Thirteenth International Workshop on Algebraic and Combinatorial Coding Theory
June 15-21, 2012, Pomorie, Bulgaria *pp. 151–156*

List decoding of Reed-Muller codes with linear complexity up to the Johnson bound

Rafaël Fourquet rafael.fourquet@gmail.com
University Paris 8 Saint-Denis, France

Abstract. We propose a deterministic list decoding algorithm for binary Reed-Muller codes $\mathrm{RM}(r,m)$ of order r in m variables. Given a decoding radius T arbitrarily close to the Johnson Bound, the algorithm outputs the list of all codewords from $\mathrm{RM}(r,m)$ which are at Hamming distance at most T from the received vector with a linear complexity in the length 2^m of the code. The memory complexity is also linear. This algorithm is a generalization to any order of the Kabatiansky-Tavernier algorithm *sums* for order 1.

1 Introduction

The binary Reed-Muller (RM) code $\mathrm{RM}(r,m)$, for integers r and m such that $0 \le r \le m$, is a linear code over $\mathbb{F}_2$ (the field with two elements) of length $n = 2^m$, dimension $k = \sum_{i=0}^{r} \binom{m}{i}$ and minimal distance $d = 2^{m-r}$. This family of codes has been studied for more than fifty years, and many decoding algorithms were proposed. However, few of them are deterministic list decoding algorithms. In this category, the most important ones are from [2, 7] for the first order RM codes, [5, 8] for the second order, and [1, 3, 6] for any order. The best complexity for any order r, which is quasi-linear in the length of the code (of order $\mathcal{O}\left(2^m m^{r-1}\right)$), was given in [3]; it applies up to the minimal distance of the code. The complexity of the proposed algorithm is linear in the length of the code, but applies only up to the Johnson bound. A similar result was obtained recently in [4] with soft-decision decoding.

2 List decoding

The set of m-variable Boolean functions $f : \mathbb{F}_2^m \to \mathbb{F}_2$ is denoted by $\mathcal{B}_m$. A Boolean function f can be uniquely represented as a polynomial belonging to $\mathbb{F}_2[x_1, \dots, x_m]/(x_1^2 - x_1, \dots, x_m^2 - x_m)$, called its algebraic normal form (ANF). The algebraic degree $\deg f$ of f is the degree of its ANF. The function f can also be uniquely represented by its *truth-table*, i.e. the binary vector $(\dots, f(x_1, \dots, x_m), \dots) \in \mathbb{F}_2^n$ of length $n = 2^m$ consisting of all values $f(x)$ for $x \in \mathbb{F}_2^m$ (a total order is fixed in $\mathbb{F}_2^m$). The RM code $\mathrm{RM}(r,m) \subset \mathbb{F}_2^n$ is the set of truth-tables associated to the Boolean functions of degrees at most r: $\mathrm{RM}(r,m) \simeq \{f \in \mathcal{B}_m \ : \ \deg f \le r\}$. We denote by $\mathrm{RM}^{\times}(r,m) \subset \mathrm{RM}(r,m)$ the subset of elements whose constant term (in the ANF) is 0 (the null function), that is $\mathrm{RM}^\times(r,m) \simeq \mathrm{RM}(r,m)/\mathrm{RM}(0,m)$. The Hamming distance $\mathrm{d}(f,g)$ between two functions f and g is defined by $\mathrm{d}(f,g) \coloneqq |\{x \in \mathbb{F}_2^m \ : \ f(x) \neq g(x)\}|$. The hamming weight of f is defined by $\mathrm{wt}(f) \coloneqq \mathrm{d}(f,0)$.

Definition 1 (List Decoding)**.** *Let $f \in \mathcal{B}_m$ be a received vector, $\varepsilon \in (0,1)$. A deterministic list decoding algorithm for the* $\mathrm{RM}(r,m)$ *code outputs the list*

$$\mathcal{L}_\varepsilon(f) \coloneqq \left\{q \in \mathrm{RM}(r,m) \ : \ \mathrm{d}(f,q) \leq 2^{m-1}(1-\varepsilon)\right\}. \tag{1}$$

We will use in the sequel a particular representation for a Boolean function in $\mathrm{RM}(r,m)$, which is related to the well-known Plotkin construction:

Definition 2 (i-coefficient and i-prefix)**.** *Let $q \in \mathrm{RM}(r,m)$. The i-coefficients $q_{[i]}(x_{i+1},\ldots,x_m) \in \mathrm{RM}(r-1,m-i)$ of q, $1 \leq i \leq m$, are uniquely defined by*

$$q(x_1,\ldots,x_m) = q(0) + \sum_{i=1}^{m} x_i q_{[i]}(x_{i+1},\ldots,x_m). \tag{2}$$

The i-prefix $q^{[i]} \in \mathrm{RM}^\times(r,m)$ of q is defined by $q^{[i]} = x_1 q_{[1]} + \cdots + x_i q_{[i]}$.

In particular, either $q = q^{[m]}$ or $q = q^{[m]} + 1$. The proposed algorithm will construct recursively potential i-prefixes of the solutions of $\mathcal{L}_\varepsilon(f)$. We denote

$$\mathcal{F}(f) \coloneqq \sum_{x \in \mathbb{F}_2^m} (-1)^{f(x)} = 2^m - 2\mathrm{d}(f,0). \tag{3}$$

We then have

$$q \text{ or } q+1 \in \mathcal{L}_\varepsilon(f) \iff |\mathcal{F}(f+q)| = \left|\mathcal{F}(f+q^{[m]})\right| \geq 2^m\varepsilon. \tag{4}$$

Let $f \in \mathcal{B}_m$, $0 \leq i \leq m$ and $\alpha \in \mathbb{F}_2^{m-i}$. We denote by $f_\alpha \in \mathcal{B}_i$ the restriction of f to the facet $\left\{(x,\alpha) \ : \ x \in \mathbb{F}_2^i\right\}$ (i.e. $\forall x \in \mathbb{F}_2^i, f_\alpha(x) = f(x,\alpha)$).

The two following lemmas are easy to prove (using the relation $q^{[i]} = q^{[i-1]} + x_i q_{[i]}$):

Lemma 1. *Let $i \leq m, x \in \mathbb{F}_2^{i-1}$ and $\alpha \in \mathbb{F}_2^{m-i}$. We have:*

$$q_\alpha^{[i]}(x,0) = q_{(0,\alpha)}^{[i-1]} \tag{5}$$

$$q_\alpha^{[i]}(x,1) = q_{(1,\alpha)}^{[i-1]} + q_{[i]}(\alpha) \tag{6}$$

Lemma 2. *Let $q = q^{[m]} \in \mathrm{RM}^{\times}(r,m)$ and $1 \le i \le m$. For all $\alpha \in \mathbb{F}_2^{m-i}$, we have*

$$q_\alpha^{[m]} = q_\alpha^{[i]} + q_\alpha(0,\dots,0), \tag{7}$$

and hence:

$$\mathcal{F}(f + q^{[m]}) = \sum_{\alpha \in \mathbb{F}_2^{m-i}} (-1)^{q_\alpha(0,\dots,0)} \mathcal{F}(f_\alpha + q_\alpha^{[i]}). \tag{8}$$

Applying the triangle inequality on (8) and using (4), we obtain:

$$q \in \mathcal{L}_\varepsilon(f) \Longrightarrow \Sigma_f^i(q) := \sum_{\alpha \in \mathbb{F}_2^{m-i}} \left| \mathcal{F}(f_\alpha + q_\alpha^{[i]}) \right| \ge 2^m \varepsilon \tag{9}$$

This quantity denoted $\Sigma_f^i(q)$ depends only on the i-prefix $q^{[i]}$ of q. The relation (9) gives a criterion to decide if a candidate $q^{[i]}$ *may* be the i-prefix of a solution $q \in \mathcal{L}_\varepsilon(f)$; we denote by $\mathcal{L}_\varepsilon^i(f)$ the list of such candidates:

$$\mathcal{L}_\varepsilon^i(f) := \left\{ q^{[i]} \in \mathrm{RM}(r,m) \;:\; \Sigma_f^i(q^{[i]}) \ge 2^m \varepsilon \right\} \tag{10}$$

From an iterative point of view, the algorithm will determine the intermediate lists $\mathcal{L}_\varepsilon^i(f)$ for $1 \le i \le m$ by computing at the i-th step the criterion $\Sigma_f^i(q^{[i]})$ only for the functions of the form $q^{[i]} = q^{[i-1]} + x_i q_{[i]}$, for all $q^{[i-1]} \in \mathcal{L}_\varepsilon^{i-1}(f)$ and $q_{[i]} \in \mathrm{RM}(r-1, m-i)$ (a function $q^{[i]}$ can be in $\mathcal{L}_\varepsilon^i(f)$ only if its prefix $q^{[i-1]}$ is in $\mathcal{L}_\varepsilon^{i-1}(f)$).

3 A recursive algorithm

For efficiency reasons, instead of computing the whole lists $\mathcal{L}_\varepsilon^i(f)$ one after the other, the algorithm computes at the i-th step, for a given prefix $q^{[i-1]} \in \mathcal{L}_\varepsilon^{i-1}(f)$, all the elements in $\mathcal{L}_\varepsilon^i(f)$ admitting $q^{[i-1]}$ as an $(i-1)$-prefix, i.e. of the form $q^{[i-1]} + x_i q_{[i]}$. The corresponding i-coefficients $q_{[i]}$ are stored in a partial list $L_i := \left\{ q_{[i]} \in \mathrm{RM}(r-1, m-i) \;:\; \Sigma_f^i(q^{[i-1]} + x_i q_{[i]}) \ge 2^m \varepsilon \right\}$, and the algorithm is recursively run on each of them; then another element in $\mathcal{L}_\varepsilon^{i-1}(f)$ is selected and the same operations are repeated.

In addition to the lists L_i, we use an array F_i of size 2^{m-i} defined by $F_i(\alpha) := \mathcal{F}(f_\alpha + q_\alpha^{[i]})$ for all $\alpha \in \mathbb{F}_2^{m-i}$ (the total memory complexity for storing the arrays F_i is then $\mathcal{O}\left(\sum_{i=1}^m 2^{m-i}\right) = \mathcal{O}(2^m)$). The values of F_i can be computed from F_{i-1} and $q_{[i]}$ with complexity $\mathcal{O}\left(2^{m-i}\right)$ using the relation[1]

$$F_i(\alpha) = F_{i-1}((0,\alpha)) + (-1)^{q_{[i]}(\alpha)} F_{i-1}((1,\alpha)) \tag{11}$$

[1]cf. equation (8). Evaluating $q_{[i]}$ on $\mathbb{F}_2^{m-i}$ has complexity in $\mathcal{O}\left(2^{m-i}\right)$ for *fixed* r.

Given $q^{[i-1]} \in \mathcal{L}_\varepsilon^{i-1}(f)$, the list L_i can be computed from F_{i-1}, with an algorithm $\Theta(\varepsilon, F_{i-1})$, precised later.

We can now give the recursive description of the proposed algorithm, denoted Γ_r. It takes as input ε, i, a prefix $q^{[i-1]}$ and F_{i-1} (corresponding to $q^{[i-1]}$). The output is the set of functions from $\mathcal{L}_\varepsilon(f)$ admitting $q^{[i-1]}$ as prefix (so the output is $\mathcal{L}_\varepsilon(f)$ itself when the algorithm is called with input $i = 1$ and $q^{[0]} = 0$).

```
ALGORITHM $\Gamma_r$
**Input:** $\varepsilon, i, q^{[i-1]}, F_{i-1}$
**Initial input:** $\varepsilon, i = 1, q^{[0]} = 0, F_0 = \left\{(-1)^{f(\alpha)} \;:\; \alpha \in \mathbb{F}_2^m\right\}$
**Output:** Functions from $\mathcal{L}_\varepsilon(f)$ admitting $q^{[i-1]}$ as $(i-1)$-prefix
**if** $i \le m$ **then:**
    compute $L_i = \Theta(\varepsilon, F_{i-1})$
    **for each** $q_{[i]} \in L_i$ **do:**
        $q^{[i]} \leftarrow q^{[i-1]} + x_i q_{[i]}$
        compute $F_i$ from $F_{i-1}$ and $q_{[i]}$ (cf. (11))
        **call** $\Gamma_r(\varepsilon, i+1, q^{[i]}, F_i)$
**if** $i = m + 1$ **then:**
    **output** $\begin{cases} q^{[m]} & \text{if } F_m(0) > 0 \\ q^{[m]} + 1 & \text{if } F_m(0) < 0 \end{cases}$
```

We now give the Θ algorithm. For $\alpha \in \mathbb{F}_2^{m-i}$, and $l \in \mathrm{RM}(r-1, m-i)$, let

$$\begin{aligned} V(0,\alpha) &:= |F_{i-1}((0,\alpha)) + F_{i-1}((1,\alpha))| \\ V(1,\alpha) &:= |F_{i-1}((0,\alpha)) - F_{i-1}((1,\alpha))| \\ M(l) &:= \sum_{\alpha \in \mathbb{F}_2^{m-i}} V(l(\alpha), \alpha) \end{aligned}$$

By definition we have $\Sigma_f^i(q^{[i-1]} + x_i l) = M(l)$. Let

$$\begin{aligned} D(\alpha) &:= (V(0,\alpha) - V(1,\alpha))/2 \\ S(\alpha) &:= (V(0,\alpha) + V(1,\alpha))/2 \ge 0. \end{aligned}$$

Then

$$V(l(\alpha), \alpha) = S(\alpha) + (-1)^{l(\alpha)} D(\alpha)$$

and hence

$$M(l) = \sum_{\alpha \in \mathbb{F}_2^{m-i}} S(\alpha) + \sum_{\alpha \in \mathbb{F}_2^{m-i}} (-1)^{l(\alpha)} D(\alpha).$$

Finally, we have

$$l \in L_i \iff M(l) \ge 2^m \varepsilon \quad \iff \sum_{\alpha \in \mathbb{F}_2^{m-i}} (-1)^{l(\alpha)} D(\alpha) \ge 2^m \varepsilon - \sum_{\alpha \in \mathbb{F}_2^{m-i}} S(\alpha).$$

By denoting $\varepsilon' := 2^i\varepsilon - 2^{i-m}\sum_{\alpha\in\mathbb{F}_2^{m-i}} S(\alpha)$, the last inequality means that the list L_i is the output of the algorithm $\Gamma_{r-1}(\varepsilon', 1, 0, D)$

4 Complexity

We first give an upper bound on the size of the lists. Let $\varepsilon > \varepsilon_r := \sqrt{1-2^{1-r}}$. Then the size of the list $\mathcal{L}_\varepsilon(f)$ is upper bounded by the well-known Johnson bound: $|\mathcal{L}_\varepsilon(f)| \le J_r := \frac{2^{1-r}}{\varepsilon^2-1+2^{1-r}}$. The crucial point in the proof of this bound is that the minimal distance in $\mathrm{RM}(r,m)$ is 2^{m-r}.

Lemma 3. *For $\varepsilon > \varepsilon_r$, $\left|\mathcal{L}^i_\varepsilon(f)\right| \le J_r$.*

Proof. Let $q^{[i]} \in \mathcal{L}^i_\varepsilon(f)$: according to (9), there exists a function $z(\alpha) \in \mathcal{B}_{m-i}$ such that $\sum_{\alpha\in\mathbb{F}_2^{m-i}} \mathcal{F}(f_\alpha + q^{[i]}_\alpha + z(\alpha)) \ge 2^m\varepsilon$. We have to upper bound the number of functions q' of the form $q' = q^{[i]} + z$ such that $\mathrm{d}(f, q') \le 2^{m-1}(1-\varepsilon)$, "modulo the functions z" [2]. We will show that the weight of such a non-null function q' (where z is chosen to minimize the weight) is at least 2^{m-r}. We could then conclude with a proof similar to the Johnson bound one. We first notice that the component[3] R of degree r in $q^{[i]}_\alpha$ does not depend on α ($\forall\alpha,\beta\in\mathbb{F}_2^{m-i}, q^{[i]}_\alpha + q^{[i]}_\beta \in \mathrm{RM}(r-1,i)$). If R is non-null, then $\mathrm{wt}(q^{[i]}_\alpha) \ge 2^{i-r}$, and hence $\mathrm{wt}(q') \ge 2^{m-r}$. Suppose now $R = 0$. For α such that $q'_\alpha \ne 0$, $\mathrm{wt}(q'_\alpha) \ge 2^{i-r+1}$ (as $\deg q^{[i]}_\alpha < r$), and it is easy to check that $q'_\alpha \ne 0$ for at least 2^{m-i-1} vectors $\alpha \in \mathbb{F}_2^{m-i}$, hence $\mathrm{wt}(q') \ge 2^{m-r}$. □

Theorem 1. *For fixed r and $\varepsilon > \sqrt{1-2^{1-r}}$, the algorithm Γ_r in $\mathrm{RM}(r,m)$ has linear complexity.*

Proof. For a given upper bound l on the sizes of the intermediate lists $\mathcal{L}^i_\varepsilon(f)$, we denote by $\gamma_{r,m}(l)$ the corresponding complexity for the algorithm Γ_r in m variables. We can show by induction on r that $\gamma_{r,m}(l) = \mathcal{O}(l^r 2^m)$, using the following relation (derived from the description page 154):

$$\gamma_{r,m}(l) = \Big(\sum_{i=1}^{m} \underbrace{l}_{\left|\mathcal{L}^{i-1}_\varepsilon(f)\right|} \times (\underbrace{\mathcal{O}\left(2^{m-i+1}\right)}_{\text{computing } F_{i-1}} + \underbrace{\gamma_{r-1,m-i}(l))}_{\text{computing } L_i}\Big)$$

We conclude by using the bound of lemma 3 for l. □

[2]Two functions $q'_1 = q^{[i]} + z_1$ and $q'_2 = q^{[i]} + z_2$ are equivalent and count for one.

[3]i.e. the sum of the monomials in the ANF of $q^{[i]}_\alpha$ which are of degree r.

5 Conclusion

The proposed deterministic list decoding algorithm has linear complexity up to the Johnson bound. Evaluating its complexity up to the minimal distance is an open problem. In practice, this algorithm can sometimes decode well beyond the Johnson bound (and the minimal distance). For example, the non-linearity profile[4] $(112, 82, 48, 22, 8, 2, 0)$ of the inverse function in 8 variables $(\mathrm{trace}(x^{254}))$ could be obtained within few minutes: this algorithms can be a powerful tool for the study of (cryptographic) Boolean functions.

References

[1] I. Dumer, G. Kabatiansky, and C. Tavernier. List decoding of reed-muller codes up to the johnson bound with almost linear complexity. In *ISIT*, 2006.

[2] I. Dumer, G. Kabatiansky, and C. Tavernier. List decoding of the first-order binary reed-muller codes. *Problems of Information Transmission*, 43(3):225–232, 2007.

[3] I. Dumer, G. Kabatiansky, and C. Tavernier. On complexity of decoding reed-muller codes within their code distance. *Proceedings ACCT-11*, juin 2008.

[4] I. Dumer, G. Kabatiansky, and C. Tavernier. Soft-decision list decoding of Reed-Muller codes with linear complexity. *IEEE symposium ISIT*, 2011.

[5] Rafaël Fourquet and Cédric Tavernier. An improved list decoding algorithm for the second order Reed-Muller codes and its applications. *Des. Codes Cryptography*, 49(1-3):323–340, 2008.

[6] P. Gopalan, A. R. Klivans, and D. Zuckerman. List-decoding reed-muller codes over small fields. *Proc. 40th ACM Symp. Theory Computing (STOC)*, pp. 265-274, 2008.

[7] G. Kabatiansky and C. Tavernier. List decoding of Reed-Muller codes of first order. In *Ninth International Workshop on Algebraic and Combinatorial Coding Theory, ACCT'2004*, pages 230–235, June 2004.

[8] G. Kabatiansky and C. Tavernier. List decoding of second order Reed-Muller codes. In *in Proc. 8th Intern. Simp. Comm. Theory and Applications, Ambleside*, June 2005.

[4]The non-linearity $\mathrm{nl}_r(f)$ of order r of a function f is defined by $\mathrm{nl}_r(f) = \min\{\mathrm{d}(f,g) : g \in \mathrm{RM}(r,m)\}$. The non-linearity profile of f is the sequence $(\mathrm{nl}_1(f), \ldots, \mathrm{nl}_{m-1}(f))$.