


Improved Soft-Aided Decoding of Product Codes With Dynamic Reliability Scores

Sisi Miao , *Student Member, IEEE*, Lukas Rapp , *Student Member, IEEE*,
and Laurent Schmalen , *Senior Member, IEEE*

Abstract—Products codes (PCs) are conventionally decoded with efficient iterative bounded-distance decoding (iBDD) based on hard-decision channel outputs which entails a performance loss compared to a soft-decision decoder. Recently, several hybrid algorithms have been proposed aimed to improve the performance of iBDD decoders via the aid of a certain amount of soft information while keeping the decoding complexity similarly low as in iBDD. We propose a novel hybrid low-complexity decoder for PCs based on error-and-erasure (EaE) decoding and dynamic reliability scores (DRSs). This decoder is based on a novel EaE component code decoder, which is able to decode beyond the designed distance of the component code but suffers from an increased miscorrection probability. The DRSs, reflecting the reliability of a codeword bit, are used to detect and avoid miscorrections. Simulation results show that this policy can reduce the miscorrection rate significantly and improves the decoding performance. The decoder requires only ternary message passing and a slight increase of computational complexity compared to iBDD, which makes it suitable for high-speed communication systems. Coding gains of up to 1.2 dB compared to the conventional iBDD decoder are observed.

Index Terms—Soft-aided hard decision decoding, product codes, optical communication.

I. INTRODUCTION

PRODUCT codes (PCs) [2] are powerful code constructions with high net coding gains (NCGs) that can be obtained with low-complexity decoders suitable for e.g., high-speed optical fiber communications. A PC codeword is a 2-D array where every row and column is protected by a component code, which is typically a Reed–Solomon (RS) code or a Bose–Chaudhuri–Hocquenghem (BCH) code. In high-throughput applications, PCs are typically decoded with iterative bounded-distance decoding (iBDD) where the component code is decoded by an efficient algebraic component code decoder based on the hard-decision channel output. iBDD is also often referred to as hard decision decoding (HDD) of PCs.

Manuscript received 1 April 2022; revised 11 July 2022 and 15 August 2022; accepted 17 August 2022. Date of publication 26 August 2022; date of current version 16 November 2022. This work was supported by the European Research Council under the European Union’s Horizon 2020 Research and Innovation Programme under Grant 101001899. Parts of this paper have been presented at the Optical Fiber Communication Conference, 2022 [1]. (*Corresponding author: Sisi Miao.*)

The authors are with the Karlsruhe Institute of Technology (KIT), Communications Engineering Lab (CEL), 76187 Karlsruhe, Germany (e-mail: sisi.miao@kit.edu; lukas.rapp3@student.kit.edu; schmalen@kit.edu).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/JLT.2022.3201951>.

Digital Object Identifier 10.1109/JLT.2022.3201951

Soft decision decoding (SDD) of PCs, also known as turbo product decoding (TPD) [3] improves the error-correcting ability of PCs by exploiting soft channel information and list-based decoding. Typically, a 1-2 dB coding gain improvement can be observed compared to HDD/iBDD [4]. However, the high internal decoder data flow required by the soft-message passing in TPD makes it challenging to adapt for ultra-high-speed optical fiber communication systems operating at throughputs of 800 Gbit/s and beyond [5]. In contrast, HDD provides a significant reduction in internal decoder data flow by only passing hard messages. Recently, several *hybrid* SDD/HDD schemes have been proposed which provide a performance/complexity trade-off. The unifying idea of these algorithms is to use soft channel information to aid the hard-decision decoder while keeping the complexity similarly low as in iBDD.

One promising approach for hybrid SDD/HDD is to use ternary messages and error-and-erasure (EaE) decoding. EaE decoding with a stall pattern analysis was studied in [6], assuming miscorrection-free decoding. In [7], a thorough analysis of iterative error-and-erasure decoding (iEaED) with extrinsic message passing (EMP) based on density evolution (DE) takes miscorrections into account. The results show that iEaED without miscorrection detection yields only small coding gains compared to HDD. The binary message passing based on EaE decoding for PCs (BEE-PC) proposed in [8] uses EaE decoding with a relatively high-cost miscorrection control and yields the best performance of hybrid SDD/HDD PC decoding so far. BEE-PC is also based on the idea of combining properly scaled soft channel reliability with the component code decoding decision, which was proposed and developed in [9], [10], [11], [12], [13].

Soft-aided bit marking (SABM) decoder was proposed in [14] and later improved to SABM with scaled reliabilities (SABM-SR) in [15] for PCs. The bits with high channel log-likelihood ratios (LLRs) are marked as highly reliable bits (HRBs) and used for miscorrection detection based on the principle that a BDD output is considered as miscorrection if it conflicts with any HRBs. However, the effect of erroneous HRBs are difficult to be eliminated without an effective update mechanism for the HRBs.

In [16], anchor decoding (AD) was proposed. Unlike the above-mentioned hybrid decoders, AD requires no soft channel output but instead operates purely on the hard-decision channel output. The anchor bits are dynamically set during decoding and used for miscorrection detection as in SABM. However, the achievable coding gain is limited due to wrongly marked anchor

bits, especially in the first decoding iterations. AD has also been shown to be useful in reducing the decoding complexity and improving the decoding performance when combined with list decoding for low rate PCs [17], [18].

In this paper, we propose a novel hybrid decoding scheme for PCs. We first propose a modified EaE decoder able to decode beyond the minimum distance of the component codes, at the cost of an increased miscorrection rate. The miscorrection problem is then solved with a novel miscorrection detection scheme which resembles a combination of AD and SABM. We introduce a new reliability measurement called dynamic reliability score (DRS) that is initialized with the soft channel output and updated during iterative decoding. The DRSs are then used to identify the anchor bits and to detect miscorrections. We present the decoder and a detailed analysis of the decoding behavior.

The remainder of the paper is organized as follows. In Section II, the preliminaries are given. In Section III, we introduce the proposed EaE decoder used as the component code decoder for PCs and calculate the decoding ability of such decoder assuming no miscorrections. In Section IV, we introduce the DRS and describe the architecture of the proposed decoding algorithm. The simulation results in Section V shows that the proposed decoder yields improved decoding performance and approaches miscorrection-free decoding. The reason of the decoding performance gain is heuristically illustrated in Section VI with an example. In Section VII, we analyze the computational and storage overhead. The last section concludes the paper.

Notation: We use boldface letters to denote vectors and matrices, e.g., \mathbf{y} and \mathbf{Y} . The i -th component of vector \mathbf{y} is denoted by y_i , and the element at the i -th row and j -th column of \mathbf{Y} is denoted by $Y_{i,j}$. Let \mathbf{y}_i be the i -th row of a matrix \mathbf{Y} . \mathbb{Z}_{32} stands for the set $\{0, 1, \dots, 31\}$. \mathbb{R} stands for the set of real numbers and $\mathbb{R}_{\geq 0}$ for the set of non-negative real numbers. For $x \in \mathbb{R}$, $\lfloor x \rfloor$ is the floor function that gives the greatest integer less than or equal to x . We use a superscript to denote the maximum number of iterations carried out by an iterative decoder, e.g., iBDD¹⁰.

II. PRELIMINARIES

A product code (PC) codeword is a two-dimensional rectangular array where every row and every column is a codeword of a component code chosen to be same (n, k, t) code \mathcal{C} .¹ We consider \mathcal{C} being either a $(2^\nu - 1, k_0, t)$ binary BCH code or its $(2^\nu - 1, k_0 - 1, t)$ even-weight subcode, both able to correct t errors with standard bounded distance decoding (BDD). Let d_{des} be the design distance of \mathcal{C} ($d_{\text{des}} \leq d_{\text{min}}$, with d_{min} the minimum Hamming distance of \mathcal{C}) and $t = \lfloor (d_{\text{des}} - 1)/2 \rfloor$. The rate of the constructed PC is $r = k^2/n^2$.

We consider a binary phase shift keying (BPSK) modulation and assume that the codewords are transmitted over a binary input additive white Gaussian noise (BI-AWGN) channel. For any transmitted bit x_i , the channel output is

$$\tilde{y}_i = (-1)^{x_i} + n_i,$$

¹We only consider PCs with identical row and column codes in this paper. However, the proposed decoding scheme naturally extends to general PCs where the row and column code are not necessarily the same.

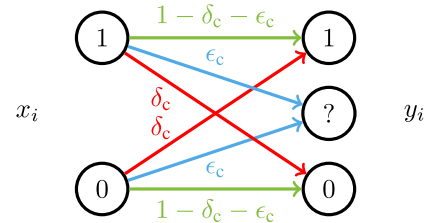


Fig. 1. Channel model of the EaE channel.

where n_i is (real-valued) AWGN with noise variance $\sigma_n^2 = (2rE_b/N_0)^{-1}$. Let $L_c := 2/\sigma_n^2$. The channel LLR for the BI-AWGN channel is given by

$$L(\tilde{Y} = \tilde{y}|X) = \ln \left(\frac{\frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(\frac{-(\tilde{y}-1)^2}{2\sigma_n^2}\right)}{\frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(\frac{-(\tilde{y}+1)^2}{2\sigma_n^2}\right)} \right) = L_c \cdot \tilde{y},$$

proportional to the magnitude of the channel output. Therefore, when performing a hard-decision at the channel output, a received value \tilde{y}_i with small magnitude is considered to be unreliable. We define T , a configurable threshold, such that values $\tilde{y}_i \in [-T, +T]$ are declared as erasures “?”. For $|\tilde{y}_i| > T$, $y_i = \text{sign}(\tilde{y}_i)$ by the usual HDD rule. Therefore, the channel outputs are mapped to three discrete values as depicted in Fig. 1 where

$$\delta_c = Q \left(\sqrt{2r \frac{E_b}{N_0}} (T + 1) \right)$$

is the error probability and

$$\epsilon_c = 1 - Q \left(\sqrt{2r \frac{E_b}{N_0}} (T - 1) \right) - Q \left(\sqrt{2r \frac{E_b}{N_0}} (T + 1) \right)$$

is the erasure probability. When $T = 0$, the EaE channel reduces to a binary symmetric channel (BSC).

An error-only BDD succeeds when its input word is in a Hamming sphere

$$\mathcal{S}_t(\mathbf{c}) = \{\mathbf{y} \in \{0, 1\}^n : d(\mathbf{y}, \mathbf{c}) \leq t\}$$

of radius t around a codeword $\mathbf{c} \in \mathcal{C}$ where $d(\mathbf{y}, \mathbf{c})$ is the Hamming distance between \mathbf{y} and \mathbf{c} .

We define the BDD decoding rule for a binary vector $\mathbf{y} \in \{0, 1\}^n$ as

$$\text{BDD}(\mathbf{y}) = \begin{cases} \mathbf{c} & \exists \mathbf{c} \in \mathcal{C} \text{ such that } \mathbf{y} \in \mathcal{S}_t(\mathbf{c}) \\ \mathbf{y} & \text{otherwise.} \end{cases}$$

Similarly, we define

$$\mathcal{S}_t^3(\mathbf{c}) := \{\mathbf{y} \in \{0, ?, 1\}^n : 2d_{\sim E(\mathbf{y})}(\mathbf{y}, \mathbf{c}) + E(\mathbf{y}) < d_{\text{des}}\},$$

as the Hamming sphere in $\{0, ?, 1\}^n$ for a codeword $\mathbf{c} \in \mathcal{C}$ where $E(\mathbf{y}) := |\{i : y_i = ?\}|$ is the number of erasures of \mathbf{y} and $d_{\sim E(\mathbf{y})}(\mathbf{y}, \mathbf{c})$ is the Hamming distance between \mathbf{y} and \mathbf{c} at the unerased coordinates of \mathbf{y} .

PCs are conventionally decoded with an iterative decoding scheme where the rows and columns of the PC block are alternately decoded with the component code decoder \mathcal{D}_C until

Algorithm 1: Iterative BDD (iBDD) of PCs.

```

1 Input:  $\mathbf{Y} \in \{0, 1\}^{n \times n}$ 
2 for  $\ell = 1, 2, \dots, L$  do //  $L$ : number of iterations
3   for  $i = 1, 2$  do
4     for  $j = 1, 2, \dots, n$  do
5        $w_j \leftarrow \text{BDD}(\mathbf{y}_j)$ 
6      $\mathbf{Y} \leftarrow \mathbf{W}^T$ 
7   if  $\mathbf{W}$  is valid codeword then
8     return  $\mathbf{W}$ 
9 Output:  $\mathbf{W} \in \{0, 1\}^{n \times n}$ 

```

Algorithm 2: Error-and-erasure Decoder (EaED).

```

1 Input:  $\mathbf{y} \in \{0, ?, 1\}^n$ 
2  $E \leftarrow E(\mathbf{y})$  // Number of erasures in  $\mathbf{y}$ 
3 if  $E \geq d_{\text{des}}$  then  $\mathbf{w} = \mathbf{y}$  // Failure
4 else
5    $\mathbf{p}^{(1)}, \mathbf{p}^{(2)} \leftarrow$  two random, complementary vectors in  $\{0, 1\}^E$ 
6    $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \in \{0, 1\}^n \leftarrow \mathbf{y}$  with erasures replaced by  $\mathbf{p}^{(1)}, \mathbf{p}^{(2)}$ 
7   for  $i = 1, 2$  do
8      $\mathbf{w}^{(i)} \leftarrow \text{BDD}(\mathbf{y}^{(i)})$ 
9     if  $\text{BDD}(\mathbf{y}^{(i)}) \in \mathcal{C}$  then  $d_i \leftarrow d_{\sim E(\mathbf{y})}(\mathbf{y}, \mathbf{w}^{(i)})$ 
10    if  $\mathbf{w}^{(1)} \notin \mathcal{C}$  and  $\mathbf{w}^{(2)} \notin \mathcal{C}$  then  $\mathbf{w} \leftarrow \mathbf{y}$  // Failure
11    else if  $\mathbf{w}^{(i)} \in \mathcal{C}$  and  $\mathbf{w}^{(j)} \notin \mathcal{C}$  ( $i, j \in \{1, 2\}, i \neq j$ ) then
12       $\mathbf{w} \leftarrow \mathbf{w}^{(i)}$ 
13    else
14      if  $d_1 > d_2$  then  $\mathbf{w} = \mathbf{w}^{(2)}$ 
15      else if  $d_2 > d_1$  then  $\mathbf{w} = \mathbf{w}^{(1)}$ 
16      else  $\mathbf{w} \leftarrow$  random choice from  $\{\mathbf{w}^{(1)}, \mathbf{w}^{(2)}\}$ 
16 Output:  $\mathbf{w} \in \mathcal{C} \cup \mathbf{y}$ 

```

the maximum number of iterations L is reached. In the conventional iBDD decoding scheme described in Algorithm 1 \mathbf{D}_C is a BDD decoder. In this paper, we replace \mathbf{D}_C by an EaE decoder described later in Section III.

Throughout this paper, we always let the result of a component decoder \mathbf{D}_C be

$$\mathbf{w} := \mathbf{D}_C(\mathbf{y}) \in \mathcal{C} \cup \mathbf{y}$$

where $\mathbf{w} = \mathbf{y}$ in case of a decoding failure.

A *miscorrection* of \mathbf{y} happens when $\mathbf{D}_C(\mathbf{y}) = \mathbf{c} \in \mathcal{C}$ but $\mathbf{c} \neq \mathbf{x}$, the transmitted codeword. For component BCH or RS codes with small t , which are often used in fiber optical communication systems, miscorrections severely degrade the decoding performance of PCs. In such systems, miscorrections are frequent and occur approximately with probability $1/t!$ [19], [20].

III. ERROR-AND-ERASURE DECODING

A. Error-and-Erasure Decoder (EaED)

In this paper, we propose the following error-and-erasure decoder (EaED), which is a modification of [21], Sec. 3.8.1]. Let $\mathbf{y} \in \{0, ?, 1\}^n$ be the received row/column vector, and let $\mathbf{w} := \text{EaED}(\mathbf{y})$ be the decoding result.

If $E(\mathbf{y}) \geq d_{\text{des}}$, the EaED does not decode and declares a failure, returning $\mathbf{w} = \mathbf{y}$, as a large number of erasures cannot be handled by the decoder.

If $E(\mathbf{y}) < d_{\text{des}}$, the erasure positions of \mathbf{y} are first filled with two complementary *random* vectors $\mathbf{p}^{(1)}, \mathbf{p}^{(2)} \in \{0, 1\}^{E(\mathbf{y})}$, i.e., $\mathbf{p}^{(1)} + \mathbf{p}^{(2)} = (1, 1, \dots, 1)$, resulting in two words $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \in \{0, 1\}^n$. The pair of vectors $(\mathbf{p}^{(1)}, \mathbf{p}^{(2)})$ is called a *filling pattern*

for the erasures. Note that $\mathbf{p}^{(1)}$ and $\mathbf{p}^{(2)}$ are not constant but generated randomly in every execution of the EaED.

Then, two BDD steps are performed. Let

$$\mathbf{w}^{(i)} := \text{BDD}(\mathbf{y}^{(i)})$$

for $i \in \{1, 2\}$. The EaED output is determined based on the two BDD outputs using the following rules:

Case 1: If both BDD steps fail, set $\mathbf{w} = \mathbf{y}$.

Case 2: If $\mathbf{w}^{(i)} \in \mathcal{C}$ for exactly one $\mathbf{w}^{(i)}$, set $\mathbf{w} = \mathbf{w}^{(i)}$.

Case 3: If both BDD steps succeed, let

$$d_i = d_{\sim E(\mathbf{y})}(\mathbf{y}, \mathbf{w}^{(i)})$$

for $i \in \{1, 2\}$. We chose $\mathbf{w} = \mathbf{w}^{(1)}$ if $d_1 < d_2$ and $\mathbf{w} = \mathbf{w}^{(2)}$ if $d_1 > d_2$; If $d_1 = d_2$, one of the codewords $\mathbf{w}^{(i)}$ is chosen at random.

The EaED algorithm is summarized in Algorithm 2. Note that the EaED reduces to a conventional BDD decoder when there are no erasures, i.e., $E(\mathbf{y}) = 0$.

Another commonly-used EaE decoder is a one-step EaE decoding algorithm² proposed by Forney in [22]. It extends the Gorenstein-Zierler algorithm [23], Ch. 6] and resolves errors and erasures at the same time by solving the key equation. It requires some modifications in the key equation solver inside the decoder while EaED uses two legacy BDDs for the BSC with some additional operations and control logic. Let \mathbf{w} denote the decoding result. Forney's EaE decoder follows the decoding rule given by

$$\mathbf{w} = \begin{cases} \mathbf{c} & \exists \mathbf{c} \in \mathcal{C} \text{ such that } \mathbf{y} \in \mathcal{S}_t^3(\mathbf{c}) \\ \mathbf{y} & \text{otherwise.} \end{cases}$$

We do not use Forney's EaE decoder in this paper because an EaED, like Forney's EaE decoder, can correct any joint EaE pattern if $\mathbf{y} \in \mathcal{S}_t^3(\mathbf{c})$ [7], Theorem 1].

Moreover, the EaED may correct some EaE patterns for $2 d_{\sim E(\mathbf{y})}(\mathbf{y}, \mathbf{c}) + E(\mathbf{y}) \geq d_{\text{des}}$ because all (or large enough number) of the erasures may possibly be filled with a correct transmit value when generating $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$. Thus, the EaED has potentially higher error-correcting capabilities than the one-step EaE decoder (see also Section III-B) but is also more prone to miscorrections without the constraint that $\mathbf{w} \in \mathcal{S}_t^3(\mathbf{y})$. Consequently, in our previous work, the EaED did not yield a satisfying decoding performance gain for PCs due to the lack of miscorrection control [7]. In this paper, we deal with the miscorrection problem to fully exploit the error-correcting potential of the EaED.

B. Failure Analysis of EaED With Ideal Miscorrection Detection

In this section, we analyze the error-correcting ability of a genie-aided *ideal* EaED following the decoding rule given by

$$\text{ideal EaED}(\mathbf{y}) = \begin{cases} \mathbf{x} & \text{if EaED}(\mathbf{y}) = \mathbf{x} \\ \mathbf{y} & \text{otherwise,} \end{cases}$$

²This decoder was previously referred to as EaED+ in [7]. We do not use this name to avoid confusion with DRSD+ proposed in the later sections.

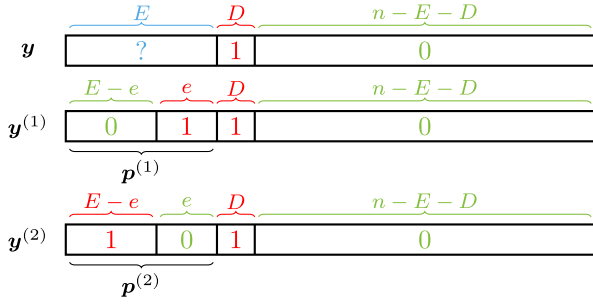


Fig. 2. Graphical illustration of the number of EaEs with a filling pattern ($\mathbf{p}^{(1)}, \mathbf{p}^{(2)}$) assuming the zero codeword was transmitted.

where \mathbf{x} is the transmit codeword. It can be seen as an EaED followed by an ideal miscorrection detection that discards all miscorrections. Although such a decoder may be impossible to realize, our simulation results in Section VI show that our proposed novel decoder approaches its performance when the number of erasures is moderate.

If no miscorrections happen, the success rate of one component code decoding by ideal EaED can be calculated combinatorically. Let $P_s(D, E)$ denote the ideal EaED success probability when decoding with D errors and E erasures.

Case 1: If $D > t$ or $E \geq d_{\text{des}}$, $P_s(D, E) = 0$. When $D > t$, miscorrection-free BDD decoding succeeds neither for $\mathbf{y}^{(1)}$ nor $\mathbf{y}^{(2)}$. When $E \geq d_{\text{des}}$, we do not decode (see Algorithm 2).

Case 2: Assume $D \leq t, E < d_{\text{des}}$, the ideal EaED succeeds if and only if

$$\exists i \in \{1, 2\}, \text{BDD}(\mathbf{y}^{(i)}) = \mathbf{w}^{(i)} = \mathbf{x}.$$

Let e denote the number of positions in $\mathbf{p}^{(1)}$ that differ from the transmit codeword \mathbf{x} . Fig. 2 illustrates the received word \mathbf{y} and both $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$ where the erasures have been replaced by the filling patterns. Under the assumption that $\mathbf{x} = \mathbf{0}$ was transmitted, e denotes the number of “1”s in $\mathbf{p}^{(1)}$. Then, the number of such positions in $\mathbf{p}^{(2)}$ is $E - e$ because $\mathbf{p}^{(1)}$ and $\mathbf{p}^{(2)}$ are complementary. The number of errors in $\mathbf{y}^{(1)}$ is $D + e$ and the number of errors in $\mathbf{y}^{(2)}$ is $D + E - e$. Therefore, the condition for correctability reduces to $D + e \leq t$ or $D + E - e \leq t$, i.e.,

$$e \in [0, t - D] \cup [E + D - t, E]. \quad (1)$$

Case 2.1: If $2D + E < d_{\text{des}} \leq 2t + 2$, $P_s(D, E) = 1$ (with $d_{\text{des}} = 2t + 1$ for BCH codes and $d_{\text{des}} = 2t + 2$ for their even-weight subcodes). Assume that (1) does not hold, i.e., $t - D < e < E + D - t$ for some e , which is the same as saying $t - D + 1 \leq e \leq E - (t - D) - 1$. This means that $2D + E \geq 2t + 2$, contradicting $2D + E < d_{\text{des}}$.

Case 2.2: If $2D + E \geq d_{\text{des}}$, we calculate $P_s(D, E)$ by counting the number of cases where (1) holds. We first show that it is impossible that $e \leq t - D$ and $e \geq E + D - t$ are true at the same time. Assume that both conditions are fulfilled, then $E + D - t \leq e \leq t - D$ for some e . This leads to $2D + E \leq 2t$. As $d_{\text{des}} > 2t$, this is impossible. Therefore, either $\mathbf{y}^{(1)}$ or $\mathbf{y}^{(2)}$ leads to the correct transmit codeword. If $e \leq t - D$, there are

TABLE I
EAED SUCCESS RATE FOR DIFFERENT EAE PATTERN FOR A $t = 2, d_{\text{des}} = 6$ COMPONENT CODE

$D \backslash E$	0	1	2	3	4	5
0	1.000	1.000	1.000	1.000	1.000	1.000
1	1.000	1.000	1.000	1.000	0.625	0.375
2	1.000	1.000	0.500	0.250	0.125	0.063

TABLE II
FORNEY'S EAE DECODER SUCCESS RATE FOR DIFFERENT EAE PATTERN FOR A $t = 2, d_{\text{des}} = 6$ COMPONENT CODE

$D \backslash E$	0	1	2	3	4	5
0	1.000	1.000	1.000	1.000	1.000	1.000
1	1.000	1.000	1.000	1.000	0	0
2	1.000	1.000	0	0	0	0

TABLE III
EAED SUCCESS RATE FOR DIFFERENT EAE PATTERN FOR A $t = 2, d_{\text{des}} = 6$ COMPONENT CODE IN $\ell = 5$ DECODING TRIALS

$D \backslash E$	0	1	2	3	4	5
0	1.000	1.000	1.000	1.000	1.000	1.000
1	1.000	1.000	1.000	1.000	0.993	0.905
2	1.000	1.000	0.969	0.762	0.487	0.276

$\sum_{e=0}^{t-D} \binom{E}{e}$ possible filling patterns. If $e \geq E + D - t$, there are $\sum_{e=E-(t-D)}^E \binom{E}{e}$ possible filling patterns. Therefore, there are

$$\sum_{e=0}^{t-D} \binom{E}{e} + \sum_{e=E-(t-D)}^E \binom{E}{e} = 2 \sum_{e=0}^{t-D} \binom{E}{e}$$

cases where decoding will succeed. In total, there are 2^E possible filling patterns for the E erasures and each pattern is equally likely. Thus, $P_s(D, E) = 2 \sum_{e=0}^{t-D} \binom{E}{e} / 2^E$ in this case.

Therefore, we summarize

$$P_s(D, E) = \begin{cases} 1 & 2D + E < d_{\text{des}}, \\ 0 & E \geq d_{\text{des}} \text{ or } D > t, \\ 2^{1-E} \sum_{e=0}^{t-D} \binom{E}{e} & \text{otherwise.} \end{cases}$$

For example, Table I gives the success rate of EaED when decoding with various numbers of EaEs for a $t = 2, d_{\text{des}} = 6$ component code. When $2D + E \geq d_{\text{des}}$, the success probability decreases with an increasing number of EaEs. However, the values are not zero as for the one-step EaE decoder by Forney (Table II). The advantage of ideal EaED becomes larger during iterative decoding, where new, independent filling patterns are chosen each time a codeword is decoded. With ideal miscorrection detection, if the success probability of one EaED step is P_s , the probability of success after ℓ decoding trials is $1 - (1 - P_s)^\ell$. For example, Table III shows the success probability after $\ell = 5$ decoding trials for EaED with ideal miscorrection detection. This also proves that random filling patterns ($\mathbf{p}^{(1)}, \mathbf{p}^{(2)}$) are crucial for the performance of the proposed decoder. As the number of EaEs is reduced during the PC decoding process, the actual success rate can be potentially even higher. Moreover, the decoding success probability $1 - (1 - P_s)^\ell$ approaches 1 with a sufficiently large number of iterations, which is not

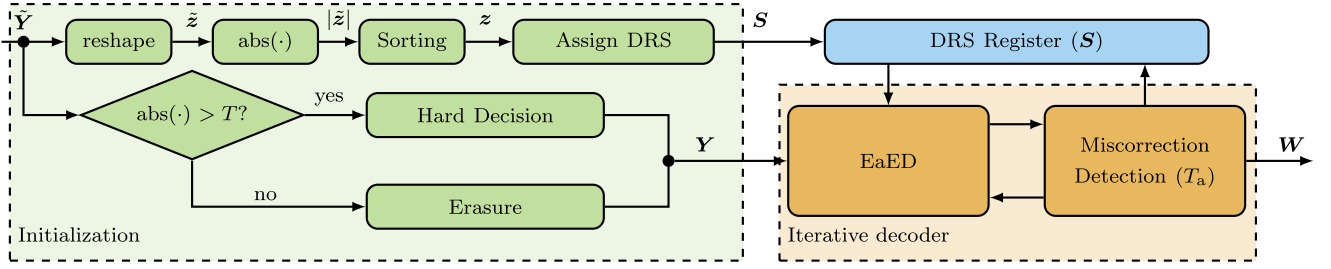


Fig. 3. Block diagram of the proposed DRSD.

Algorithm 3: Initialization of the Dynamic Reliability Scores (DRSs).

1 Input: $\tilde{Y} \in \mathbb{R}^{n \times n}$
2 $\tilde{z} \leftarrow \text{reshape}(\tilde{Y})$ // (2)
3 $z \leftarrow \text{sort } |\tilde{z}| \text{ ascendingly}$
4 $\sigma(\cdot) \leftarrow \text{permutation function of the indices}$
5 $S \leftarrow \text{assign DRS according to } z$ // (3)
6 Output: $S \in \{9, 10, \dots, 24\}^{n \times n}$

realistic in practice. Thus, we only use 10 decoding iterations of the ideal EaED when using it as a benchmark.

IV. DYNAMIC RELIABILITY SCORE (DRS) DECODING

To represent and update the reliability of a bit in the PC block, we introduce a reliability measure called DRS, which is stored in an additional register separately from the PC codeword. The DRS reflects the reliability of a bit from both its channel LLR and its behavior during decoding. We propose to represent the DRSs by 5-bit integers in the range $[0, 31]$ (as we observe that further increasing the number of representation levels does not improve the performance of the proposed decoder significantly). We introduce an anchor threshold T_a . All bits with a $\text{DRS} > T_a$ are classified as *anchor bits* (following [16]) during decoding and are not allowed to be flipped by a component code decoder. Note that the DRS is not a fully-accurate measurement of the reliability, i.e., we cannot say that a bit with higher DRS must be more reliable than a bit with lower DRS. However, we can say that bits with DRS above the threshold T_a can be considered as correct with high confidence. The DRS is used for miscorrection detection in iterative decoding with EaED.

The block diagram and workflow of the proposed dynamic reliability score decoder (DRSD) is shown in Fig. 3.

At the initialization, the received unquantized real-valued channel output of the PC block $\tilde{Y} \in \mathbb{R}^{n \times n}$ is fed to two paths. In the upper path, the initial value of the DRSs S is calculated for the PC block. To do this, we first reshape \tilde{Y} into a vector $\tilde{z} = (\tilde{z}_1, \tilde{z}_2, \dots, \tilde{z}_{n^2}) \in \mathbb{R}^{n^2}$ where

$$\tilde{z}_{n(i-1)+j} = \tilde{Y}_{i,j} \quad (2)$$

Then the absolute value of \tilde{z} is taken and sorted ascendingly, resulting in a vector $z = (z_1, z_2, \dots, z_{n^2}) \in \mathbb{R}_{\geq 0}^{n^2}$ with

$$z_i \geq z_j, \forall i > j,$$

Algorithm 4: Dynamic Reliability Score Decoder (DRSD) and the DRSD+ Termination Option.

1 Input: $\tilde{Y} \in \mathbb{R}^{n \times n}$
2 $Y \leftarrow 0^{n \times n}$
3 **for** $i = 1, 2, \dots, n$ **do** // Initialization
4 **for** $j = 1, 2, \dots, n$ **do**
5 **if** $|\tilde{Y}_{i,j}| \leq T$ **then** $y_{i,j} = ?$
6 **else if** $\tilde{Y}_{i,j} > T$ **then** $y_{i,j} = 0$
7 **else** $y_{i,j} = 1$
8 $S \leftarrow \text{initial DRSs}$ // Algorithm 3
9 **for** $\ell = 1, 2, \dots, L - L/5$ **do** // Decoding with DRS
10 **for** $i = 1, 2$ **do**
11 **for** $j = 1, 2, \dots, n$ **do**
12 **if** $\text{Syndrome}(y_j) = 0$ **then**
13 **for** $m \in \{1, 2, \dots, n\}$ **do**
14 $S_{j,m} = \min(S_{j,m} + 1, 31)$
15 **continue**
16 $w_{\text{tmp}} \leftarrow \text{EaED}(y_j)$ // Algorithm 2
17 $k \leftarrow 0^n$
18 **for** $m = 1, 2, \dots, n$ **do**
19 **if** $Y_{j,m} \neq ?$ **then** $k_m = w_{\text{tmp},m} \oplus Y_{j,m}$
20 $a \in \{0, 1\}^n$, $a_m = \mathbb{1}_{\{S_{j,m} > T_a\}}$
21 **if** $\text{sum}(k \cdot a^T) = 0$ **then** // Not miscorrection
22 $w_j \leftarrow w_{\text{tmp}}$
23 **for** $m \in \{1, 2, \dots, n\}$ **do**
24 **if** k_m **then**
25 $S_{j,m} = \max(S_{j,m} - 1, 0)$
26 $Y \leftarrow W^T$, $S \leftarrow S^T$
27 **if** W is valid codeword **then return** W
28 **if** $\ell \bmod 5 = 0$ **then** // Increase anchor threshold
29 $T_a = T_a + 1$
30 **for** $\ell = 1, 2, \dots, L/5$ **do** // Termination
31 **if** DRSD **then**
32 **for** $i = 1, 2$ **do** // Plain iEaED
33 **for** $j = 1, 2, \dots, n$ **do**
34 $w_j \leftarrow \text{EaED}(y_j)$ // Algorithm 2
35 $Y \leftarrow W^T$
36 **if** W is valid codeword **then return** W
37 **if** DRSD+ **then**
38 **repeat** line 10-27 **with** $T_a = T_a^*$
39 Replace erasures in W with random binary value
40 Output: $W \in \{0, 1\}^{n \times n}$

where

$$z_{\sigma(i)} = |\tilde{z}_i|$$

and $\sigma(\cdot)$ is a permutation of the indices corresponding to the sorting operation. Then, the initial DRSs $S \in \mathbb{Z}_{32}^{n \times n}$ are assigned

by

$$S_{i,j} = 9 + \left\lfloor \frac{16 \cdot (\sigma(n(i-1) + j) - 1)}{n^2} \right\rfloor, \quad (3)$$

such that the initial DRSs are in the range of [9,24]. This process is summarized in Algorithm 3. This initial DRS value is stored in the DRS register. The anchor threshold T_a is set as the optimal value found during simulation (see Section V).

In the lower branch, erasures are marked, i.e., $Y_{i,j} = ?$ if $|\tilde{Y}_{i,j}| < T$, with T (defined in Section II) being an optimizable threshold. For non-erased bits, a usual hard-decision is performed. The initial block $\mathbf{Y} \in \{0, ?, 1\}^{n \times n}$ is passed to the iterative row and column decoder.

During iterative decoding, the rows and columns are decoded alternately until the maximal number of iterations L is reached, as in a conventional iBDD decoder. We replace the BDD decoder by an EaED. Moreover, every EaED decoding step is followed by the miscorrection detection step evaluating whether an anchor bit is flipped by the EaED output. In this case, this decoder output is discarded and the DRS for all the anchor bits in conflict with this EaED output is reduced by 1. If a decoding step does not flip any anchor bit, this EaED output will be accepted while the DRS of all flipped bits is reduced by 1. If a vector is already a codeword and thus no decoding is performed, the DRS of every bit in it is increased by 1. This is based on the observation that the longer a bit stays constant during iterative decoding, the more reliable it is. A similar idea has been exploited in AD [16]. The messages used for updating the DRS are all set to 1 to enable a simple hardware implementation of the DRS register avoiding the need for full adders. The DRS values will be clipped to 0 and 31, respectively. In the case of an EaED decoding failure, neither the codeword nor the DRS is changed. In addition, the anchor threshold T_a is increased by 1 after every five decoding iterations, such that a small penalty is given for words that fail to decode consistently. With the update of the DRSs, the anchor bits are reevaluated every iteration. For DRSD^L, the last $L/5$ iterations are performed with plain iEaED. This is to eliminate the effect of erroneous anchor bits with high DRS. After L iterations, the erased values which have not been resolved are replaced with a random binary value. We call the proposed decoder DRSD and the complete algorithm of the proposed decoding scheme is summarized in Algorithm 4.

Furthermore, when L is sufficiently large, (e.g. $L = 20$), we propose an alternative termination option where the few final decoding iterations are no longer plain iEaED but DRSD with a very high and constant but optimizable anchor threshold T_a^* (see Algorithm 4, lines 37-38). We call the resulting algorithm DRSD+. The DRSD+ is motivated by the observation that most of the correct bits will have very high DRSs after a sufficiently large number of iterations (see Section VI). Consequently, when a large T_a^* is used, the erroneous anchor bits are removed while a relatively large set of correct anchor bits are still preserved, resulting in a decoding performance improvement.

V. SIMULATION RESULTS

A. Preparation

We verify and demonstrate the performance of DRS decoding using numerical examples. For the proposed DRSD, we consider both $L = 10$ and $L = 20$ as 10 iterations are often used in the literature for soft-aided PC codewords, but we observe that DRSD can benefit from more than 10 iterations.

We determine the *noise threshold* as the minimal E_b/N_0 with which the target bit error rate (BER) of 10^{-4} after a fixed number of iterations is achieved. For two decoders, we compare their performance by calculating the *noise threshold difference (gain)* $\Delta(E_b/N_0)^*$. The noise thresholds are calculated numerically by a Monte Carlo approach along with a binary search and both the optimal erasure threshold T (T_{opt}) and the optimal initial anchor threshold T_a are found during the search. As the initial DRSs are in the range of [9,24], the initial T_a is also in this range. Moreover, as a too small number of anchor bits does not contribute to the miscorrection detection significantly, we narrow down the search space for T_a to [9,15]. Then, the parameter optimization for both T and T_a can be carried out by a grid search. The anchor threshold T_a depends mostly on t . For $L = 20$, the estimated optimal initial value of the anchor threshold is $T_a = 9$ for $t = 2$, $T_a = 10$ for $t = 3$, and $T_a = 12$ for $t = 4$. For the $(127, k, 4)$ component codes, we exceptionally find $T_a = 14$. For $L = 10$, the initial anchor threshold T_a is reduced by 1 for all component codes. The optimal erasure thresholds T_{opt} are as shown in Fig. 4. We observe that slightly varying T (within $\pm 10\%$) does not degrade the performance severely.

B. Noise Threshold Gain $\Delta(E_b/N_0)^*$

The noise threshold gain $\Delta(E_b/N_0)^*$ that a DRSD²⁰ achieves compared to an iBDD²⁰ decoder is calculated and shown in Fig. 4 with the non-transparent markers. We additionally show the noise threshold gain of EaED with ideal miscorrection detection and $L = 16$ iterations compared to iBDD²⁰ with the transparent markers (the optimal threshold T may differ slightly but is not plotted for the sake of clarity). We use $L = 16$ for EaED with ideal miscorrection detection as it does not require the final 4 iterations of EaED to mitigate the risk of erroneous anchor bits. The dotted line depicts the capacity gain of the BI-AWGN channel compared to the hard-decision BSC. Note that when a data point exceeds the dotted line, the code does not operate beyond capacity. This simply means that the noise threshold gain of DRSD compared to iBDD is larger than the capacity gain of the BI-AWGN channel compared to the BSC. In general, for PCs with various component codes, the noise threshold gain of DRSD is similarly large as the noise threshold gain of EaED with ideal miscorrection detection. This means that our proposed decoder is nearly miscorrection-free. For larger t and higher code rate, the gap between DRSD and the ideal case becomes even smaller.

C. BER Results

In Fig. 5, we compare the residual post-decoding BER for different decoders. We construct two PCs from even-weight

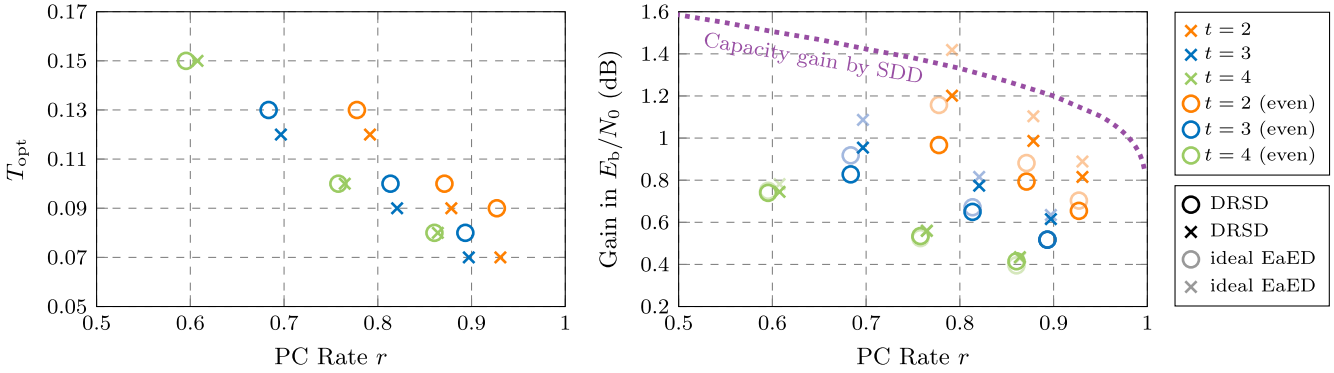


Fig. 4. Simulation results of the parameter analysis: The optimal erasure threshold T_{opt} for DRSD²⁰ found during numerical simulation is plotted in the left subplot. The noise threshold gain that the DRSD²⁰ achieves compared to iBDD²⁰ is plotted in the right subplot with the non-transparent markers. The component codes are (n, k, t) -BCH codes with $n \in \{127, 255, 511\}$ and $t \in \{2, 3, 4\}$ or their even-weight subcodes. The abscissa corresponds to the rate r of the constructed PCs. Additionally on the right subplot, the transparent markers corresponds to the respective EaED with ideal miscorrection detection and $L = 16$ iterations (denoted as “ideal EaED”). The dotted curve marks the capacity gain due to soft decision decoding (SDD) on the binary-input AWGN channel.

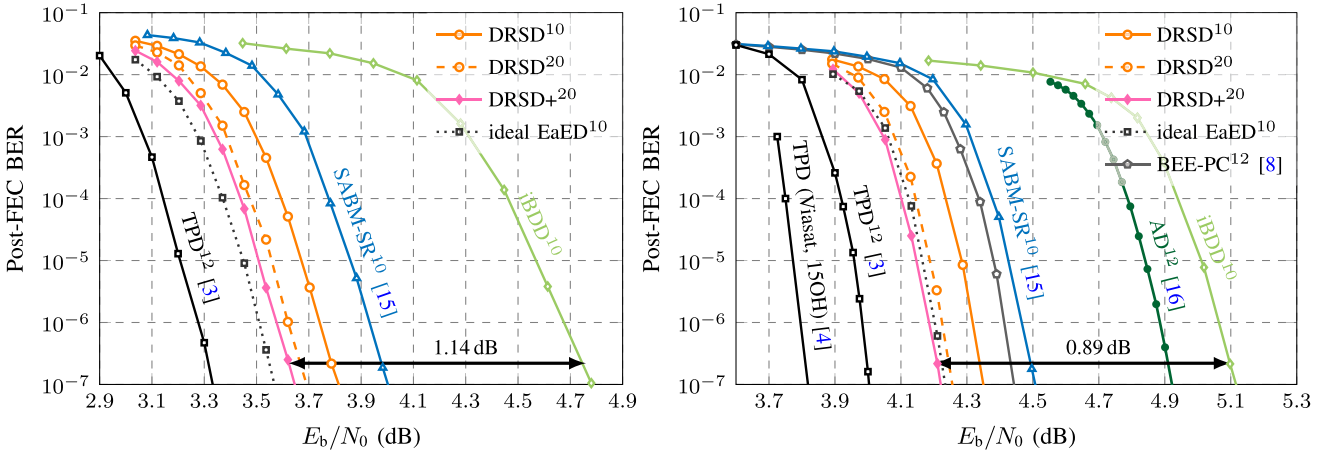


Fig. 5. BER vs. E_b/N_0 (in dB) curve for product codes of rate 0.78 (28% overhead, component code C_1 , left plot) and 0.87 (15% overhead, component code C_2 , right plot). The superscript of the labels denotes the maximum number of decoding iterations L .

subcodes of BCH codes denoted by $C_1(127,112,2)$ and $C_2(255,238,2)$ with PC rates of 0.78 (28% overhead) and 0.87 (15% overhead), respectively. For reference, we show the results of TPD [3], SABM-SR [15], BEE-PC [8], and AD [16] with the data points provided in [8] and [15] where the component codes are singly-extended BCH codes with parameters $(128,113,2)$ and $(256,239,2)$ (hence with a small, negligible rate difference compared to our codes). For iBDD, we observe that the decoding results of $L = 10$ and $L = 20$ do not have a noticeable difference. Thus, we only show the results of iBDD¹⁰. For both PCs, an obvious gain compared to iBDD can be observed with DRSD and the performance is further improved by running 20 iterations. With twice the number of iterations, our decoder behaves very closely to an EaED with ideal miscorrection detection (denoted as “ideal EaED^L” in the figure) and has only a small gap to the significantly more complex TPD. For the rate 0.78 PC, we estimate, at a residual BER of 10^{-15} , an NCG of 10.88 dB (DRSD¹⁰) and 11.03 dB (DRSD²⁰). The NCGs are obtained by extrapolating the simulation results from Fig. 5 using `berfit`

in MATLAB assuming no error floor. For the rate 0.87 PC, the conjectured NCGs are 10.51 dB and 10.59 dB respectively. The modified DRSD+²⁰ provides an additional 0.05 dB coding gain compared to DRSD²⁰ without extra complexity. The optimal erasure threshold for DRSD+²⁰ is $T_{\text{opt}} + 0.2$ with T_{opt} the optimal erasure threshold of DRSD²⁰ and the optimized $T_a^* = 24$. Compared to a plain iBDD decoder, the DRSD+²⁰ yields 1.14 dB and 0.89 dB decoding gain for the rate 0.78 and rate 0.87 PCs, respectively. (As DRSD¹⁰ does not improve the decoding performance significantly, its results are not shown).

We observe that DRSD yields significantly better performance when 20 iterations are used. The reason behind this is that: first, the decoder needs to obtain a both large and reliable set of anchor bits in the first few iterations of decoding as we will show in Section VI. Second, assuming no miscorrections, the use of random filling of the erasures ensures that the probability of correctly decoding words with erasures rises with the number of iterations as shown in Section III. Moreover, as Section VII will show, using 20 iterations does not increase the decoding

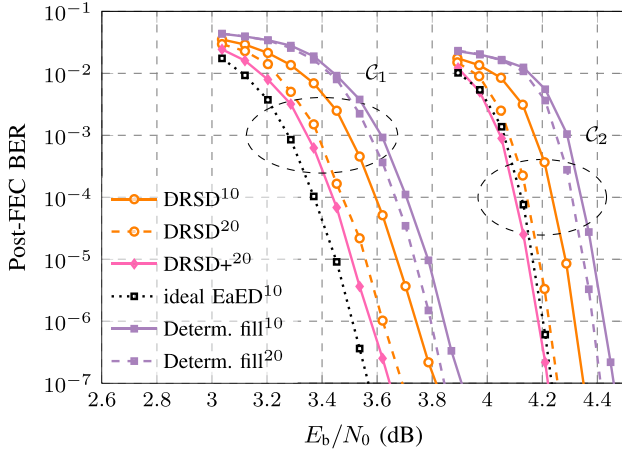


Fig. 6. BER vs. E_b/N_0 (in dB) curve for product codes with component code C_1 and C_2 when decoded with various modifications of DRSD. The superscript of the labels denotes the maximum number of decoding iterations L .

complexity significantly. We conclude that it is much more beneficial to use 20 decoding iterations.

D. The Importance of Random Filling Patterns

In Fig. 6, additionally to the BER results after decoding with the proposed decoding scheme, we show the performance when using deterministic filling vectors for the erased positions within the EaED. The purple curves depict the decoding performance using a DRSD but we replace the random filling pattern $(\mathbf{p}^{(1)}, \mathbf{p}^{(2)})$ in Algorithm 2 (line 5) with $(\mathbf{0}, \mathbf{1})$ as in [21], Sec. 3.8.1]. This degrades the decoding performance by more than 0.1 dB for both PCs. Moreover, we observe that when using a deterministic value for $\mathbf{p}^{(1)}$ and $\mathbf{p}^{(2)}$, allowing 20 iterations does not yield performance gains as large as we observe with DRSD. This is due to the fact that with identical filling patterns, row/column words which cause a decoding failure in early iterations will still cause a decoding failure if its EaEs are not corrected by the corresponding column/row code.

E. Choice of Parameters

In Section IV, we gave the algorithm description with a predefined number of 32 DRS representation levels and an initial DRS range of [9,24]. In this section, we provide some intuition and numerical simulation results which support this choice of parameters.

Figure 7 shows the decoding performance of DRSD²⁰ with different DRS representation levels. We consider the number of levels in terms of the number of bits required for storage. All thresholds inside the algorithm are scaled relatively to the number of representation levels. If the number of representation levels is smaller than 32, we observe a degradation of the decoding performance. However, if a smaller storage space is preferred, the DRSD algorithm can still be used with some loss in decoding performance. As using 64 levels does not improve upon 32 levels significantly, we choose 32 levels for the proposed DRSD throughout this paper.

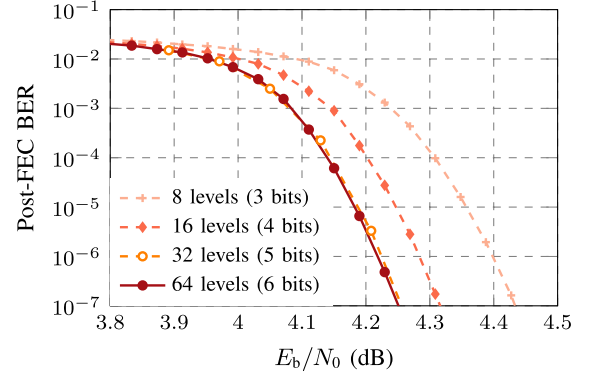


Fig. 7. DRSD²⁰ decoding performance for a PC based on the (255,238,2) even-weight BCH subcode with different DRS representation levels.

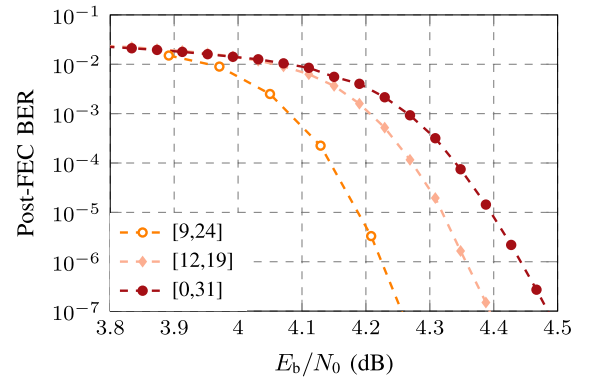


Fig. 8. DRSD²⁰ decoding performance for a PC based on the (255,238,2) even-weight BCH subcode with different initial DRS ranges $[a, b]$.

Next, we consider the range into which the initial DRS are distributed: first, the range should not be too small so that the channel output reliability information can be fully exploited. Second, it should not be too big either, otherwise, the updating mechanism during iterative decoding would have too little impact on the scores and thus cannot efficiently update the anchor bits. We exemplarily show the performance for some ranges $[a, b]$ in Fig. 8 for 32 DRS representation levels. We observe that an initial range that is too big or too small degrades the decoding performance. Hence, we choose to use the initial range of [9,24], as it is a good compromise between representing the soft channel output and leaving sufficient room for increasing and decreasing the scores.

VI. HEURISTIC PERFORMANCE ANALYSIS

In this section, we investigate the iterative decoding process to show the connection between anchor bits and miscorrections. This also illustrates the advantage of the DRSD+ termination variant. Figure 9 shows the fraction of anchor bits, erroneous anchor bits, and miscorrections during the decoding of a PC with component code C_2 and $L = 20$ iterations. The quantities are provided for each half iteration which is defined as the processing of decoding all the rows *or* all the columns of a PC block once. The values are calculated by averaging the respective

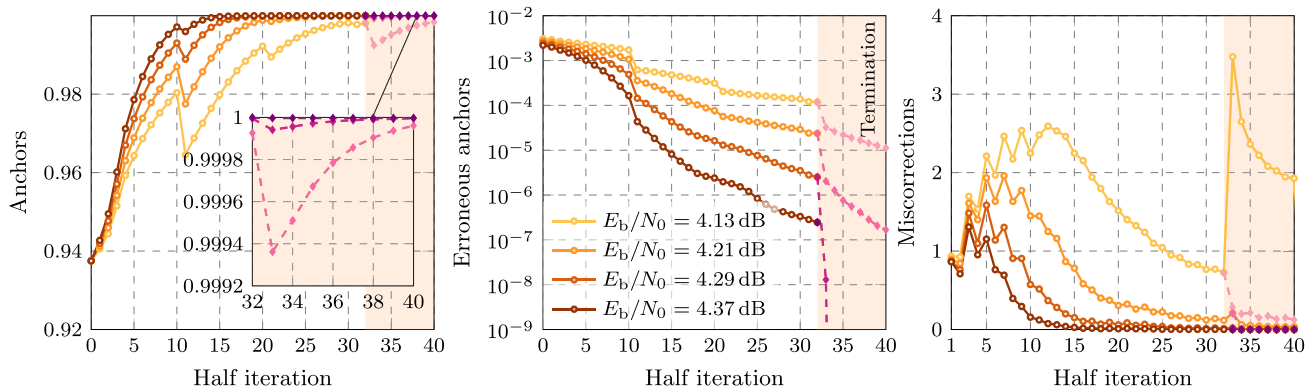


Fig. 9. Percentage of marked anchor bits, percentage of wrongly marked anchor bits, and average number of miscorrections in every iteration in the decoding of a (255,238,2) even-weight BCH subcode for various E_b/N_0 . The solid orange lines stand for DRSD²⁰ and the dashed pink lines for DRSD⁺²⁰.

quantities during the Monte-Carlo simulations which produced Fig. 5.

The leftmost subfigure depicts the fraction of anchors in each iteration and the middle subfigure depicts the fraction of wrongly marked anchor bits for different values of E_b/N_0 (the 0-th half iteration stands for the state after initialization and before iterative decoding). According to Section V, we set the initial anchor threshold to $T_a = 9$. Since there are 16 representation levels (in the interval [9,24]) for the initial DRSs, the initial fraction of anchor bits ($\text{DRS} > T_a$) is always around $15/16 \approx 93.8\%$. The number of erroneous anchor bits decreases as E_b/N_0 increases. The ratio of anchor bits increases during iterative decoding due to the raise of DRSs for the bits that are successfully decoded. At the same time, the DRS of the bits where the row and column decoders disagree is reduced. This reduces the number of erroneous anchor bits. T_a is increased by 1 after every 10 half-iterations, corresponding to the drop in both overall and erroneous anchor bits. As the last 4 iterations of DRSD are plain iEaED and the DRSs are not used, the solid lines stop after 32 half-iterations.

The rightmost subfigure in Fig. 9 shows the number of miscorrections in each half iteration. For a poor E_b/N_0 with a high post-decoding BER, the DRS updates fail to raise the ratio of anchor bits and decrease the ratio of erroneous anchor bits sufficiently. Consequently, the number of miscorrections is high and even increases during decoding (due to error propagation). However, in the waterfall region, our decoder effectively prevents miscorrections. When $E_b/N_0 \geq 4.29$ dB, nearly all decoded bits are marked correctly as anchor bits after a few iterations, resulting in almost no miscorrections, whereas a plain iEaED without miscorrection detection will have more than a hundred miscorrections in each iteration given the same E_b/N_0 value, as we observed during experiments.

As shown in Fig. 9, the number of miscorrections in the last few plain iEaED decoding iterations is relatively high. However, we observe that without these termination iterations, the decoding performance suffers from a high error floor due to erroneous anchor bits. The DRSD+ termination option removes almost

all erroneous anchors with the high anchor threshold T_a^* . This reduces the number of miscorrections in the last iterations while also eliminating the effect of erroneous anchor bits, leading to the improved decoding performance.

VII. COMPLEXITY ANALYSIS

The proposed decoder does not require soft message passing. The DRS register sends a binary message to the EaED indicating whether a bit is an anchor or not. The EaED signals to the DRS register if DRSs should be increased or decreased. During EaED decoding, ternary messages are passed.

The major computational overhead of the proposed DRSD compared to iBDD is the usage of EaED, where the presence of erasures in a component word entails two BDD steps. This increases the overall number of BDD steps, especially in the first few iterations, before the erasures are resolved. Words without erasures can be decoded with simple BDD. The additional storage for DRSs is relatively small as the DRSs are stored using 5-bit integers. As the main cause of the computational overhead is the increase in the number of BDD steps, we show in Fig. 10 an exemplary comparison of the BDD steps in the decoding of a PC with component code C_2 . See Fig. 5 for the decoding performance of this code. We record the number of BDD steps used in three decoding schemes. One is the conventional iBDD¹⁰, which serves as a baseline. Then we plot the results of DRSD¹⁰ and DRSD²⁰ respectively. All the results are normalized by $(255 \cdot 20)$, which is the total number of BDD steps in the iBDD¹⁰ decoding process. Two facts cause the normalized value to be less than 1: one is early termination due to the decoding success of the entire PC block. Another one is that we do not count the decoding of words with zero-syndrome, as the actual decoding algorithm does not need to be carried out. In the low E_b/N_0 region, the DRSD decoder requires a relatively large number of BDD steps. However, for larger E_b/N_0 , the number of decoding steps is within the same order of magnitude as for iBDD. We also observe that the effective number of BDD steps in the DRSD²⁰ scheme increases only slightly compared to DRSD¹⁰. Other soft-aided decoding schemes such as the

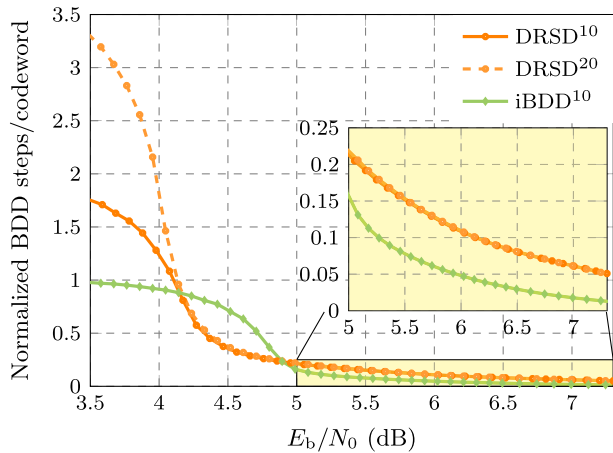


Fig. 10. Normalized number of BDD steps in decoding of a (255,238,2) even-weight BCH code-based PC with iBDD and DRSD.

SABM-SR decoder [15] also require a number of BDD steps that is likely within the same order of magnitude than DRSD.

VIII. CONCLUSION

We proposed a novel hybrid decoding scheme for PCs and simulation results show that the proposed DRS decoder significantly improves the decoding performance of the conventional hard decision iBDD for PCs with near miscorrection-free EaE decoding. For two exemplary PCs of rate 0.87 and rate 0.78, DRSD closes roughly 80% of the gap between hard-decision iBDD and soft-decision TPD which outperforms other soft-aided HDD schemes. Complexity analysis shows that the decoding complexity is similarly low as for conventional HDD iBDD decoder. The high NCGs make this scheme a promising candidate for future low-complexity optical communication systems.

One of the open questions is the error floor of the proposed DRSD. With a proper choice of the parameters, we conjecture that the proposed DRSD should not have a worse error floor than a plain iterative EaE decoder, which has been investigated in [6] for PCs with relatively small block lengths. The reason is that DRSD does not introduce additional miscorrections compared with the plain iterative EaE decoder. However, the EaE decoder used in this work is different from the EaE decoder studied in [6]. Therefore, accurately calculating the error floor for decoding the PCs with large block lengths that are used in this work remains difficult and is left as future work.

ACKNOWLEDGMENT

The authors would like to thank Alex Alvarado and Alireza Sheikh for providing the numerical BEE-PC simulation results used in Fig. 5.

REFERENCES

- [1] S. Miao, L. Rapp, and L. Schmalen, "Improved soft-aided error-and-erasure decoding of product codes with dynamic reliability scores," in *Proc. Opt. Fiber Commun. Conf.*, 2022, Paper W3H.2.
- [2] P. Elias, "Coding for noisy channels," in *Proc. IRE Conv. Rec., Part IV*, 1955, pp. 37–46.
- [3] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1003–1010, Aug. 1998.
- [4] V. Inc., "ECC66100 series SD-FEC encoder/decoder cores," 2017, Accessed: Jun. 01, 2022. [Online]. Available: https://www.viasat.com/content/dam/us-site/semiconductors/documents/ecc_66100_datasheet_2_pgr_007_web_0_1126494.pdf
- [5] H. Sun et al., "800 G DSP ASIC design using probabilistic shaping and digital sub-carrier multiplexing," *J. Lightw. Technol.*, vol. 38, no. 17, pp. 4744–4756, 2020.
- [6] D. K. Soma, A. K. Pradhan, and K. Narayanan, "Errors and erasures decoding of product codes for optical transport networks," *IEEE Commun. Lett.*, vol. 25, no. 8, pp. 2482–2486, Aug. 2021.
- [7] L. Rapp and L. Schmalen, "Error-and-erasure decoding of product and staircase codes," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 32–44, Jan. 2022.
- [8] A. Sheikh, A. G. i. Amat, and A. Alvarado, "Novel high-throughput decoding algorithms for product and staircase codes based on error-and-erasure decoding," *J. Lightw. Technol.*, vol. 39, no. 15, pp. 4909–4922, 2021.
- [9] A. Sheikh, A. G. i. Amat, and G. Liva, "Iterative bounded distance decoding of product codes with scaled reliability," in *Proc. IEEE Eur. Conf. Opt. Commun.*, 2018, pp. 1–3.
- [10] A. Sheikh, A. G. i. Amat, and G. Liva, "Binary message passing decoding of product-like codes," *IEEE Trans. Commun.*, vol. 67, no. 12, pp. 8167–8178, Dec. 2019.
- [11] A. Sheikh, A. G. 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.
- [12] A. Sheikh, A. G. i. Amat, G. Liva, C. Häger, and H. D. Pfister, "On low-complexity decoding of product codes for high-throughput fiber-optic systems," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process.*, 2018, pp. 1–5.
- [13] A. Sheikh, A. G. i. Amat, and G. Liva, "Binary message passing decoding of product codes based on generalized minimum distance decoding," in *Proc. IEEE 53rd Annu. Conf. Inf. Sci. Syst.*, 2019, pp. 1–5.
- [14] Y. Lei et al., "Improved decoding of staircase codes: The soft-aided bit-marking (SABM) algorithm," *IEEE Trans. Commun.*, vol. 67, no. 12, pp. 8220–8232, Dec. 2019.
- [15] G. Liga, A. Sheikh, and A. Alvarado, "A novel soft-aided bit-marking decoder for product codes," in *Proc. Eur. Conf. Opt. Commun.*, 2019, pp. 1–4.
- [16] C. Häger and H. D. Pfister, "Approaching miscorrection-free performance of product codes with anchor decoding," *IEEE Trans. Commun.*, vol. 66, no. 7, pp. 2797–2808, Jul. 2018.
- [17] C. Senger, "Improved iterative decoding of product codes based on trusted symbols," in *Proc. Int. Symp. Inf. Theory*, 2019, pp. 1342–1346.
- [18] C. Senger, "On list decoding of generalized Reed-Solomon codes under partial codeword knowledge," in *Proc. IEEE 12th Int. ITG Conf. Syst., Commun. Coding*, 2019, pp. 1–5.
- [19] R. McEliece and L. Swanson, "On the decoder error probability for Reed-Solomon codes (corresp.)," *IEEE Trans. Inf. Theory*, vol. 32, no. 5, pp. 701–703, Sep. 1986.
- [20] J. Justesen, "Performance of product codes and related structures with iterated decoding," *IEEE Trans. Commun.*, vol. 59, no. 2, pp. 407–415, Feb. 2011.
- [21] T. K. Moon, *Error Correction Coding - Mathematical Methods and Algorithms*. Hoboken, NJ, USA: Wiley, 2005.
- [22] G. Forney, "On decoding BCH codes," *IEEE Trans. Inf. Theory*, vol. 11, no. 4, pp. 549–557, Oct. 1965.
- [23] R. M. Roth, *Introduction to Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2006.