List Decoding of Reed-Muller Codes

Grigory Kabatiansky and Cédric Tavernier

Abstract

We construct list decoding algorithms for first order Reed-Muller codes $RM[1,m]$ of length $n = 2^m$ correcting up to $n(\frac{1}{2} - \epsilon)$ errors with complexity $\mathcal{O}(n\epsilon^{-3})$. Considering probabilistic approximation of these algorithms leads to randomized list decoding algorithms with characteristics similar to Goldreich-Levin algorithm, namely, of complexity $\mathcal{O}(m^2\epsilon^{-7}\log\frac{1}{\epsilon}(\log\frac{1}{\epsilon} + \log\frac{1}{P_{err}} + \log m))$, where P_{err} is the probability of wrong list decoding.

1 Introduction

Following P.Elias definition [1] list decoding algorithm of decoding radius T should produce for any received vector y the list $L_T(y) = \{c \in C : d(y,c) \leq T\}$ of all vectors c from a code C which are at distance at most T apart from y. Recently very efficient list decoding algorithms were proposed for Reed-Solomon codes and algebraic-geometry codes (see [2]). Until very recently (see[7]) efficient list decoding algorithms were not known for Reed-Muller codes, despite that these codes are generalization of Reed-Solomon codes (by considering multivariate polynomials instead of univariate). At the same time, very efficient but probabilistic algorithm of list decoding for Reed-Muller codes of order 1 was known from 1989 [3], i.e. much before deterministic ones for RS-codes. In this paper we propose two deterministic list decoding algorithms for first order Reed-Muller codes of decoding radius $T = n(\frac{1}{2} - \epsilon)$ with complexity $\mathcal{O}(n/\epsilon^3)$. We consider also their probabilistic approximation and evaluate the performance of these and related probabilistic algorithms [3],[4].

2 Deterministic list decoding algorithms for Reed-Muller codes of order 1

Binary Reed-Muller code $RM(1,m)$ of order 1 and length $n = 2^m$ consists of vectors $\mathbf{f} = (..., f(x_1,...,x_m), ...)$ where $f(x_1,...,x_m) = f_0 + f_1x_1 + \ldots + f_mx_m$ is a linear Boolean function and $(x_1,...,x_m)$ runs over all 2^m points of the m-dimensional Boolean cube. It is well-known that $RM(1,m)$ is an optimal code consisting of $2n$ vectors with the minimal code distance $d = n/2$. For these codes there are well-known ML decoding algorithm (FFT) of complexity $\mathcal{O}(n \log n)$ as well as bounded distance decoding algorithm [5] of complexity $\mathcal{O}(n)$. The later algorithm can be considered as a list decoding algorithm of decoding radius $t = \frac{n}{4} - 1$. Our goal is to construct a list decoding algorithm of $RM(1,m)$ with decoding radius $T = n(\frac{1}{2} - \epsilon)$ almost twice larger and with the same (asymptotically) complexity.

Let $\mathbf{y}$ be a received vector and $L_\epsilon(\mathbf{y}) = \{\mathbf{f} \in RM(1,m) : d(\mathbf{y}, \mathbf{f}) \leq n(\frac{1}{2} - \epsilon)\}$ be the desired list. The proposed algorithm works recursively by finding on the i-th step a list $L_\epsilon^i(\mathbf{y})$ of "candidates" which should (but may not) coincide with i-prefix of some $f(x_1, ..., x_m) = f_0 + f_1 x_1 + \ldots + f_m x_m \in L_\epsilon(\mathbf{y})$. The main idea is to approximate the Hamming distance between the received vector $\mathbf{y}$ and an arbitrary "propagation" of a candidate $c^{(i)}(x_1, \ldots, x_m) = c_1 x_1 + \ldots + c_i x_i$ by the sum of Hamming distances over all i-dimensional "facets" of the m-dimensional Boolean cube.

Let $S_j = \{(x_1, \ldots, x_i, s_1, \ldots, s_{m-i})\}$ be one of i-dimensional facets, where $(x_1, \ldots, x_i)$ runs over all 2^i binary i-dimensional vectors, $s_1, \ldots, s_{m-i}$ are fixed and $j = s_1 + \ldots + s_{m-i} 2^{m-i-1}$ is the number of this facet. Consider restrictions of the received vector $\mathbf{y}$ and the candidate $c^{(i)}(x_1, \ldots, x_m) = c_1 x_1 + \ldots + c_i x_i$ on facet S_j and denote $d_{S_j}(\mathbf{y}, \mathbf{c}^{(i)})$ the Hamming distance between these two vectors (of length 2^i). Clearly that for any linear function $c(x_1, \ldots, x_m)$ such that $c^{(i)}(x_1, \ldots, x_m) = c_1 x_1 + \ldots + c_i x_i$ is its prefix, i.e., $c(x_1, \ldots, x_m) = c_0 + c^{(i)}(x_1, \ldots, x_m) + c_{i+1} x_{i+1} + \ldots + c_m x_m$, we have that $d_{S_j}(\mathbf{y}, \mathbf{c})$ equals either $d_{S_j}(\mathbf{y}, \mathbf{c}^{(i)})$ or $d_{S_j}(\mathbf{y}, \mathbf{c}^{(i)} \oplus \mathbf{1})$. Define "$i$"-th distance $\Delta^{(i)}(\mathbf{y}, \mathbf{c}^{(i)})$ between $\mathbf{y}$ and $\mathbf{c}^{(i)}$ by

$$\Delta^{(i)}(\mathbf{y}, \mathbf{c}^{(i)}) = \sum_{j=0}^{2^{m-i}-1} \Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}), \tag{1}$$

where $\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}) = \min\{d_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}), d_{S_j}(\mathbf{y} \oplus \mathbf{1}, \mathbf{c}^{(i)})\}$. Then the following result is obvious.

Lemma 1 *For any linear function $\mathbf{c} = c(x_1, \ldots, x_m)$ and any its prefix $\mathbf{c}^{(\mathbf{i})} = c^{(i)}(x_1, \ldots, x_m)$*

$$d(\mathbf{y}, \mathbf{c}) \geq \Delta^{(i)}(\mathbf{y}, \mathbf{c}^{(i)}).$$

This Lemma leads us to the following natural criteria of acceptance a candidate. Namely, a candidate $\mathbf{c}^{(i)} = c_1 x_1 + \ldots + c_i x_i$ is accepted iff $\Delta^{(i)}(\mathbf{y}, \mathbf{c}^{(i)}) \leq n(\frac{1}{2} - \epsilon)$. Saying without words : $L_\epsilon^i(\mathbf{y}) = \{\mathbf{c}^{(i)} : \Delta^{(i)}(\mathbf{y}, \mathbf{c}^{(i)}) \leq n(\frac{1}{2} - \epsilon)\}$. We call the corresponding algorithm as *Sums-Algorithm.*

To work effectively any list decoding algorithm should generate rather small list(s). To prove it for Sums-Algorithm we need the following simple

Lemma 2 . *Let $\mathbf{c} = c_0 + c_1 x_1 + \ldots + c_m x_m$ be an affine function such that $d(\mathbf{y}, \mathbf{c}) \leq n(\frac{1}{2} - \epsilon)$ and let $\mathbf{c}^{(i)} = c_1 x_1 + \ldots + c_i x_i$ its i-th prefix. Then for every $i \in [1, \ldots, m]$ there is a fraction of at least $2(\epsilon - \epsilon')$ facets S_j which satisfy $2^{-i} \Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}) \leq \frac{1}{2} - \epsilon'$.*

Proof. Denote $p_z = 2^{i-m}|\{j : 2^{-i}\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}) = \frac{1}{2} - z\}|$. We shall prove that $P = P_{\epsilon'}(\mathbf{c}^{(i)}) = \sum_{z \geq \epsilon'} p_z$ is greater or equal to $2(\epsilon - \epsilon')$. On the one hand,

$$\Delta^{(i)}(\mathbf{y}, \mathbf{c}^{(i)}) = \sum_{j=0}^{2^{m-i}-1} \Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}) = 2^m \sum p_z \left(\frac{1}{2} - z\right) = n\left(\frac{1}{2} - \sum p_z z\right)$$

since $\sum p_z = 1$. Then by Lemma 1 we have that $\Delta^{(i)}(\mathbf{y}, \mathbf{c}^{(i)}) \leq d(\mathbf{y}, \mathbf{c}) \leq n\left(\frac{1}{2} - \epsilon\right)$ and hence $\sum p_z z \geq \epsilon$. On the other hand,

$$\sum p_z z \leq \sum_{z<\epsilon'} p_z \epsilon' + \sum_{z \geq \epsilon'} p_z z \leq \epsilon' + \frac{P}{2}$$

because $\max z = 1/2$. We conclude that $P \geq 2(\epsilon - \epsilon')$. □

This Lemma applying for $\epsilon' = \epsilon/2$ motivates introducing another list(s) $R^i_\epsilon(\mathbf{y}) = \{\mathbf{c}^{(i)} : P_{\epsilon/2}(\mathbf{c}^{(i)}) \geq \epsilon\}$, i.e., consisting of all prefixes $\mathbf{c}^{(i)}$ such that for at least ϵ fraction of all facets S_j we have

$$2^{-i}\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}) \leq \frac{1}{2} - \frac{\epsilon}{2}.$$

The corresponding list decoding algorithm which we call *Ratio-Algorithm* works in a way similar to *Sums-Algorithm* but with another criteria of acceptance. Namely, a candidate $\mathbf{c}^{(i)} = c_1 x_1 + \ldots + c_i x_i$ is accepted iff $\mathbf{c}^{(i)} \in R^i_\epsilon(\mathbf{y})$. Note that after performing all m steps *Ratio-Algorithm* should do an extra step by checking and then outputting only such vectors $\mathbf{c}$ from the last list for which $d(\mathbf{y}, \mathbf{c}) \leq n(\frac{1}{2} - \epsilon)$.

Lemma 2 means that $L^i_\epsilon(\mathbf{y}) \subseteq R^i_\epsilon(\mathbf{y})$. Now we can estimate the size of any intermediate list for both algorithms.

Lemma 3 *For any received vector* $\mathbf{y}$ *and for every* $i \in [1, \ldots, m]$

$$|L^i_\epsilon(\mathbf{y})| \leq |R^i_\epsilon(\mathbf{y})| \leq 2\epsilon^{-3} \tag{2}$$

Proof. Denote $A(\mathbf{c}^{(i)}) = |\{j : 2^{-i}\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}) \leq \frac{1}{2} - \hat{\epsilon}\}| = 2^{m-i}P_{\hat{\epsilon}}(\mathbf{c}^{(i)})$. If $\mathbf{c}^{(i)} \neq \hat{\mathbf{c}}^{(i)}$ then their restrictions on any i-dimensional facet S_j are distinct codevectors of $RM(1, i)$ and therefore $|\{\mathbf{c}^{(i)} : 2^{-i}\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}) \leq \frac{1}{2} - \hat{\epsilon}\}| = |\{\mathbf{c}^{(i)} : 2^{-i}d_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}) \leq \frac{1}{2} - \hat{\epsilon}\}| + |\{\mathbf{c}^{(i)} : 2^{-i}d_{S_j}(\mathbf{y} \oplus \mathbf{1}, \mathbf{c}^{(i)}) \leq \frac{1}{2} - \hat{\epsilon}\}| \leq \frac{1}{2\hat{\epsilon}^2}$ where the last inequality follows from Johnson bound (applied for $d = n'/2$ and $w \leq n'(\frac{1}{2} - \hat{\epsilon})$, where $n' = 2^i$ is the length of $RM(1, i)$). Then

$$\sum_{all \mathbf{c}^{(i)}} A(\mathbf{c}^{(i)}) = \sum_{j=0}^{2^{m-i}-1} |\{\mathbf{c}^{(i)} : 2^{-i}\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}) \leq \frac{1}{2} - \hat{\epsilon}\}| \leq 2^{m-i}\frac{1}{2\hat{\epsilon}^2}$$

Hence the number of $\mathbf{c}^{(i)}$ such that $A(\mathbf{c}^{(i)}) \geq \tilde{\epsilon}2^{m-i}$ cannot exceed $\frac{1}{2\tilde{\epsilon}\hat{\epsilon}^2}$. Since this number for $\tilde{\epsilon} = \epsilon$ and $\hat{\epsilon} = \frac{\epsilon}{2}$ equals to $|R^i_\epsilon(\mathbf{y})|$ it concludes the proof. □

3 Complexity

Performing of the proposed algorithms demands the following elementary subroutines: summation of two i-bits integers, its complexity equals $c_1 i$;
taking minimum of two i-bits integers, its complexity equals $c_2 i$.
We need also to add 2^k i-bits integers. The complexity of this subroutine equals $\sum_{l=1}^{k} c_1(i+l-1)2^{k-l} = c_1 2^k(\sum_{l=1}^{k}(i-1)2^{-l} + \sum_{l=1}^{k} l2^{-l}) < c_1(i+1)2^k$.
Surely we shall use the recursive structure of both algorithms. The result of i-th step will be the lists $L_\epsilon^i(\mathbf{y})$ or $R_\epsilon^i(\mathbf{y})$ together with assigned to every "survived" $\mathbf{c}^{(i)}$ a collection (vector) of all values $\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)})$ and $C^i(j)$, where $C^i(j) = 0$ if $\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}) = \min\{d_{S_j}(\mathbf{y}, \mathbf{c}^{(i)}), d_{S_j}(\mathbf{y} \oplus \mathbf{1}, \mathbf{c}^{(i)})\} = d_{S_j}(\mathbf{y}, \mathbf{c}^{(i)})$ and $C^i(j) = 1$ otherwise. We can consider $C^i(j)$ as our guess of c_0 based on the received vector $\mathbf{y}$ restricted to S_j.
For performing $i+1$-th step observe that any $i+1$-dimensional facet $S_j = \{(x_1, \dots, x_i, x_{i+1}, s_1, \dots, s_{m-i-1})\}$ is the union of two i-dimensional facets $S_{j_0} = \{(x_1, \dots, x_i, 0, s_1, \dots, s_{m-i-1})\}$ and $S_{j_1} = \{(x_1, \dots, x_i, 1, s_1, \dots, s_{m-i-1})\}$. To calculate $\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i+1)})$ consider at first the case $c_{i+1} = 0$ what means that the prefix c^i and its prolongation c^{i+1} coincide. If $C^i(j_0) = C^i(j_1)$ then $\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i+1)}) := \Delta_{S_{j_0}}(\mathbf{y}, \mathbf{c}^{(i)}) + \Delta_{S_{j_1}}(\mathbf{y}, \mathbf{c}^{(i)})$ and $C^{i+1}(j) := C^i(j_0)$. Otherwise let $\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i+1)}) := \Delta_{S_{j_0}}(\mathbf{y}, \mathbf{c}^{(i)}) + (2^i - \Delta_{S_{j_1}}(\mathbf{y}, \mathbf{c}^{(i)}))$ and $C^{i+1}(j) := C^i(j_0)$ if $\Delta_{S_{j_0}}(\mathbf{y}, \mathbf{c}^{(i)}) \le \Delta_{S_{j_1}}(\mathbf{y}, \mathbf{c}^{(i)})$, or let $\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i+1)}) := \Delta_{S_{j_1}}(\mathbf{y}, \mathbf{c}^{(i)}) + (2^i - \Delta_{S_{j_0}}(\mathbf{y}, \mathbf{c}^{(i)}))$ and $C^{i+1}(j) := C^i(j_1)$ if $\Delta_{S_{j_1}}(\mathbf{y}, \mathbf{c}^{(i)}) \le \Delta_{S_{j_0}}(\mathbf{y}, \mathbf{c}^{(i)})$.
In the case $c_{i+1} = 1$ we have that the prefix $\mathbf{c}^{(i)}$ and its prolongation $\mathbf{c}^{(i+1)}$ coincide on S_{j_0}, and S_{j_1}, one of them is the inversion of another. This observation means that we can put $C^i(j_1) := C^i(j_1) \oplus 1$ and then apply the above described algorithm.
Hence for performing of $i+1$-th step for any prefix $\mathbf{c}^{(i)}$ we need to add $2^{m-(i+1)}$ pairs of $i-1$-bits integers and take the same number of minimums to calculate every $\Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i+1)})$. Then we need to take sum $\sum_{j=0}^{2^{m-i-1}-1} \Delta_{S_j}(\mathbf{y}, \mathbf{c}^{(i+1)})$ for *Sums-Algorithm* or take sum of zeroes and ones for *Ratio-Algorithm* to accept or not the prolongation $\mathbf{c}^{(i+1)}$. By Lemma 3 the number of "survived" prefixes $\mathbf{c}^{(i)}$ does not exceed $2\epsilon^{-3}$, hence the total amount of calculations for performing $i+1$-th step is at most $2\epsilon^{-3}(2^{m-(i+1)}(c_1 i + c_2 i) + c_1(i+1)2^{m-(i+1)})$. Hence the total amount of calculation for the whole algorithm does not exceed

$$2\epsilon^{-3}\sum_{i=1}^{m}(2c_1 + c_2)i2^{m-i} < 4\epsilon^{-3}(2c_1 + c_2)2^m.$$

We prove

Theorem 1 *For any received vector* $\mathbf{y}$ *both Sums-Algorithm and Ratio-Algorithm evaluate with complexity* $\mathcal{O}(n\epsilon^{-3})$ *the list of all vectors* $\mathbf{c} \in RM(1, m)$ *such that* $d(\mathbf{y}, \mathbf{c}) \le n(\frac{1}{2} - \epsilon)$.

4 Probabilistic approximation of deterministic list decoding algorithms

Probabilistic list decoding algorithm for $RM(1,m)$ was first suggested in [3] and later was reformalized in a larger context in [4]. This algorithm intends to produce a list $PrL_\epsilon(\mathbf{y})$ which contains:
1) all vectors $\mathbf{c} \in RM(1,m) : d(\mathbf{y},\mathbf{c}) \leq n(\frac{1}{2}-\epsilon)$;
2) no vectors $\mathbf{c} \in RM(1,m) : d(\mathbf{y},\mathbf{c}) \geq n(\frac{1}{2}-\frac{\epsilon}{4})$.
This algorithm being probabilistic has as errors of the first and the second order, namely, with probability P_1 there is some "good" codevector $\mathbf{c}$ (i.e. $d(\mathbf{y},\mathbf{c}) \leq n(\frac{1}{2}-\epsilon)$) which does not belong to $PrL_\epsilon(\mathbf{y})$, and, on the other hand, with probability P_2 there is some "bad" codevector $\mathbf{c}$ (i.e., $d(\mathbf{y},\mathbf{c}) \geq n(\frac{1}{2}-\frac{\epsilon}{4})$) which belongs to $PrL_\epsilon(\mathbf{y})$. Sum of these probabilities $P_{err} = P_1 + P_2$ is called "error probability". The designed in [3], [4] probabilistic list decoding algorithm has complexity $poly(1/\epsilon, m, 1/logP_{err})$. In this section we show that randomized version of *Sums-Algorithm* and *Ratio-Algorithm* do the same as the algorithm of [3], [4] with complexity

$$\mathcal{O}(m^2\epsilon^{-7}\log\frac{1}{\epsilon}(\log m + \log\frac{1}{\epsilon} + \log\frac{1}{P_{err}})). \tag{3}$$

To get randomized version of *Ratio-Algorithm* and *Sums-Algorithm* we estimate ratio $P_\epsilon(\mathbf{c}^{(i)})$ (or $\Delta^{(i)}(\mathbf{y},\mathbf{c}^{(i)})$, correspondingly) by choosing randomly N facets. Then to estimate $\Delta_{S_j}(\mathbf{y},\mathbf{c}^{(i)}) = \min\{d_{S_j}(\mathbf{y},\mathbf{c}^{(i)}), d_{S_j}(\mathbf{y}\oplus\mathbf{1},\mathbf{c}^{(i)})\}$ for every of N chosen facets we take randomly M points from S_j. We choose M sufficiently large to distinguish between "good" facets S_j, where $\Delta_{S_j}(\mathbf{y},\mathbf{c}^{(i)}) \leq 2^i(\frac{1}{2}-\epsilon)$, and "bad" facets S_j, where $\Delta_{S_j}(\mathbf{y},\mathbf{c}^{(i)}) \geq 2^i(\frac{1}{2}-\frac{\epsilon}{4})$. Chernoff inequality guarantees that the probability of incorrect distinguishing between good and bad facets is less than $e^{-\mathcal{O}(\epsilon^2 M)}$. The corresponding analysis for facets and the size of the lists leads to (4). Note that the complexity of these randomised algorithms evaluated in number of *bit* operations and for the *worst* case (not "in average").

5 Conclusion

The proposed list decoding algorithm of linear complexity for $RM(1,m)$ can be generalized for decoding of biorthogonal code in Euclidian space and for q-ary RM codes with the corresponding decoding radius $T_q = n(1-q^{-1}-\epsilon)$. The very recent paper [7] provides list decoding algorithm for q-ary RM codes of arbitrary order s. That algorithm is in fact GS-decoding [2] of the corresponding BCH-code containing a given RM-code and therefore its decoding radius $T' = n(1-\sqrt{d/n})$. For $d/n \ll 1$, i.e for the case of growing (with m) order s, $T' \approx d/2$, and there is known algorithm of complexity $n \cdot \min(s, m-s)$ (hense at most $n\log n$) correcting $d/2$ errors [6]. For RM-codes of fixed order algorithm [7] is better than bounded distance decoding, but for $RM(1,m)$-codes is much weaker both in decoding radius ($1-\frac{1}{\sqrt{q}}$ instead of $1-\frac{1}{q}-\epsilon$) and in complexity ($\mathcal{O}(n^3)$ instead of $\mathcal{O}(n/\epsilon^3)$) comparing with the proposed algorithm. Note that Dumer's algorithms for RM-codes of any fixed order correct with linear complexity *almost* all errors within decoding radius $T = n(\frac{1}{2} - \epsilon)$, see [8],[9]. Currently we do not know if there exist similar list decoding algorithm.

6 Aknowledgement

The authors are grateful to Ilya Dumer for helpful discussions.

References

[1] P. Elias, "List decoding for noisy channels" *1957-IRE WESCON Convention Record*, Pt. 2, pp. 94–104, 1957.

[2] V.Guruswami and M.Sudan, "Improved decoding of Reed-Solomon and algebraic-geometry codes ," *IEEE Trans. on Information Theory*, vol. 45, pp. 1757–1767, 1999.

[3] O.Goldreich and L.A.Levin, "A hard-core predicate for all one-way functions", *Proceedings of 21-st ACM Symp. on Theory of Computing*, pp. 25–32, 1989.

[4] O. Goldreich, R. Rubinfeld and M. Sudan, "Learning polynomials with queries: the highly noisy case", *SIAM J. on Discrete Math.*, pp. 535–570, 2000.

[5] S. Litsyn and O.Shekhovtsov, "Fast decoding algorithm for first order Reed-Muller codes ", *Problems of Information Transmission*,vol. 19, pp. 87–91, 1983.

[6] G. A. Kabatianskii, "On decoding of Reed-Muller codes in semi continuous channels," *Proc.* 2^{nd} *Int. Workshop "Algebr. and Comb. Coding theory"* , Leningrad, USSR,pp. 87-91, 1990.

[7] Ruud Pellikaan and Xin-Wen Wu, "List decoding of q-ary Reed-Muller Codes", *IEEE Trans. on Information Theory*, vol. 50, pp. 679-682, 2004.

[8] I.Dumer, "Recursive decoding of Reed-Muller codes",*Proceedings of 37th Allerton Conf. on Commun.,Contr. and Comp.* , pp. 61–69, 1999.

[9] I.Dumer, "Recursive decoding and its performance for low-rate Reed-Muller codes",*IEEE Trans. on Information Theory*, vol. 50, pp. 811-823, 2004.