Recursive Subproduct Codes with Reed-Muller-like Structure

Aditya Siddheshwar, Lakshmi Prasad Natarajan, Prasad Krishnan

***Abstract*—We study a family of subcodes of the m-dimensional product code $\mathcal{C}^{\otimes m}$ ('subproduct codes') that have a recursive Plotkin-like structure, and which include Reed-Muller (RM) codes and Dual Berman codes as special cases. We denote the codes in this family as $\mathcal{C}^{\otimes[r,m]}$, where $0 \leq r \leq m$ is the 'order' of the code. These codes allow a 'projection' operation that can be exploited in iterative decoding, viz., the sum of two carefully chosen subvectors of any codeword in $\mathcal{C}^{\otimes[r,m]}$ belongs to $\mathcal{C}^{\otimes[r-1,m-1]}$. Recursive subproduct codes provide a wide range of rates and block lengths compared to RM codes while possessing several of their structural properties, such as the Plotkin-like design, the projection property, and fast ML decoding of first-order codes. Our simulation results for first-order and second-order codes, that are based on a belief propagation decoder and a local graph search algorithm, show instances of subproduct codes that perform either better than or within 0.5 dB of comparable RM codes and CRC-aided Polar codes.**

The extended version of this paper containing the proofs of all claims is available in [1].

I. Introduction

The product construction [2] is a well-known technique to obtain new codes from existing codes. When used m times repeatedly on an $[n, k, d]$ code $\mathcal{C}$ we obtain an m-fold product code $\mathcal{C}^{\otimes m}$ with parameters $[n^m, k^m, d^m]$. The parity-check constraints of product codes can be expressed through a factor graph with generalized check nodes. Hence, it is natural to use iterative techniques to decode product codes [3], [4]. The product construction has been a popular technique for designing coding schemes for the additive white Gaussian noise (AWGN) channel; examples of some recent work in this direction include [5]–[7]. Several variants of the product construction are known that have yielded codes with good error correction properties, such as subcodes of product codes ('subproduct codes') that satisfy certain symmetry properties [4], [8], [9], and constructions using ideas from convolutional codes [10], [11].

The objective of this work is to use the product construction to identify codes that satisfy a *projection* property: the sum of two carefully chosen subvectors of any codeword lies in another code for which a fast decoder is available. In such cases, the latter code can be used as a generalized check node in the factor graph for iterative decoding. Reed-Muller (RM) codes [12], [13] enjoy such a projection property, and this was used to design the recursive projection aggregation (RPA) decoder [14] and a related belief propagation (BP) decoder [15] that use the fact that $\mathrm{RM}(1, m)$ codes have a fast ML decoder [16], [17]. A similar decoding technique for extended cyclic codes was proposed in [18].

Starting from any $[n, k \geq 2, d]$ code $\mathcal{C}$ which contains the all-one codeword $\mathbb{1}$, we identify a family of subproduct codes $\{\mathbb{1}, \mathbf{0}\} = \mathcal{C}^{\otimes[0,m]} \subset \mathcal{C}^{\otimes[1,m]} \subset \cdots \subset \mathcal{C}^{\otimes[m,m]} = \mathcal{C}^{\otimes m}$. The parameters of $\mathcal{C}^{\otimes[r,m]}$, where the 'order' $r \in \{0, 1, \ldots, m\}$, are $\left[n^m, \sum_{l=0}^{r} \binom{m}{l}(k-1)^l, d^r n^{(m-r)}\right]$. These codes include the RM codes and Dual Berman codes [19]–[21] as special cases, corresponding to, $\mathcal{C} = \mathbb{F}_2^2$ and $\mathcal{C} = \mathbb{F}_2^n$, $n \geq 3$, respectively. These subproduct codes satisfy a recursive structure which generalizes the Plotkin construction of RM codes. As with RM codes, a projection operation can be applied on $\mathcal{C}^{\otimes[r,m]}$ to yield $\mathcal{C}^{\otimes[r-1,m-1]}$, and further, the code $\mathcal{C}^{\otimes[1,m]}$ can be ML decoded efficiently.

We describe the construction and basic properties of the recursive subproduct codes $\mathcal{C}^{\otimes[r,m]}$, including the identification of their minimum weight codewords, in Section II. A fast ML decoder and a soft-output max-log-MAP decoder for the first-order codes $\mathcal{C}^{\otimes[1,m]}$ is described in Section III. In Section IV we introduce the projection operation, describe a BP decoder that uses projections, and an improvement using a local graph search algorithm [22] for second-order codes $\mathcal{C}^{\otimes[2,m]}$. We present some simulation results for first- and second-order codes in Section V to show that it is possible to design subproduct codes that perform either better than or within 0.5 dB of comparable RM codes and CRC-aided Polar codes.

Recursive subproduct codes provide a wider range of rates and block lengths compared to RM codes while possessing several of their structural properties. We believe that these codes warrant further investigation, such as in the context of low-capacity channels [23], designing more efficient decoders, improving the performance using high-rate outer codes, and application in private information retrieval [24], [25].

Notation: The symbol $\otimes$ denotes the Kronecker product. For $\ell > 0$, let $[\![\ell]\!] = \{0, 1, \ldots, \ell - 1\}$. We use capital letters to denote matrices (such as G), and small bold letters to denote row vectors (such as $\boldsymbol{g}$). For two vectors $\boldsymbol{a}, \boldsymbol{b}$, their concatenation is denoted by $(\boldsymbol{a} \,|\, \boldsymbol{b})$. The dimension of a code $\mathcal{C}$ is $\mathsf{dim}(\mathcal{C})$ and its minimum distance is $d_{\min}(\mathcal{C})$. The Hamming weight of a vector $\boldsymbol{a}$ is denoted by $w_H(\boldsymbol{a})$, and support by $\mathsf{supp}(\boldsymbol{a})$. We use span to denote the span of a collection of vectors.

Aditya Siddheshwar and Prasad Krishnan are with the Signal Processing and Communications Research Center, International Institute of Information Technology, Hyderabad 500032, India (email: aditya.siddheshwar@research.iiit.ac.in, prasad.krishnan@iiit.ac.in).

Lakshmi Prasad Natarajan is with the Department of Electrical Engineering, Indian Institute of Technology Hyderabad, Sangareddy 502284, India (email: lakshminatarajan@iith.ac.in).

This work was supported by the Qualcomm 6G University Research India Program.

II. Recursive Subproduct Codes

Denote the all-one row vector in $\mathbb{F}_2^n$ by $\mathbb{1}_n$. Let $\mathcal{C}$ be any $[n, k \geq 2, d]$ binary linear code with $\mathbb{1}_n \in \mathcal{C}$. Consider any basis $\{\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{k-1}\}$ for $\mathcal{C}$ such that $\boldsymbol{g}_0 = \mathbb{1}_n$. For every $\boldsymbol{j} = (j_0, \ldots, j_{m-1}) \in [\![k]\!]^m$, define the vector $\boldsymbol{b}_{\boldsymbol{j}} \in \mathbb{F}_2^{n^m}$ as $\boldsymbol{b}_{\boldsymbol{j}} = \boldsymbol{g}_{j_0} \otimes \boldsymbol{g}_{j_1} \otimes \cdots \otimes \boldsymbol{g}_{j_{m-1}}$. Note that $\boldsymbol{b}_{\boldsymbol{j}}$, $\boldsymbol{j} \in [\![k]\!]^m$, are precisely the rows of $G \otimes \cdots \otimes G = G^{\otimes m}$, where $G = [\boldsymbol{g}_0^T, \cdots, \boldsymbol{g}_{k-1}^T]^T$ is a generator matrix of $\mathcal{C}$. Since G, and hence $G^{\otimes m}$, have linearly independent rows, we see that $\{\boldsymbol{b}_{\boldsymbol{j}} : \boldsymbol{j} \in [\![k]\!]^m\}$ is linearly independent. We define

$$\mathcal{C}^{\otimes[r,m]} \triangleq \mathsf{span}\left(\{\boldsymbol{b}_{\boldsymbol{j}} : \boldsymbol{j} \in [\![k]\!]^m, w_H(\boldsymbol{j}) \leq r\}\right)$$

for $0 \leq r \leq m$. Since $G^{\otimes m}$ is a generator matrix of the m-fold product code $\mathcal{C}^{\otimes m}$, it is clear that $\mathcal{C}^{\otimes[r,m]} \subseteq \mathcal{C}^{\otimes m}$. Also, $\dim(\mathcal{C}^{\otimes[r,m]}) = |\{\boldsymbol{j} : w_H(\boldsymbol{j}) \leq r\}| = \sum_{l=0}^{r} (k-1)^l \binom{m}{l}$. It can be shown that the code $\mathcal{C}^{\otimes[r,m]}$ is independent of the choice of $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{k-1}$ as long as $\{\mathbb{1}_n, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{k-1}\}$ is a basis of $\mathcal{C}$, see [1]. We will refer to r as the 'order' of the code.

Remark 1: Consider $\mathcal{C} = \mathbb{F}_2^2$, i.e., $n = k = 2$, with $\boldsymbol{g}_0 = (1,1)$, $\boldsymbol{g}_1 = (0,1)$. Then $\boldsymbol{b}_{\boldsymbol{j}} : \boldsymbol{j} \in \{0,1\}^m$, are the rows of the $2^m \times 2^m$ matrix $G^{\otimes m}$ where $G = [\boldsymbol{g}_0^T, \boldsymbol{g}_1^T]^T$. Note that $w_H(\boldsymbol{b}_{\boldsymbol{j}}) = \prod_{\ell \in [\![m]\!]} w_H(\boldsymbol{g}_{j_\ell}) = 2^{(m - w_H(\boldsymbol{j}))}$. Thus, $\mathcal{C}^{\otimes[r,m]}$ is the span of all the rows of $G^{\otimes m}$ with weight at least $2^{(m-r)}$, which is a well known construction of $\mathrm{RM}(r,m)$, i.e., $\mathcal{C}^{\otimes[r,m]} = \mathrm{RM}(r,m)$ when $\mathcal{C} = \mathbb{F}_2^n$. Further, if $\mathcal{C} = \mathbb{F}_2^n$, $n \geq 3$, then $\mathcal{C}^{\otimes[r,m]}$ is the Dual Berman code [19]–[21] of order r and length n^m, see [1]. □

Lemma 1: (Recursive Plotkin-like structure of $\mathcal{C}^{\otimes[r,m]}$)

$$\mathcal{C}^{\otimes[r,m]} = \Big\{ \sum_{i=0}^{k-1} \boldsymbol{d}_i \otimes \boldsymbol{g}_i : \boldsymbol{d}_0 \in \mathcal{C}^{\otimes[r,m-1]}, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{k-1} \in \mathcal{C}^{\otimes[r-1,m-1]} \Big\}. \quad (1)$$

Further, for each codeword in $\mathcal{C}^{\otimes[r,m]}$, there exists a unique choice of $\boldsymbol{d}_i : i \in [\![k]\!]$ in the decomposition in (1).

Proof Idea: Let $G_{r,m}$ denote the generator matrix of $\mathcal{C}^{\otimes[r,m]}$ composed of rows $\boldsymbol{b}_{\boldsymbol{j}} : \boldsymbol{j} \in [\![k]\!]^m, w_H(\boldsymbol{j}) \leq r$. Using the Kronecker product structure of $\boldsymbol{b}_{\boldsymbol{j}}$ it can be seen that

$$G_{r,m} = \begin{bmatrix} G_{r,m-1} \otimes \mathbb{1}_n \\ G_{r-1,m-1} \otimes G_{sub} \end{bmatrix} \quad (2)$$

up to a reordering of rows, where $G_{sub} = [\boldsymbol{g}_1^T, \ldots, \boldsymbol{g}_{k-1}^T]^T$. Considering the row span of (2) proves the lemma [1]. ■

Remark 2: For RM codes, i.e., $\mathcal{C} = \mathbb{F}_2^2$, $\boldsymbol{g}_0 = (1,1)$, $\boldsymbol{g}_1 = (0,1)$, the decomposition (1) is $\boldsymbol{c} = \boldsymbol{d}_0 \otimes (1,1) + \boldsymbol{d}_1 \otimes (0,1)$. Reordering the symbols in $\boldsymbol{c}$ we obtain $(1,1) \otimes \boldsymbol{d}_0 + (0,1) \otimes \boldsymbol{d}_1 = (\boldsymbol{d}_0, \boldsymbol{d}_0 + \boldsymbol{d}_1)$, the $(u|u+v)$ Plotkin construction. □

We now present the characterization of the minimum distance and the minimum weight codewords of $\mathcal{C}^{\otimes[r,m]}$. The derivation of these results is based on the following observation. We express $\boldsymbol{c} \in \mathcal{C}^{\otimes[r,m]}$ as the concatenation of n^{m-1} vectors of length-n each, $\boldsymbol{c} = (\boldsymbol{c}_1 | \boldsymbol{c}_2 | \ldots | \boldsymbol{c}_{n^{m-1}})$. From (1) we have $\boldsymbol{c} = \sum_{i=0}^{k-1} \boldsymbol{d}_i \otimes \boldsymbol{g}_i$, for some unique codewords $\boldsymbol{d}_0 = (d_{0,1}, \ldots, d_{0,n^{m-1}}) \in \mathcal{C}^{\otimes[r,m-1]}$, and $\boldsymbol{d}_i = (d_{i,1}, \ldots, d_{i,n^{m-1}}) \in \mathcal{C}^{\otimes[r-1,m-1]}$, $i = 1, \ldots, k-1$. Observe that, for any position $t \in \{1, \ldots, n^{m-1}\}$, we have

$$\boldsymbol{c}_t = \sum_{i=0}^{k-1} d_{i,t} \cdot \boldsymbol{g}_i \in \mathcal{C}.$$

Since $\{\boldsymbol{g}_i : i \in [\![k]\!]\}$ is a basis of $\mathcal{C}$, we see that $\boldsymbol{c}_t \in \mathcal{C} \setminus \{\boldsymbol{0}\}$ if at least one of the symbols $d_{i,t} : i \in [\![k]\!]$ is non-zero. Using this observation, and due to the fact $w_H(\boldsymbol{g}_0) = w_H(\mathbb{1}_n) = n$, we immediately obtain

$$w_H(\boldsymbol{c}) = \textstyle\sum_{t=1}^{n^{m-1}} w_H(\boldsymbol{c}_t) \geq |\mathsf{supp}(\boldsymbol{d}_0) \setminus S|\, n \;+\; |S|\, d, \quad (3)$$

where $d = d_{\min}(\mathcal{C})$ and $S = \cup_{i=1}^{k-1} \mathsf{supp}(\boldsymbol{d}_i)$.

Lemma 2: The minimum distance of $\mathcal{C}^{\otimes[r,m]}$ is $d^r n^{m-r}$.

Proof Idea: To prove $d_{\min} \geq d^r n^{m-r}$, apply induction on m using (3) and the known boundary cases $d_{\min}(\mathcal{C}^{\otimes[0,m]}) = n^m$ (repetition code) and $d_{\min}(\mathcal{C}^{\otimes[m,m]}) = d^m$ (product code). For the induction step, prove for cases $S = \emptyset$ and $S \neq \emptyset$. We then prove $d_{\min} \leq d^r n^{m-r}$, again by induction, by identifying codewords in $\mathcal{C}^{\otimes[r,m]}$ of this weight using (1). ■

The following result, whose proof is based on (3), provides a recursive characterization of minimum weight codewords of $\mathcal{C}^{\otimes[r,m]}$. For any linear code $\mathcal{C}$ let $\mathcal{A}_{\min}(\mathcal{C})$ be the set of its minimum weight (non-zero) codewords.

Claim 1: $\mathcal{A}_{\min}\left(\mathcal{C}^{\otimes[r,m]}\right) = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$, where

$$\begin{aligned}
\mathcal{A}_1 &= \{\tilde{\boldsymbol{d}} \otimes \boldsymbol{g} \colon \forall \tilde{\boldsymbol{d}} \in \mathcal{A}_{\min}(\mathcal{C}^{\otimes[r-1,m-1]}), \forall \boldsymbol{g} \in \mathcal{A}_{\min}(\mathcal{C})\},\\
\mathcal{A}_2 &= \{\boldsymbol{d} \otimes \mathbb{1}_n + \tilde{\boldsymbol{d}} \otimes \tilde{\boldsymbol{g}} \colon \forall \tilde{\boldsymbol{d}} \in \mathcal{A}_{\min}(\mathcal{C}^{\otimes[r-1,m-1]}),\\
&\qquad \forall \boldsymbol{d} \in \mathcal{C}^{\otimes[r,m-1]} \text{ s.t } \emptyset \neq \mathsf{supp}(\boldsymbol{d}) \subsetneq \mathsf{supp}(\tilde{\boldsymbol{d}}),\\
&\qquad \forall \tilde{\boldsymbol{g}} \in \mathcal{A}_{\min}(\mathcal{C}) \text{ s.t. } \mathbb{1}_n + \tilde{\boldsymbol{g}} \in \mathcal{A}_{\min}(\mathcal{C})\}, \text{ and}\\
\mathcal{A}_3 &= \{\boldsymbol{d} \otimes \mathbb{1}_n \colon \forall \boldsymbol{d} \in \mathcal{A}_{\min}(\mathcal{C}^{\otimes[r,m-1]})\}.
\end{aligned}$$

Note that $\mathcal{A}_2 \neq \emptyset$ only if there exists a $\tilde{\boldsymbol{g}} \in \mathcal{A}_{\min}(\mathcal{C})$ such that $\mathbb{1}_n + \tilde{\boldsymbol{g}} \in \mathcal{A}_{\min}(\mathcal{C})$, i.e., $n = 2d_{\min}(\mathcal{C}) = 2d$. If $n \neq 2d$, then $\mathcal{A}_{\min}(\mathcal{C}^{\otimes[r,m]}) = \mathcal{A}_1 \cup \mathcal{A}_3$. We now explicitly identify $\mathcal{A}_{\min}(\mathcal{C}^{\otimes[r,m]})$ when $n \neq 2d$.

Lemma 3: $\mathcal{A}_{\min}(\mathcal{C}^{\otimes[r,m]}) = \bigcup_{J \subseteq [\![m]\!] : |J| = r} \mathcal{A}_{\min}(\mathcal{C}^{\otimes[r,m]}, J)$ if $n \neq 2d_{\min}(\mathcal{C})$, where $\mathcal{A}_{\min}(\mathcal{C}^{\otimes[r,m]}, J)$ is defined as

$$\left\{ \bigotimes_{j \in [\![m]\!]} \boldsymbol{h}_j : \boldsymbol{h}_j \in \mathcal{A}_{\min}(\mathcal{C}), \forall j \in J, \text{ and } \boldsymbol{h}_j = \mathbb{1}_n, \forall j \notin J \right\}.$$

Proof Idea: We use $\mathcal{A}_{\min}(\mathcal{C}^{\otimes[r,m]}) = \mathcal{A}_1 \cup \mathcal{A}_3$ inductively together with the known results for the boundary cases, viz., repetition code ($r = 0$) and product code ($r = m$) [26]. ■

In Section IV-C we use a local graph search decoder [22] to improve the performance of the BP decoder for $\mathcal{C}^{\otimes[2,m]}$. The complexity of this search depends on $|\mathcal{A}_{\min}(\mathcal{C}^{\otimes[2,m]})|$. From Lemma 3, if $n \neq 2d$, this equals $\binom{m}{2} |\mathcal{A}_{\min}(\mathcal{C})|^2 = \mathcal{O}\left(\log^2 N\right)$ as a function of the length $N = n^m$. The graph used in this search decoder for $\mathcal{C}^{\otimes[2,m]}$ is a connected graph when $\mathsf{span}(\mathcal{A}_{\min}(\mathcal{C}^{\otimes[2,m]})) = \mathcal{C}^{\otimes[2,m]}$ [22], which is guaranteed by

Claim 2: If $n \neq 2d_{\min}(\mathcal{C})$ and $\mathcal{C} = \mathsf{span}(\mathcal{A}_{\min}(\mathcal{C}))$, then for every $r \in \{0, \ldots, m\}$, $\mathcal{C}^{\otimes[r,m]} = \mathsf{span}(\mathcal{A}_{\min}(\mathcal{C}^{\otimes[r,m]}))$.

III. Fast ML Decoding of First-Order Codes

We consider maximum-likelihood (ML) decoding of the first-order codes $\mathcal{C}^{\otimes[1,m]}$ in a binary-input memoryless channel $W(y|x)$. The naive ML decoder for $\mathcal{C}^{\otimes[1,m]}$ has complexity order $N\,2^{\dim(\mathcal{C}^{\otimes[1,m]})} = N\,2^{1+m(k-1)} = 2N^{1+\alpha}$, where $N = n^m$ is the block length of the code and $\alpha = (k-1)/\log_2 n$. In contrast, using ideas similar to the fast ML decoder for $\mathrm{RM}(1,m)$ [16], [17], we use the recursive structure of the code to implement the ML decoder with complexity $\mathcal{O}(\max\{N, N^\alpha\})$ when $\alpha \neq 1$ and complexity $\mathcal{O}(N \log N)$ if $\alpha = 1$. We also propose an efficient soft-output max-log-MAP decoder with the same complexity order as the fast ML decoder.

Let $\boldsymbol{y} = (y_1, \ldots, y_N)$ be the channel output and $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_N)$ denote the vector of channel log likelihood ratios (LLRs), where $\ell_i = \log_e\left(W(y_i|0)/W(y_i|1)\right)$. For any $\boldsymbol{c} = (c_1, \ldots, c_N) \in \mathcal{C}^{\otimes[1,m]}$, let us denote the bipolar representation of the bit c_i as $c_i^b = (-1)^{c_i}$. Similarly, for a codeword $\boldsymbol{c}$ define its bipolar representation $\boldsymbol{c}^b = \left(c_1^b, \ldots, c_N^b\right) \in \{\pm 1\}^N$. Note that the ML decoder chooses the codeword whose bipolar representation $\boldsymbol{c}^b$ has the largest correlation $\langle \boldsymbol{c}^b, \boldsymbol{\ell}\rangle = \sum_i c_i^b \ell_i$ with the LLR vector $\boldsymbol{\ell}$.

A. Efficient ML Decoding

Let $\mathcal{C}_{sub} = \mathsf{span}(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{k-1})$, a subcode of $\mathcal{C}$. Using Lemma 1 and the fact that $\mathcal{C}^{\otimes[0,m-1]}$ is the repetition code of length n^{m-1}, we observe that $\mathcal{C}^{\otimes[1,m]}$ equals

$$\left\{\boldsymbol{d} \otimes \mathbb{1}_n + \mathbb{1}_{n^{(m-1)}} \otimes \boldsymbol{a} : \boldsymbol{d} \in \mathcal{C}^{\otimes[1,m-1]}, \boldsymbol{a} \in \mathcal{C}_{sub}\right\}, \quad (4)$$

where $\mathbb{1}_{n^{(m-1)}}$ is the all-one vector of length n^{m-1}. For two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ of same length let $\boldsymbol{u} \odot \boldsymbol{v}$ denote the componentwise product of the entries of $\boldsymbol{u}$ and $\boldsymbol{v}$. From (4), we deduce that for all $\boldsymbol{c} \in \mathcal{C}^{\otimes[1,m]}$, $\boldsymbol{c}^b = \left(\boldsymbol{d}^b \otimes \mathbb{1}_n\right) \odot \left(\mathbb{1}_{n^{(m-1)}} \otimes \boldsymbol{a}^b\right)$, where $\boldsymbol{d} \in \mathcal{C}^{\otimes[1,m-1]}$, $\boldsymbol{a} \in \mathcal{C}_{sub}$. Using $\boldsymbol{d} = (d_1, \ldots, d_{n^{m-1}})$ and $\boldsymbol{a} = (a_1, \ldots, a_n)$ to denote the components of $\boldsymbol{d}$ and $\boldsymbol{a}$,

$$\boldsymbol{c}^b = \boldsymbol{d}^b \otimes \boldsymbol{a}^b = \left(d_1^b \boldsymbol{a}^b, d_2^b \boldsymbol{a}^b, \ldots, d_{n^{(m-1)}}^b \boldsymbol{a}^b\right). \quad (5)$$

Let us similarly split the n^m-length LLR vector $\boldsymbol{\ell}$ into n^{m-1} subvectors of length n each, $\boldsymbol{\ell} = (\boldsymbol{\ell}_1 \,|\, \cdots \,|\, \boldsymbol{\ell}_{n^{(m-1)}})$, where $\boldsymbol{\ell}_j = (\ell_{j,1}, \ldots, \ell_{j,n})$ for $j = 1, \ldots, n^{(m-1)}$. Then

$$\langle \boldsymbol{c}^b, \boldsymbol{\ell}\rangle = \sum_{j=1}^{n^{m-1}} \sum_{i=1}^{n} d_j^b a_i^b \ell_{j,i} = \sum_{j=1}^{n^{m-1}} d_j^b \sum_{i=1}^{n} a_i^b \ell_{j,i} = \langle \boldsymbol{d}^b, \boldsymbol{\ell}(\boldsymbol{a})\rangle,$$

where $\langle \boldsymbol{d}^b, \boldsymbol{\ell}(\boldsymbol{a})\rangle$ is the correlation between the two $n^{(m-1)}$-length vectors $\boldsymbol{\ell}(\boldsymbol{a}) \triangleq \left(\sum_{i=1}^n a_i^b \ell_{1,i}, \ldots, \sum_{i=1}^n a_i^b \ell_{n^{m-1},i}\right)$ and $\boldsymbol{d}^b$. While $\boldsymbol{c} \in \mathcal{C}^{\otimes[1,m]}$, we note that $\boldsymbol{d} \in \mathcal{C}^{\otimes[1,m-1]}$. This relation allows us to implement the ML decoder for $\mathcal{C}^{\otimes[1,m]}$ recursively by calling the ML decoder for $\mathcal{C}^{\otimes[1,m-1]}$. For each of the $2^{(k-1)}$ choices of $\boldsymbol{a} \in \mathcal{C}_{sub}$ we use the ML decoder for $\mathcal{C}^{\otimes[1,m-1]}$ to find the codeword $\boldsymbol{d}^b$ that maximizes $\langle \boldsymbol{d}^b, \boldsymbol{\ell}(\boldsymbol{a})\rangle$. Then, among all these $2^{(k-1)}$ choices of $(\boldsymbol{d}^b, \boldsymbol{a})$ we pick the one with the maximum correlation. For $m = 1$ we use the brute-force decoder with complexity $n2^k$. This recursive decoder has complexity order $\max\{N, N^\alpha\}$ if $\alpha \neq 1$, and complexity order $N \log N$ if $\alpha = 1$, see [1].

B. Efficient Max-Log-MAP Decoding

An efficient implementation of the optimal MAP (maximum a posteriori probability) decoder for $\mathrm{RM}(1,m)$ has been described in [17], which can be extended to $\mathcal{C}^{\otimes[1,m]}$. However, this decoder is not numerically stable since it operates in the probability domain (not log domain), and it uses expensive functions log and exp. An *approximation* to the max-log-MAP decoder for $\mathrm{RM}(1,m)$ was proposed in [6] which operates completely in the log domain. In comparison we provide a low-complexity recursive algorithm for *exact* max-log-MAP decoding for $\mathcal{C}^{\otimes[1,m]}$ (which applies to $\mathrm{RM}(1,m)$ also).

Note that (5) partitions $\boldsymbol{c}^b$ into n^{m-1} subvectors each of length n. The i^{th} symbol in the j^{th} subvector is $d_j^b a_i^b$, where $i = 1, \ldots, n$ and $j = 1, \ldots, n^{m-1}$. We have thus indexed the code bits in $\boldsymbol{c}$ using (j,i). The max-log-MAP decoder outputs

$$L_{j,i} = \log_e\Big(\max_{\boldsymbol{c}: c_{j,i}^b = +1} \mathbb{P}[\boldsymbol{c}|\boldsymbol{y}]\Big) - \log_e\Big(\max_{\boldsymbol{c}: c_{j,i}^b = -1} \mathbb{P}[\boldsymbol{c}|\boldsymbol{y}]\Big)$$

for all (j,i). Here, $L_{j,i}$ is the max-log approximation of the true log APP (a posteriori probability) ratio $\log_e\left(\mathbb{P}[c_{j,i}^b = +1|\boldsymbol{y}] \,/\, \mathbb{P}[c_{j,i}^b = -1|\boldsymbol{y}]\right)$. Following a procedure similar to [27, equation (63)] it is rather straightforward to show that $L_{j,i} = \left(L_{j,i}^{(+1)} - L_{j,i}^{(-1)}\right)/2$, where

$$L_{j,i}^{(+1)} = \max_{\boldsymbol{c}: c_{j,i}^b = +1} \langle \boldsymbol{c}^b, \boldsymbol{\ell}\rangle \quad \text{and} \quad L_{j,i}^{(-1)} = \max_{\boldsymbol{c}: c_{j,i}^b = -1} \langle \boldsymbol{c}^b, \boldsymbol{\ell}\rangle.$$

Using (5), and the facts $\langle \boldsymbol{c}^b, \boldsymbol{\ell}\rangle = \langle \boldsymbol{d}^b, \boldsymbol{\ell}(\boldsymbol{a})\rangle$, $c_{j,i}^b = d_j^b\, a_i^b$, it is easy to observe that

$$L_{j,i}^{(s)} = \max_{\boldsymbol{a} \in \mathcal{C}_{sub}} \left\{ \max_{\boldsymbol{d} \in \mathcal{C}^{\otimes[1,m-1]}: d_j^b = s a_i^b} \langle \boldsymbol{d}^b, \boldsymbol{\ell}(\boldsymbol{a})\rangle \right\} \text{ for } s \in \{\pm 1\}.$$

Observe that the inner maximization is a computation that would be used to perform max-log-MAP decoding of the code $\mathcal{C}^{\otimes[1,m-1]}$ where the channel LLRs are $\boldsymbol{\ell}(\boldsymbol{a})$. For $s \in \{\pm 1\}$, by using the notation $L_j^{(s)}(\boldsymbol{a}) = \max\left\{\langle \boldsymbol{d}^b, \boldsymbol{\ell}(\boldsymbol{a})\rangle : \boldsymbol{d} \in \mathcal{C}^{\otimes[1,m-1]}, d_j^b = s\right\}$, we see that

$$L_{j,i}^{(+1)} = \max_{\boldsymbol{a} \in \mathcal{C}_{sub}} L_j^{(a_i^b)}(\boldsymbol{a}), \; L_{j,i}^{(-1)} = \max_{\boldsymbol{a} \in \mathcal{C}_{sub}} L_j^{(-a_i^b)}(\boldsymbol{a}). \quad (6)$$

In our recursive algorithm, we use a procedure for the code $\mathcal{C}^{\otimes[1,m-1]}$ to compute $L_j^{(s)}(\boldsymbol{a})$, $s = \pm 1$, for each $\boldsymbol{a} \in \mathcal{C}_{sub}$. We then use these values in (6) to compute the $L_{j,i}^{(+1)}$ and $L_{j,i}^{(-1)}$ for $\mathcal{C}^{\otimes[1,m]}$, and then finally, $L_{j,i}$ as half their difference. For $m = 1$, we use a brute-force approach to compute $L_{j,i}^{(+1)}, L_{j,i}^{(-1)}$. This max-log-MAP decoder has the same complexity order as the recursive ML decoder in Section III-A, see [1].

IV. Decoding Second-Order Codes

We introduce a notation for indexing the code bits using $[\![n]\!]^m = \{0, \ldots, n-1\}^m$ as well as for puncturing codewords, and then we present the projection operation. Borrowing ideas from the literature on decoding RM codes [14], [15], [28], [29] we propose a BP decoder for $\mathcal{C}^{\otimes[2,m]}$ that exploits this projection. We then use a local graph search algorithm [22] to improve the performance of the BP decoder.

A. Projection Operation

To create a new indexing we simply replace the natural index $i \in \{1,\dots,n^m\}$ with the coefficients $i_0,\dots,i_{m-1}$ of the base-n expansion of $i-1=\sum_{l=0}^{m-1} i_l n^{m-1-l}$, where $\boldsymbol{i}=(i_0,\dots,i_{m-1}) \in [\![n]\!]^m$. Recall that the $\boldsymbol{j}^{\text{th}}$ basis vector is $\boldsymbol{b_j}=\boldsymbol{g}_{j_0}\otimes\cdots\otimes\boldsymbol{g}_{j_{m-1}}$, $\boldsymbol{j}\in[\![k]\!]^m$, $w_H(\boldsymbol{j})\le r$. Representing the entries of $\boldsymbol{g}_{j_\ell}$ as $(g_{j_\ell,0},\dots,g_{j_\ell,n-1})$ we observe that the $\boldsymbol{i}^{\text{th}}$ entry of $\boldsymbol{b_j}$ is $b_{\boldsymbol{j},\boldsymbol{i}}=g_{j_0,i_0}\times\cdots\times g_{j_{m-1},i_{m-1}}=\prod_{\ell\in[\![m]\!]} g_{j_\ell,i_\ell}$. If $\boldsymbol{c}\in\mathcal{C}^{\otimes[r,m]}$ then there exist coefficients $a_{\boldsymbol{j}}\in\mathbb{F}_2$ such that $\boldsymbol{c}=\sum_{\boldsymbol{j}:w_H(\boldsymbol{j})\le r} a_{\boldsymbol{j}}\boldsymbol{b_j}$. Thus, for each $\boldsymbol{i}\in[\![n]\!]^m$, the $\boldsymbol{i}^{\text{th}}$ entry of $\boldsymbol{c}$ is

$$c_{\boldsymbol{i}}=\sum_{\boldsymbol{j}\in[\![k]\!]^m:w_H(\boldsymbol{j})\le r} a_{\boldsymbol{j}}b_{\boldsymbol{j},\boldsymbol{i}}=\sum_{\boldsymbol{j}\in[\![k]\!]^m:w_H(\boldsymbol{j})\le r} a_{\boldsymbol{j}}\prod_{\ell\in[\![m]\!]} g_{j_\ell,i_\ell}. \quad (7)$$

We intend to puncture a codeword onto the collection of indices $\boldsymbol{i}$ where some of the coordinates of $\boldsymbol{i}$ are frozen. Let $\mathcal{F}\subset[\![m]\!]$, with $|\mathcal{F}|=f$, and $\boldsymbol{i}_{\mathcal{F}}=(i_\ell:\ell\in\mathcal{F})$ be the corresponding subvector of $\boldsymbol{i}$. Let $\boldsymbol{u}\in[\![n]\!]^f$ denote the frozen values of $\boldsymbol{i}_{\mathcal{F}}$. Then $\mathcal{H}=\{\boldsymbol{i}\in[\![n]\!]^m:\boldsymbol{i}_{\mathcal{F}}=\boldsymbol{u}\}$ is the collection of indices where $\boldsymbol{i}_{\mathcal{F}}=\boldsymbol{u}$. We denote the vector obtained by puncturing a codeword $\boldsymbol{c}$ by only retaining the indices in $\mathcal{H}$ as $\mathcal{P}_{\mathcal{H}}(\boldsymbol{c})=(c_{\boldsymbol{i}}:\boldsymbol{i}\in\mathcal{H})$. The length of $\mathcal{P}_{\mathcal{H}}(\boldsymbol{c})$ is $|\mathcal{H}|=n^{(m-f)}$. This punctured vector can be indexed using $[\![n]\!]^{(m-f)}$ as follows. For any $\boldsymbol{i}'=(i'_0,\dots,i'_{m-f-1})\in[\![n]\!]^{(m-f)}$, the $\boldsymbol{i}'^{\text{th}}$ entry of $\mathcal{P}_{\mathcal{H}}(\boldsymbol{c})$ is equal to the $\boldsymbol{i}^{\text{th}}$ entry of $\boldsymbol{c}$ where $\boldsymbol{i}$ is defined as $\boldsymbol{i}_{\mathcal{F}}=\boldsymbol{u}$ and $\boldsymbol{i}_{[\![m]\!]\setminus\mathcal{F}}=\boldsymbol{i}'$.

In order to apply projection on $\mathcal{C}^{\otimes[r,m]}$ we choose $\mathcal{F},\boldsymbol{u},\tilde{\boldsymbol{u}}$ as follows, $\mathcal{F}\subset[\![m]\!]$, $|\mathcal{F}|=f$, $\boldsymbol{u},\tilde{\boldsymbol{u}}\in[\![n]\!]^{(m-f)}$ with $\boldsymbol{u}\neq\tilde{\boldsymbol{u}}$. For any $f=1,\dots,m-r+1$, there are $\binom{m}{f}\binom{n^f}{2}$ choices of $\mathcal{F},\boldsymbol{u},\tilde{\boldsymbol{u}}$, and each such choice provides a projection operation. Define $\mathcal{H}=\{\boldsymbol{i}\in[\![n]\!]^m:\boldsymbol{i}_{\mathcal{F}}=\boldsymbol{u}\}$ and $\widetilde{\mathcal{H}}=\{\boldsymbol{i}\in[\![n]\!]^m:\boldsymbol{i}_{\mathcal{F}}=\tilde{\boldsymbol{u}}\}$ for puncturing a codeword $\boldsymbol{c}$ to the indices $\mathcal{H}$ and $\widetilde{\mathcal{H}}$.

Lemma 4: (Projection) For every $\boldsymbol{c}\in\mathcal{C}^{\otimes[r,m]}$,

$$\mathcal{P}_{\mathcal{H}}(\boldsymbol{c})+\mathcal{P}_{\widetilde{\mathcal{H}}}(\boldsymbol{c})\in\mathcal{C}^{\otimes[r-1,m-f]}.$$

Proof Idea: For $\boldsymbol{i}'\in[\![n]\!]^{(m-f)}$ we use the expansion (7) to express the $\boldsymbol{i}'^{\text{th}}$ entry of $\mathcal{P}_{\mathcal{H}}(\boldsymbol{c})+\mathcal{P}_{\widetilde{\mathcal{H}}}(\boldsymbol{c})$ as a sum indexed by $\boldsymbol{j}\in[\![k]\!]^m$, $w_H(\boldsymbol{j})\le r$. We then observe that if $\boldsymbol{j}$ is such that $j_\ell=0$ for all $\ell\in\mathcal{F}$ then the corresponding term in the sum is zero (using the fact $\boldsymbol{g}_{j_\ell}=\mathbb{1}_n$ if $j_\ell=0$). The remaining terms correspond to the choices of $\boldsymbol{j}$ such that $w_H((j_\ell:\ell\notin\mathcal{F}))\le w_H(\boldsymbol{j})-1\le r-1$. This sum can be reduced to a form similar to (7) for the code $\mathcal{C}^{\otimes[r-1,m-f]}$ (instead of $\mathcal{C}^{\otimes[r,m]}$). See [1] for full proof. ∎

RM codes enjoy a richer set of projection operations than Lemma 4 which include the projections described in Lemma 4 as a strict subset, see [14, Lemma 1].

B. Belief Propagation Decoding of Second-Order Codes

We use an approach identical to the BP decoder proposed in [15] for RM codes to decode $\mathcal{C}^{\otimes[2,m]}$. Our factor graph uses two types of generalized check nodes. The first arise from the projection operations from Lemma 4 using $f=1$. The second arise from the condition that every $\boldsymbol{c}\in\mathcal{C}^{\otimes[2,m]}$ belongs to the product code $\mathcal{C}^{\otimes m}$ (these are useful only when $\mathcal{C}$ is non-trivial, i.e., $k<n$). These constraints can be stated as follows: for any $\mathcal{F}\subset[\![m]\!]$ with $|\mathcal{F}|=m-1$ and any $\boldsymbol{u}\in[\![n]\!]^{m-1}$, we have $\mathcal{P}_{\mathcal{H}}(\boldsymbol{c})\in\mathcal{C}$, where $\mathcal{H}=\{\boldsymbol{i}:\boldsymbol{i}_{\mathcal{F}}=\boldsymbol{u}\}$. The factor graph contains the following nodes (see Fig. 1):

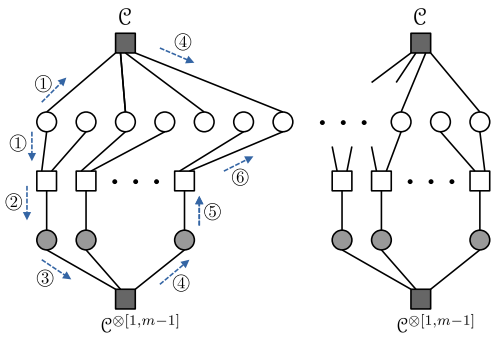

Fig. 1. Factor graph for BP decoding of $\mathcal{C}^{\otimes[2,m]}$. Numbers next to the dashed arrows represent the sequence of messages in each iteration. Empty circles are V, empty squares are C, filled circles are $\mathsf{V_h}$, filled squares are $\mathsf{C_g}$.

(i) The set V of n^m variable nodes denoting the code bits.
(ii) Set C of check nodes that denote the addition of code bits in the projection operation. Each such node is connected to two variable nodes and one hidden variable node.
(iii) Hidden variable nodes $\mathsf{V_h}$ denoting the (projected) code bits of $\mathcal{C}^{\otimes[1,m-1]}$.
(iv) Two types of generalized check nodes $\mathsf{C_g}$: nodes of type $\mathcal{C}^{\otimes[1,m-1]}$ are connected to n^{m-1} hidden variable nodes, and nodes of type $\mathcal{C}$ are connected to n variable nodes.

We use a weighted variable node update rule at nodes $v\in\mathsf{V}$

$$\lambda_{v\to c}^{(t)}=\ell_v+\gamma\sum_{c'\in\partial v\cap\mathsf{C}:c'\neq c}\lambda_{c'\to v}^{(t-1)}+\gamma_{\mathsf{g}}\sum_{c'\in\partial v\cap\mathsf{C_g}:c'\neq c}\lambda_{c'\to v}^{(t-1)}$$

for all $c\in\partial v$, where $\lambda_{v\to c}$, $\lambda_{c\to v}$ are messages from and to v, respectively, superscript is the iteration index, ∂ represents the set of neighbors of a node, $\gamma,\gamma_{\mathsf{g}}>0$ represent weights, and ℓ_v is the channel LLR of code bit v. Generalized check nodes perform max-log-MAP decoding to output the extrinsic LLRs (fast decoder for $\mathcal{C}^{\otimes[1,m-1]}$, brute-force for $\mathcal{C}$). We use the standard BP rule at other nodes. We run up to $T_{\max}$ iterations or till convergence to a codeword, whichever is earlier. The complexity order of each iteration is $N\log^2 N$ for $\alpha=1$, and $\max\{N,N^\alpha\}\log N$ otherwise [1].

C. Improving the Performance via Local Graph Search

Let $\mathcal{G}$ be a graph whose vertex set is $\mathcal{C}^{\otimes[2,m]}$. Two codewords are neighbors if the distance between them is $d_{\min}(\mathcal{C}^{\otimes[2,m]})$. As proposed in [22], we trace a path in $\mathcal{G}$ of length P_{LGS} as follows: for $p=1,\dots,P_{\mathsf{LGS}}$ we choose the p^{th} codeword in the path $\boldsymbol{c}^{(p)}$ as the codeword with the largest likelihood among all neighbors of $\boldsymbol{c}^{(p-1)}$ that do not already lie on the path traced so far. If no such neighbor is found, we terminate. We set $\boldsymbol{c}^{(0)}$ as the BP decoder's output. The output of the algorithm is the most likely codeword among all the nodes in the traced path. If $n\neq 2d_{\min}(\mathcal{C})$, the degree of each node in $\mathcal{G}$ is small, in the order of

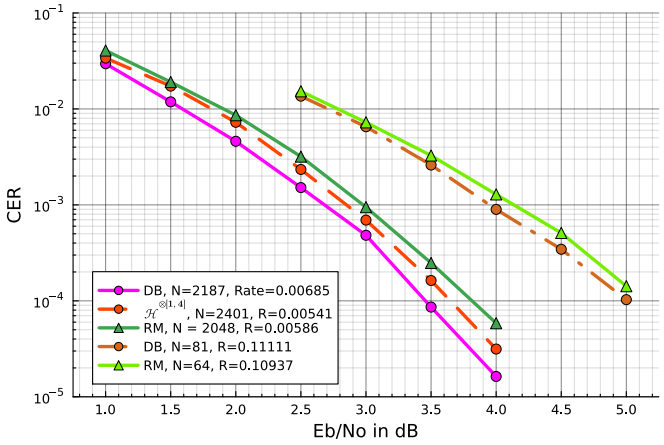

Fig. 2. Performance of first-order codes under ML decoding. 'DB' denotes Dual Berman codes, and $\mathcal{H}$ is the $[7, 4, 3]$ Hamming code.

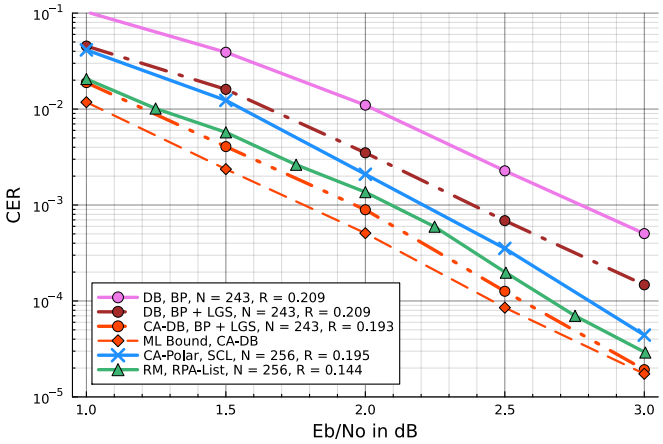

Fig. 3. Performance of codes with block lengths close to 250. Here, 'DB' denotes the Dual Berman code, and 'CA-DB' is its CRC-aided version.

$\log^2 N$. In this case, the complexity order of this algorithm is $P_{\mathsf{LGS}} \log^2 N \max\left\{N, \log^2 N \log P_{\mathsf{LGS}}\right\}$, see [1].

In general, we use this algorithm only if the output $\boldsymbol{c}_{\mathsf{BP}}$ of the BP decoder belongs to $\mathcal{C}^{\otimes[2,m]}$. For Dual Berman codes ($\mathcal{C} = \mathbb{F}_2^n$), we use $\boldsymbol{c}^{(0)} = \mathcal{D}(\boldsymbol{c}_{\mathsf{BP}})$ for any $\boldsymbol{c}_{\mathsf{BP}} \in \mathbb{F}_2^{n^m}$ where $\mathcal{D} : \mathbb{F}_2^{n^m} \to \mathcal{C}^{\otimes[2,m]}$ is a low-complexity decoder from [21].

V. Simulation Results

We present the codeword error rate (CER) of a few recursive subproduct codes $\mathcal{C}^{\otimes[r,m]}$ with $r = 1, 2$ in the binary-input AWGN channel benchmarked against $\mathrm{RM}(1, m)$ (fast ML decoding), $\mathrm{RM}(2, m)$ (RPA-list decoding [14], list size 16, with near-ML performance) and CRC-aided Polar (CA-Polar) codes (successive cancellation list (SCL) decoding, list size 32). We used the python library [30], [31] (8-bit CRC) and the Matlab library [32] (5G NR uplink codes with 11-bit CRC) for CA-Polar codes. In all presented results we observe that the CER of the recursive subproduct codes are either better than or within 0.5 dB of the performances of RM and CA-Polar codes.

Fig. 2 compares first-order codes under ML decoding. Here 'DB' denotes the two Dual Berman codes, viz., $\mathcal{C}^{\otimes[1,m]}$ with $\mathcal{C} = \mathbb{F}_2^3$ and $m = 4, 7$. The symbol $\mathcal{H}$ (for the code $\mathcal{H}^{\otimes[1,4]}$) denotes the $[7, 4, 3]$ Hamming code. The superior performance of the subproduct codes over RM codes could be because of the small number of minimum weight codewords ($\mathcal{O}(\log N)$ for these subproduct codes, see Lemma 3, versus $\mathcal{O}(N)$ for $\mathrm{RM}(1, m)$).

Fig. 3 compares the $[243, 51, 27]$ Dual Berman code $\mathcal{C}^{\otimes[2,5]}$, where $\mathcal{C} = \mathbb{F}_2^3$, with $\mathrm{RM}(2, 8)$ and the CA-Polar code simulated using [30]. The rightmost curve is $\mathcal{C}^{\otimes[2,m]}$ with BP decoding ($T_{\max} = 5, \gamma = 0.12$; there are no generalized check nodes of type $\mathcal{C}$ in the factor graph). The improvement obtained via the local graph search ('BP+LGS') is also shown ($P_{\mathsf{LGS}} = 512$). We strengthen this code using a 4-bit CRC (polynomial x^4+x+1) at the cost of a slight reduction in rate. We use the same 'BP+LGS' decoder as before with two minor modifications while determining the graph search output: we consider only those codewords in the traced path that satisfy the CRC, and we use $P_{\mathsf{LGS}} = 2^{13}$. We also show the ML lower bound [14], [33] of this CRC-aided Dual Berman (CA-DB) code. Observe that the CA-DB code has a lower CER than $\mathrm{RM}(2, 8)$ (however, please note that these two codes have different rates).

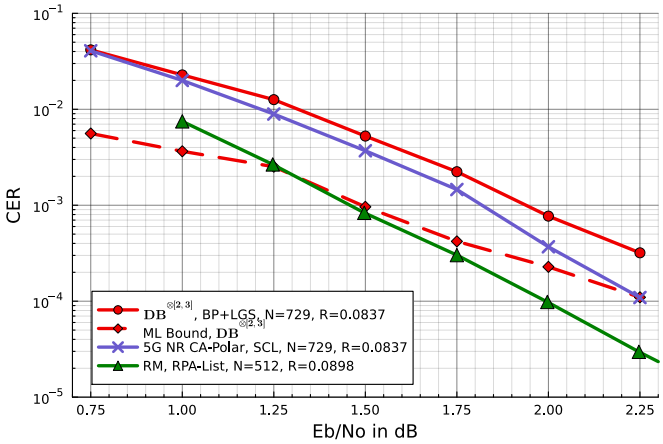

Fig. 4. Codes with rates close to 0.085. Here DB denotes the $[9, 5, 3]$ Dual Berman code.

Fig. 4 compares codes with rates close to 0.085. The second-order RM code of length 512 and the 5G NR CA-Polar code of length 729 and dimension 61 (this length is attained via rate-matching in 5G NR) are used as benchmarks. The recursive subproduct code presented in this figure is the $[729, 61, 81]$ code $\mathcal{C}^{\otimes[2,3]}$, where $\mathcal{C} = \left(\mathbb{F}_2^3\right)^{\otimes[1,2]}$ is the $[9, 5, 3]$ Dual Berman code. BP decoding (using $\gamma = 0.0175$, $\gamma_{\mathsf{g}} = 0.3$, $T_{\max} = 200$) with local graph search decoding ($P_{\mathsf{LGS}} = 2^{13}$) was used. Its performance is about 0.5 dB away from that of the RM code when the CER is 10^{-3}. A lower bound on the ML decoding performance of $\mathcal{C}^{\otimes[2,3]}$ is also shown.

Please see [1] for a simulation result showing the CER of $\mathcal{H}^{\otimes[2,3]}$, $\mathcal{H} = [7, 4, 3]$, against the 5G NR CA-Polar code.

REFERENCES

[1] A. Siddheshwar, L. P. Natarajan, and P. Krishnan, "Recursive subproduct codes with Reed-Muller-like structure," *CoRR*, 2024. [Online]. Available: https://arxiv.org/abs/2401.15678

[2] P. Elias, "Error-free coding," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 29–37, 1954.

[3] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[4] J. Justesen, "Performance of product codes and related structures with iterated decoding," *IEEE Transactions on Communications*, vol. 59, no. 2, pp. 407–415, 2011.

[5] M. C. Coşkun, T. Jerkovits, and G. Liva, "Successive cancellation list decoding of product codes with Reed-Muller component codes," *IEEE Commun. Lett.*, vol. 23, no. 11, pp. 1972–1976, 2019.

[6] M. V. Jamali, M. Fereydounian, H. Mahdavifar, and H. Hassani, "Low-complexity decoding of a class of Reed-Muller subcodes for low-capacity channels," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 123–128.

[7] M. C. Coşkun, G. Liva, A. Graell i Amat, M. Lentmaier, and H. D. Pfister, "Successive cancellation decoding of single parity-check product codes: Analysis and improved decoding," *IEEE Trans. Inf. Theory*, vol. 69, no. 2, pp. 823–841, 2023.

[8] T. Mittelholzer, T. Parnell, N. Papandreou, and H. Pozidis, "Symmetry-based subproduct codes," in *2015 IEEE International Symposium on Information Theory (ISIT)*, 2015, pp. 251–255.

[9] H. D. Pfister, S. K. Emmadi, and K. Narayanan, "Symmetric product codes," in *2015 Information Theory and Applications Workshop (ITA)*, 2015, pp. 282–290.

[10] A. J. Feltstrom, D. Truhachev, M. Lentmaier, and K. S. Zigangirov, "Braided block codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2640–2658, 2009.

[11] B. P. Smith, A. Farhood, A. Hunt, F. R. Kschischang, and J. Lodge, "Staircase codes: FEC for 100 Gb/s OTN," *Journal of Lightwave Technology*, vol. 30, no. 1, pp. 110–117, 2012.

[12] D. E. Muller, "Application of boolean algebra to switching circuit design and to error detection," *Transactions of the I.R.E. Professional Group on Electronic Computers*, vol. EC-3, no. 3, pp. 6–12, 1954.

[13] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 38–49, 1954.

[14] M. Ye and E. Abbe, "Recursive projection-aggregation decoding of Reed-Muller codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 8, pp. 4948–4965, 2020.

[15] M. Lian, C. Häger, and H. D. Pfister, "Decoding Reed–Muller codes using redundant code constraints," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 42–47.

[16] Y. Be'ery and J. Snyders, "Optimal soft decision block decoders based on fast Hadamard transform," *IEEE Trans. Inf. Theory*, vol. 32, no. 3, pp. 355–364, 1986.

[17] A. Ashikhmin and S. Litsyn, "Simple MAP decoding of first-order Reed-Muller and Hamming codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1812–1818, 2004.

[18] B. Zhang and Q. Huang, "Derivative descendants and ascendants of binary cyclic codes, and derivative decoding," in *2022 IEEE Globecom Workshops (GC Wkshps)*, 2022, pp. 535–540.

[19] T. Blackmore and G. Norton, "On a family of abelian codes and their state complexities," *IEEE Trans. Inf. Theory*, vol. 47, no. 1, pp. 355–361, 2001.

[20] S. Berman, "Semisimple cyclic and Abelian codes. II," *Cybernetics*, vol. 3, no. 3, pp. 17–23, 1967.

[21] L. P. Natarajan and P. Krishnan, "Berman codes: A generalization of Reed–Muller codes that achieve BEC capacity," *IEEE Trans. Inf. Theory*, vol. 69, no. 11, pp. 6956–6980, 2023.

[22] M. Kamenev, "On decoding of Reed-Muller codes using a local graph search," *IEEE Trans. Commun.*, vol. 70, no. 2, pp. 739–748, 2022.

[23] M. Fereydounian, H. Hassani, M. V. Jamali, and H. Mahdavifar, "Channel coding at low capacity," *IEEE Journal on Selected Areas in Information Theory*, vol. 4, pp. 351–362, 2023.

[24] R. Freij-Hollanti, O. W. Gnilke, C. Hollanti, A.-L. Horlemann-Trautmann, D. Karpuk, and I. Kubjas, "t-Private information retrieval schemes using transitive codes," *IEEE Trans. Inf. Theory*, vol. 65, no. 4, pp. 2107–2118, 2019.

[25] S. Kale, K. Agarwal, and P. Krishnan, "t-PIR schemes with flexible parameters via star products of Berman codes," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 1348–1353.

[26] R. Miller, "Number of minimum-weight code words in a product code," *Electronics Letters*, vol. 14, pp. 642–643(1), September 1978. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/el_19780431

[27] 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, 1996.

[28] V. M. Sidel'nikov and A. Pershakov, "Decoding of Reed-Muller codes with a large number of errors," *Problemy Peredachi Informatsii*, vol. 28, no. 3, pp. 80–94, 1992.

[29] B. Sakkour, "Decoding of second order Reed-Muller codes with a large number of errors," in *IEEE Information Theory Workshop*, 2005.

[30] M. Rowshan, A. Burg, and E. Viterbo, "List decoder for Polar codes, CRC-Polar codes, and PAC codes," https://github.com/mohammad-rowshan/List-Decoder-for-Polar-Codes-and-PAC-Codes, 2022.

[31] ——, "Polarization-adjusted convolutional (PAC) codes: Sequential decoding vs list decoding," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1434–1447, 2021.

[32] The MathWorks Inc., "5G New Radio Polar Coding ," https://in.mathworks.com/help/5g/gs/polar-coding.html, Natick, Massachusetts, United States.

[33] I. Dumer and K. Shabunov, "Soft-decision decoding of Reed-Muller codes: recursive lists," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1260–1266, 2006.