Thirteenth International Workshop on Algebraic and Combinatorial Coding Theory
June 15-21, 2012, Pomorie, Bulgaria *pp. 7–14*

The Goldreich-Levin algorithm with reduced complexity [1]

Ameriah Salem Abdouli ameirah.abdouli@kustar.ac.ae
Khalifa University of Science, Technology and Research(KUSTAR)& NKC,UAE
Ilya Dumer dumer@ee.ucr.edu
University of California at Riverside, USA
Grigory Kabatiansky kaba@iitp.ru
KUSTAR, UAE, on leave from ITTP RAS, Moscow
Cedric Tavernier tavernier.cedric@gmail.com
NKC,UAE

Abstract. The celebrated Goldreich-Levin algorithm performs randomized list decoding of the Reed-Muller codes $\mathrm{RM}(1,m)$ of length $n=2^m$ within the decoding radius $\frac{n}{2}(1-\epsilon)$ for any $\epsilon>0$, and achieves a low decoding error probability of 2^{-s} with a polylogarithmic complexity sm/ϵ^4. The deterministic, error-free, Green algorithm performs complete maximum likelihood decoding with complexity $n\ln^2 n$. We combine the two algorithms and reduce the complexity of the Goldreich-Levin algorithm to the order of sm/ϵ^2. Here the output can include the information blocks of ϵ^{-2} reconstructed codewords, which require m/ϵ^2 bits, and the order of sm/ϵ^2 gives the lowest complexity known for any given ϵ. Another particular case is near-capacity decoding, wherein we take parameter ϵ of a vanishing rate $m/\sqrt{n}$. Then the proposed algorithm has complexity order of $n\ln^2 n$, instead of the quadratic order n^2 of the Goldreich-Levin algorithm.

1 Introduction

Decoding of the first-order Reed-Muller codes $\mathrm{RM}(1,m)$ has been addressed in many papers since 1960s. In particular, the maximum likelihood algorithm designed by Green [1] for codes $\mathrm{RM}(1,m)$ uses Fast Fourier Transform (FFT) and sorts out all codewords with respect to their distances from the received vector. The algorithm requires the order of $n\ln^2 n$ bit operations, where $n=2^m$ is the code length. For any $\epsilon\in(0,1)$, the deterministic algorithm of [6] performs error-free list decoding of any received vector $\mathbf{g}$ within the decoding radius of $\frac{n}{2}(1-\epsilon)$ and requires a linear complexity order of $n\ln^2\epsilon$.

In the area of randomized algorithms, a major breakthrough was achieved by Goldreich and Levin [2], with later refinements of [3]- [5]. For any received

[1]The work of I. Dumer was supported by NSF grant 1102074 and ARO grant W911NF-11-1-0027. The work of G.Kabatiansky was supported by RFBR grants 11-01-00735 and 12-01-00905.

vector $\mathbf{g}$, any fixed positive number $s > 1$ and fixed number $\epsilon \in (0, 1)$, the latest version [5] of the Goldreich-Levin (GL) algorithm has low poly-logarithmic complexity $(sm \ln m)/\epsilon^4$ as $m \to \infty$, and with high probability $1-2^{-s}$ retrieves all codewords of RM$(1, m)$ within the decoding radius $\frac{n}{2}(1-\epsilon)$.

For fixed parameters ϵ and s, the GL algorithm drastically reduces the asymptotic complexity of the error-free algorithms [1] and [6]. On the other hand, note that near channel capacity, we can consider the decoding radius $\frac{n}{2}(1-\epsilon)$ of the code RM(1,m) with vanishing parameter $\epsilon = m/\sqrt{n}$ as $n \to \infty$. In this case, the GL algorithm has unacceptably high complexity of quadratic order $n^2 s$. Therefore, our main goal is to reduce complexity of the GL algorithm in parameter ϵ. Below we consider general parameters $\epsilon = \epsilon_m$ and $s = s_m$ that are not necessarily fixed as $m \to \infty$. All logarithms are taken base 2. Our main goal is to prove the following

Theorem 1. *Let $m \to \infty$, $n = 2^m$, $\epsilon \in \left(\frac{m}{\sqrt{n}}, 1\right)$ and let*

$$1 < s \le n\epsilon^2/80. \tag{1}$$

Then for any received vector $\mathbf{g} \in \{0,1\}^n$, list decoding of the first-order code RM$(1, m)$ *can reconstruct all codewords located within distance $\frac{n}{2}(1-\epsilon)$ from $\mathbf{g}$ with probability no less than $1 - 2^{-s}$ and poly-logarithmic complexity*

$$\mathcal{O}\left(\left(s + \log \frac{m}{\epsilon^2}\right) \frac{m}{\epsilon^2} \log^2 \epsilon\right). \tag{2}$$

In particular, for highly-noisy case of $\epsilon = m/\sqrt{n}$, the proposed algorithm removes the complexity gap between maximum likelihood decoding of the Green machine [1] and the randomized decoding of the GL algorithms of [2]- [5].

Note that it is indicated in [3] that the complexity order in ϵ can be reduced by using FFT of linear functions. We use a similar approach in our proof of Theorem 1 and derive the explicit restrictions for the varying paramaters s and ϵ that will help us to reduce the complexity order from ϵ^{-4} to ϵ^{-2}. We also conjecture that the complexity order m/ϵ^2 of Theorem 1 is minimal in both parameters m and ϵ since any algorithm needs at least $m + 1$ operations to output the information bits of any single codeword of RM$(1, m)$ and, on the other hand, the output list can include [6] ϵ^{-2} different codewords for infinitely many m, $\mathbf{g}$, and ϵ.

2 Two versions of the GL algorithm

Below, we describe two simplified versions $GL^{(1)}$ and $GL^{(2)}$ of the original Goldreich-Levin algorithm [2]. The variations of the first algorithm $GL^{(1)}$ are

presented in [3] and [4] (where it is termed A' along with the two other versions, A and A''), and $GL^{(2)}$ is outlined in [5]. We will use both algorithms $GL^{(1)}$ and $GL^{(2)}$ in this section to proceed with further improvements in Section 3. We also modify most estimates of [4] and [5] to address the general setting with varying parameters ϵ and s. In particular, it will be helpful for us to consider an arbitrary level 2^{-s} of the error probability instead of the fixed level 3/4 set in the algorithm $GL^{(2)}$.

Consider an m-dimensional Boolean cube $\mathbb{F}_2^m$ that includes $n = 2^m$ points $x = (x_1, ..., x_m)$. Given any binary string $a = (a_1, \ldots, a_m)$, define a linear Boolean function $a(x)$ as the dot product

$$a(x) = ax = \sum_{j=1}^{m} a_j x_j.$$

Following [4] and [5], in this section we consider binary Hadamard codes $\mathcal{H}(m)$ [1] of length $n = 2^m$ and size n instead of codes $\mathrm{RM}(1, m)$. Each linear function $a(x)$ is represented in $\mathcal{H}(m)$ by the vector $\boldsymbol{a}$ with symbols $a(x)$ obtained as x runs through $\mathbb{F}_2^m$. For any subset $X \subseteq \mathbb{F}_2^m$, let $\boldsymbol{a}(X)$ be the subvector of $\boldsymbol{a}$ defined on positions $x \in X$.

Let parameters m, ϵ, s and the received word $\mathbf{g} \in \mathbb{F}_2^n$ be given. Our goal is to reconstruct the list of functions

$$L_\epsilon(\mathbf{g}) = \{a(x) : d(\mathbf{g}, \boldsymbol{a}) \le \frac{n}{2}(1 - \epsilon)\}.$$

Here each function $a(x)$ will be retrieved as the set of its coefficients $(a_1, .., a_m)$. The algorithm $GL^{(1)}(\mathbf{g}, m, \epsilon, s)$ performs as follows. Let vector $\mathbf{j} = (0...010...0) \in \mathbb{F}_2^m$ have symbol 1 in position j ($j \in \{1, ..., m\}$) and 0s elsewhere. Let

$$l = \left\lceil \log \frac{ms}{\epsilon^2} \right\rceil. \tag{3}$$

We also assume that $l < m$, in which case it suffices to take $s < n\epsilon^2/m$ and $\epsilon > \sqrt{m/n}$. Now we independently and uniformly pick up l vectors $X = \left\{x_{(1)}, ..., x_{(l)}\right\}$ from $\mathbb{F}_2^m$ and consider the linear subspace

$$\mathbb{X} = \left\{ \sum_{i=1}^{l} h_i x_{(i)} \,|\, h_i = 0, 1 \right\}. \tag{4}$$

If $\mathrm{Rank}(\mathbb{X}) < l$ or $\mathbb{X}$ contains some unit vector $\mathbf{j}$, we pick a new subset X. It is easy to verify that l random vectors $x_{(1)}, ..., x_{(l)}$ in $\mathbb{F}_2^m$ are linearly dependent with probability

$$P_X \stackrel{\triangle}{=} \Pr\{\mathrm{Rank}(\mathbb{X}) < l\} \le 2^{l-m}. \tag{5}$$

Performing $i = \lceil \log_2 s \rceil$ trials of choosing $\mathbb{X}$, we obtain an insignificant probability of failure $P_X^i \le s^{l-m} \le s^{-1}$, since $l < m$. Each trial takes about ml^2 operations to verify condition $\mathrm{Rank}(\mathbb{X}) = l$.

Given a string $b = (b_1, ..., b_l) \in \mathbb{F}_2^l$, we will seek any function

$$a_b(x) \in L_\epsilon(\mathbf{g}) : \boldsymbol{a}_b(X) = b. \tag{6}$$

The $GL^(1)$ algorithm inspects all 2^l strings $b \in \mathbb{F}_2^l$ to find all such functions $a_b(x)$. Note that any linear function $a_b(x)$ defined on a basis X is also known at every point x of its span $\mathbb{X}$:

$$\text{if} \quad x = \sum_{i=1}^{l} h_i x_{(i)} \quad \text{then} \quad a_b(x) = \sum_{i=1}^{l} h_i b_i. \tag{7}$$

Our goal is to find the coefficients $a_{j,b}$ of $a_b(x)$, which in fact are the values of the function $a_b(x)$ at the points $\mathbf{j}$:

$$a_{j,b} = a_b(\mathbf{j}), \quad j = 1, ..., m$$

Note that each coefficient $a_b(\mathbf{j})$ satisfies 2^l different equalities

$$a_b(\mathbf{j}) = a_b(x) + a_b(x + \mathbf{j}), \quad x \in \mathbb{X}, \quad j = 1, ..., m. \tag{8}$$

Here the unknown outputs $a_b(x+\mathbf{j})$ will be approximated, i.e., replaced by the channel outputs $\mathbf{g}(x+\mathbf{j})$. The algorithm $GL^{(1)}$ then estimates each $a_b(\mathbf{j})$ taking the 2^l-majority vote

$$\tilde{a}_b(\mathbf{j}) = \mathsf{Maj}_{x \in \mathbb{X}} \left\{ a_b(x) + \mathbf{g}(x + \mathbf{j}) \right\}, \quad j = 1, ..., m. \tag{9}$$

By running all $b \in \mathbb{F}_2^l$, the GL^1 algorithm outputs the entire list of functions

$$\{\tilde{a}_b(x) = \sum_{j=1}^{m} \tilde{a}_b(\mathbf{j}) x_j \mid b \in \mathbb{F}_2^l\}. \tag{10}$$

The proof of the following result can be derived mostly following [4] and standard usage of the Chebyshev inequality.

Theorem 2. *Let $m \to \infty$, $n = 2^m$, $\epsilon \in \left(\sqrt{\frac{m}{n}}, 1\right)$, and $s \in (1, n\epsilon^2/m)$. For any received vector $\mathbf{g} \in \{0,1\}^n$ the algorithm $GL^{(1)}(\mathbf{g}, m, \epsilon, s)$ retrieves any linear function $\boldsymbol{a}(x)$ located within distance $\frac{n}{2}(1-\epsilon)$ from $\mathbf{g}$ with probability no less than $1 - s^{-1}$ and complexity $\mathcal{O}(m^3 s^2 \epsilon^{-4})$.*

Note that Theorem 2 upper-bounds the failure probability $1/s$ for a specific function $a_b(x)$ but not the entire list $L_\epsilon(\mathbf{g})$, which according to the Johnson bound, has size

$$|L_\epsilon(\mathbf{g})| \le \epsilon^{-2}.$$

We now turn to another algorithm, see [5], $GL^{(2)}(\mathbf{g}, m, \epsilon, s)$, that outputs the entire list $L_\epsilon(\mathbf{g})$ with high probability. It also improves the complexity of $GL^{(1)}$. We will also impose some restrictions on parameters m, ϵ, s to obtain exponentially declining error probability 2^{-s}. We take $\epsilon \ge m/n^{1/2}$ and use parameters

$$\lambda = \log \epsilon^{-2} + 4, \quad l = \lceil \lambda \rceil, \quad k = 2\left(s + \log \frac{m}{\epsilon^2}\right). \tag{11}$$

Again, note that $l < m$ for large m. To obtain an l-dimensional subspace $\mathbb{X}$, we use s independent trials to pick up its basis X. This adds insignificant complexity $ml^2 s$ and has low error probability $P_X^s \le 2^{(l-m)s}$.

Next, we substantially increase the number of estimates used in $GL^{(1)}$ to obtain $\tilde{a}_b(\mathbf{j})$. Note that $\mathbb{X}$ has 2^{m-l}cosets in $\mathbb{F}_2^m$. We then perform $9k$ or fewer trials picking up randomly and uniformly random points from $\mathbb{F}_2^m$ (to shorten notation, we use k instead of $\lceil k \rceil$). Later, we will show that with high probability, we can choose k points $y_{(i)}$ among $9k$ points that fall into different cosets of $\mathbb{X}$. Given the sets $\mathbb{X}$ and $Y = \{y_{(i)},\, i = 1, ..., k\}$, the algorithm proceeds as follows.

We first replace the unit vector $\mathbf{j}$ in equalities (8) and (9) with any point $y_{(i)}$. For any b, a function $a_b(x)$ can be estimated at any point $y = y_{(i)}$ as

$$\tilde{a}_b(y_{(i)}) = \mathsf{Maj}_{x \in \mathbb{X}} \left\{ a_b(x) + \mathbf{g}(x + y_{(i)}) \right\}. \tag{12}$$

Similarly, given b and j, we evaluate the same function $a_b(x)$ at the point $y = y_{(i)} + \mathbf{j}$,

$$\tilde{a}_b(y_{(i)} + \mathbf{j}) = \mathsf{Maj}_{x \in \mathbb{X}} \left\{ a_b(x) + \mathbf{g}(x + y_{(i)} + \mathbf{j}) \right\} \tag{13}$$

Given b and j, we can now estimate the coefficient $a_b(\mathbf{j})$ as a function of the point $y = y_{(i)}$ as follows

$$\tilde{a}_{b,i}(\mathbf{j}) = \tilde{a}_b(y_{(i)}) + \tilde{a}_b(y_{(i)} + \mathbf{j}) \tag{14}$$

Now we can combine k estimates $\tilde{a}_{b,i}(\mathbf{j})$ obtained for different i in one majority vote

$$\tilde{a}_b(\mathbf{j}) = \mathsf{Maj}_{i=1,...,k}\, \tilde{a}_{b,i}(\mathbf{j}) \tag{15}$$

The following theorem can be derived from [5].

Theorem 3. *Let* $m \to \infty$, $n = 2^m$, $\epsilon \in \left(\frac{m}{\sqrt{n}}, 1\right)$, *and let positive number* s *satisfy (1). Then for any received vector* $\mathbf{g} \in \{0,1\}^n$, *the algorithm* $GL^{(2)}(\mathbf{g}, m, \epsilon, s)$ *reconstructs the entire list* $L_\epsilon(\mathbf{g})$ *of linear functions located within distance* $\frac{n}{2}(1-\epsilon)$ *from* $\mathbf{g}$ *with probability no less than* $1 - 2^{-s}$ *and complexity*

$$\mathcal{O}\left(m\epsilon^{-4}s + m\epsilon^{-4}\log\frac{m}{\epsilon^2}\right).$$

3 A modified GL algorithm

The GL algorithm has substantially advanced the entire theory of hard-core predicates. However, its complexity grows as ϵ^{-4}, which makes it less efficient from the coding perspective. In particular, recall that its complexity has quadratic order $n^2 s$ if decoder operates near channel capacity. Thus, it substantially exceeds the complexity $\mathcal{O}\left(n\ln^2 n\right)$ of the Green machine, which performs the same task error-free. In what follows, we reduce complexity to the order of $m\epsilon^{-2}$ in parameters m and ϵ. In some sense, our improvement based on the following obvious remark

Lemma 1. *Linear functions* $\{a(x)\}$ *defined on* $\mathbb{F}_2^m$ *form a Hadamard code* $\mathcal{H}(l)$ *on any* l*-dimensional linear subspace* $\mathbb{X} \subset \mathbb{F}_2^m$.

Next, consider the majority voting performed on the vector $\mathbf{g}(\mathbb{X} + y_{(i)})$ in (12). Given any $y = y_{(i)}$, we choose in favor of some constant $\tilde{a}_b(y)$ in (12) instead of $\tilde{a}_b(y) + 1$ if the corresponding affine function $a_b(x) + \tilde{a}_b(y)$ is closer to $\mathbf{g}(\mathbb{X} + y)$ than the opposite function $a_b(x) + \tilde{a}_b(y) + 1$. Thus, the estimates $\tilde{a}_b(y)$ can be derived simultaneously for different $b \in \mathbb{F}_2^l$, by decoding vector $\mathbf{g}(\mathbb{X} + y)$ into the list of $L = 2^l$ closest affine functions

$$\{b(h) + \tilde{a}_b(y),\ b \in \mathbb{F}_2^l\}.$$

Here exactly one function appears for each b. In other words, $\mathbf{g}(\mathbb{X} + y)$ is L-list decoded into the biorthogonal code $\mathrm{RM}(1, l)$ of size $2L$. Thus, we obtain L strings of coefficients $(b_1, ..., b_l, \tilde{a}_b(y))$, which in turn can be rewritten as a vector $\tilde{A}(y)$ that has a symbol $\tilde{a}_b(y)$ in each "position" $b = (b_1, ..., b_l)$ of $\mathbb{F}_2^l$.

More generally, for each $i = 1, ..., k$ and $j = 0, ..., m$, we perform list-decoding F_L of a vector $\mathbf{g}(\mathbb{X} + y_{(i)} + \mathbf{j})$ into L closest affine functions and form a vector $\tilde{A}(y_{(i)} + \mathbf{j})$ of length L that is formed by the free terms of these affine functions. This operation is equivalent to (13). In turn, for each $j \geq 1$, two vectors $\tilde{A}(y_{(i)})$ and $\tilde{A}(y_{(i)} + \mathbf{j})$ give a new vector

$$\tilde{A}_i(\mathbf{j}) = \tilde{A}(y_{(i)}) + \tilde{A}(y_{(i)} + \mathbf{j}) \tag{16}$$

that includes all symbols $\tilde{a}_{b,i}(\mathbf{j})$ obtained earlier in (14). Finally, for each $j = 1, ..., m$, we perform majority voting on k vectors $\tilde{A}_i(\mathbf{j})$ which give m vectors

$$\tilde{A}(\mathbf{j}) = \mathsf{Maj}_{i=1,...,k} \tilde{A}_i(\mathbf{j}) \tag{17}$$

Thus, vectors $\tilde{A}(\mathbf{j})$ form an $m \times 2^l$ matrix, and give the coefficients $\tilde{a}_b = (\tilde{a}_{1,b}, ..., \tilde{a}_{m,b})$ of the function $\tilde{a}_b(x)$ in any column $b \in \mathbb{F}_2^l$.
Algorithm $GL^{(2)}$ is now modified as follows.

Algorithm $GL^{(2)}_{\rm mod}(\mathbf{g}, m, \epsilon, s)$ for code $\mathcal{H}(m)$.
Input: numbers m, ϵ, s, vector $\mathbf{g} \in \{0,1\}^n$.
Pick up an l-dimensional subspace $\mathbb{X}$ in $\leq s$ trials.
Pick up points $y_{(i)}$, $i = 1, ..., k$ in $\leq 9k$ trials.
1. For each i and $j = 0, ..., m$, decode vector
$\mathbf{g}(\mathbb{X} + y_{(i)} + \mathbf{j})$ into a vector $\tilde{A}(y_{(i)} + \mathbf{j})$.
2. Find vector $\tilde{A}_i(\mathbf{j}) = \tilde{A}(y_{(i)}) + \tilde{A}(y_{(i)} + \mathbf{j})$.
For each $j \geq 1$, find $\tilde{A}(\mathbf{j}) = \mathsf{Maj}_{i=1,...,k} \tilde{A}_i(\mathbf{j})$.
3. Output the list $\{\tilde{a}_b(x)\}$ of (10).

Proof of Theorem 1. The two algorithms $GL^{(2)}$ and $GL^{(2)}_{\rm mod}$ have the same outputs in each decoding step, and therefore have the same probabilities of failure. Thus, we only need to find the complexity $\Phi_{\rm mod}$ of the modified algorithm. Note that the list decoding F_L of each vector $\mathbf{g}(\mathbb{X} + y_{(i)} + \mathbf{j})$ can be performed using the FFT-algorithm of the Green machine. This algorithm (see [1] or [6] for more details) represents each affine function with coefficients $b_1, ..., b_l, \tilde{a}_b$ as a path in a tree of depth $m+1$. In step $j = 1, ..., m$, the algorithm recursively derives the distance from the received vector to every codeword generated by a function $b_1, ..., b_j, 0, ..., 0$. In the end, it outputs the distance corresponding to the full function $b_1, ..., b_l, \tilde{a}_b$, or chooses the value of the free term $\tilde{a}_b = 0, 1$ that gives the shorter distance for any prefix $b_1, ..., b_l$. The algorithm requires the order of $l^2 2^l$ operations.

Thus, we need the order of $mkl^2 2^l$ operations for decoding all mk vectors $\mathbf{g}(\mathbb{X} + y_{(i)} + \mathbf{j})$. Finally, we calculate each vector $\tilde{A}_i(\mathbf{j})$, which requires 2^l operations per vector. Therefore, the algorithm $GL^{(2)}_{\rm mod}$ has the overall complexity

$$\Phi_{\rm mod} \sim mkl^2 2^l = \mathcal{O}\left(\left(s + \log \frac{m}{\epsilon^2}\right) \frac{m}{\epsilon^2} \log^2 \epsilon\right).$$

This completes our proof for the Hadamard code $\mathcal{H}(m)$.

For the code $RM(1,m)$, we can perform $GL^{(2)}_{\rm mod}$ for the received vector $\mathbf{g}$ and the shifted vector $\mathbf{g}+\mathbf{1}$, and then combine both lists. Thus, the former complexity $\Phi_{\rm mod}$ is at most doubled, whereas the bound 2^{-s} is not changed. Indeed, the same Johnson bound $|L_\epsilon(\mathbf{g})| \le \epsilon^{-2}$ holds for both codes $\mathcal{H}(m)$ and $RM(1,m)$, and can be used for the latter in the same way it was for the former. $\square$

Appendix. One open problem is to refine the estimates used for pairwise independent random variables in Theorems 1 to 3. In turn, this could relax restriction (1), inherent in Theorems 1 and 3. Namely, we raise the following

Open Problem. Let $E \subset \mathbb{F}_2^m$ be an arbitrary subset of size $|E| = 2^{m-1}(1-\epsilon)$ but without the null vector $\mathbf{0}$, and $\mathbb{X} \subset \mathbb{F}_2^m$ be any l-dimensional space, $l < m/2$. We say that $\mathbb{X}$ is in error and write $\mathbb{X}_{\rm err}$ if $|E \cap \mathbb{X}| \ge \frac{1}{2}|\mathbb{X}|$. Maximize the fraction

$$f(m,\epsilon,l) = \max_E \frac{\text{number of } \mathbb{X}_{\rm err}}{\text{number of } \mathbb{X}}$$

over subsets E. Is $f(m,\epsilon,l) < 2^{-l}$ for any $\epsilon \in (0,1)$?

References

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

[2] O. Goldreich and L. A. Levin, "A hard-core predicate for all one-way functions", *21st ACM Symp. Theory of Computing,* Seattle, WA, USA, May 14 - 17, 1989, pp. 25–32.

[3] L. A. Levin, "Randomness and Nondeterminism," J. Symb. Logic, vol. 58, pp. 1102-1103, 1993.

[4] O. Goldreich, *Foundations of Cryptography*, vol. 1, Cambridge, New York, 2001.

[5] L. Trevisan, "Some applications of coding theory in computational complexity", *Quaderni di matematica,* vol. 13, pp. 347-424, 2004.

[6] I. Dumer, G. Kabatiansky, C. Tavernier, "List decoding of Reed-Muller codes of the first order, " *Problems Info. Transmission,* vol. 43, no. 3 pp. 46-54, 2007.