, M. Paskov, and P. Bayvel, "Improved soft-decision forward error correction via post-processing of mismatched log-likelihood ratios," in *Proc. Eur. Conf. Optical Commun.* (ECOC), Düsseldorf, Germany, Sep. 2016, pp. 1–3.
- [9] I. Land, "Reliability information in channel decoding practical aspects and information theoretical bounds," Ph.D. dissertation, Kiel University, Kiel, Germany, 2005.
- [10] G. Kaplan and S. Shamai, "Information rates and error exponents of compound channels with application to antipodal signaling in a fading environment," *AEU. Archiv für Elektronik und Übertragungstechnik*, vol. 47, no. 4, pp. 228–239, 1993.
- [11] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially coupled turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 10, pp. 6199– 6215, Oct. 2017.
- [12] Y.-Y. Jian, H. D. Pfister, and K. R. Narayanan, "Approaching capacity at high-rates with iterative hard-decision decoding," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5752–5773, Sep. 2017.
- [13] A. Sheikh, A. Graell i Amat, G. Liva, and A. Alvarado, "Refined reliability combining for binary message passing decoding of product codes," *J. Lightw. Technol.*, vol. 39, no. 15, pp. 4958–4973, Aug. 2021.
- [14] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
- [15] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low-density parity check codes," in *Proc. Int. Symp. Inf. Theory* (*ISIT*), Sorrento, Italy, Jun. 2000, p. 199.