Reduced-Complexity Projection-Aggregation List Decoder for Reed-Muller Codes

Jiajie Li, Huayi Zhou, Marwan Jalaleddine and Warren J. Gross

***Abstract*—Projection-aggregation decoders have been used in conjunction with a list structure to achieve near maximum-likelihood decoding for short-length and low-rate Reed-Muller (RM) codes but suffer from high computational complexity. We reduce the worst-case computational complexity of projection-aggregation (PA) decoders by more than 50% using a scheduling scheme compared to PA decoders without the scheduling scheme, and propose a redesigned syndrome check pattern to avoid repeated syndrome computations in the decoder. A latency model based on the existing hardware architecture is proposed. Input distribution aware (IDA) decoding is adopted as a pre-possessing tool, and the average list size when using IDA decoding is analytically derived under additive white Gaussian noise and uncorrelated normalized Rayleigh fading channels. Using IDA, the average list size is reduced by 30% with less than 0.1 dB loss. The proposed list decoders require a smaller computational complexity than the state-of-the-art iterative decoder, automorphism ensemble decoding with the belief propagation constituent decoder (AED-BP) for decoding RM$(7, 3)$ and RM$(8, 3)$ codes. Based on the developed latency models, the PA list decoder has a smaller latency than the AED-BP and the successive cancellation list decoder to reach near maximum-likelihood decoding performance.**

***Index Terms*—IDA decoding, list decoders, Reed-Muller codes, projection-aggregation decoders**

I. Introduction

Reed-Muller (RM) codes [1] have recently attracted a lot of interest in the research community due to their close structural similarities [2] with capacity-achieving polar codes [3] that have been adopted into the 5G communication standard [4]. Similar to polar codes, recent research works show that RM codes achieve the capacity of various channels, such as the binary erasure channel [5], the binary symmetric channel [6], [7], and the binary memoryless symmetric channel [8], [9]. Additionally, RM codes are proven to have a polarization effect, a property that contributes to the capacity-achieving ability of polar codes [10].

The first decoder for RM codes is Reed's majority-vote decoder [11] that can correct error patterns with a weight less than half of the minimum distance. RM codes have excellent maximum likelihood (ML) decoding performance because of their large minimum distance [12]. Achieving this ML decoding performance for first-order RM codes is possible with the use of the low computational complexity fast Hadamard transform (FHT) decoder [13], [14]. For higher-order RM codes, Dumer's list decoder, which is structurally similar to successive cancellation (SC) list (SCL) decoding [15], can be used to approach the ML performance at the cost of a large enough list size [16]. There are other sequential decoders that can achieve the ML decoding performance with an average complexity equal to one SC decoding attempt but these decoders have their own drawbacks. For example, the SC stack (SCS) decoder [17]–[21] requires a large memory complexity to achieve ML decoding performance, and the SC-Fano [22], [23] and the SC ordered-search (SCOS) decoders [24], [25] require multiple backtracking attempts that cause large decoding latency in the worst-case scenario. Details of the summary of the existing techniques are in Section II.

When decoding short-length and low-rate RM codes, a recently proposed recursive projection-aggregation (RPA) decoder shows superior decoding performance than Dumer's list decoder, and the RPA list decoder extension can achieve ML decoding performance under various rates and lengths [26]. However, since the computational complexity of RPA increases with the order of RM codes, RPA decoders can be seldom used with higher-order RM codes [26].

Many techniques and variants have been developed to reduce the complexity of RPA decoders. For RPA decoders, an early stopping criterion based on the syndrome check and a scheduling scheme based on the sign change per iteration are proposed to reduce the average complexity and the worse-case complexity respectively [27]. A sparse multi-decoder variant of the RPA decoder (SRPA) reduces the complexity by using a subset of subspaces [28]. The collapsed projection-aggregation (CPA) decoder removes the recursive structure in the RPA decoder [29]. For CPA decoders, a pruning metric [30], which can be used to construct a pruned CPA (PCPA) decoder, and an optimized pruning method [31] are proposed to remove subspaces that are likely to return similar decoding results. For the ease of explanation in this work, the RPA decoder and its variants are referred as projection-aggregation (PA) decoders.

When extending the PA decoder to a Chase-type list decoder, Reed's majority-vote decoder is used as the post-processing decoder [26]. The output from the Reed's decoder is a RM codewords, which ensures all decoded codewords are RM codewords when performing the ML-in-the-list selection in the list decoder. A simplified list decoder proposed in [31] uses a syndrome check to replace the Reed's decoder, and

Jiajie Li, Marwan Jalaleddine and Warren J. Gross are with the Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada. Huayi Zhou was with McGill University, and is now with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210018, China, and the Purple Mountain Laboratories, Nanjing 210023, China. (e-mail: {jiajie.li, huayi.zhou, marwan.jalaleddine}@mail.mcgill.ca, warren.gross@mcgill.ca)

Huayi Zhou is personally supported in part by National Science Foundation of China (NSFC) under Grants 62201395 and Jiangsu Provincial Natural Science Foundation under Grants BK20210044.

decoded codewords, which have an all-zeros syndrome, will be used by the ML-in-the-list selection in the simplified list decoder.

Prior work focused on reducing the computational complexity of PA decoding without attempting to reduce the complexity of the PA list decoding. In this work, we introduce several modifications to reduce the computational complexity of the PA list decoder. It should be noted that parts of the techniques used in this work were previously discussed in [27], [31] whereby the syndrome check pattern, the scheduling scheme, the optimized criterion for pruning the CPA, the optimized PCPA (Opt. PCPA), and the simplified list decoder, are devised. This article builds on the earlier work in the following ways:

1) We introduce a redesigned syndrome check pattern, use this new pattern alongside the scheduling scheme [27], and test the pattern with the Opt. PCPA decoder. These techniques not only reduce the complexities but also return the syndrome check result alongside the decoded codeword to avoid the repeated syndrome computation when extended to the simplified list decoder.
2) The Opt. PCPA decoder with the scheduling scheme reduces the worst-case complexity by more than 50% compared to the Opt. PCPA decoder without the scheduling scheme. The Opt. PCPA decoder with redesigned syndrome check pattern and scheduling scheme is extended to the simplified list decoder.
3) A latency model based on the existing hardware architecture is proposed for the list decoder.
4) We use the input-distribution-aware (IDA) method [32], [33] to determine the list size for the list decoder based on the received soft information.
5) We propose a method to compute the average list size of the IDA decoding given the channel model and assumptions on the transmitted data. This differs from the prior works [32], [33] that determine the list size through Monte Carlo simulations.

A RM code with the order r and the code length parameter m is defined as RM(m, r). Numerical experiments are conducted on RM$(7, 3)$, RM$(8, 3)$, and RM$(7, 4)$ codes. By using IDA decoding as a pre-processing tool, the average list size of list decoders can be reduced by 30% with less than 0.1 dB performance loss at a FER of around 10^{-4}. Average list sizes returned from the proposed method are similar to analytical results. When decoding RM$(7, 3)$ and RM$(8, 3)$ codes, the proposed reduced-complexity PA list decoder requires a smaller average complexity than the state-of-the-art automorphism ensemble decoding (AED) with the belief propagation (BP) constituent decoder (AED-BP) [34]. Also, the proposed list decoder has 0.4 dB gain compared to the AED-BP decoder at a FER of 2×10^{-4} when decoding the RM$(8, 3)$ code under the same list/ensemble size of 16. The simplified list decoder has $6.8\times$, $3.5\times$, and $4.7\times$ smaller worst-case complexity than AED-BP when decoding RM$(7, 3)$, RM$(8, 3)$, and RM$(7, 4)$ codes respectively. According to the latency model based on the state-of-the-art hardware architecture, the proposed list decoder has a smaller latency than the AED-BP and the SCL decoder.

This work is structured as the following. Section II gives a summary of existing sequential decoding techniques. Section III provides necessary backgrounds of RM codes, PA decoders, and some complexity reduction methods. Section IV presents a systematic method to calculate the number of operations required by each functional operation in the PA and its list decoders which helps analyze the methods proposed in this work. Section V introduces the redesigned syndrome check pattern and discusses how to select subspaces that are kept in the scheduling scheme. Section VI analyzes the complexity reduction brought by the IDA decoding, and explains the proposed latency model. Experimental results are shown in Section VII. Conclusions are drawn in Section VIII.

II. Summary of related works

The SCL/recursive list decoders [15], [16] can return near ML decoding performance. It is empirically [16] and theoretically [35] observed that the recursive list decoder with a list size of 1024 is required to achieve near ML decoding performance for some RM codes [16]. Methods like the adaptive list decoder [36] reduce the average complexity when a large list size is needed for decoding. However, a large worst-case complexity and a large decoding latency still exist for the adaptive list decoder [36].

Many other decoder types are also developed alongside the list decoding approach. The idea of adopting stack into the recursive and sequential decoders for RM codes is considered in [17], an improved stack-based decoder with the "Look-Ahead" (incorporate not-yet-processed bits/nodes) evaluation metric is proposed in [18], and these decoders achieve near ML decoding performance given a sufficient stack size. Similar research trends appear for polar codes, where the SCS decoder [19], [20] is used to improve the decoding performance and the SCS decoder with a "Look-Ahead" (incorporate not-yet-processed frozen bits) metric is proposed in [21]. While having a lower average complexity than the SCL/recursive list decoder, it is observed that a stack size ≥ 100 is required to achieve near ML decoding performance for some RM and polar codes, which incurs a large memory complexity.

For RM and polar codes, another type of sequential decoders, the SC-Fano decoder achieves near ML decoding performance as well [22], [23]. The SC-Fano decoder has a smaller memory requirement compared to the SCS decoder, and it has competitively low average complexity equal to one SC decoding attempt. However, the SC-Fano decoder for RM and polar codes requires a larger time complexity than SC, SCL, and SCS decoders [37], and it also has a larger decoding latency than the SCL decoder in the hardware implementation [38].

A recently proposed ordered-search variant [24], [25] of the SC decoder also achieves ML decoding performance of RM codes using competitively low average complexity such as an average complexity equal to one SC decoding attempt [24]. These SC ordered-search variants are much easier to tune to achieve ML decoding performance compared to the SC-Fano decoders. In the worst-case scenario, the SC ordered-search decoders [24], [25] require multiple backward tree

search/node-visits in the sequel to reach the ML decoding performance, and have a complexity that is equal to several (e.g., ≥ 100) SC decoding attempts. These sequential attempts might cause high worst-case latency in the hardware implementation, so the impact due to the worst-case complexity needs to be further investigated.

The decoding structure of the PA decoder allows parallel implementation [26], which is a crucial factor that affects the decoding latency and throughput [26]. Also, the PA decoder requires a small memory complexity (e.g., $4n$ for the RPA decoder, $5n$ for the RPA list decoder, and n is the code length) [26]. Moreover, unlike the hard-decision output nature of many SC/recursive decoders mentioned above, the soft-in/soft-out nature of the PA decoder is suitable for iterative detection and decoding [39] according to [34]. A hardware implementation of a PA decoder, iterative PA decoder, without any early stopping criterion [40] shows a higher throughput and a lower decoding latency than a state-of-the-art SCL decoder implementation [41] for polar codes. Hence, the PA and PA list decoders are good complements to existing techniques mentioned above. However, the implementation of the iterative PA decoder has a higher power consumption than the SCL decoder implementation [40], which is partially due to the high computational complexity (both the average and the worst-case complexity). Our work aims to provide a systematic approach for reducing the complexity/power consumption of the PA decoder.

III. Preliminaries

A. Notations

Matrices are denoted as bold upper-case letters ($\boldsymbol{M}$), and vectors are denoted as bold lower-case letters ($\boldsymbol{v}$) unless stated otherwise. The transpose operator is denoted as $^\top$, and a projection based on the coset of a subspace $\mathbb{B}_i$ is represented by the subscript $/\mathbb{B}_i$. The index in binary representation is denoted as z.

B. RM Codes

RM codes can be defined by two parameters m and r, and $0 \leq r \leq m$. The generator matrix for the RM(m, m) code ($\boldsymbol{G}(m, m)$) can be obtained by applying the m-th Kronecker power of the base matrix $\boldsymbol{F}$ [2]:

$$\boldsymbol{G}(m, m) = \boldsymbol{F}^{\otimes m}, \ \boldsymbol{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{1}$$

The generator matrix for the RM(m, r) code ($\boldsymbol{G}(m, r)$) is composed of rows that have the $k = \sum_{i=0}^{i=r} \binom{m}{i}$ largest Hamming weight in $\boldsymbol{G}(m, m)$, and the rate of the RM(m, r) code is $R = k/n$, where $n = 2^m$. The parity check matrix $\boldsymbol{H}$ of the RM(m, r) code is the generator matrix of the RM$(m, m - r - 1)$ code ($\boldsymbol{H} = \boldsymbol{G}(m, m - r - 1)$) [42], and valid codewords $\boldsymbol{c} \in$ RM(m, r) have an all-zeros syndrome vector $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{c}^\top = \boldsymbol{0}$.

C. Projection-Aggregation Decoder

PA decoders are composed of three phases: *i)* projection; *ii)* decoding of $r = 1$ RM sub-codes; and *iii)* aggregation. These three phases are repeated iteratively to produce the decoded codeword.

i): Order r of RM(m, r) codewords is reduced by projecting to s-dimensional subspaces until reaching RM$(m - r + 1, 1)$ sub-codes, where $1 \leq s \leq r - 1$. The projection function for the LLR vector $\boldsymbol{L}$ is

$$\boldsymbol{L}_{/\mathbb{B}_i}(T) = 2 \tanh^{-1} \left(\prod_{z \in T} \tanh \left(\frac{\boldsymbol{L}(z)}{2} \right) \right), \tag{2}$$

where T is the coset of the s-dimensional subspace $\mathbb{B}_i$, and the coset T has a size 2^s. Two main PA decoders, the RPA decoder and the CPA decoder, perform the projection step as the following:

- RPA decoders project the received codeword or the log-likelihood ratio (LLR) vector to $n_{\mathbb{B}} = n - 1$ one-dimensional subspaces, produce $n - 1$ different RM$(m - 1, r - 1)$ sub-codes, and reduce the order r by 1 in every recursion.

- CPA decoders project the received codeword or the LLR vector to $n_{\mathbb{B}} = \binom{m}{r-1}_2$ different $(r - 1)$-dimensional subspaces, and produce RM$(m - r + 1, 1)$ sub-codes without recursions.

ii): RM$(m - r + 1, 1)$ sub-codes are decoded by the FHT decoder. The code bit $\hat{\boldsymbol{y}}_{/\mathbb{B}_i}(T)$ of decoded RM$(m - r + 1, 1)$ sub-codes is an estimation of the parity check of code bits in a coset T.

iii): In the aggregation phase, the estimation of the received LLR vector of the RM(m, r) codeword ($\hat{\boldsymbol{L}}$) is recovered. For both the RPA decoder and the CPA decoder, each LLR is computed according to the $\hat{\boldsymbol{y}}_{/\mathbb{B}_i}(T)$ and all other LLRs in the same coset. Then, results from all subspaces are aggregated:

$$\boldsymbol{L}_{cumu}(z) = \sum_{i=1}^{n_B} -1^{\hat{\boldsymbol{y}}_{/\mathbb{B}_i}(T)} \left(2 \tanh^{-1} \left(\prod_{z_i \in T \setminus z} \tanh \left(\frac{\boldsymbol{L}(z_i)}{2} \right) \right) \right). \tag{3}$$

The average of the aggregated LLR vector ($\hat{\boldsymbol{L}} = \boldsymbol{L}_{cumu}/n_{\mathbb{B}}$) is either fed to the next iteration, or hard decisions of the results are used to output the decoded codeword.

According to [31], a saturation-based early stopping is devised to stop decoding if two consecutive iterations return the same result and if the difference of LLR vectors between these consecutive iterations is smaller than a pre-defined threshold:

$$||\hat{\boldsymbol{L}} - \boldsymbol{L}||_2 < \theta ||\boldsymbol{L}||_2, \text{ and } \hat{\boldsymbol{c}}_i = \hat{\boldsymbol{c}}_{i-1} \ i \in [1, N_{max}], \tag{4}$$

where θ is a small constant, $||\cdot||_2$ denotes the L2 norm, and N_{max} is the maximum number of iterations. The LLR vector of the previous iteration is denoted by $\boldsymbol{L}$. The LLR of the current iteration is denoted by $\hat{\boldsymbol{L}}$. The hard decision of LLR vectors for current and previous iterations are denoted as $\hat{\boldsymbol{c}}_i$

and $\hat{\boldsymbol{c}}_{i-1}$ respectively. Furthermore, a syndrome-based early stopping can be used in the PA decoder [27], whereby the syndrome is checked after aggregating a number of decoded results from subspaces, and the decoding stops when

$$\boldsymbol{H}\hat{\boldsymbol{c}}^{\top} = \boldsymbol{0}. \tag{5}$$

If early-stopping criteria are not satisfied, $\hat{\boldsymbol{L}}$ is used as the LLR vector $\boldsymbol{L}$ for the next iteration. The hard decision of $\hat{\boldsymbol{L}}$ is used as the decoded codeword when the maximum number of iteration N_{max} is reached or early stopping criteria are satisfied [26], [29], [27].

D. Approximated PA Decoder

According to [27], [31], the equation of projection (2) can be replaced by the min-sum approximation [43]. In this work, $\text{sign}(\cdot)$ represents the function of computing the sign of inputs, and the LLR magnitude is denoted by $|\boldsymbol{L}(z)|$. The min-sum approximation of (2) is

$$\boldsymbol{L}_{/\mathbb{B}_i}(T) \approx (\prod_{z\in T} \text{sign}(\boldsymbol{L}(z)) \min\{|\boldsymbol{L}(z)|, \forall\ z \in T\}. \tag{6}$$

The hyperbolic tangent and the inverse hyperbolic tangent portion of the aggregation (3) can be approximated by the min-sum accordingly,

$$\boldsymbol{L}_{cumu}(z) \approx \sum_{i=1}^{n_B} -1^{\hat{\boldsymbol{y}}_{/\mathbb{B}_i}(T)}(\prod_{z_i\in T\setminus z} \text{sign}(\boldsymbol{L}(z_i)) \cdot \min(\{|\boldsymbol{L}(z_i)|, \forall\ z_i \in T \setminus z\}) \tag{7}$$

E. Optimized PCPA Decoder

It is shown in [30] that projections to subspaces might have similar error patterns if their subspaces are similar, and the probability of having similar error patterns is

$$P = \frac{1}{2}[1 + (1-2\epsilon)^{(2^{(r-1)+1}-2|\mathbb{B}_i \bigcap \mathbb{B}_j|)}], \tag{8}$$

where $\mathbb{B}_i$ and $\mathbb{B}_j$ are two subspaces, and ϵ is the probability of an independent error occurs in the received vector. Based on (8), a correlation coefficient is proposed to measure the similarity of subspaces in the CPA decoder:

$$r_S := \sum_{i=1}^{|S|}\sum_{j=1}^{|S|} r_{ij} \text{ and } r_{ij} := \frac{\dim(\mathbb{B}_i \bigcap \mathbb{B}_j)}{r-1}, \tag{9}$$

where r_{ij} is the pair-wised correlation between $\mathbb{B}_i$ and $\mathbb{B}_j$, S is the set of subspaces, r_S is the set correlation for S, and $\dim(\cdot)$ is the function to compute the dimension of the intersection of two subspaces. In equation (9), i and j index subspaces in S. $|S|$ denotes the size of the set of subspaces S. Then, pruning can be performed based on the correlation metric, and a subset that has the least similar subspaces is constructed based on (9). The smaller the r_S, the better the subspace collection S, and finding a S with a small r_S can be transformed into a mixed-integer quadratic programming problem [31]:

$$\min_{\boldsymbol{u}} \boldsymbol{u}\boldsymbol{R}\boldsymbol{u}^{\top}, \text{ s.t. } u_i \in \{0,1\}\ \forall u_i \in \boldsymbol{u},\ \boldsymbol{1}\boldsymbol{u}^{\top} = |S|, \tag{10}$$

where $u_i = 1$ if the subspace $\mathbb{B}_i \in S$, and $u_i = 0$ otherwise. $\boldsymbol{R}$ is a matrix that stores all values r_{ij}, and it is symmetric and positive semi-definite if diagonal entries are properly scaled by adding a diagonal positive semi-definite matrix [44]. The all-ones vector is denoted by $\boldsymbol{1}$. Methods have been proposed to solve (10) and produce Opt. PCPA decoders in [31].

F. Projection-Aggregation List Decoder

The list decoder extension of PA decoders is a Chase decoder [26]. First, positions corresponding to t smallest LLR value in terms of magnitude ($|\boldsymbol{L}(z)|$) and the maximum LLR magnitude of $\boldsymbol{L}$ ($\max_z(|\boldsymbol{L}(z)|)$) are found, where $\boldsymbol{L}(z) \in \boldsymbol{L}$. Then, LLRs in these t positions are replaced with $\pm\max_z(|\boldsymbol{L}(z)|)$ or $\pm 2\max_z(|\boldsymbol{L}(z)|)$. There are 2^t possible cases, so the list size is 2^t. These 2^t cases are decoded by the PA decoder independently. But the decoded codeword of RPA/CPA decoders is not necessarily a RM codeword. Reed's decoder is used to correct the decoded codewords to valid RM codewords [26], or a syndrome check is used to filter out decoded codewords that are not RM codewords [31]. The codewords returned from the Reed's decoder or PA decoder's results with an all-zeros syndrome vector are included in the list, and the codewords with the largest posterior probability in the list will be returned from the list decoder [26], [31].

IV. Complexity Analysis of Projection-Aggregation Decoders

Given all proposed complexity reduction techniques, a throughout complexity analysis is needed to demonstrate the effectiveness of proposed techniques and parameter tuning. In this work, the complexity analysis follows a weighted count modified from the rule proposed in [34], and the weight is shown in Table I. In [34], BP decoding is assumed to use the box plus and the box minus approximation [45]. In this work, the PA decoder is assumed to use the min-sum approximation, when the complexity is counted. This work considers projection, FHT decoding, aggregation, and syndrome checks as key functional operations of the PA decoder. Weights of these functional operations are computed as the following.

Projection: From the approximated projection (6), signs of LLR in the same coset are multiplied, and the minimum LLR magnitude is searched among the coset. When projecting to d-dimensional subspaces, each coset contains $D = 2^d$ LLRs, where $d = r-1$ for the CPA/PCPA decoder. For a coset size D, $D-1$ pair-wised multiplications of signs ($\text{sign}(x)\text{sign}(y)$) and pair-wised comparisons for finding the minimum magnitude ($\min(|x|,|y|)$) are needed. When projecting to a d-dimensional subspace, the projection operation is repeated 2^{m-d} times for 2^{m-d} different cosets. A sign change ($\text{sign}(x)y$) is performed after the sign of the parity check and the minimum LLR magnitude are found for each coset. For example, the projection of the RM$(7,3)$ code requires $(2^2-1)\times 2^{7-2} = 96$ pair-wised multiplications of signs and comparisons, and $2^{7-2} = 32$ sign changes. The projection of the RM$(7,4)$ code requires $(2^3-1)\times 2^{7-3} = 112$ pair-wised multiplications of signs and comparisons, and $2^{7-3} = 16$ sign changes. Operations repeat N times when projecting to N subspaces.

TABLE I
WEIGHTS OF KEY FUNCTIONAL OPERATIONS OF PA AND ITS LIST DECODERS, WHERE $d = r-1$, $D = 2^d$, N IS THE NUMBER OF DECODED SUBCODES, N_{SYN} IS THE NUMBER OF SYNDROME CHECKS, AND $N_{\text{ITER.}}$ IS THE NUMBER OF ITERATIONS.

Operation	Weight	2-input $\boxplus$ [34]	Projection	FHT	Aggregation	ML out of L	FFG BP Stopping [34]	Syndrome Stopping
$\text{sign}(x)\text{sign}(y)$	1	N	$N(D-1)2^{m-d}$	$N(\frac{(m-d)2^{m-d}}{2})$	$N(D-2)n$	0	$N_{\text{iter.}}(\frac{mn}{2}+2n-1)$	0
$\text{sign}(x)y$	1	N	$N2^{m-d}$	0	$N2n$	Ln	0	0
$\min(\lvert x\rvert,\lvert y\rvert)$	1	N	$N(D-1)2^{m-d}$	0	$N(D-2)n$	0	0	0
$\max(x,y)$	1	0	0	$N(2^{m-d}-1)$	0	$L-1$	0	0
$f_{\pm}(\lvert x\rvert)$ LUT	1	$2N$	0	0	0	0	0	0
$x+y$, $x-y$	1	$4N$	0	$N(2^{m-d}(m-d))$	$n(N-1)$	$L(n-1)$	0	0
$x \cdot y$	3	0	0	0	$N_{\text{iter.}}n$	0	0	0
$\text{AND}(x,y)$	1	0	0	0	0	0	0	$N_{\text{SYN}}(k(n-k)+\frac{n-k}{2}(n-k+1))$
$\text{XOR}(x,y)$	1	0	0	0	0	0	0	$N_{\text{SYN}}((k-1)(n-k)+\frac{n-k}{2}(n-k+1))$
Weighted total	-	$9N$	$N(2D-1)2^{m-d}$	$N((\frac{3(m-d)}{2}+1)2^{m-d}-1)$	$(N(2D-1)-1+3N_{\text{iter.}})n$	$2Ln-1$	$N_{\text{iter.}}(\frac{mn}{2}+2n-1)$	$N_{\text{SYN}}(2k(n-k)+(n-k)^2)$

FHT decoding: The FHT decoding is performed on the projected codeword that has a code length of 2^{m-d}, and the FHT decoding has three steps [46]: i), perform the FHT; ii), find the index corresponding to the element with the largest magnitude in the FHT output; iii), encode the message vector corresponding to that index back to the first-order RM code. The FHT needs $2^{m-d}(m-d)$ additions and subtractions in total. $2^{m-d}-1$ pair-wised comparisons ($\max(x,y)$) are need to search the index of the maximum magnitude in the FHT output with a length 2^{m-d}. The maximum complexity of fast encoding for polar codes is $\frac{1}{2}n\log n$ [47], [48], where n is the code length. Due to the similarity between polar codes and RM codes, the same encoding technique can apply to RM codes, and the encoding complexity of RM codes is assumed to be the same of polar codes. As the encoding operation is viewed as the sign multiplication and the complexity of the fast encoding is used as the weight in the prior work [34], we follow the same convention and the weight of the encoding of the order-1 RM subcodes is $\left((m-d)\,2^{m-d}\right)/2$ sign multiplication operations. FHT decoding of the RM$(7,3)$ code requires $2^5\times 5 = 160$ additions and subtractions, $2^5-1=31$ pairwised comparisons, and $\left(2^5\times 5\right)/2 = 80$ sign multiplications. FHT decoding of the RM$(7,4)$ code requires $2^4\times 4 = 64$ additions and subtractions, $2^4-1=15$ pairwised comparisons, and $\left(2^4\times 4\right)/2 = 32$ sign multiplications. FHT decoding is done N times to decode N subcodes.

Aggregation: From (7), each decoded LLR is estimated based on all other LLRs in the same coset, so the min-sum approximation (7) is performed among $D-1$ elements. This results in $D-2$ pair-wised multiplications of signs ($\text{sign}(x)\text{sign}(y)$) and pair-wised comparisons($\min(|x|,|y|)$). A sign change operation ($\text{sign}(x)y$) is performed after the sign of the parity check and the minimum LLR magnitude are found. Also, an additional sign change is performed based on the decoded first-order codeword ($-1^{\hat{\boldsymbol{y}}/\mathbb{B}_i(T)}$). Thus, two sign change operations are needed by the aggregation. These operations are repeated n times to get estimations of n LLRs in the received LLR vector. Hence, the estimation of the received LLR vector of a RM$(7,3)$ code requires $2\times 2^7 = 256$ pair-wised multiplications of signs and comparisons, and $2\times 2^7 = 256$ sign change operations. The estimation of the received LLR vector of a RM$(7,4)$ code requires $6\times 2^7 = 768$ pair-wised multiplications and comparisons, and $2\times 2^7 = 256$ sign change operations. After computing results from N subspaces, $N-1$ additions are needed to accumulate results from all subspaces. In the aggregated LLR vector, n estimated LLRs are normalized in every iteration, which takes n multiplications $(x\cdot y)$.

Syndrome early stopping: Since the parity check matrix of the RM(m,r) code is $\boldsymbol{H} = \boldsymbol{G}(m, m-r-1)$, the construction process of $\boldsymbol{H}$ can be viewed as removing first k rows in $\boldsymbol{G}(m,m)$ with a weight $< 2^{r+1}$. Hence, the structure of $\boldsymbol{H}$ can be approximately viewed as the concatenation of a $(n-k)\times k$ sub-matrix on the left and a $(n-k)\times(n-k)$ lower triangular sub-matrix on the right. To compute the ith element in the length $n-k$ syndrome vector, $k+i$ multiplications and $k+i-1$ additions are needed. In total, the number of multiplications is

$$\sum_{i=1}^{n-k} k+i = k\times(n-k)+\frac{n-k}{2}(n-k+1),$$

and the number of additions is

$$\sum_{i=1}^{n-k} (k-1)+i = (k-1)\times(n-k)+\frac{n-k}{2}(n-k+1).$$

Hence, the parity check of the RM$(7,3)$ code requires 6112 additions and 6176 multiplications. The parity check of the RM$(7,4)$ code requires 3277 additions and 3306 multiplications. Operations mentioned above are repeated N_{syn} times if N_{syn} syndrome checks are performed. Moreover, the multiplication and the addition in the binary field are viewed as AND and XOR operations respectively, which are assumed to have a unit cost.

V. REDUCED-COMPLEXITY CPA/PCPA DECODERS

In this section, the syndrome-based early stopping (SYN) and the scheduling scheme (SCH) method are applied to the CPA/PCPA decoder. First, the syndrome check pattern is redesigned, and this redesigned pattern aims to avoid repeated syndrome computations when extending the decoder to the simplified list decoder. Also, the redesigned syndrome check pattern should address corner cases generated by the characteristics of the CPA decoder and the proposed scheduling scheme (SCH) for reducing the worst-case complexity. Second, to apply SCH in the CPA/PCPA decoder, a discussion on subspaces that are kept in the scheduling scheme is presented in this work.

A. Redesigned Syndrome Check Pattern

The syndrome can be checked after receiving the LLR vectors and aggregating δ decoding results as proposed in [27].

This proposed syndrome check pattern does not take several corner cases into consideration, and repeated syndrome computations might be performed when extending to the simplified list decoder. Corner cases addressed in this work are the following:

1) The number of subspaces in the PA decoder might not be divisible by the frequency δ, so the syndrome check might be skipped at the end of the iteration.
2) When adopting the SCH [27] into the decoder, it is likely that the number of subspaces is smaller than the syndrome check frequency δ or that the number of subspaces is not divisible by δ.
3) A syndrome check result should be returned alongside the decoded codeword to avoid repeated syndrome computation in the simplified list decoder.

Because of the corner cases mentioned above, the syndrome check pattern should be redesigned. Our simulation results show that, on average, several FHT decoding attempts are needed to output a codeword at our target FER, and we hypothesize that error-free received LLR vectors are rare. So, the redesigned syndrome check pattern removes the syndrome check on the received LLR vector. Also, we propose an adaptive frequency to handle corner cases, which ensures that every returned decoded codeword will be accompanied by a syndrome check result. Assume $n_{\mathbb{B}}$ is the number of subspaces used in the current iterations, and $n_{\mathbb{B}}$ is either not divisible by the δ or smaller than δ. Given a frequency δ, we compute the sequence that contains the number of aggregated decoding results of subspaces before the syndrome check will perform as follows:

Step 1: The total number of syndrome checks is calculated as $\lceil \frac{n_{\mathbb{B}}}{\delta} \rceil$, and a sequence $\boldsymbol{f} = [1, 2, ..., \lceil \frac{n_{\mathbb{B}}}{\delta} \rceil]$, which records the subspace's indexes of where the syndrome check should be performed, is generated.

Step 2: Before performing a syndrome check, a number of subspace results should be aggregated. The sequence that stores the number of aggregated subspaces is calculated by $\delta \boldsymbol{f}$.

Step 3: The elements in $\boldsymbol{f}$ are clipped to $n_{\mathbb{B}}$ ($\{\min(\delta \times i, n_{\mathbb{B}}), \forall i \in \{1, ..., \lceil \frac{n_{\mathbb{B}}}{\delta} \rceil\}\}$).

Two examples are given to demonstrate how the adaptive scheme handles corner cases mentioned above.

Example 1: Assume that a RM$(7, 2)$ code is decoded by the CPA decoder and the syndrome check frequency $\delta = 32$. When $r = 2$, the CPA decoder is equivalent to the RPA decoder. Hence, the CPA decoder needs to decode 127 subspaces. The number of syndrome checks is $\lceil \frac{n_{\mathbb{B}}}{\delta} \rceil = \lceil \frac{127}{32} \rceil = 4$, and a sequence $[1, 2, 3, 4]$ is created. Then, the sequence is multiplied with the syndrome check frequency δ to get indexes when the syndrome check should perform ($\delta \boldsymbol{f} = [32, 64, 96, 128]$). Lastly, indexes that are larger than $n_{\mathbb{B}}$ are clipped to $n_{\mathbb{B}}$, and the sequence becomes $[32, 64, 96, 127]$. Hence, the syndrome check would be performed after aggregating 32, 64, 96, and 127 results of subspaces respectively.

Example 2: Assume that a RM$(7, 2)$ code is decoded by the CPA decoder with a syndrome check frequency $\delta = 32$, the reduction factor for SCH is $d_{\text{SCH}} = 2$, and $N_{max} = 4$. When $r = 2$, the CPA decoder is equivalent to the RPA decoder. Hence, the CPA decoder needs to decode 127 subspaces in the first iteration, 64 subspaces in the second iteration, 32 subspaces in the third iteration, and 16 subspaces in the last iteration. It can be seen that a syndrome check can be outputted alongside the decoded codeword for the first three iterations, and the decoder at the iteration four cannot output a syndrome check alongside the decoded codeword as $n_{\mathbb{B}}$ at iteration four is smaller than δ. Hence, by the adaptive scheme at iteration four, the number of the syndrome check is $\lceil \frac{n_{\mathbb{B}}}{\delta} \rceil = \lceil \frac{4}{32} \rceil = \lceil 0.125 \rceil = 1$, and the generated sequence is $[1]$. Then, the sequence is multiplied by δ to get the index ($[1 \times \delta] = [32]$), and addresses the corner case by clipping to $n_{\mathbb{B}}$ ($[\min(32, n_{\mathbb{B}})] = [\min(32, 16) = 16]$).

B. Scheduling Scheme

The scheduling scheme [27] can be applied to the CPA/PCPA decoder to reduce the worst-case complexity. The maximum number of subspaces/projections is reduced by a factor of d_{SCH} after every iteration. If we choose to use the number of instances of FHT decoding to estimate the worst-case complexity, then the worst-case complexity of the CPA/PCPA decoder with SCH is $\sum_{i=0}^{N_{max}-1} \lceil \frac{n_{\mathbb{B}}}{d_{\text{SCH}}^i} \rceil$ instances of FHT decoding, where $n_{\mathbb{B}}$ is the number of subspaces used by the CPA/PCPA decoder. The other problem that should be answered is what subspaces should be kept for later iterations if the SCH is used in the PCPA decoder.

In [27], assume there are $n_{\mathbb{B}}$ subspaces in the RPA decoder, and these $n_{\mathbb{B}}$ subspaces are firstly shuffled. At every iteration i, the RPA$_{\text{SCH}}$ decoder uses the first $\lceil \frac{n_{\mathbb{B}}}{d_{\text{SCH}}^i} \rceil$ subspaces in the decoding process, where $i \in \{0, 1, ..., N_{max} - 1\}$. A greedy search method is proposed to generate the Opt. PCPA decoder in [31], which firstly collects a set of subspaces that have zero pairwise correlation with each other, and then, a new subspace that incurs a minimum increment of set correlation with respect to the current set of subspaces is added:

$$\text{argmin}_i = \sum_{\mathbb{B}_j \in S} \frac{\dim(\mathbb{B}_i \bigcap \mathbb{B}_j)}{r - 1}, \forall \mathbb{B}_j \in S \text{ and } \forall \mathbb{B}_i \notin S. \tag{11}$$

Hence, for the subspace order generated by the greedy search method, keeping the first $\lceil \frac{n_{\mathbb{B}}}{d_{\text{SCH}}^i} \rceil$ subspaces naturally forms a set of subspaces with a small set correlation.

Simulations on PCPA decoders with the shuffled and the greedy search generated subspace order are performed, and the pseudo-code of the PCPA decode with the new syndrome check pattern and SCH is shown in Algorithm 1. Fig. 1 (a), (c), and (e) show the FER results of PCPA decoders with and without the syndrome check and the SCH. The greedy search generated subspace order (Ord.) and three different shuffled subspace orders (Rand.) are used in the simulation. The frequency δ for the syndrome check is set to 16 for the RM $(7, 3)$ code, 8 for the RM $(7, 4)$ code, 64 for the RM $(8, 3)$, and the reduction factor $d_{\text{SCH}} = 2$. The reasoning for selecting the parameters δ is explained in Section VII. PCPA denotes the PCPA decoder without the syndrome check early stopping and the SCH. Ord. denotes keeping the first

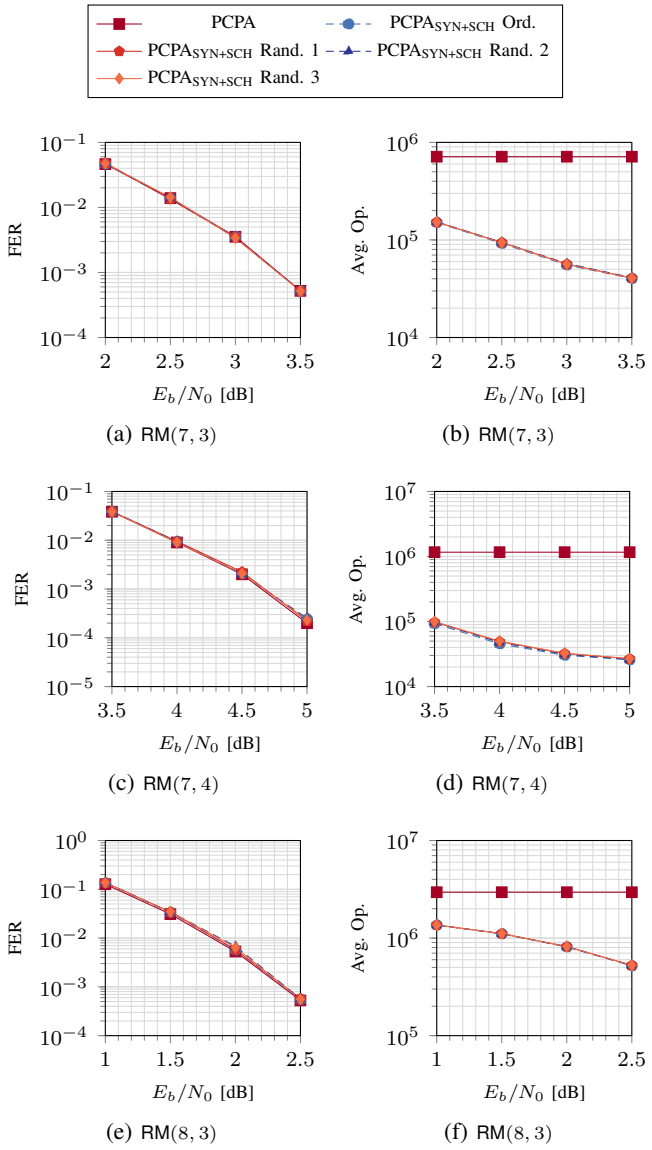


Fig. 1. FERs and numbers of operations of PCPA decoders.

$\lceil \frac{n_{\mathbb{B}}}{d^i_{\text{SCH}}} \rceil$ subspaces with greedy search generated subspace order. Rand. 1, 2, and 3 denote keeping the first $\lceil \frac{n_{\mathbb{B}}}{d^i_{\text{SCH}}} \rceil$ subspaces with three different shuffled orders. PCPA decoders with different scheduling scheme orders have similar error correction performance as the PCPA decoder without the syndrome check and the SCH. Observed from Fig. 1 (b), (d), and (f), a similar average complexity is required by the PCPA decoder with the Ord. reduction order, compared to random reduction orders. In this work, the PCPA decoder with the greedy search generated subspace order is used for all simulations.

Methods have been proposed to reduce the number of FHT decoding attempts in the worst case. We assume all recursive layers in the RPA decoder use the same N_{max}. The original RPA decoder [26] uses all $n-1$ subspaces in every recursive layer, so the worst-case complexity of the RPA decoder is $\prod_{i=0}^{r-2} N_{max}(2^{m-i}-1)$. The SRPA decoder employs multiple sparse RPA decoders that use only a portion of the subspaces [28]. If one-eighth of subspaces are used in each recursive layer, the worst-case complexity of the SRPA decoder is $(N_{max})^{r-1}\prod_{i=0}^{r-2}\frac{2^{m-i}}{8}q_{r-i}$ FHT decoding attempts, where q_{r-i} is the number of sparse decoders in the order-$(r-i)$ recursive layer. The RPA decoder with SCH employs a decay factor d_{SCH} in each recursion layer, and the maximum number of subspaces in each recursion layer is reduced by a factor d_{SCH} in the next iteration [27], which requires $\prod_{i=0}^{r-2}\sum_{j=0}^{N_{max}-1}\lceil\frac{2^{m-i}-1}{d^j_{\text{SCH}}}\rceil$ FHT decoding attempts in the worst case. The RPA decoder with the multi-factor pruning strategy (RPA$_{\text{MF}}$) uses different pruning factors for different layers and iterations [49], and its worst-case complexity is $\lambda_{\text{RPA}_{\text{MF}}}(m, r, \gamma, \delta_{\text{iter}}, \delta_{\text{rec}}, N_{max})$

$$= \sum_{i=1}^{N_{max}} \lceil \gamma\delta_{\text{iter}}^{i-1}\delta_{\text{rec}}^{r-2}(2^m-1))\rceil \lambda_{\text{RPA}_{\text{MF}}}(m-1, r-1, \gamma\delta_{\text{iter}}^{i-1},$$

$\delta_{\text{iter}}, \delta_{\text{rec}}, N_{max})$

FHT decoding attempts, where γ is the pruning factor at the first iteration of the order-r recursive layer, and δ_{iter} and δ_{rec} are pruning factors for every iteration and recursion.

Since the pruning criteria proposed in [30] only works for RM codes with $r > 2$, the comparisons of worst-case complexity are performed on the RM$(7, 3)$ code. From literature [27], [28], [49], all these worst-case complexity reduction techniques incur negligible performance loss. Table II presents results of several worst-case complexity reduction methods. The reduction factor d_{SCH} for RPA$_{\text{SCH}}$ and PCPA$_{\text{SCH}}$ with 128 subspaces is 2. $1/8$ of total subspaces are used by the SRPA decoder, and the number of decoders that only use a subset of subspaces are 2 and 4 for recursive layers with $r = 3$ and $r = 2$ respectively. The parameter setting for the RPA with the multi-factor pruning is $(\gamma, \delta_{iter}, \delta_{rec}) = (\frac{3}{4}, \frac{1}{3}, \frac{3}{4})$. From Table II, the worst-case complexity has been reduced by a significant portion by previous work. The PCPA decoder with SCH leverages the non-recursive structure, pruning, and the SCH, so it requires significantly fewer instances of FHT decoding compared to other methods. By using the SCH with the PCPA decoder, the worst-case complexity is reduced by 50% compared to the PCPA decoder without the SCH.

VI. Proposed Modifications to List Decoder

Due to the large computational complexity measured by the number of FHT decoding ($O(n^r \log n)$ for RPA decoders [26] and $O(n \log n)$ with a big multiple $\binom{m}{r-1}_2$ for CPA decoders [29]), PA list decoders are less attractive compared to other low-complexity decoders for RM codes such as the AED for RM codes [34]. Reductions in complexity are needed for PA list decoders to compete with state-of-the-art decoders. IDA decoding is used to reduce the high computational complexity of the list decoder in this work. Also, based on the existing hardware architecture [40], a latency model is proposed for the PA list decoder.

A. IDA Assisted Reduced Complexity List Decoder

IDA decoding is proposed by Condo et al. to reduce the average complexity of list decoders [32], [33] with little decoding performance degradation. It is observed that most received soft information can be decoded using small list sizes [33]. Thus, by using IDA as a pre-processing tool, the

TABLE II
WORST-CASE COMPLEXITY OF PA DECODERS. THE WORST-CASE COMPLEXITY IS MEASURED BY THE MAXIMUM NUMBER OF FHT DECODING ATTEMPTS IN THE DECODER.

RM codes	RPA [26]	RPA_{SCH} [27]	SRPA [28]	CPA [29]	RPA_{MF} [49]	PCPA-128 [31]	$\text{PCPA-128}_{\text{SCH}}$
RM$(7,3)$	128016	28800	16384	10668	5879	512	240

Algorithm 1: $\text{PCPA}_{\text{SYN+SCH}}$

Input: $\boldsymbol{L}$, m, r, N_{max}, $n_{\mathbb{B}}$, d_{SCH}, δ, θ
Output: $\hat{\boldsymbol{c}}$

$\boldsymbol{H} \leftarrow \boldsymbol{G}(m, m-r-1)$
$\hat{\boldsymbol{c}}_{old} \leftarrow \boldsymbol{0}$
$\hat{\boldsymbol{L}}_{old} \leftarrow \boldsymbol{0}$
for $i = 1, ..., N_{max}$ **do**
  for $i = 1, ..., \lceil n_{\mathbb{B}}/\delta \rceil$ **do**
    $\boldsymbol{f}(i) \leftarrow \min(\delta \times i, n_{\mathbb{B}})$
  $\boldsymbol{L}_{cumu}(z) \leftarrow 0 \ \forall z \in \{0,1\}^m$
  for $j = 1, ..., n_{\mathbb{B}}$ **do**
    $\boldsymbol{L}_{/\mathbb{B}_j} \leftarrow \texttt{Projection}(\boldsymbol{L}, \mathbb{B}_j)$
    $\hat{\boldsymbol{y}}_{/\mathbb{B}_j} \leftarrow \texttt{FHTDecoding}(\boldsymbol{L}_{/\mathbb{B}_j})$
    $\boldsymbol{L}_{cumu} += \texttt{Aggregation}(\boldsymbol{L}, \hat{\boldsymbol{y}}_{/\mathbb{B}_j})$
    if $j \in \boldsymbol{f}$ **then**
      $\hat{\boldsymbol{c}} \leftarrow \texttt{HardDecision}(\boldsymbol{L}_{cumu})$
      $\boldsymbol{s} \leftarrow \boldsymbol{H}\hat{\boldsymbol{c}}^{\top}$
      if $\boldsymbol{s} == \boldsymbol{0}$ **then**
        break
  $\hat{\boldsymbol{L}} \leftarrow \frac{\boldsymbol{L}_{cumu}}{j}$
  $n_{\mathbb{B}} \leftarrow \lceil n_{\mathbb{B}}/d_{\text{SCH}} \rceil$
  $\hat{\boldsymbol{c}} \leftarrow \texttt{HardDecision}(\hat{\boldsymbol{L}})$
  if $(||\hat{\boldsymbol{L}}_{old} - \hat{\boldsymbol{L}}||_2 < \theta||\hat{\boldsymbol{L}}||_2$ ***and*** $\hat{\boldsymbol{c}}_{old} == \hat{\boldsymbol{c}})$ ***or*** $\boldsymbol{s} == \boldsymbol{0}$ **then**
    break
  $\hat{\boldsymbol{L}}_{old}, \boldsymbol{L} \leftarrow \hat{\boldsymbol{L}}$
  $\hat{\boldsymbol{c}}_{old} \leftarrow \hat{\boldsymbol{c}}$
return $\hat{\boldsymbol{c}}$, $\boldsymbol{s}$

Algorithm 2: IDA/M-IDA-Simp. List Decoder

Input: $\boldsymbol{L}$, m, r, N_{max}, θ, t, γ, φ, δ, d_{SCH}
Output: $\hat{\boldsymbol{c}}$

$\boldsymbol{H} \leftarrow \texttt{GenerateParityMatrix}(\boldsymbol{G}(m, m-r-1))$
$\tilde{\boldsymbol{L}} \leftarrow \boldsymbol{L}$
ListSize $\leftarrow \texttt{IDA/M-IDA}(\boldsymbol{L}, \gamma, \varphi)$
$t \leftarrow \log_2(\text{ListSize})$
$(z_1, ..., z_t) \leftarrow$ indices of the t smallest $|\boldsymbol{L}(z)|$, $z \in \mathbb{E}$
$L_{max} \leftarrow 2\max(\{|\boldsymbol{L}(z)|, \forall z \in \mathbb{E}\})$
$i = 1$
for $\boldsymbol{l} \in \{L_{max}, -L_{max}\}^t$ **do**
  $(\boldsymbol{L}(z_1), \boldsymbol{L}(z_2), ..., \boldsymbol{L}(z_t)) \leftarrow \boldsymbol{l}$
  $\hat{\boldsymbol{c}}$, $\boldsymbol{s} \leftarrow \text{PCPA}_{\text{SYN+SCH}}(\boldsymbol{L}, m, r, N_{max}, \theta, \delta, d_{\text{SCH}})$
  if $\boldsymbol{s} == \boldsymbol{0}$ **then**
    $\tilde{\boldsymbol{C}}(i, :) \leftarrow \hat{\boldsymbol{c}}$
    $i = i + 1$
index $\leftarrow \text{argmax}_i \sum_{z \in \{0,1\}^m} ((-1)^{\tilde{\boldsymbol{C}}(i,z)} \tilde{\boldsymbol{L}}(z))$
return $\hat{\boldsymbol{c}} \leftarrow \tilde{\boldsymbol{C}}(\textit{index}, :)$

list size can be determined based on the statistical information of received LLRs.

The IDA decoding determines the list size by counting the number of LLRs with a magnitude that is smaller than a threshold γ [32]. If the number of LLRs, which have a magnitude that is smaller or equal to γ, is smaller than the threshold φ, a small list size (L_{low}) is used [32]. Otherwise, a large list size (L_{high}) is used. A low-complexity IDA decoding for Chase decoders, M-IDA decoding, is proposed to use L_{low} if the $(t-1)$-th smallest LLR magnitude of the received vector is larger than the threshold γ [33]. Otherwise, L_{high} is used. The M-IDA can be seen as a low-complexity variant of the IDA, as it requires fewer comparisons and counting [33]. From [33], the M-IDA decoder is more suitable for the Chase list decoder compared to another low-complexity variant, MD-IDA. It is mentioned in [32] that a large γ is used for a large L_{low}, and a small γ is used for a small L_{low}.

As IDA and M-IDA decoding have low complexities [32], [33], using IDA or M-IDA as a pre-possessing tool will not increase the complexity of the list decoder significantly. In this work, the IDA decoder is used as a pre-processing tool for the PA list decoder, and the pseudo-code of the IDA/M-IDA simplified list (IDA/M-IDA-Simp. List) decoder with proposed complexity reduction techniques is shown in Algorithm 2.

B. Computation of Complexity Reduction for IDA Decoding

By modifying equation (1) in [32], the average/expected list size can be expressed as

$$\bar{L} = \delta L_{low} + (1-\delta) L_{high}, \tag{12}$$

where δ is the fraction of time that L_{low} is used to decode [32]. In [32] and [33], the fraction of time δ and the average list size $\bar{L}$ are found through the Monte Carlo simulation. In this work, we postulate that δ is the probability of receiving an LLR vector that can be decoded using L_{low}, and $\bar{L}$ can be found using the computed δ. For the additive white Gaussian noise channel (AWGN) channel, the LLR l is computed as [46]

$$l = \frac{2}{\sigma^2} y = 4 \times R \times 10^{0.1 \times E_b/N_0} \times y, \tag{13}$$

where y is the channel output, E_b/N_0 is in decibel [dB], and R is the code rate. The channel output corresponding to the LLR threshold γ is

$$y_\gamma = \frac{\gamma}{(4 \times R \times 10^{E_b/N_0 \times 0.1})}. \tag{14}$$

As independent and identically distributed (i.i.d) code bits are transmitted, and the symmetric AWGN channel and the binary phase-shift keying (BPSK) modulation are used, the all-zeros transmitted codeword can be used to compute the probability of receiving a soft information vector that can be decoded by L_{low}. For IDA decoding, the probability of receiving an LLR

vector that can be decoded with L_{low} is

$$\begin{aligned}\delta &= \sum_{\varphi_i=0}^{\varphi-1} \binom{n}{\varphi_i} P(|l| > \gamma | x=1)^{n-\varphi_i} P(|l| \le \gamma | x=1)^{\varphi_i} \\ &= \sum_{\varphi_i=0}^{\varphi-1} \binom{n}{\varphi_i} (1 - P(|l| \le \gamma | x=1))^{n-\varphi_i} P(|l| \le \gamma | x=1)^{\varphi_i},\end{aligned} \tag{15}$$

where l is the LLR corresponding to the channel output y, and

$$P(|l| \le \gamma | x=1) = \int_{-y_\gamma}^{y_\gamma} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y-1)^2}{2\sigma^2}) \mathrm{d}y. \tag{16}$$

For M-IDA, the probability of receiving an LLR vector that can be decoded with L_{low} is

$$\begin{aligned}\delta &= \sum_{t_i=0}^{t-2} \binom{n}{t_i} P(|l| > \gamma | x=1)^{(n-t_i)} P(|l| \le \gamma | x=1)^{t_i} \\ &= \sum_{t_i=0}^{t-2} \binom{n}{t_i} (1 - P(|l| \le \gamma | x=1))^{(n-t_i)} P(|l| \le \gamma | x=1)^{t_i}.\end{aligned} \tag{17}$$

Let the list size L be a random variable with two realizations L_{low} and L_{high}, and the corresponding probabilities are δ and $1-\delta$. The list size only depends on the current received LLR vector, and it does not depend on other list sizes generated by previous and later received LLR vectors. So, in the simulation, all list sizes generated by IDA or M-IDA decoding are mutually independent and hence uncorrelated. It can be concluded that the list size L is an i.i.d random variable. Also, the expected value of L^2 is bounded ($\mathbb{E}[L^2] = \delta L_{low}^2 + (1-\delta) L_{high}^2 \le L_{high}^2 \in \mathbb{R}$). By the strong law of larger numbers [50], the empirical mean of the list size returned from simulations converges to the expected value of the list size almost surely as the number of simulation runs goes to infinity.

The complexity reduction produced by IDA decoding is found through simulations in previous work [32], [33]. The analytical computation for the reduced average list size helps speed-up the process of parameter tuning for IDA decoding, and it is also possible to extend to other channel models.

The uncorrelated normalized Rayleigh fading channel is used as a demonstration. The normalized Rayleigh fading channel distorts the transmitted signal as the following:

$$y = r \times x + \mathcal{N}, \tag{18}$$

where r follows a normalized Rayleigh distribution [51]:

$$P(r) = 2r \exp(-r^2),$$

$x \in \{-1, 1\}$ is the symbol of the BPSK modulation, and $\mathcal{N}$ follows the Gaussian distribution with mean 0 and variance σ^2. With the perfect state information (SI), the fading factor r is treated as a constant [52]. Without SI, the received signal y follows the following distribution [52]:

$$\begin{aligned}P(y|x) &= \int_0^\infty \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-rx)^2}{2\sigma^2}\right) P(r) \mathrm{d}r \\ &= \frac{xy \exp\left(\frac{x^2y^2}{2\sigma^2(x^2+2\sigma^2)} - \frac{y^2}{2\sigma^2}\right) \operatorname{erfc}\left(\frac{-xy}{\sqrt{2\sigma^2(x^2+2\sigma^2)}}\right)}{(x^2+2\sigma^2)^{\frac{3}{2}}} \\ &\quad + \frac{\sqrt{2}\,\sigma \exp(-\frac{y^2}{2\sigma^2})}{\sqrt{\pi}\,(x^2+2\sigma^2)}.\end{aligned} \tag{19}$$

where $\operatorname{erfc}(\cdot)$ is the complementary error function.

When there is no SI, the LLR of the normalized Rayleigh fading channel can be computed as [52]:

$$\begin{aligned}l &= \ln \frac{P(y|x=1)}{P(y|x=-1)} \\ &= \ln \frac{\xi(y)\Phi\left(\frac{y}{\sqrt{2\sigma^2(1+2\sigma^2)}}\right)}{\xi(y)\Phi\left(-\frac{y}{\sqrt{2\sigma^2(1+2\sigma^2)}}\right)},\end{aligned} \tag{20}$$

where

$$\xi(y) = \frac{\sqrt{2}\sigma \exp(-\frac{y^2}{2\sigma^2})}{\sqrt{\pi}(1+2\sigma^2)},$$

$\Phi(h) = 1 + \sqrt{\pi} h e^{h^2} \operatorname{erfc}(-h)$, and $h = \frac{y}{\sqrt{2\sigma^2(1+2\sigma^2)}}$. It can be easily checked that

$$P(-y|x=-1) = \xi(-y)\Phi\left(\frac{y}{\sqrt{2\sigma^2(1+2\sigma^2)}}\right) = P(y|x=1),$$

so the normalized Rayleigh fading channel is symmetric and we can assume the all-zeros codeword is transmitted.

Similar to computing the average list size for the AWGN channel, we need to convert the LLR threshold γ to the corresponding received signal y_γ. However, the explicit form for the inverse of equation (20) is hard to derive, so it is not straightforward to know whether there is a inverse function between the threshold γ and a received signal y_γ. The following theorem shows that equation (20) is bijective.

Theorem 1. *Let $h = \frac{y}{\sqrt{2\sigma^2(1+2\sigma^2)}}$ and the LLR under the normalized Rayleigh fading channel is*

$$\ln \frac{\xi(y)\Phi(h)}{\xi(y)\Phi(-h)} = \ln \frac{\Phi(h)}{\Phi(-h)} = l, \tag{21}$$

and equation (21) *is bijective.*

Proof. The proof is composed of two steps to show equation (21) is bijective. The first step shows there exists a solution h for arbitrary value l (i.e., surjection), and the second step shows that the solution h for the given l is unique (i.e., injection).

Surjection: Given that l is a function of h (equation (21)), at least one h exists for each and every l.

Injection: Instead of analyzing equation (21), it is suffice to

analyze the function after taking the exponential on both sides of the equation (21), and we have

$$\frac{\Phi(h)}{\Phi(-h)} = \exp(l).$$

By taking the derivative with respect to (w.r.t) h, we have

$$\frac{d}{dh}\left(\frac{\Phi(h)}{\Phi(-h)}\right) = \frac{2\sqrt{\pi}\exp(h^2)}{(\sqrt{\pi}\exp(h^2)h\ \mathrm{erfc}(h)-1)^2}, \tag{22}$$

which implies equation (22) > 0 and $\frac{\Phi(h)}{\Phi(-h)}$ is monotonically increasing w.r.t. h. Hence, equation (21) is also monotonically increasing w.r.t. h, and the solution h for the given l is one-to-one.

Conclusion: Equation (21) is bijective. □

Corollary 1.1. *Since equation* (21) *is bijective, equation* (21) *is invertible.*

Since the explicit form for the inverse of the simplified LLR equation (21) is still hard to derive, in this work, the solution h_γ is found by solving the root finding problem given a positive LLR threshold $+\gamma$:

$$\frac{\Phi(h_{+\gamma})}{\Phi(-h_{+\gamma})} - \exp(+\gamma) = 0. \tag{23}$$

The root finding problem can be solved by the Newton's method given the expression in (23) and the derivative (22). Once we obtain the value of $h_{+\gamma}$ using Newton's method, the received signal $y_{+\gamma}$ can be derived by

$$y_{+\gamma} = h_{+\gamma} \times \sqrt{2\sigma^2(1+2\sigma^2)}. \tag{24}$$

The corresponding received signal for the negative LLR threshold $-\gamma$ ($y_{-\gamma}$) can be derived following the same procedure mentioned above. Since we can assume the all-zeros codeword is transmitted, the probability of receiving a LLR with a magnitude less or equal to γ under the normalized Rayleigh fading channel is

$$P(|l| \le \gamma | x = 1) = \int_{y_{-\gamma}}^{y_{+\gamma}} \xi(y)\Phi\left(\frac{y}{\sqrt{2\sigma^2(1+2\sigma^2)}}\right)\mathrm{d}y. \tag{25}$$

The probability δ of using L_{low} can be computed using the same procedure for the AWGN channel.

C. Latency Analysis of the PA List Decoder

Based on the latency model of the hardware architecture of the RPA decoder [40], we analyze the latency caused by the key functional operations used in our proposed list decoder. In order to use the latency model in [40], the following theorem shows that the projection and aggregation function for the s-dimensional subspace can be decomposed into a series of compositions of projection to 1-dimensional subspace. The projection to the 1-dimensional subspace is a functional operation supported by the architecture [40].

Theorem 2. *The projection function* (2) *for the s-dimensional subspace can be decomposed into an s-level composition of the projection to* 1*-dimensional subspaces.*

Proof. For an arbitrary s-dimensional subspace, there are 2^s LLRs in the same coset, and we denote the projection function for the s-dimensional subspace as

$$h_{2^s}(l_1, l_2, ..., l_{2^s}) = 2\tanh^{-1}\left(\prod_{i=1}^{2^s}\tanh\left(\frac{l_i}{2}\right)\right). \tag{26}$$

Equation (26) can be re-grouped as the following:

$$\begin{aligned}
&h_{2^s}(l_1, l_2, ..., l_{2^s}) \\
&= 2\tanh^{-1}\left(\prod_{i=1}^{2^s}\tanh\left(\frac{l_i}{2}\right)\right) \\
&= 2\tanh^{-1}\left(\tanh\left(\frac{1}{2}2\tanh^{-1}\left(\prod_{i=1}^{2^{s-1}}\tanh\left(\frac{l_i}{2}\right)\right)\right)\right. \\
&\left.\tanh\left(\frac{1}{2}2\tanh^{-1}\left(\prod_{i=2^{s-1}+1}^{2^s}\tanh\left(\frac{l_i}{2}\right)\right)\right)\right) \\
&= 2\tanh^{-1}\left(\tanh\left(\frac{1}{2}h_{2^{s-1}}\left(l_1, l_2, ..., l_{2^{s-1}}\right)\right)\right. \\
&\left.\tanh\left(\frac{1}{2}h_{2^{s-1}}\left(l_{2^{s-1}+1}, l_{2^{s-1}+2}, ..., l_{2^s}\right)\right)\right) \\
&= h_2(h_{2^{s-1}}(l_1, l_2, ..., l_{2^{s-1}}), h_{2^{s-1}}(l_{2^{s-1}+1}, l_{2^{s-1}+2}, ..., l_{2^s})),
\end{aligned}$$

where $h_2(A, B) = 2\tanh^{-1}(\tanh(A/2)\tanh(B/2))$ denotes the 1-dimensional projection function. If we repeat the above derivation for $h_{2^{s-1}}(\cdot)$, we have

$$\begin{aligned}
&h_{2^{s-1}}(l_1, l_2, ..., l_{2^{s-1}}) \\
&= h_2(h_{2^{s-2}}(l_1, l_2, ..., l_{2^{s-2}}), \\
&h_{2^{s-2}}(l_{2^{s-2}+1}, l_{2^{s-2}+2}, ..., l_{2^{s-1}})),
\end{aligned}$$

and

$$\begin{aligned}
&h_{2^{s-1}}(l_{2^{s-1}+1}, l_{2^{s-1}+2}, ..., l_{2^s}) \\
&= h_2(h_{2^{s-2}}(l_{2^{s-1}+1}, l_{2^{s-1}+2}, ..., l_{2^{s-1}+2^{s-2}}), \\
&h_{2^{s-2}}(l_{2^{s-1}+2^{s-2}+1}, l_{2^{s-1}+2^{s-2}+2}, ..., l_{2^s})).
\end{aligned}$$

By induction, we have

$$h_{2^s} = h_2(h_2(h_2...), h_2(h_2...)), \tag{27}$$

which is an s-level composition of the 1-dimensional projection function. □

Since $-\tanh^{-1}(x) = \tanh^{-1}(-x)$ and $-1^{\hat{\boldsymbol{y}}_{/\mathbb{B}_i}(T)} \in \{1, -1\}$, the aggregation function can be rewritten as

$$\begin{aligned}
&-1^{\hat{\boldsymbol{y}}_{/\mathbb{B}_i}(T)}\left(2\tanh^{-1}\left(\prod_{z_i\in T\setminus z}\tanh\left(\frac{\boldsymbol{L}(z_i)}{2}\right)\right)\right) \\
&= 2\tanh^{-1}\left(-1^{\hat{\boldsymbol{y}}_{/\mathbb{B}_i}(T)}\prod_{z_i\in T\setminus z}\tanh\left(\frac{\boldsymbol{L}(z_i)}{2}\right)\right) \\
&= 2\tanh^{-1}\left(\tanh\left(\frac{+\infty\times -1^{\hat{\boldsymbol{y}}_{/\mathbb{B}_i}(T)}}{2}\right)\right. \\
&\left.\prod_{z_i\in T\setminus z}\tanh\left(\frac{\boldsymbol{L}(z_i)}{2}\right)\right),
\end{aligned} \tag{28}$$

and there are 2^s $\tanh(\cdot)$ functions inside $\tanh^{-1}(\cdot)$. By Theorem 2, the equation (28) can be reorganized as the composition of the 1-dimensional projection function as well.

Given the new form of the projection function shown in Theorem 2 and the aggregation function (28), the projection unit in [40] can be used to realize the projection and the aggregation function for the CPA decoder with SCH. Hence, the latency of all PA decoders using the structure of the CPA decoder with SCH is

$$t_{(m,r)} = N_{max} * (2 * s * t_{proj} + t_{FHT} + t_{divider}) + \sum_{i=0}^{N_{max}-1} \left(\lceil \frac{1}{P} \lceil \frac{n_{\mathbb{B}}}{d^i_{\text{SCH}}} \rceil \rceil - 1 \right), \quad (29)$$

where $t_{proj} = 1$, $t_{FHT} = 3$, and $t_{divider} = \lceil \log_2(n_{\mathbb{B}}) \rceil$ are the latencies of the projection unit, the FHT decoder, and averaging the aggregated results respectively. The number of processing units implemented for the CPA decoder is denoted as $P \in \{2^p, p \in \{0, 1, ..., m\}\}$. The latency is measured by the number of clock cycles.

Compared to the RPA implementation in [40], the latency caused by inserting input to the $(r-1)$-level and the repeated use of divider in the $(r-1)$-level are removed because of the non-recursive structure of the CPA/PCPA decoder. Since the latency of the projection unit and the aggregation unit is the same in [40], the latency of the aggregation function used by the CPA decoder is the same as the latency of the aggregation function used by the RPA implementation in [40]. The latency caused by the input/output register is omitted in this analysis because the latency caused by the decoding process is the main focus of this work.

The latency of the list decoder for the CPA decoder with SCH is

$$tl_{(m,r)} = m + \lceil \frac{L}{P_{\text{CPA}}} \rceil * (t_{(m,r)} + 1) + \log_2(L), \quad (30)$$

where m clock cycles are used by the bitonic sorter [53] to construct the Chase-type list decoding, P_{CPA} is the number of the CPA decoder built inside the list decoder, the one extra cycle added to $t_{(m,r)}$ is used to perform the syndrome check and compute the posterior probability in parallel, and $\log_2(L)$ cycles are used to select the codeword with the largest posterior probability in the list when using a single-elimination tournament implementation.

VII. Experimental Results

A. Experimental Setup

Experiments are performed over the BPSK modulation and the AWGN channel while only experiments in Fig.4 are performed over the normalized Rayleigh fading channel. The Opt. PCPA decoder, which is implemented according to [29], [31], with the saturation early stopping (4) is extended to list decoders. For the Opt. PCPA decoder, 128 subspaces are used for RM codes with $m = 7$, and 256 subspaces are used for RM codes with $m = 8$. The maximum number of iterations is $N_{max} = \lceil \frac{m}{2} \rceil$. The parameter of SCH ($d_{\text{SCH}}$) is set to 2, and the syndrome check frequency δ is selected to achieve

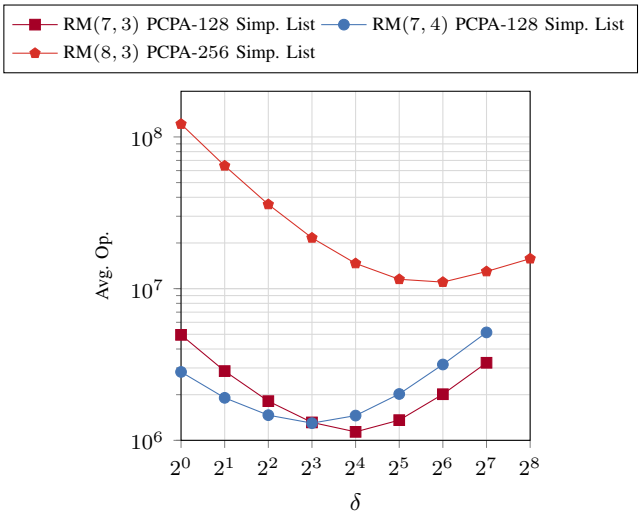


Fig. 2. Average number of required operations for RM$(7,3)$, RM$(8,3)$, and RM$(7,4)$ measured at $E_b/N_0 = 3.5$ dB, 2.5 dB, and 5 dB respectively.

TABLE III
Parameters for IDA and M-IDA decoding.

	RM$(7,3)$		RM$(7,4)$		RM$(8,3)$	
	IDA	M-IDA	IDA	M-IDA	IDA	M-IDA
γ	4.5	0.2	4	1	4	0.05
φ	66	–	14	–	190	–

the minimum average complexity measured by the weighted counting rule shown in Table I for the list decoder. From Fig. 2, the minimum average complexity is achieved at $\delta = 16$, 64, and 8 for the RM$(7,3)$, the RM$(8,3)$, and the RM$(7,4)$ codes respectively at selective E_b/N_0s. The ML lower bounds, computed by the method used in [15], are provided as a reference for simulated RM codes, and they are computed using our list decoder without IDA/M-IDA (Simp. list) decoder with a list size of 16.

From Fig. 3 and 4, computed average list sizes (Ana.) match average list sizes returned from simulations (Sim.) under both the AWGN and the normalized Rayleigh fading channel. Since the decoding performance of list decoders with IDA can always be improved by tightening the parameter settings, a fair comparison between list decoders with and without IDA should conduct under a fixed complexity reduction. The parameters of IDA and M-IDA decoding are searched in the range shown in Fig. 3 such that a minimum degradation is incurred at a selected E_b/N_0 (3.5 dB for RM$(7,3)$ code, 2.5 dB for RM$(8,3)$ code, and 5 dB for RM$(7,4)$ code) given a 30% average list size reduction. These parameters are shown in Table III.

B. Results of Proposed List Decoders

From Fig. 5 (b), (e), and (h), it can be observed that the average list size can be reduced to around 11 at 3.5 dB for RM$(7,3)$ codes, 2.5 dB for RM$(8,3)$ codes, and 5 dB for RM$(7,4)$ codes, which is roughly a 30% reduction in the average list size. From Fig. 5 (a), (d), and (g), all decoders

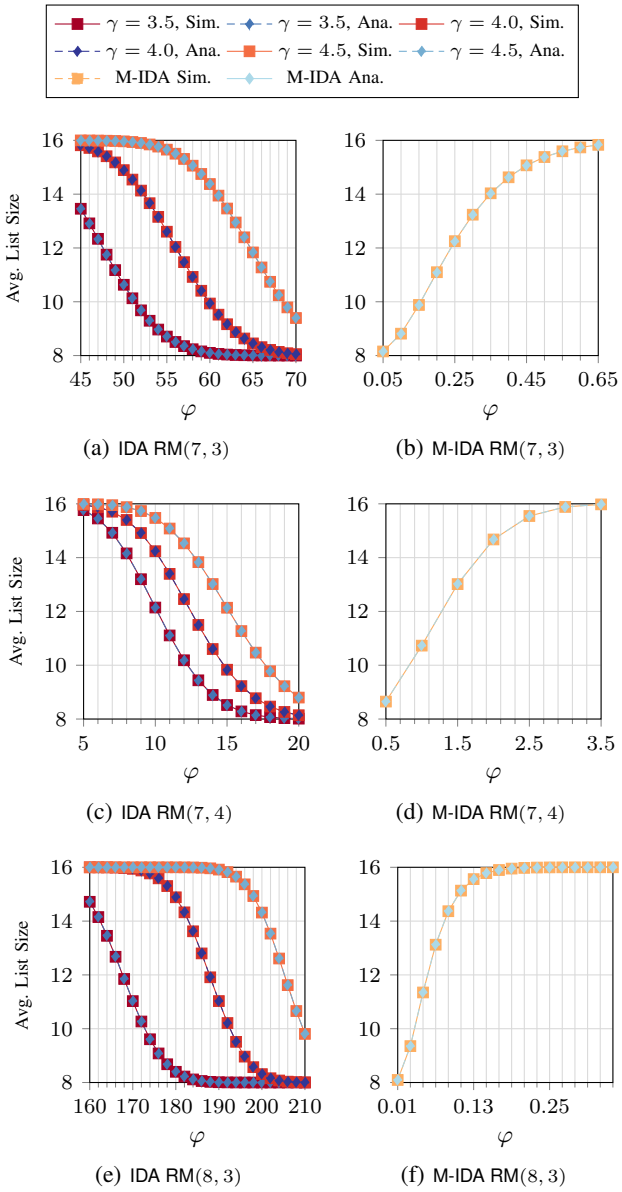


Fig. 3. $\bar{L}$ of IDA-Simp. List and M-IDA-Simp. List for RM$(7,3)$, RM$(8,3)$, and RM$(7,4)$ codes at $E_b/N_0 = 3.5$ dB, 2.5 dB, and 5.0 dB respectively.

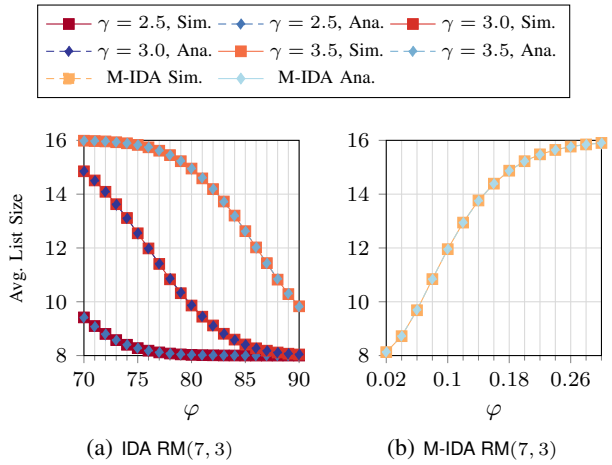


Fig. 4. $\bar{L}$ of IDA-Simp. List and M-IDA-Simp. List for the RM$(7,3)$ code at $E_b/N_0 = 3.5$ dB under the normalized Rayleigh fading channel.

with IDA or M-IDA decoding have less than 0.1 dB loss in decoding performance at a FER of 10^{-4}.

C. Comparisons with the AED-BP Decoder

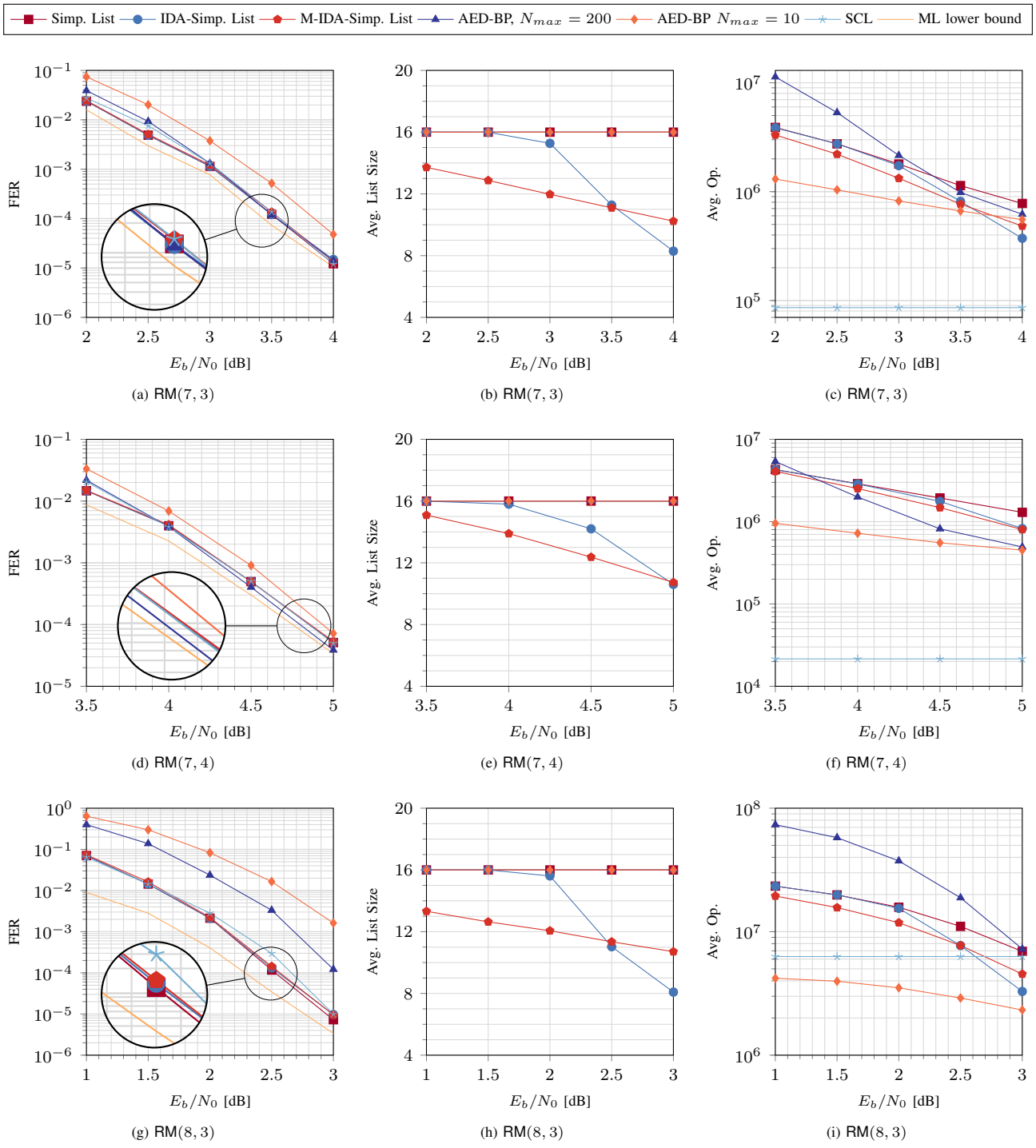
Our proposed reduced-complexity PA list decoders are compared with the state-of-the-art AED-BP decoder with a maximum number of iterations of 200 and the reduced factor graph [34]. The number of ensembles used by the AED-BP is 16. The AED-BP with $N_{max} = 200$ is used to compare the complexity with the proposed decoders. The AED-BP with $N_{max} = 10$ is used to compare the latency, which will be explained in the later section. Fig. 5 (c), (f), and (i) show the average number of operations using weights shown in Table I. It can be observed that, for RM$(7,3)$ and RM$(8,3)$ codes, the complexity of proposed list decoders is smaller than the AED-BP decoder at $E_b/N_0 = 3.5$ dB and $E_b/N_0 = 2.5$ dB respectively. For RM$(7,4)$ codes, the AED-BP decoder requires fewer operations than list decoders at $E_b/N_0 = 5.0$ dB. From Fig.5 (g), the proposed list decoder has 0.4 dB gain compared to the AED-BP decoder at a FER of 2×10^{-4} when decoding the RM$(8,3)$ code.

Besides comparing the average number of operations, we investigate the number of operations in the worst-case scenario (decoding without early stopping mechanisms) in Fig. 6. When extended to the list decoder, the PCPA decoder with 128 subspaces and $d_{\text{SCH}} = 2$ is used for $m = 7$ RM codes, and the PCPA decoder with 256 subspaces and $d_{\text{SCH}} = 2$ is used for the RM$(8,3)$ code. The BP decoder with 200 iterations and the reduced factor graph are used in the AED. The list size for the simplified list decoder is 16, and 16 ensembles are used in the AED. It can be observed that, in the worst-case scenario, the simplified list decoder uses $6.8\times$ fewer operations compared to AED-BP when decoding RM$(7,3)$ codes, $3.5\times$ fewer operations compared to AED-BP when decoding RM$(8,3)$ codes, and uses $4.7\times$ fewer operations compared to AED-BP when decoding RM$(7,4)$ codes.

D. Comparisons with the SCL Decoder

The complexity of the SCL decoder is counted using the same rule in [54], which counts the operations of the f and g functions, and the number of XORs for the partial sum. In [54], the f function takes the min-sum form, which requires three operations (two sign changes, and one operation for finding the minimum), and the g function takes two operations (one sign change and one addition).

List sizes of SCL decoders are tuned to achieve similar decoding performance as the Simp. List decoder. From Fig. 5, SCL decoders with list sizes of 32, 1024, and 8 can approach similar decoding performance to the Simp. List decoders for RM$(7,3)$, RM$(8,3)$, and RM$(7,4)$ codes respectively. It can be observed that SCL decoders use an order of magnitude lower numbers of operations compared to Simp. List decoders and AED-BP decoders when decoding RM codes with rates ≥ 0.5 (RM$(7,3)$ and RM$(7,4)$ codes). When decoding low-rate RM codes such as the RM$(8,3)$ code, the proposed list



(a) RM$(7, 3)$ (b) RM$(7, 3)$ (c) RM$(7, 3)$

(d) RM$(7, 4)$ (e) RM$(7, 4)$ (f) RM$(7, 4)$

(g) RM$(8, 3)$ (h) RM$(8, 3)$ (i) RM$(8, 3)$

Fig. 5. FERs, average list sizes, and average numbers of operations of list decoders. Proposed list decoders (without IDA or M-IDA) and the AED-BP use a list/ensemble size of 16. The SCL decoder uses list sizes of 32, 1024, and 8 for decoding RM$(7, 3)$, RM$(8, 3)$, and RM$(7, 4)$ codes respectively.

TABLE IV
THE E_b/N_0 REACHING A FER$= 10^{-4}$, THE N_{max} FOR AED-BP AND PCPA$_{\text{SCH}}$ LIST DECODERS, THE LIST SIZE L FOR THE SCL DECODER, AND THE LATENCY ESTIMATION IN THE NUMBER OF CLOCK CYCLES BASED ON EXISTING HARDWARE ARCHITECTURES. THE NOTATION $-$ MEANS THE DECODER DOES NOT REACH THE FER$= 10^{-4}$ WITHIN THE RANGE OF THE SIMULATION.

RM Codes	AED-BP [56]			AED-BP [56]			SR-List [41]			PCPA$_{\text{SCH}}$ List		
	E_b/N_0 [dB]	N_{max}	Latency	E_b/N_0 [dB]	N_{max}	Latency	E_b/N_0 [dB]	L	Latency	E_b/N_0 [dB]	N_{max}	Latency
RM$(7,3)$	3.55	200	1207	3.85	10	67	3.55	32	86	3.55	4	68
RM$(7,4)$	4.80	200	1207	4.92	10	67	4.85	8	88	4.85	4	76
RM$(8,3)$	-	200	1407	-	10	77	2.66	1024	121	2.53	4	73

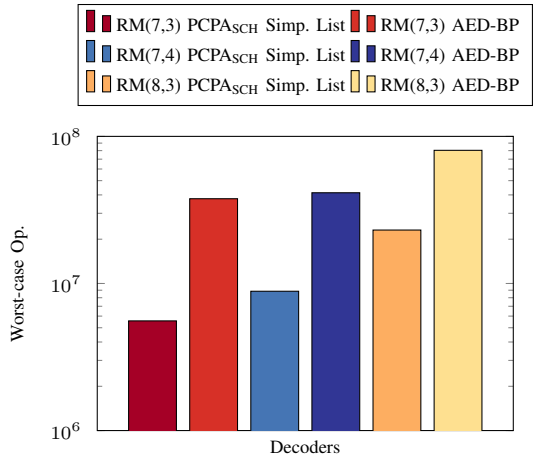


Fig. 6. Comparisons of the number of operations in the worst-case scenario (no early stopping) between the simplified list decoder extended from the PCPA decoder with $d_{\text{SCH}} = 2$ and the AED-BP with 200 iterations. 128 and 256 subspaces are used for $m = 7$ and $m = 8$ RM codes respectively. The list size and the number of the ensemble are 16.

decoders require a similar or smaller number of operations to the SCL decoder at a high E_b/N_0 (e.g., $\geq$ 2.5 dB) region that is interested by the actual applications such as the ultra-reliable low-latency communication [55]. However, a list size of 1024 is required by the SCL decoder to achieve a similar decoding performance to the Simp. List decoder, which causes a large decoding latency due to the serial nature of the SCL decoders.

E. Latency Analysis

The latency of the SCL decoder can be characterized into two parts [57], [58]: I), Latency induced by decoding; II), Latency induced by the sorting process. The decoding latency is modeled by the state-of-the-art low-latency hardware architecture for the node-based SCL decoder, the sequential repetition (SR) list (SR-List) decoder [41]. The latency of performing the f function, the g function, and the partial sum for the SC decoder is one clock cycle. The sorting process is embedded into the computation inside the special nodes [41].

The latency of the AED-BP is estimated based on the latency model of the hardware architecture for the BP list (BPL) decoder [56]. Based on the hardware architecture for the BPL decoder [56], the latency is composed of three parts: the permutation and the inverse permutation, the computation of the likelihood of returned codeword (one clock cycle), and the decoding latency ($N_{max}(m-1)$ clock cycles). The architecture in [56] uses a sequential implementation, and it does not use the likelihood to select the final codeword. In this work, we assume that the likelihood is used to select the final codeword, and $\log_2(L)$ cycles are need to select the codeword with the largest likelihood among the ensemble when using a single-elimination tournament implementation. Since the permutation unit in [56] is for the factor graph permutation, we assume a specific hardware unit is implemented for each permutation for the AED-BP, and this unit uses one clock cycle for both the permutation and the inverse permutation.

In this latency analysis, full parallelism is assumed on the AED-BP, the PCPA list decoder, and the SR-list decoder. Table IV shows the latency estimation when decoding RM$(7,3)$, RM$(8,3)$, and RM$(7,4)$ codes. Under the fully parallel implementation, the proposed PCPA list decoder has lower latency than the SCL decoder and the AED-BP decoder with $N_{max} = 200$. In actual implementation, a small N_{max} is given to the BP decoder. We found that the AED-BP decoder has a similar latency to the proposed list decoder when $N_{max} = 10$. However, compared to other decoders, Fig. 5 and Table IV show that the degradation in the decoding performance will occur when setting $N_{max} = 10$. Hence, we conclude that the proposed list decoder requires a smaller decoding latency to reach the near ML decoding performance.

VIII. CONCLUSION

In this work, we propose complexity-reduction techniques for the PA list decoder. A redesigned syndrome check pattern and the SCH are first applied to the CPA/PCPA decoder. The CPA/PCPA decoder with the redesigned syndrome check pattern can return the syndrome check result alongside the returned codeword. To reduce the average computational complexity of the list decoder, we use the recently proposed IDA decoding as a pre-possessing tool that adaptively determines the list size based on the received soft information. We give an analytical derivation to the average list size of list decoders with IDA, and analytically derived results match the empirical results. For list decoders with IDA decoding, the average list size can be reduced by 30% with less than 0.1 dB decoding performance loss at a FER of around 10^{-4}. The proposed reduced-complexity list decoders requires a smaller average number of operations than the AED-BP decoder when decoding RM$(7,3)$ and RM$(8,3)$ codes. In the worst-case scenario, the simplified list decoder uses a smaller number of operations compared to the AED-BP decoder when decoding RM$(7,3)$, RM$(8,3)$, and RM$(7,4)$ codes. Proposed list decoders require

a smaller latency than the AED-BP and the SCL decoder to reach near ML decoding performance.

ACKNOWLEDGMENT

The authors would like to thank Marzieh Hashemipour-Nazari for the insight of calculating the number of FHT decoding used in the RPA decoder with multi-factor pruning.

REFERENCES

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

[2] E. Arikan, "A survey of Reed-Muller codes from polar coding perspective," in *2010 IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*, 2010, pp. 1–5.

[3] ——, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inform. Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[4] "TS 38.212 NR; multiplexing and channel coding V17.1.0," 3GPP, Technical Specification (TS), Mar. 2022.

[5] S. Kudekar, S. Kumar, M. Mondelli, H. D. Pfister, E. Şaşoğlu, and R. L. Urbanke, "Reed–Muller codes achieve capacity on erasure channels," *IEEE Trans. Inform. Theory*, vol. 63, no. 7, pp. 4298–4316, 2017.

[6] E. Abbe, A. Shpilka, and A. Wigderson, "Reed–Muller codes for random erasures and errors," *IEEE Trans. Inform. Theory*, vol. 61, no. 10, pp. 5229–5252, 2015.

[7] O. Sberlo and A. Shpilka, "On the performance of Reed-Muller codes with respect to random errors and erasures," in *Proc. of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '20. USA: Society for Industrial and Applied Mathematics, 2020, p. 1357–1376.

[8] G. Reeves and H. D. Pfister, "Reed–Muller codes on BMS channels achieve vanishing bit-error probability for all rates below capacity," *IEEE Trans. Inform. Theory*, pp. 1–1, 2023.

[9] E. Abbe and C. Sandon, "A proof that Reed-Muller codes achieve Shannon capacity on symmetric channels," in *IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, 2023, pp. 177–193.

[10] E. Abbe and M. Ye, "Reed-Muller codes polarize," *IEEE Trans. Inform. Theory*, vol. 66, no. 12, pp. 7311–7332, 2020.

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

[12] E. Arikan, H. Kim, G. Markarian, U. Ozgiir, and E. Poyraz, "Performance of short polar codes under ML decoding," in *Proc. ICT Mobile Summit 2009, (Santander, Spain)*, 10-12 June 2009.

[13] R. R. Green, "A serial orthogonal decoder," *JPL Space Programs Summary*, vol. 37, pp. 247–253, 1966.

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

[15] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inform. Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

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

[17] N. Stolte, U. Sorger, and G. Sessler, "Sequential stack decoding of binary Reed-Muller codes," *ITG FACHBERICHT*, pp. 63–70, 2000.

[18] N. Stolte and U. Sorger, ""Look-Ahead" soft-decision decoding of binary Reed-Muller codes," in *International Symposium on Information Theory and Its Applications, Honolulu, HA, Nov.*, 2000.

[19] K. Niu and K. Chen, "Stack decoding of polar codes," *Electronics letters*, vol. 48, no. 12, pp. 695–697, 2012.

[20] ——, "CRC-aided decoding of polar codes," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1668–1671, 2012.

[21] V. Miloslavskaya and P. Trifonov, "Sequential decoding of polar codes," *IEEE Commun. Lett.*, vol. 18, no. 7, pp. 1127–1130, 2014.

[22] M.-O. Jeong and S.-N. Hong, "SC-Fano decoding of polar codes," *IEEE Access*, vol. 7, pp. 81 682–81 690, 2019.

[23] E. Arıkan, "From sequential decoding to channel polarization and back again," *arXiv preprint arXiv:1908.09594*, 2019.

[24] P. Yuan and M. C. Coşkun, "Complexity-adaptive maximum-likelihood decoding of modified GN-coset codes," in *IEEE Information Theory Workshop (ITW)*, 2021, pp. 1–6.

[25] S. A. Hashemi, N. Doan, W. J. Gross, J. Cioffi, and A. Goldsmith, "A tree search approach for maximum-likelihood decoding of Reed-Muller codes," in *IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–6.

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

[27] J. Li, S. M. Abbas, T. Tonnellier, and W. J. Gross, "Reduced complexity RPA decoder for Reed-Muller codes," in *11th International Symposium on Topics in Coding (ISTC)*, 2021, pp. 1–5.

[28] D. Fathollahi, N. Farsad, S. A. Hashemi, and M. Mondelli, "Sparse multi-decoder recursive projection aggregation for Reed-Muller codes," in *IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 1082–1087.

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

[30] Q. Huang and B. Zhang, "Pruned collapsed projection-aggregation decoding of Reed-Muller codes," *CoRR*, vol. abs/2105.11878, 2021. [Online]. Available: https://arxiv.org/abs/2105.11878

[31] J. Li and W. J. Gross, "Optimization and simplification of PCPA decoder for Reed-Muller codes," *IEEE Commun. Lett.*, vol. 26, no. 6, pp. 1206–1210, 2022.

[32] C. Condo, "Input-distribution-aware successive cancellation list decoding of polar codes," *IEEE Commun. Lett.*, vol. 25, no. 5, pp. 1510–1514, 2021.

[33] C. Condo and A. Nicolescu, "Input-distribution-aware parallel decoding of block codes," in *11th International Symposium on Topics in Coding (ISTC)*, 2021, pp. 1–5.

[34] M. Geiselhart, A. Elkelesh, M. Ebada, S. Cammerer, and S. ten Brink, "Automorphism ensemble decoding of Reed–Muller codes," *IEEE Trans. Commun.*, vol. 69, no. 10, pp. 6424–6438, 2021.

[35] M. C. Coşkun and H. D. Pfister, "An information-theoretic perspective on successive cancellation list decoding and polar code design," *IEEE Trans. Inform. Theory*, vol. 68, no. 9, pp. 5779–5791, 2022.

[36] B. Li, H. Shen, and D. Tse, "An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check," *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 2044–2047, 2012.

[37] M. Rowshan, A. Burg, and E. Viterbo, "Polarization-adjusted convolutional (PAC) codes: Sequential decoding vs list decoding," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1434–1447, 2021.

[38] A. Mozammel, "Hardware implementation of Fano decoder for polarization-adjusted convolutional (PAC) codes," *IEEE Trans. Circuits Syst. II*, vol. 69, no. 3, pp. 1632–1636, 2022.

[39] M. Ebada, S. Cammerer, A. Elkelesh, M. Geiselhart, and S. t. Brink, "Iterative detection and decoding of finite-length polar codes in Gaussian multiple access channels," in *54th Asilomar Conference on Signals, Systems, and Computers*, 2020, pp. 683–688.

[40] M. Hashemipour-Nazari, Y. Ren, K. Goossens, and A. Balatsoukas-Stimming, "Pipelined architecture for soft-decision iterative projection aggregation decoding for RM codes," *IEEE Trans. Circuits Syst. I*, pp. 1–0, 2023.

[41] Y. Ren, A. T. Kristensen, Y. Shen, A. Balatsoukas-Stimming, C. Zhang, and A. Burg, "A sequence repetition node-based successive cancellation list decoder for 5G polar codes: Algorithm and implementation," *IEEE Trans. Signal Processing*, vol. 70, pp. 5592–5607, 2022.

[42] E. Abbe, A. Shpilka, and M. Ye, "Reed–Muller codes: Theory and algorithms," *IEEE Trans. Inform. Theory*, vol. 67, no. 6, pp. 3251–3277, 2021.

[43] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, 2005.

[44] C. Bliek, P. Bonami, and A. Lodi, "Solving mixed-integer quadratic programming problems with IBM-CPLEX: a progress report," in *Proceedings of the Twenty-Sixth RAMP Symposium*, Hosei University, Tokyo, Oct. 16-17 2014.

[45] T. Clevorn and P. Vary, "The box-minus operator and its application to low-complexity belief propagation decoding," in *IEEE 61st Vehicular Technology Conference*, vol. 1, 2005, pp. 687–691 Vol. 1.

[46] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. USA: Wiley-Interscience, 2005.

[47] E. Arikan, "Systematic polar coding," *IEEE Commun. Lett.*, vol. 15, no. 8, pp. 860–862, 2011.

[48] L. Li and W. Zhang, "On the encoding complexity of systematic polar codes," in *28th IEEE International System-on-Chip Conference (SOCC)*, 2015, pp. 415–420.

[49] M. Hashemipour-Nazari, K. Goossens, and A. Balatsoukas-Stimming, "Multi-factor pruning for recursive projection-aggregation decoding of RM codes," in *IEEE Workshop on Signal Processing Systems (SiPS)*, 2022, pp. 1–6.

[50] K. L. Chung, *A course in probability theory*. Academic press, 2001.

[51] R. Asvadi, A. H. Banihashemi, M. Ahmadian-Attari, and H. Saeedi, "LLR approximation for wireless channels based on Taylor series and its application to BICM with LDPC codes," *IEEE Trans. Commun.*, vol. 60, no. 5, pp. 1226–1236, 2012.

[52] R. Yazdani and M. Ardakani, "Linear LLR approximation for iterative decoding on wireless channels," *IEEE Trans. Commun.*, vol. 57, no. 11, pp. 3278–3287, 2009.

[53] K. E. Batcher, "Sorting networks and their applications," in *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, 1968, pp. 307–314.

[54] H. Zhou, W. Song, W. J. Gross, Z. Zhang, X. You, and C. Zhang, "An efficient software stack sphere decoder for polar codes," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1257–1266, 2020.

[55] M. Shirvanimoghaddam, M. S. Mohammadi, R. Abbas, A. Minja, C. Yue, B. Matuz, G. Han, Z. Lin, W. Liu, Y. Li, S. Johnson, and B. Vucetic, "Short block-length codes for ultra-reliable low latency communications," *IEEE Commun. Mag.*, vol. 57, no. 2, pp. 130–137, 2019.

[56] Y. Ren, Y. Shen, L. Zhang, A. T. Kristensen, A. Balatsoukas-Stimming, E. Boutillon, A. Burg, and C. Zhang, "High-throughput and flexible belief propagation list decoder for polar codes," *IEEE Trans. Signal Processing*, vol. 72, pp. 1158–1174, 2024.

[57] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Trans. Signal Processing*, vol. 63, no. 19, pp. 5165–5179, 2015.

[58] Y. Tao, S.-G. Cho, and Z. Zhang, "A configurable successive-cancellation list polar decoder using split-tree architecture," *IEEE J. Solid-State Circuits*, vol. 56, no. 2, pp. 612–623, 2021.