List Decoding of Reed-Muller Codes Based on a Generalized Plotkin Construction

Kenji Yasunaga
Institute of Systems, Information Technologies and Nanotechnologies
Fukuoka, Japan
Email: yasunaga@isit.or.jp

***Abstract*—Gopalan, Klivans, and Zuckerman proposed a list-decoding algorithm for Reed-Muller codes. Their algorithm works up to a given list-decoding radius. Dumer, Kabatiansky, and Tavernier improved the complexity of the algorithm for binary Reed-Muller codes by using well-known Plotkin construction. In this study, we propose a list-decoding algorithm for non-binary Reed-Muller codes as a generalization of Dumer et al.'s algorithm. Our algorithm is based on a generalized Plotkin construction. Since Dumer et al.'s and our algorithms can be applied to more general codes than Reed-Muller codes, we give a condition for codes under which these list-decoding algorithms work.**

I. Introduction

A list-decoding problem has been paid increasing attention in coding theory and theoretical computer science since the breakthrough work of efficient list-decoding algorithms for Hadamard codes [3] and Reed-Solomon codes [10]. For the surveys of recent list-decoding algorithms and its applications, see [11], [5], [6], [7].

In 2008, Gopalan, Klivans, and Zuckerman [4] proposed a list-decoding algorithm for Reed-Muller codes with small alphabet size. Their algorithm works up to any given list-decoding radius. Dumer, Kabatiansky, and Tavernier [2] improved the complexity of the algorithm of [4] for the binary case by using well-known Plotkin construction.

In this study, we propose a list-decoding algorithm for non-binary Reed-Muller codes. Our algorithm is a generalization of that of [2] to the case of non-binary codes. Since the Plotkin construction is used only for constructing binary codes, we consider a generalized Plotkin construction to deal with non-binary cases. Then we propose a list-decoding algorithm for non-binary Reed-Muller codes based on the generalized Plotkin construction. Unfortunately, we failed to analyze the performance of our algorithm, including the list-decoding radius and the time complexity. Nevertheless, since our algorithm allows parallel computation in several steps, whereas the algorithm of [4] need to compute these steps sequentially, we believe that our algorithm could be faster for some range of parameters, in particular small q and relatively large r.

Next, we explore codes to which the list-decoding algorithms of [4], [2] and ours (for short GKZ-type algorithm) can be applied. *Polar codes* are introduced by Arikan [1] as a family of capacity achieving codes for any symmetric binary-input discrete memoryless channels. We see polar codes as a generalization of binary Reed-Muller codes, and give a sufficient condition for them under which the GKZ-type algorithm can be applied.

II. List decoding for Reed-Muller codes over $\mathbb{F}_q$

A. Reed-Muller codes

Fix a finite field $\mathbb{F}_q$. Let $\mathrm{RM}_q(r,m)$ denote the r-th order Reed-Muller code with m variables over $\mathbb{F}_q$. The message space of $\mathrm{RM}_q(r,m)$ is the set of polynomials of total degree at most r in m variables over $\mathbb{F}_q$. The codeword of a polynomial $p(x_1,x_2,\ldots,x_m)$ consists of its evaluation at every point in $\mathbb{F}_q^m$. Let $\mathbb{F}_q=\{a_0,a_1,\ldots,a_{q-1}\}$. Without loss of generality, we assume that an evaluation point for $(x_1,x_2,\ldots,x_m)$ is lexicographically ordered in the codeword vector. That is, the codeword of p is a vector in $\mathbb{F}_q^{q^m}$ represented in (1).

Using the same correspondence, any function $f:\mathbb{F}_q^m\to\mathbb{F}_q$ has the corresponding vector representation. Also we can see any vector in $\mathbb{F}_q^{q^m}$ as a function that maps $\mathbb{F}_q^m$ to $\mathbb{F}_q$. Hereafter, we use a vector representation and a function representation interchangeably.

B. A Generalized Plotkin construction

Plotkin construction [9] is a method for constructing a binary code C from two binary codes C_1,C_2 with the same code length such that

$$C=\{u\circ(u+v):u\in C_1,v\in C_2\},$$

where, for $x_1\in\mathbb{F}_q^{n_1}$ and $x_2\in\mathbb{F}_q^{n_2}$, $x_1\circ x_2\in\mathbb{F}_q^{n_1+n_2}$ represents the concatenation of x_1 and x_2. It is well known that binary Reed-Muller codes have this construction [8, Chapter 13, Theorem 2].

A list-decoding algorithm in [2] uses the following fact.

$$\begin{aligned}\mathrm{RM}_2(r,m)&=\{u\circ(u+v):u\in\mathrm{RM}_2(r,m-1),\\&\qquad v\in\mathrm{RM}_2(r-1,m-1)\}\\&=\{(u+v)\circ u:u\in\mathrm{RM}_2(r,m-1),\\&\qquad v\in\mathrm{RM}_2(r-1,m-1)\}.\end{aligned}$$

Equivalently,

$$\begin{aligned}&\mathrm{RM}_2(r,m)\\&\quad=\{p_0(x_1,\ldots,x_{m-1})+(x_m-b)p_1(x_1,\ldots,x_{m-1})\\&\quad:p_0\in\mathrm{RM}_2(r,m-1),p_1\in\mathrm{RM}_2(r-1,m-1)\},\end{aligned}\quad(2)$$

where $b\in\mathbb{F}_2$.

$$(p(a_0,a_0,\ldots,a_0),p(a_1,a_0,\ldots,a_0),\ldots,p(a_{q-1},a_{q-1},\ldots,a_{q-1},a_0),$$
$$p(a_0,a_0,\ldots,a_1),p(a_1,a_0,\ldots,a_1),\ldots,p(a_{q-1},a_{q-1},\ldots,a_{q-1},a_1),$$
$$\ldots,$$
$$p(a_0,a_0,\ldots,a_{q-1}),p(a_1,a_0,\ldots,a_{q-1}),\ldots,p(a_{q-1},a_{q-1},\ldots,a_{q-1})) \quad (1)$$

We generalize the above construction of binary Reed-Muller codes to non-binary cases. Fix $\mathbb{F}_q = \{b_0, b_1, \ldots, b_{q-1}\}$. Then any $p(x_1,\ldots,x_m) \in \mathrm{RM}_q(r,m)$ with $r \geq q-1$ can be represented as

$$\begin{aligned}
&p(x_1,\ldots,x_m)\\
&= p_0(x_1,\ldots,x_{m-1}) + (x_m - b_0)p_1(x_1,\ldots,x_{m-1})\\
&\quad + (x_m-b_0)(x_m-b_1)p_2(x_1,\ldots,x_{m-1}) + \cdots\\
&= \sum_{i=0}^{q-1}\left(p_i(x_1,\ldots,x_{m-1})\prod_{j=0}^{i-1}(x_m-b_j)\right), \qquad (3)
\end{aligned}$$

where $p_i(x_1,\ldots,x_{m-1}) \in \mathrm{RM}_q(r-i,m-1)$. Hence, for $r \geq q-1$,

$$\mathrm{RM}_q(r,m) = \left\{\sum_{i=0}^{q-1} p_i(x_1,\ldots,x_{m-1})\prod_{j=0}^{i-1}(x_m-b_j) : p_i(x_1,\ldots,x_{m-1}) \in \mathrm{RM}_q(r-i,m-1)\right\}.$$

Since there are $q!$ choices of $\{b_0,b_1,\ldots,b_{q-1}\}$, $\mathrm{RM}_q(r,m)$ can be represented in $q!$ ways.

If we take the vector representation, any $p(x_1,\ldots,x_m) \in \mathrm{RM}_q(r,m)$ can be represented as

$$\begin{aligned}
&p(x_1,\ldots,x_m)\\
&= p(x_1,\ldots,x_{m-1},a_0)\circ p(x_1,\ldots,x_{m-1},a_1)\\
&\quad \circ\cdots\circ p(x_1,\ldots,x_{m-1},a_{q-1})\\
&= \bigcirc_{i=0}^{q-1} p(x_1,\ldots,x_{m-1},a_i)\\
&= \bigcirc_{i=0}^{q-1}\left(\sum_{j=0}^{q-1} p_j(x_1,\ldots,x_{m-1})\prod_{k=0}^{j-1}(a_i-b_k)\right), \qquad (4)
\end{aligned}$$

where $\bigcirc_{i=0}^{n} v_i = v_0 \circ v_1 \circ \cdots \circ v_n$ for vectors $v_0,\ldots,v_n$, and the last equality follows from (3).

C. List decoding algorithm

For two functions x, y, define the relative distance between x and y as

$$\Delta(x,y) = \frac{1}{q^m}\left|\{a \in \mathbb{F}_q^m : x(a) \neq y(a)\}\right|$$

and the relative distance between x and y with respect to $x_i = b$ as

$$\begin{aligned}
&\Delta_{x_i=b}(x,y)\\
&= \frac{1}{q^{m-1}}\cdot\left|\{a=(a_1,\ldots,a_m)\in\mathbb{F}_q^m : x(a)\neq y(a), a_i = b\}\right|.
\end{aligned}$$

Note that

$$\frac{1}{q}\cdot\left(\sum_{j=0}^{q-1}\Delta_{x_m=b_j}(p,y)\right) = \Delta(p,y).$$

Define the list-decoding size of $\mathrm{RM}_q(r,m)$ with radius η as

$$l_q(r,m,\eta) = \max_{y\in\mathbb{F}_q^{q^m}}\left|\{p\in\mathrm{RM}_q(r,m) : \Delta(p,y)\leq\eta\}\right|.$$

The following simple observation is used in our algorithm.

Lemma 1. *Let p, y be two polynomials. If*

$$\Delta_{x_m=b_0}(p,y) \leq \Delta_{x_m=b_1}(p,y) \leq \cdots \leq \Delta_{x_m=b_{q-1}}(p,y) \quad (5)$$

holds for $\mathbb{F}_q = \{b_0,b_1,\ldots,b_{q-1}\}$, then for $0 \leq i \leq q-1$,

$$\sum_{j=0}^{i}\Delta_{x_m=b_j}(p,y) \leq (i+1)\Delta(p,y).$$

Proof: Since $(1/q)\cdot(\sum_{j=0}^{q-1}\Delta_{x_m=b_j}(p,y)) = \Delta(p,y)$, it follows from the condition (5) that $(1/(i+1))\cdot(\sum_{j=0}^{i}\Delta_{x_m=b_j}(p,y)) \leq \Delta(p,y)$. ∎

Let $y : \mathbb{F}_q^m \to \mathbb{F}_q$ be a received word and $p \in \mathrm{RM}_q(r,m)$ such that $\Delta(p,y) \leq \eta$ and $y = p + e$ for an error $e : \mathbb{F}_q^m \to \mathbb{F}_q$. Decompose y and e into $y = y_0 \circ y_1 \circ \cdots \circ y_{q-1}$ and $e = e_0 \circ e_1 \circ \cdots \circ e_{q-1}$, where $y_i, e_i : \mathbb{F}_q^{m-1} \to \mathbb{F}_q$ for $0 \leq i \leq q-1$. Let assume that $(b_0,b_1,\ldots,b_{q-1}) = (a_{i_0},a_{i_1},\ldots,a_{i_{q-1}})$. Then, by considering (4), we have that

$$\begin{bmatrix} y_0 \\ \vdots \\ y_{q-1}\end{bmatrix} = M\begin{bmatrix} p_0 \\ \vdots \\ p_{q-1}\end{bmatrix} + \begin{bmatrix} e_0 \\ \vdots \\ e_{q-1}\end{bmatrix}, \qquad (6)$$

where $p = p_0 \circ p_1 \circ \cdots \circ p_{q-1}$ and

$$M = \begin{bmatrix} 1 & a_0 - a_{i_0} & \cdots & \prod_{j=0}^{q-2}(a_0 - a_{i_j}) \\ 1 & a_1 - a_{i_0} & \cdots & \prod_{j=0}^{q-2}(a_1 - a_{i_j}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & a_{q-1} - a_{i_0} & \cdots & \prod_{j=0}^{q-2}(a_{q-1} - a_{i_j}) \end{bmatrix}.$$

By arranging the coordinates, (6) can be represented as

$$\begin{bmatrix} y_{i_0} \\ \vdots \\ y_{i_{q-1}}\end{bmatrix} = \tilde{M}\begin{bmatrix} p_0 \\ \vdots \\ p_{q-1}\end{bmatrix} + \begin{bmatrix} e_{i_0} \\ \vdots \\ e_{i_{q-1}}\end{bmatrix}, \qquad (7)$$

where $\tilde{M}$ is defined in (8).

$$\tilde{M} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & a_{i_1} - a_{i_0} & 0 & \cdots & 0 & 0 \\ 1 & a_{i_2} - a_{i_0} & \prod_{l=0}^{1}(a_{i_2} - a_{i_l}) & \cdots & 0 & 0 \\ \vdots & & \vdots & \ddots & \vdots & \vdots \\ 1 & a_{i_{q-2}} - a_{i_0} & \prod_{l=0}^{1}(a_{i_{q-2}} - a_{i_l}) & \cdots & \prod_{l=0}^{q-3}(a_{i_{q-2}} - a_{i_l}) & 0 \\ 1 & a_{i_{q-1}} - a_{i_0} & \prod_{l=0}^{1}(a_{i_{q-1}} - a_{i_l}) & \cdots & \prod_{l=0}^{q-3}(a_{i_{q-1}} - a_{i_l}) & \prod_{l=0}^{q-2}(a_{i_{q-1}} - a_{i_l}) \end{bmatrix} \tag{8}$$

Next we describe the idea of our algorithm. Suppose (5) holds. We have $y_{i_0} = p_0 + e_{i_0}$ from (7). By Lemma 1, p_0 can be recovered if we use a list-decoder for $\mathrm{RM}_q(r, m-1)$ with radius η and take y_{i_0} as an input to the decoder. Next, we consider recovering p_1 from $y_{i_1} = p_0 + (a_{i_1} - a_{i_0})p_1 + e_{i_1}$ without using p_0. We use y_{i_0} instead of p_0. Namely, in order to recover p_1, we set $(y_{i_1} - y_{i_0})/(a_{i_1} - a_{i_0})$ as an input to the decoder. Then the error radius will be at most 2η by Lemma 1. Thus p_1 can be recovered if we use a list-decoder for $\mathrm{RM}_q(r-1, m-1)$ with radius 2η. Generally, we can recover p_j from $y_{i_0}, \ldots, y_{i_j}$ without using $p_0, \ldots, p_{j-1}$ if we use a list-decoder for for $\mathrm{RM}_q(r-j, m-1)$ with radius $(j+1)\eta$.

In the above argument, we use the following relations.

$$\begin{bmatrix} w_0 \\ \vdots \\ w_{q-1} \end{bmatrix} = \tilde{M}^{-1} \begin{bmatrix} y_{i_0} \\ \vdots \\ y_{i_{q-1}} \end{bmatrix} = \begin{bmatrix} p_0 \\ \vdots \\ p_{q-1} \end{bmatrix} + \tilde{M}^{-1} \begin{bmatrix} e_{i_0} \\ \vdots \\ e_{i_{q-1}} \end{bmatrix} \tag{9}$$

where $[w_0, \ldots, w_{q-1}]$ is the tuplet of w_j, which is the input to a list-decoder with radius $(j+1)\eta$ for recovering p_j. Since $\tilde{M}$ is lower triangular and non-singular, the inverse $\tilde{M}^{-1}$ exists and is lower triangular.

To implement the above idea, since we do not know which $(b_0, b_1, \ldots, b_{q-1})$ satisfies (5), we test all $q!$ permutations of $(b_0, b_1, \ldots, b_{q-1})$ in the algorithm. Our list-decoding algorithm is presented in Algorithm 1. In the base case when $r = 1$, we use the brute-force algorithm.

Algorithm 1 LIST-DEC$_q(r, m, \eta, y)$.

Input: $r, m, \eta, y : \mathbb{F}_q^m \to \mathbb{F}_q$.
Output: All $p : \mathbb{F}_q^m \to \mathbb{F}_q$ such that $\deg(p) \le r$ and $\Delta(p, y) \le \eta$.

1: For all permutations $(b_0, b_1, \ldots, b_{q-1})$,
2: Calculate $[w_0, \ldots, w_{q-1}]$ in (9).
3: For $i = 0, \ldots, q-1$,
4: $L_i \leftarrow$ LIST-DEC$_q(r-i, m-1, (i+1)\eta, w_i)$.
5: For all possible $p_i \in L_i$,
6: Set $p(x_1, \ldots, x_m) = \sum_{i=0}^{q-1} p_i(x_1, \ldots, x_{m-1}) \prod_{j=0}^{i-1}(x_m - b_j)$.
7: If $\Delta(p, y) \le \eta$, then add p to L.
8: Return L.

The difference between our algorithm and that of [4] is in Steps 3–6 of Algorithm 1. Our algorithm allows parallel computation in these steps. As described above, each input w_i to the decoder is computed as in (9). Therefore, the q calls of the decoder can be computed in parallel. In the algorithm of [4], each input to the decoder must be computed sequentially, but their algorithm can handle less errorneous inputs than ours.

In our algorithm, LIST-DEC$_q(r, m, \eta, y)$ calls LIST-DEC$_q(r-i, m-1, (i+1)\eta, w_i)$ for $0 \le i \le q-1$ recursively. To analyze the performance of the algorithm, including the error radius and the time complexity, we need to estimate the performance of LIST-DEC$_q(r-i, m-1, (i+1)\eta, w_i)$ for $0 \le i \le q-1$. In the algorithm of [4], where the algorithm for $\mathrm{RM}_q(r, m)$ with radius η calls the sub-algorithms for $\mathrm{RM}_q(r-i, m-1)$ for $0 \le i \le q-1$, the radius each sub-algorithm needs to handle is $\frac{q}{q-i}\eta$, whereas our sub-algorithm needs to handle the radius $(i+1)\eta$. It is shown in [4] that the list size for sub-algorithms does not increase; Namely, they show that

$$l_q(r-i, m-1, \frac{q}{q-i}\eta) \le l_q(r, m, \eta)$$

for any r, m, and η. Therefore, they can bound the time complexity of their algorithm by using the list size $l_q(r, m, \eta)$. However, we could not find any such bound for $l_q(r-i, m-1, (i+1)\eta)$, so we failed to estimate the performance of our algorithm in terms of list size. Let $T_q(r, m, \eta)$ be the time complexity for our algorithm LIST-DEC$_q(r, m, \eta, y)$. Then we have that

$$\begin{aligned} &T_q(r, m, \eta) \\ &\le (q!)\Bigg(O(q^3) + \sum_{j=0}^{q-1} T_q\big(r-j, m-1, (j+1)\eta\big) \\ &\quad + \Bigg(\prod_{j=0}^{q-1} l_q\big(r-j, m-1, (j+1)\eta\big)\Bigg) \cdot O(q^m) \cdot O(q^m) \Bigg). \end{aligned}$$

We need a further study to bound the time complexity. Since the gap between $l_q(r-i, m-1, \frac{q}{q-i}\eta)$ and $l_q(r-i, m-1, (i+1)\eta)$ is small when q is small (in particular, when $q = 3$, the gap is only between $l_3(r-1, m-1, \frac{3}{2}\eta)$ and $l_3(r-1, m-1, 2\eta)$), we believe that our algorithm could be faster for some range of parameters, say small q and relatively large r.

III. LIST-DECODING FOR POLAR CODES

In this section, we explore codes to which the list-decoding algorithm presented in [4], [2] and the previous section, for short the GKZ-type list-decoding, can be applied. Here we consider codes only over $\mathbb{F}_2$.

Polar codes are the capacity achieving codes introduced by Arikan [1] for arbitrary symmetric binary-input discrete memoryless channels. Polar codes can be seen as a generalization of Reed-Muller codes. Let $G = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$. A polar code of length 2^n is a code with generator matrix whose rows are chosen from rows of $G^{\otimes n}$, where $\otimes n$ is the n-th power of Kronecker product. The generator matrix of $\mathrm{RM}_2(n,n)$ is equal to $G^{\otimes n}$. Interestingly, using the correspondence between vectors and polynomials described in Section II-A, the set of rows of $G^{\otimes n}$ is the set of monomials with total degree at most n. Then $\mathrm{RM}_2(r,n)$ is the polar code of length 2^n whose generator matrix consists of monomials with total degree at most r.

We say a code $C \subseteq \mathrm{RM}_2(r,n)$ is *decomposable* if there are $C_0 \subseteq \mathrm{RM}_2(r,n-1)$ and $C_1 \subseteq \mathrm{RM}_2(r-1,n-1)$ such that

$$C = \{p_0(x_1,\ldots,x_{n-1}) + (x_n - b)p_1(x_1,\ldots,x_{n-1}) : \\ p_0 \in C_0, p_1 \in C_1\}$$

for $b \in \mathbb{F}_2$. Furthermore, if C_0 and C_1 are also decomposable, we say C is *recursively decomposable*. It is not difficult see that if C is recursively decomposable, then we can apply the GKZ-type list-decoding to C.

Let $C \subseteq \mathrm{RM}_2(n,n)$ be a code of length 2^n. Let $G(C)$ denote the set of vectors of a generator matrix for C. Note that $G(C)$ consists of monomials of total degree at most n. Define $\deg(C)$ to be the minimum integer r such that $C \subseteq \mathrm{RM}_2(r,n)$. Let define $C(i,j) = C \cap \mathrm{RM}_2(i,j)$ for $0 \le i \le \deg(C)$ and $n - \deg(C) \le j \le n$. We characterize decomposable codes as follows.

Lemma 2. *Let $C \subseteq \mathrm{RM}_2(n,n)$ be a code of length 2^n. Then C is decomposable if $x_n p(x_1,\ldots,x_{n-1}) \in G(C)$ implies $p(x_1,\ldots,x_{n-1}) \in G(C)$.*

Proof: Suppose $\deg(C) = r$. Let $C_0 = C(r,n-1)$ and $C_1 = C(r-1,n-1)$. Then any polynomial $p \in C$ is uniquely represented as $p = p_0 + x_n p_1$ for some $p_0 \in C_0$ and $p_1 \in \mathrm{RM}_2(r-1,n-1)$. Then it follows that $p_1 \in C_1$ from the assumption that $x_n p' \in G(C)$ implies $p' \in G(C)$. Also p can be represented as $p = p'_0 + (x_n - 1)p_1$ for some $p'_0 \in C_0$ since $C \supseteq C_0 \supseteq C_1$. Thus C is decomposable. ∎

From the above lemma, we conclude as follows.

Theorem 1. *Let $C \subseteq \mathrm{RM}_2(r,n)$ be a code of length 2^n and $\deg(C) = r$. If $C(i,j)$ is such that $x_i p(x_1,\ldots,x_{i-1}) \in G(C(i,j))$ implies $p(x_1,\ldots,x_{i-1}) \in G(C(i,j))$ for all $0 \le i \le \deg(C)$ and $n - \deg(C) \le j \le n$, then the GKZ-type list-decoding can be applied to C.*

IV. Conclusion

In this paper, we have proposed a list-decoding algorithm for non-binary Reed-Muller codes and have explored the possibilities of application of the decoding to more general class of codes, in particular polar codes. Future work is to analyze the performance of our algorithm and to present interesting classes of codes that the GKZ-type algorithm can be applied to.

References

[1] E. Arikan, "Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[2] I. Dumer, G. Kabatiansky, and C. Tavernier, "On the complexity of decoding Reed-Muller codes within their code distance," in Proc. *Eleventh International Workshop on Algebraic and Combinatorial Coding Theory*, pp. 82–85, 2008.

[3] O. Goldreich and L. Levin, "A hard-core predicate for all one-way functions," in Proc. *the 21st ACM Symposium on Theory of Computing (STOC)*, pp. 25–32, 1989.

[4] P. Gopalan, A.R. Klivans, and D. Zuckerman, "List-decoding Reed-Muller codes over small fields," in Proc. *the 40th ACM Symposium on Theory of Computing (STOC)*, pp. 265–274, 2008.

[5] V. Guruswami, "List decoding of error-correcting codes," *Lecture Notes in Computer Science*, vol. 3282, Springer, 2004.

[6] V. Guruswami, "Algorithmic results in list decoding," *Foundations and Trends in Theoretical Computer Science*, vol. 2, no. 2, 2007.

[7] V. Guruswami, "List decoding of binary codes – a brief survey of some recent results," in Proc. *Second International Workshop, Coding and Cryptology (IWCC 2009)*, *Lecture Notes in Computer Science*, vol. 5557, pp. 97–106, 2009.

[8] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977.

[9] M. Plotkin, "Binary codes with specified minimum distance," *IRE Transactions on Information Theory*, vol. 6, no. 4, pp. 445–450, 1960.

[10] M. Sudan, "Decoding of Reed-Solomon codes beyond the error-correction bound," *Journal of Complexity*, vol. 13, no. 1, pp. 180–193, 1997.

[11] M. Sudan, "List decoding: algorithms and applications," *ACM SIGACT News*, vol. 31, pp. 16–27, 2000.