

# A Tree Search Approach for Maximum-Likelihood Decoding of Reed-Muller Codes

Seyyed Ali Hashemi\*, Nghia Doan<sup>†</sup>, Warren J. Gross<sup>†</sup>, John Cioffi\*, Andrea Goldsmith<sup>‡</sup>

\*Department of Electrical Engineering, Stanford University, USA

<sup>†</sup>Department of Electrical and Computer Engineering, McGill University, Canada

<sup>‡</sup>Department of Electrical and Computer Engineering, Princeton University, USA

ahashemi@stanford.edu, nghia.doan@mail.mcgill.ca, warren.gross@mcgill.ca, cioffi@stanford.edu, goldsmith@princeton.edu

**Abstract**—A low-complexity tree search approach is presented that achieves the maximum-likelihood (ML) decoding performance of Reed-Muller (RM) codes. The proposed approach generates a bit-flipping tree that is traversed to find the ML decoding result by performing successive-cancellation decoding after each node visit. A depth-first search (DFS) and a breadth-first search (BFS) scheme are developed and a log-likelihood-ratio-based bit-flipping metric is utilized to avoid redundant node visits in the tree. Several enhancements to the proposed algorithm are presented to further reduce the number of node visits. Simulation results confirm that the BFS scheme provides a lower average number of node visits than the existing tree search approach to decode RM codes.

## I. INTRODUCTION

The Reed-Muller (RM) code family [1], [2] is one of the oldest channel coding schemes. It has regained attention recently due to its similarity with polar codes [3]. Both RM and polar codes are constructed by selecting rows of a generator matrix. The row selection for RM codes is such that the minimum distance among all the codewords is maximized, while the row selection for polar codes minimizes the error probability when the low-complexity successive cancellation (SC) decoder is used. Unlike polar codes, RM codes have the advantage of channel-independent code construction that is particularly beneficial when they are used in rapidly changing environments, e.g. in mmWave and THz bands, which are being considered for 5G and other wireless systems. In addition, RM codes can achieve the capacity of binary erasure channels under maximum-likelihood (ML) decoding [4]. Since ML decoding is generally impractical, RM codes are often decoded using decoders that can approach ML decoding performance, such as recursive list decoding [5], decoding on several factor graph permutations [6]–[8], decoding with minimum-weight parity-checks [9], and recursive projection-aggregation decoding [10], [11].

SC decoding was first used to decode RM codes in [12]. However, the error-correction performance of RM codes under SC decoding is far from satisfactory. It was observed in [13] that the resulting error probability of each information bit under SC decoding can be used to perform a bit-flipping tree search to develop a SC ordered search (SCOS) decoding algorithm that achieves ML decoding performance with adaptive complexity. The advantage of the SCOS decoding algorithm over the bit-flipping decoding algorithms for polar

codes [14], [15] is that it does not require a cyclic redundancy check (CRC) to determine if the decoding has succeeded. However, the tree search algorithm in [13] relies on two costly computations: the calculation of a metric that has to be performed using empirical methods; and the sorting of the metrics to determine which branch of the tree is explored next.

In this paper, a modified tree search decoding algorithm is presented that, unlike the SCOS decoder in [13], does not require the empirical calculation of a metric, and does not need the sorting operation to determine the next branch traversal. A bit-flipping tree is developed and two tree search algorithms are used to find the ML decoding result: a depth-first search (DFS) approach that has a simple recursive structure; and a breadth-first search (BFS) approach that obtains a lower average decoding complexity than the DFS method. The key idea in the proposed method is to follow the natural bit index order rather than sorting the bit error probabilities. Simulation results show that the proposed methods achieve ML decoding performance while the BFS method requires fewer tree node visits than the SCOS decoder. To further reduce the average and worst-case decoding complexity of the DFS and the BFS schemes, several enhancements are proposed: 1) the depth of the tree is limited to a certain predefined value that provides near-ML decoding performance while significantly reducing the average and worst-case decoding complexity; 2) redundant node visits are eliminated to reduce the worst-case decoding complexity; and 3) a metric is utilized to order the node visits that reduces the average decoding complexity.

## II. PRELIMINARIES

### A. Reed-Muller Codes

A RM code of length  $N = 2^m$  and order  $r$ , ( $0 \leq r \leq m$ ) is denoted as  $\mathcal{RM}(r, m)$ . The dimension (number of information bits) of  $\mathcal{RM}(r, m)$  is  $K = \sum_{i=0}^r \binom{m}{i}$  and it has a minimum distance  $d = 2^{m-r}$ . RM codes can be constructed by converting the input bits  $\mathbf{u} = \{u_0, u_1, \dots, u_{N-1}\}$  to the encoded bits  $\mathbf{x} = \{x_0, x_1, \dots, x_{N-1}\}$  by a linear transformation as  $\mathbf{x} = \mathbf{u}\mathbf{G}^{\otimes m}$ , where  $\mathbf{G}^{\otimes m}$  is the  $m$ -th Kronecker power of the matrix  $\mathbf{G} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$  [16]. The construction of RM codes involves generating two sets based on  $w_i$ , the weight of the  $i$ -th row of  $\mathbf{G}^{\otimes m}$ : a frozen set  $\mathcal{F} = \{i | 0 \leq i < N, w_i < d\}$ , where  $u_i = 0 \ \forall i \in \mathcal{F}$ ; and an information set  $\mathcal{I} = \{i | 0 \leq i < N, w_i \geq d\}$ , where  $u_i \ \forall i \in \mathcal{I}$  carries the information bits.

Both of the sets  $\mathcal{I}$  and  $\mathcal{F}$  are known to the transmitter and the receiver.

This paper considers binary phase-shift keying (BPSK) modulation on the vector  $\mathbf{x}$  and an additive white Gaussian noise (AWGN) channel model. Thus the received signal is  $\mathbf{y} = (\mathbf{1} - 2\mathbf{x}) + \mathbf{z}$ , where  $\mathbf{1}$  is an all-one vector of size  $N$ , and  $\mathbf{z} \in \mathbb{R}^N$  is the Gaussian noise vector with zero mean and variance  $\sigma^2$ . The log-likelihood ratio (LLR) values associated with the received vector  $\mathbf{y}$  can be calculated as  $\alpha_m = \ln \frac{\mathbb{P}(\mathbf{x}=0|\mathbf{y})}{\mathbb{P}(\mathbf{x}=1|\mathbf{y})} = \frac{2\mathbf{y}}{\sigma^2}$ .

### B. Successive-Cancellation Decoding and Its Variants

SC decoding follows a bit-by-bit decoding schedule. Each bit  $u_i$  is estimated as the ML decision based on  $\alpha_m$ , considering all of the previous bits  $u_0, \dots, u_{i-1}$  are decoded correctly and all of the future bits  $u_{i+1}, \dots, u_{N-1}$  are unknown. If  $i \in \mathcal{F}$ , then  $\hat{u}_i$ , the estimation of  $u_i$ , is set to 0. Otherwise, the LLR value of  $u_i$  is calculated as

$$\alpha_i = \ln \frac{\mathbb{P}(u_i = 0 | \alpha_m, \hat{u}_0, \dots, \hat{u}_{i-1})}{\mathbb{P}(u_i = 1 | \alpha_m, \hat{u}_0, \dots, \hat{u}_{i-1})}, \quad (1)$$

and  $u_i$  is estimated as 0 if  $\alpha_i \geq 0$ , and as 1 otherwise. To reduce the complexity in the implementation of the SC decoding algorithm, the min-sum approximation [17] is used to carry out (1).

SC decoding is well-suited to work with polar codes of large lengths. However, since SC decoding does not use any available information about the future bits, its error-correction performance on polar codes of moderate lengths and RM codes is not satisfactory. In fact, it was shown in [3] that RM codes are asymptotically unreliable under SC decoding. Therefore, variants of SC decoding such as SC list (SCL) decoding and SC flip (SCF) decoding are used for polar codes and RM codes [5], [14], [15], [18].

SCL decoding is based on running multiple SC decoders in parallel by estimating each information bit as either 0 or 1. To avoid exponential growth in the complexity of SCL decoding, at each bit estimation step, only the  $L$  most reliable candidate paths are kept in a list. Therefore, SCL decoding with list size  $L = 2^K$  guarantees ML decoding performance. A path metric is used to select the  $L$  most reliable candidates at each step that can be well approximated as [19]

$$\text{PM}_i = \sum_{j=0}^i \beta_j |\alpha_j|, \quad (2)$$

where  $\alpha_i$  is the LLR value of  $u_i$  and

$$\beta_i = \begin{cases} 1 & \text{if } \hat{u}_i \neq \frac{1 - \text{sgn}(\alpha_i)}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

When  $u_{N-1}$  is decoded, the path with the smallest path metric is selected as the decoding result.

SCF decoding is developed for polar codes and it relies on an outer CRC code to detect if the initial SC decoding attempt succeeds or not. In case of a failure, the SCF decoder sequentially flips one bit at a time and performs SC decoding

in an attempt to pass the CRC test. In [14], the absolute value of the LLR value for each  $u_i$  is selected as a metric to determine the bits that need to be flipped. It was shown in [15] that a dynamic SCF decoding can perform multiple bit-flips at a time. An enhanced bit-flip selection metric based on the LLR value of previous bits is developed that improved the performance of dynamic SCF decoding. However, the dependence of SCF decoding on a CRC renders it unattractive for decoding RM codes.

The SCOS decoding [13] achieves the ML decoding performance by following a sequential schedule that starts with a round of SC decoding. The result of SC decoding is used as a potential ML candidate and the path metric associated with this candidate is stored as  $\text{PM}_{\text{best}}$  using (2). Then, the estimated values of the information bits  $\hat{u}_i$ ,  $i \in \mathcal{I}$ , are flipped as  $\hat{u}_i \oplus 1$  and their path metrics are calculated considering all the previous bit estimations are unchanged. Then SC decoding is performed to generate a new candidate. If the path metric of the new candidate is smaller than  $\text{PM}_{\text{best}}$ , the candidate is stored as the potential ML candidate and  $\text{PM}_{\text{best}}$  is updated. To reduce the number of bit-flips, SCOS decoding uses the fact that the path metric calculation in (2) is non-decreasing in  $i$ . Therefore, if the path metric associated with a bit-flip is larger than  $\text{PM}_{\text{best}}$ , that bit-flip and any subsequent bit-flips cannot result in the ML candidate. SCOS decoding thus creates a list of candidates whose path metrics are less than  $\text{PM}_{\text{best}}$ . In order to determine which bit is flipped next in the list of candidates, a score parameter is calculated for each  $\hat{u}_i$  as

$$S_i = \text{PM}_i + \sum_{j=0}^i \ln(1 - p_j), \quad (3)$$

where  $p_j$  is the probability that in SC decoding, the first error occurs in  $u_j$ . After each SC decoding, the candidates in the list are sorted based on their score and the bit with the minimum score is flipped. To provide a complexity-performance trade-off, SCOS decoding allows for limiting the list size at the cost of no longer guaranteeing ML decoding performance.

There are two main issues with SCOS decoding. First, the calculation of the score in (3) depends on estimating  $p_j$  that is approximated either by massive Monte-Carlo simulations or by density evolution. This is especially cumbersome in rapidly changing environments where the score must be updated frequently. Second, the sorting of scores is a computationally intensive task, especially when the list size increases. The next section presents a tree search approach to tackle SCOS decoding issues.

## III. THE TREE SEARCH APPROACH

### A. Proposed Algorithm

The tree search approach relies on the generation of a bit-flipping tree as shown in Fig. 1. Each node in the tree represents a SC decoding attempt after a bit-flipping is performed on the result of the SC decoder in its parent node. The proposed decoding scheme starts by an initial SC decoding attempt, which is represented as the root node in the tree.

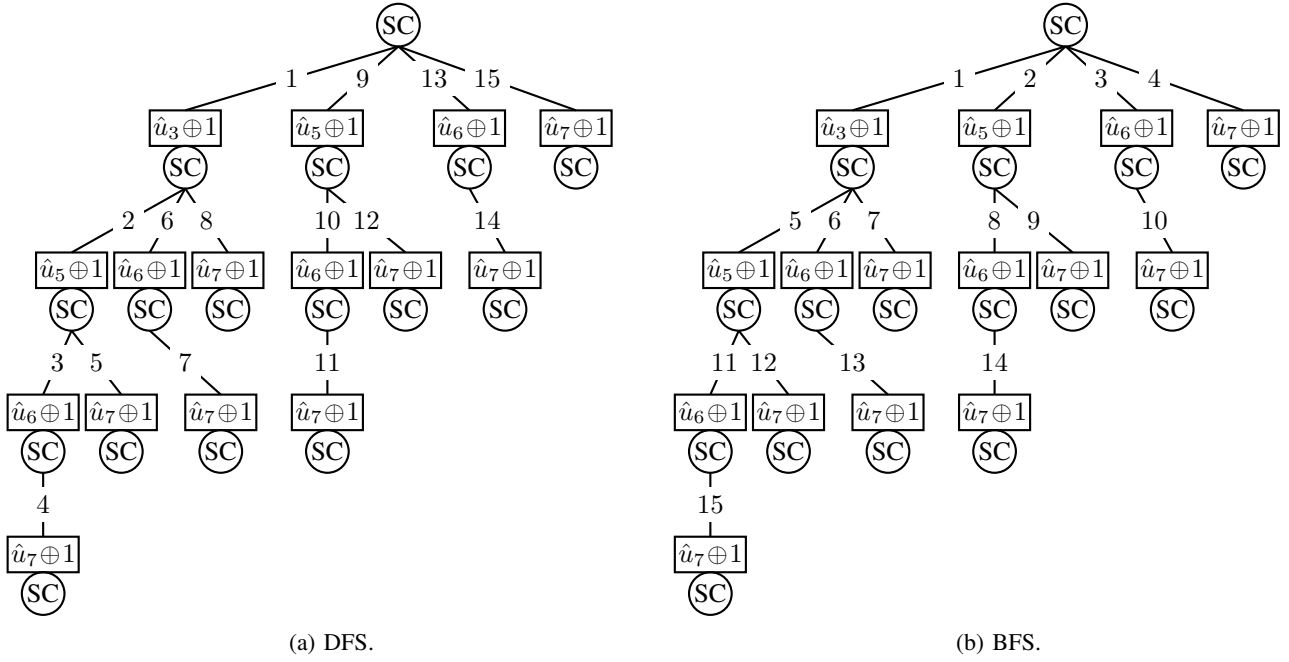


Fig. 1: Bit-flipping tree for  $\mathcal{RM}(1, 3)$  with  $\mathcal{F} = \{0, 1, 2, 4\}$ .

The corresponding candidate result is stored as  $\hat{\mathbf{u}}_{\text{best}}$ , and the corresponding path metric is calculated based on (2) and is stored as  $\text{PM}_{\text{best}}$ . The decoder then proceeds by traversing the bit-flipping tree until all the nodes are visited. Let the complexity of the proposed decoding scheme be the number of SC decoding attempts. It can be seen that the worst-case complexity of the proposed decoder is equivalent to the number of nodes in the bit-flipping tree, which is

$$\mathcal{C}_1 = 2^K. \quad (4)$$

To reduce the average complexity of the decoder, The tree needs to be pruned to limit the number of node visits. Therefore, an approach similar to the SCOS decoder in [13] is adopted. In particular, before each node visit, the path metric associated with the bit-flipping at that node is calculated and compared with  $\text{PM}_{\text{best}}$ . The node is visited only if the path metric is less than  $\text{PM}_{\text{best}}$ . Otherwise, that node and any node that is a descendent of that node is not visited. In the case of a node visit, the corresponding path metric is calculated based on (2) and is compared with  $\text{PM}_{\text{best}}$ . If the path metric is less than  $\text{PM}_{\text{best}}$ , then  $\text{PM}_{\text{best}}$  is updated with the new path metric and the new ML candidate is stored as  $\hat{\mathbf{u}}_{\text{best}}$ .

To traverse the bit-flipping tree without calculating an additional metric and without any sorting operations, the natural order in the bit indices are considered and two tree traversal approaches are adopted. First, a DFS approach, in which the priority is given to the nested bit-flipping operations as shown in Fig. 1a for  $\mathcal{RM}(1, 3)$ . Second, a BFS approach, in which the priority is given to bit-flips at the same levels of the tree as shown in Fig. 1b for  $\mathcal{RM}(1, 3)$ . Note that the order in which the nodes are visited is indicated by the numbers on the edges of the bit-flipping tree. Algorithm 1 and Algorithm 2

summarize the functions  $\text{TS-DFS}(\cdot)$  and  $\text{TS-BFS}(\cdot)$ , which represent the bit-flipping tree search based on the DFS and the BFS strategies, respectively.

In Algorithm 1, the inputs to the  $\text{TS-DFS}(\cdot)$  function are  $\mathbf{y}$ , the set of bit-flipping indices  $\mathcal{E}$ , the ML candidate  $\hat{\mathbf{u}}_{\text{best}}$ , and the path metric associated with the ML candidate  $\text{PM}_{\text{best}}$ . Before the function is called,  $\mathcal{E} = \emptyset$  and  $\text{PM}_{\text{best}} = \infty$ . The function starts by a round of SC decoding using the function  $\text{SC}(\cdot)$ , which takes  $\mathbf{y}$  and  $\mathcal{E}$  as inputs, and results in a candidate  $\hat{\mathbf{u}}$  and its corresponding LLR values  $\alpha$ . The path metric for  $\hat{\mathbf{u}}$  is then calculated and stored in  $\text{PM}_{\hat{\mathbf{u}}}$ . The ML candidate is updated only if  $\text{PM}_{\hat{\mathbf{u}}} < \text{PM}_{\text{best}}$ . Then for any information bit whose index is larger than the maximum index in  $\mathcal{E}$ , denoted by  $i_{\mathcal{E}}^{\max}$ , a metric associated with the bit-flip at the  $i$ -th index is calculated and is stored in  $\text{PM}_{\text{tmp}}$ . A recursive call to  $\text{TS-DFS}(\cdot)$  is conducted only if  $\text{PM}_{\text{tmp}} < \text{PM}_{\text{best}}$ . The  $\text{TS-BFS}(\cdot)$  function in Algorithm 2 depends on storing a set  $\mathcal{S}$  of LLR values and bit-flipping positions as  $\{\alpha, \mathcal{E}\}$ . Since the tree search is conducted by giving priority to the node visits in the same level of the tree, all the candidates at the same level of tree with  $\text{PM}_{\text{tmp}} < \text{PM}_{\text{best}}$  are stored in an auxiliary set  $\mathcal{S}_{\text{next}}$ . The set  $\mathcal{S}$  is updated by  $\mathcal{S}_{\text{next}}$  and the function runs until the set  $\mathcal{S}$  becomes empty.

While the worst-case complexity of the DFS and the BFS schemes for searching the tree is exponential in  $K$ , simulation results in Section IV show that the average complexity of the proposed method is small, especially when the channel experiences low levels of noise.

### B. Complexity Considerations

1) *Limiting the Tree Search:* To reduce the number of node visits in the proposed decoding algorithm, the tree is pruned

**Algorithm 1: TS-DFS( $\cdot$ )**


---

```

1 Function TS-DFS ( $\mathbf{y}, \mathcal{E}, \hat{\mathbf{u}}_{\text{best}}, \text{PM}_{\text{best}}$ ) :
2    $\alpha, \hat{\mathbf{u}} \leftarrow \text{SC}(\mathbf{y}, \mathcal{E})$ 
3    $\text{PM}_{\hat{\mathbf{u}}} \leftarrow \sum_{j \in \mathcal{F}} \frac{1 - \text{sgn}(\alpha_j)}{2} |\alpha_j| + \sum_{j \in \mathcal{E}} |\alpha_j|$ 
4   if  $\text{PM}_{\hat{\mathbf{u}}} < \text{PM}_{\text{best}}$  then
5      $\text{PM}_{\text{best}} \leftarrow \text{PM}_{\hat{\mathbf{u}}}$ 
6      $\hat{\mathbf{u}}_{\text{best}} \leftarrow \hat{\mathbf{u}}$ 
7   for  $i \in \{\mathcal{I} | i_{\mathcal{E}}^{\text{max}} < i\}$  do
8      $\text{PM}_{\text{tmp}} \leftarrow \sum_{j \in \{\mathcal{F} | j < i\}} \frac{1 - \text{sgn}(\alpha_j)}{2} |\alpha_j| + \sum_{j \in \mathcal{E} \cup i} |\alpha_j|$ 
9     if  $\text{PM}_{\text{tmp}} < \text{PM}_{\text{best}}$  then
10       $\hat{\mathbf{u}}_{\text{best}}, \text{PM}_{\text{best}} \leftarrow \text{TS-DFS}(\mathbf{y}, \mathcal{E} \cup i, \hat{\mathbf{u}}_{\text{best}}, \text{PM}_{\text{best}})$ 
11 return  $\hat{\mathbf{u}}_{\text{best}}, \text{PM}_{\text{best}}$ 

```

---

such that the search is only performed up to a specific depth in the tree. Let  $\omega$  denote the maximum depth of the tree up to which the search is carried out and let  $\text{TS-DFS-}\omega(\cdot)$  and  $\text{TS-BFS-}\omega(\cdot)$  denote  $\text{TS-DFS}(\cdot)$  and  $\text{TS-BFS}(\cdot)$  functions that search the tree up to depth  $\omega$ , respectively. Fig. 2 shows an example of this tree-pruning strategy for  $\mathcal{RM}(1, 3)$  with  $\mathcal{F} = \{0, 1, 2, 4\}$ , considering the maximum depth of the tree is  $\omega = 2$ , for both DFS (Fig. 2a) and BFS (Fig. 2b) schemes. Note that this tree-pruning strategy uses the fact that it is more likely to achieve the ML decoding performance by performing a small number of bit-flips [15], which is equivalent to limiting the depth of the bit-flipping tree. The worst-case complexity of this scheme is calculated as

$$\mathcal{C}_2 = \sum_{i=0}^{\omega} \binom{K}{i}, \quad (5)$$

where  $\mathcal{C}_2 < \mathcal{C}_1$  for  $\omega < K$ . It is worth noting that while this tree-pruning strategy significantly reduces the complexity of decoding, it does not guarantee the ML decoding performance. In addition, when  $\omega = 0$ , the proposed decoding algorithm is equivalent to a single round of SC decoding, and when  $\omega = 1$ , the DFS and the BFS strategies are equivalent.

2) *Eliminating Redundant Node Visits*: The path metric calculation in (2) significantly reduces the worst-case complexity by only considering the information bits that are located before the last frozen bit in the code. This is due to the fact that the path metrics in SC decoding (2) can only increase if the estimation of a bit does not follow the sign of its LLR value. Since the SC decoding algorithm is such that the estimation of each bit follows the sign of its LLR value, only frozen bits with fixed values can change the path metric in (2). The bit-by-bit schedule of SC decoding, thus, makes the estimation of any information bit after the last frozen bit in the code redundant [20]. Let  $K_l$  denote the number of information bits after the last frozen bit. The worst-case complexity of the proposed method is

$$\mathcal{C}_3 = 2^{K-K_l}. \quad (6)$$

**Algorithm 2: TS-BFS( $\cdot$ )**


---

```

1 Function TS-BFS ( $\mathbf{y}$ ) :
2    $\mathcal{E} \leftarrow \emptyset$ 
3    $\alpha, \hat{\mathbf{u}}_{\text{best}} \leftarrow \text{SC}(\mathbf{y})$ 
4    $\text{PM}_{\text{best}} \leftarrow \sum_{j \in \mathcal{F}} \frac{1 - \text{sgn}(\alpha_j)}{2} |\alpha_j|$ 
5    $\mathcal{S} \leftarrow \{\{\alpha, \mathcal{E}\}\}$ 
6   while  $|\mathcal{S}| > 0$  do
7      $\mathcal{S}_{\text{next}} \leftarrow \emptyset$ 
8     for  $\{\alpha, \mathcal{E}\} \in \mathcal{S}$  do
9       for  $i \in \{\mathcal{I} | i_{\mathcal{E}}^{\text{max}} < i\}$  do
10         $\text{PM}_{\text{tmp}} \leftarrow \sum_{j \in \{\mathcal{F} | j < i\}} \frac{1 - \text{sgn}(\alpha_j)}{2} |\alpha_j| + \sum_{j \in \mathcal{E} \cup i} |\alpha_j|$ 
11        if  $\text{PM}_{\text{tmp}} < \text{PM}_{\text{best}}$  then
12           $\alpha_{\text{tmp}}, \hat{\mathbf{u}}_{\text{tmp}} \leftarrow \text{SC}(\mathbf{y}, \mathcal{E} \cup i)$ 
13           $\text{PM}_{\hat{\mathbf{u}}_{\text{tmp}}} \leftarrow \sum_{j \in \mathcal{F}} \frac{1 - \text{sgn}(\alpha_{\text{tmp}_j})}{2} |\alpha_{\text{tmp}_j}| + \sum_{j \in \mathcal{E} \cup i} |\alpha_{\text{tmp}_j}|$ 
14          if  $\text{PM}_{\hat{\mathbf{u}}} < \text{PM}_{\text{best}}$  then
15             $\text{PM}_{\text{best}} \leftarrow \text{PM}_{\hat{\mathbf{u}}_{\text{tmp}}}$ 
16             $\hat{\mathbf{u}}_{\text{best}} \leftarrow \hat{\mathbf{u}}_{\text{tmp}}$ 
17          if  $\text{PM}_{\text{tmp}} < \text{PM}_{\text{best}}$  then
18             $\mathcal{S}_{\text{next}} \leftarrow \mathcal{S}_{\text{next}} \cup \{\alpha_{\text{tmp}}, \mathcal{E} \cup i\}$ 
19         $\mathcal{S} \leftarrow \mathcal{S}_{\text{next}}$ 
20 return  $\hat{\mathbf{u}}_{\text{best}}$ 

```

---

Fig. 3 shows the pruned bit-flipping tree for  $\mathcal{RM}(1, 3)$  with  $\mathcal{F} = \{0, 1, 2, 4\}$ . It can be seen that since bits  $u_5, u_6$ , and  $u_7$  are located after the last frozen bit ( $u_4$ ), the ML decoding result can be found by a maximum of two rounds of SC decoding. Note that since this optimization scheme can guarantee ML decoding performance, it is used in this paper in all the proposed methods.

3) *Ordering the Node Visits*: To reduce the average number of node visits in the proposed tree search approach, a metric is assigned to each set of bit-flips  $\mathcal{E}$  and the node that results in the smallest metric is given priority at a specific level of the bit-flipping tree. This paper uses the metric developed in [21] as

$$M_i = \sum_{j \in \mathcal{E} \cup i} |\alpha_j| + \sum_{j \in \{\mathcal{I} | j \leq i\}} \max(0, \beta - |\alpha_j|), \quad (7)$$

where  $\beta$  is a parameter that is empirically selected to be 0.8. This approach ensures the bit-flipping set  $\mathcal{E}$  that more likely caused the error is identified in the earlier stages of the tree search method, which in turn reduces the number of node visits. Let  $\text{TS-DFS-O}(\cdot)$  and  $\text{TS-BFS-O}(\cdot)$  denote  $\text{TS-DFS}(\cdot)$  and  $\text{TS-BFS}(\cdot)$  functions that use (7) to order the node visits, and  $\text{TS-DFS-O-}\omega(\cdot)$  and  $\text{TS-BFS-O-}\omega(\cdot)$  denote  $\text{TS-DFS-}\omega(\cdot)$  and  $\text{TS-BFS-}\omega(\cdot)$  functions that use

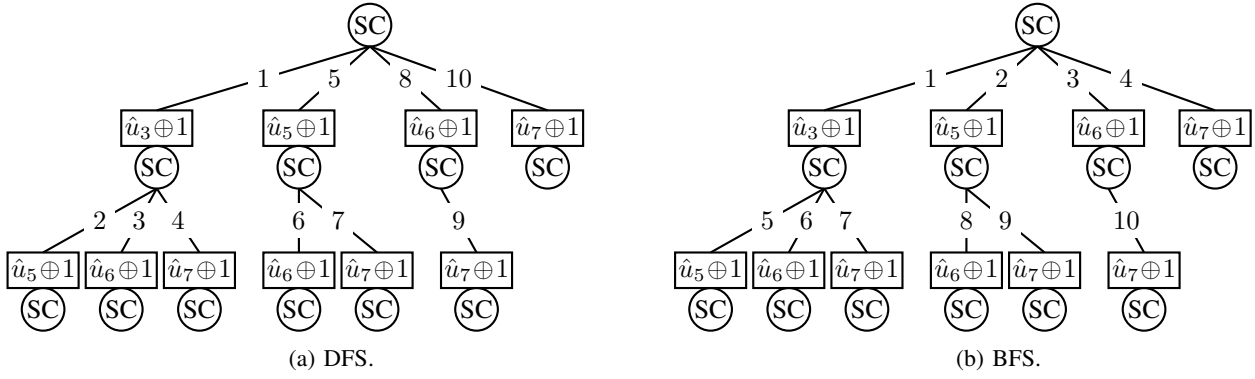


Fig. 2: Pruned bit-flipping tree for  $\mathcal{RM}(1,3)$  with  $\mathcal{F} = \{0, 1, 2, 4\}$ , considering a maximum tree depth of  $\omega = 2$ .

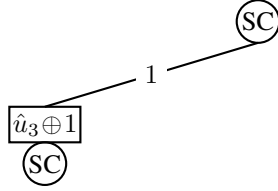


Fig. 3: Pruned bit-flipping tree for  $\mathcal{RM}(1,3)$  with  $\mathcal{F} = \{0, 1, 2, 4\}$ . Since there is only one information bit ( $u_3$ ) before the last frozen bit ( $u_4$ ), a maximum of two rounds of SC decoding results in ML decoding performance.

(7) to order the node visits. Note that while this approach on average reduces the number of node visits, a maximum of  $K$  metric values in (7) need to be sorted at each level of the bit-flipping tree.

#### IV. NUMERICAL RESULTS

This section evaluates the frame error rate (FER) performance and the complexity of the proposed tree search methods and compares them with the SCOS decoder in [13]. The functions TS-DFS( $\cdot$ ), TS-BFS( $\cdot$ ), TS-DFS-O( $\cdot$ ), and TS-BFS-O( $\cdot$ ) achieve the ML decoding performance for any RM code. Fig. 4 shows the FER performance of  $\mathcal{RM}(3,7)$ ,  $\mathcal{RM}(4,7)$ , and  $\mathcal{RM}(2,8)$  using the functions TS-DFS- $\omega(\cdot)$  and TS-BFS- $\omega(\cdot)$  for different values of  $\omega$ . It can be seen that while both TS-DFS- $\omega(\cdot)$  and TS-BFS- $\omega(\cdot)$  decoders provide the same FER performance at the same value of  $\omega$ , selecting  $\omega = 4$  for  $\mathcal{RM}(3,7)$ ,  $\omega = 3$  for  $\mathcal{RM}(4,7)$ , and  $\omega = 5$  for  $\mathcal{RM}(2,8)$  results in an FER performance that is very close to the lower bound of ML decoding performance.

Fig. 5 shows the average decoding complexity  $\mathcal{C}_{\text{avg}}$  in terms of the number of SC decoding attempts for different tree search methods proposed in this paper and compares it with that of the SCOS decoder in [13]. For TS-DFS- $\omega(\cdot)$ , TS-BFS- $\omega(\cdot)$ , TS-DFS-O- $\omega(\cdot)$ , and TS-BFS-O- $\omega(\cdot)$  functions,  $\omega = 4$  is selected for  $\mathcal{RM}(3,7)$ ,  $\omega = 3$  is selected for  $\mathcal{RM}(4,7)$ , and  $\omega = 5$  is selected for  $\mathcal{RM}(2,8)$  to ensure near ML decoding performance. It can be seen that limiting the depth of the bit-flipping tree provides significant average complexity reduction for both DFS and BFS schemes. Moreover, the DFS scheme

benefits from further average complexity reduction when the node visits are ordered. However, ordering the node visits results in negligible average complexity reduction for the BFS scheme. This is due to the fact that the BFS scheme naturally starts with smaller number of bit-flips. In fact, the BFS scheme achieves a lower average complexity than the SCOS decoder in [13]. Note that all the proposed decoders approach the complexity of a single SC decoding attempt for large values of  $E_b/N_0$ .

#### V. CONCLUSION

This paper proposed a tree search approach that achieves the ML decoding performance for RM codes. Two tree search methods are adopted: a DFS scheme that has a simple recursive structure; and a BFS scheme that results in a low decoding complexity. Several enhancements to reduce the worst-case and the average complexity of the proposed tree search approach are developed: 1) limiting the depth of the tree; 2) eliminating redundant node visits; and 3) ordering the node visits. Simulation results confirm the effect of the proposed methods to reduce complexity and show that the BFS scheme provides lower average decoding complexity than the existing tree search approach to decode RM codes.

#### REFERENCES

- [1] 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.
- [2] 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.
- [3] E. Arkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.
- [4] 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. Inf. Theory*, vol. 63, no. 7, pp. 4298–4316, Jul. 2017.
- [5] 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.
- [6] J. D. Key, T. P. McDonough, and V. C. Mavron, "Reed-Muller codes and permutation decoding," *Discrete mathematics*, vol. 310, no. 22, pp. 3114–3119, 2010.
- [7] S. A. Hashemi, N. Doan, M. Mondelli, and W. J. Gross, "Decoding Reed-Muller and polar codes by successive factor graph permutations," in *IEEE Int. Symp. on Turbo Codes & Iterative Inf. Process.*, 2018, pp. 1–5.

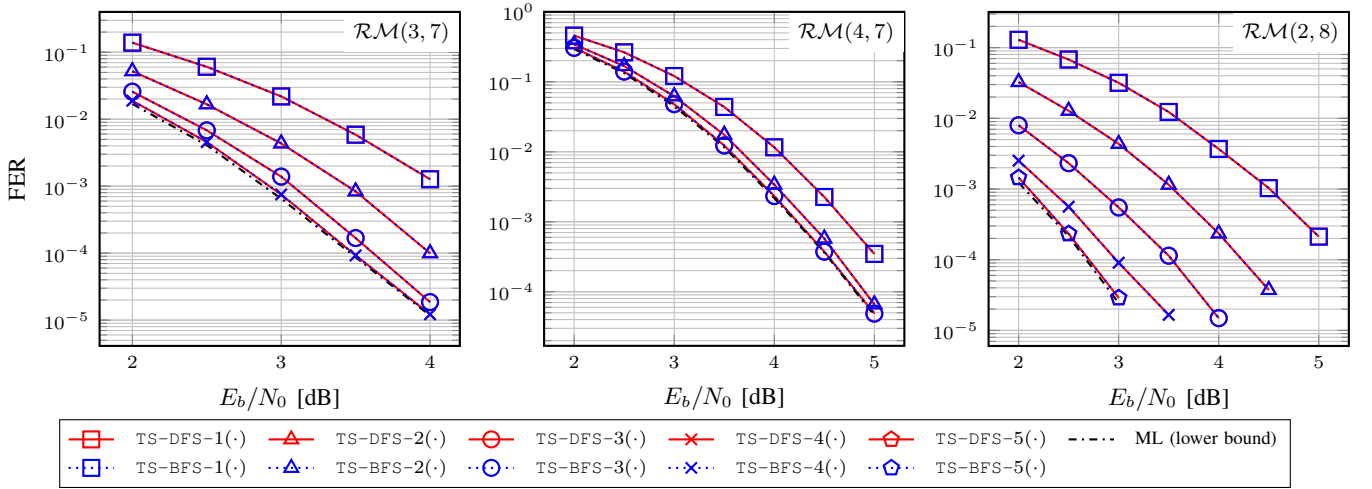


Fig. 4: Error-correction performance of  $\mathcal{RM}(3,7)$ ,  $\mathcal{RM}(4,7)$ , and  $\mathcal{RM}(2,8)$  under TS-DFS- $\omega(\cdot)$  and TS-BFS- $\omega(\cdot)$  for different values of  $\omega$ .

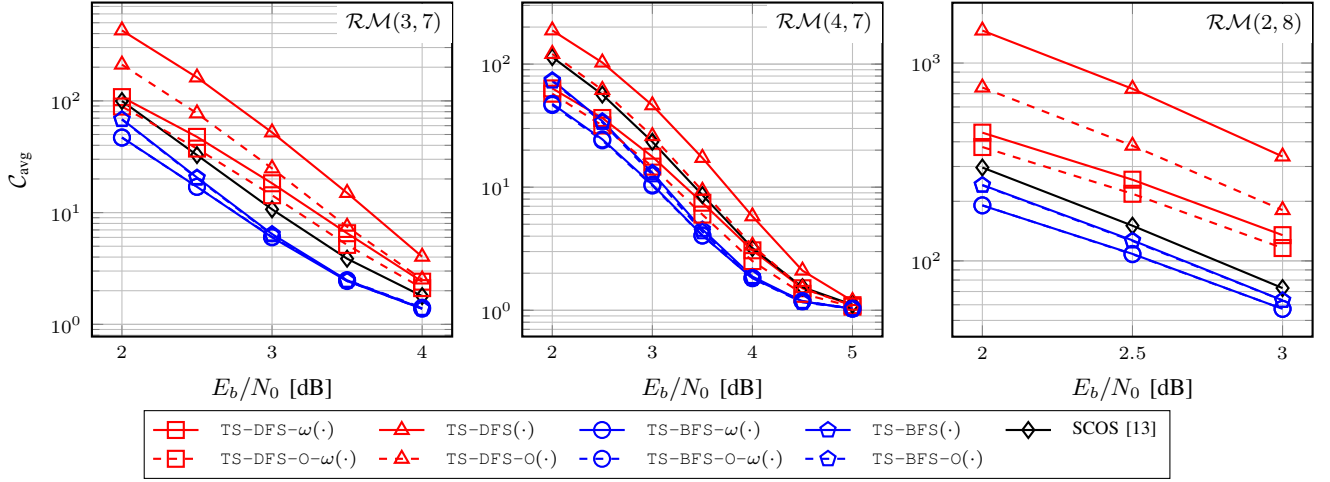


Fig. 5: Average decoding complexity in terms of the number of SC decoding attempts for  $\mathcal{RM}(3,7)$ ,  $\mathcal{RM}(4,7)$ , and  $\mathcal{RM}(2,8)$  under different tree search methods in this paper in comparison with the SCOS decoder of [13]. Note that  $\omega = 4$  for  $\mathcal{RM}(3,7)$ ,  $\omega = 3$  for  $\mathcal{RM}(4,7)$ , and  $\omega = 5$  for  $\mathcal{RM}(2,8)$ .

- [8] M. Kameney, Y. Kameney, O. Kurmaev, and A. Maevskiy, "A new permutation decoding method for Reed-Muller codes," in *IEEE Int. Symp. on Inf. Theory*, 2019, pp. 26–30.
- [9] E. Santi, C. Häger, and H. D. Pfister, "Decoding Reed-Muller codes using minimum-weight parity checks," in *IEEE Int. Symp. on Inf. Theory*, 2018, pp. 1296–1300.
- [10] 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.
- [11] D. Fathollahi, N. Farsad, S. A. Hashemi, and M. Mondelli, "Sparse multi-decoder recursive projection aggregation for Reed-Muller codes," in *IEEE Int. Symp. on Inf. Theory*, 2021. [Online]. Available: <https://arxiv.org/abs/2011.12882>
- [12] I. Dumer, "Recursive decoding and its performance for low-rate Reed-Muller codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 5, pp. 811–823, 2004.
- [13] P. Yuan and M. C. Coşkun, "Complexity-adaptive maximum-likelihood decoding of modified  $G_N$ -coset codes," *CoRR*, vol. abs/2105.04048, 2021. [Online]. Available: <https://arxiv.org/abs/2105.04048>
- [14] O. Afisiadis, A. Balatsoukas-Stimming, and A. Burg, "A low-complexity improved successive cancellation decoder for polar codes," in *48th Asilomar Conf. on Sig., Sys. and Comp.*, Nov. 2014, pp. 2116–2120.
- [15] L. Chandesris, V. Savin, and D. Declercq, "Dynamic-SCFlip decoding of polar codes," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2333–2345, Jun. 2018.
- [16] E. Ankan, "A survey of Reed-Muller codes from polar coding perspective," in *IEEE Inf. Theory Work. on Inf. Theory*, 2010, pp. 1–5.
- [17] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A semi-parallel successive-cancellation decoder for polar codes," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 289–299, Oct. 2013.
- [18] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, Mar. 2015.
- [19] A. Balatsoukas-Stimming, M. Bastani Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5165–5179, Oct. 2015.
- [20] S. A. Hashemi, M. Mondelli, S. H. Hassani, C. Condo, R. L. Urbanke, and W. J. Gross, "Decoder partitioning: Towards practical list decoding of polar codes," *IEEE Trans. Commun.*, vol. 66, no. 9, pp. 3749–3759, 2018.
- [21] N. Doan, S. A. Hashemi, F. Ercan, T. Tonnellier, and W. J. Gross, "Neural successive cancellation flip decoding of polar codes," *J. Sig. Proc. Sys.*, vol. 93, no. 6, pp. 631–642, Jun. 2021.