Successive Cancellation Permutation Decoding of Extended BCH codes

Nikolai Iakuba and Peter Trifonov
ITMO University
{nyakuba, pvtrifonov}@itmo.ru

***Abstract*—BCH codes are used in many applications, including optical transport networks, flash memory and video broadcasting. This paper introduces soft decision permutation decoding algorithm for extended BCH codes based on it's representation as polar codes with dynamic frozen symbols. The proposed algorithm outperforms bounded distance hard decision decoding and provides flexible tradeoff between performance and decoding complexity.**

I. Introduction

Bose-Chaudhuri-Hocquengham (BCH) codes are still extensively used in many practical applications, such as optical transport networks, flash memory and video broadcasting. A BCH code with constructive minimum distance δ can be decoded using bounded distance hard decision decoders, which can correct $t = \lfloor \frac{\delta-1}{2} \rfloor$ errors.

Several one-pass schemes of Chase decoding were proposed [1], which evaluate error-locator polynomials for all test vectors using a single pass of the Berlekamp's algorithm. A good overview of various soft decision algebraic decoding methods developed for BCH codes is given in [2].

Other soft decision decoding methods based on Viterbi and ordered statictics decoding algorithms can provide maximum-likelihood decoding at cost of substantial increase in decoding complexity [3], [4].

In this paper a soft-input decoding algorithm based on successive cancellation decoding for primitive extended BCH codes is presented. It uses the representation of extended BCH codes as polar codes with dynamic frozen symbols, and employs permutation decoding techniques to enhance performance.

The proposed decoding algorithm is based on the successive cancellation list decoder developed in [5] for polar codes, and provides a flexible tradeoff between decoding performance and complexity. Simulations show, that the proposed decoder performs better than hard-decision decoder, however it does not guarantee bounded distance decoding of BCH codes.

II. BCH and Reed-Muller codes

In this paper we consider extended primitive narrow-sense BCH codes (eBCH codes) over $\mathbb{F}_2$. Generator polynomial of a BCH code of length $n = 2^m - 1$ with constructive minimum distance δ has roots $\alpha, \alpha^2, \ldots, \alpha^{\delta-1}$, where α denotes primitive element of $\mathrm{GF}(2^m)$.

Hence, the check matrix of $\mathrm{eBCH}(2^m, k, d \geq \delta)$ code is defined as a binary image of the matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 0 & 1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \alpha^{\delta-2} & \alpha^{2(\delta-2)} & \ldots & \alpha^{(n-1)(\delta-2)} \\ 0 & 1 & \alpha^{\delta-1} & \alpha^{2(\delta-1)} & \ldots & \alpha^{(n-1)(\delta-1)} \end{pmatrix}, \quad (1)$$

where all linearly dependent rows are eliminated.

It can be seen that BCH codes are closely related to Reed-Muller and polar codes. Polar (n, k) code is defined as a set of vectors

$$\mathcal{C} = \left\{ u_0^{n-1} A^{\otimes m} \mid u_0^{n-1} \in \mathbb{F}_2^n, u_i = 0, \forall i \in \mathcal{F} \right\}, \quad (2)$$

where $A = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$, $\otimes m$ denotes m-times Kronecker product of the matrix A with itself, and $\mathcal{F}, |\mathcal{F}| = k$ denotes set of frozen symbols, which depends on a transmission channel.

A Reed-Muller code of length 2^m and order r can be defined as a special case of polar codes:

$$\mathrm{RM}(r, m) = \left\{ u_0^{n-1} A^{\otimes m} \mid u_i = 0, \forall i : \mathrm{wt}(i) < m - r \right\}, \quad (3)$$

where $\mathrm{wt}(i)$ denotes Hamming weight of the index.

Both polar and Reed-Muller codes can be decoded using successive cancellation (SC) algorithm. Suppose, that the codeword of $(n = 2^m, k)$ polar (or Reed-Muller) code $c_0^{n-1} = u_0^{n-1} A^{\otimes m}$ is transmitted through the channel W with transition probabilities $W(y|x)$, and the noisy vector y_0^{n-1} is passed to the decoder.

The decoding algorithm is based on recursive computation of transition probabilities

$$\mathcal{W}_i(y_0^{n-1}, u_0^{i-1} | u_i) \triangleq \sum_{u_{i+1}^{n-1} \in \mathbb{F}_2^{n-i-1}} \frac{1}{2^{n-1}} \prod_{i=0}^{n-1} W(y_i | u_i). \quad (4)$$

At the receiver end symbols $\hat{u}_\phi, \phi = 0, \ldots, n-1$ can be estimated one by one:

$$\hat{u}_\phi = \begin{cases} \arg\max_{u_\phi \in \mathbb{F}_2} \mathcal{W}_\phi(y_0^{n-1}, \hat{u}_0^{\phi-1} | u_\phi), \phi \notin \mathcal{F} \\ 0, \phi \in \mathcal{F}. \end{cases} \quad (5)$$

Due to the recursive structure of the matrix $A^{\otimes m}$ encoding and decoding can be done in $O(n \log n)$ operations.

A. Successive cancellation list decoding algorithm

Unfortunately, SC decoding of polar codes of moderate lengths leads to rather poor performance, since the estimates $\hat{u}_\phi$ are computed without taking into account frozen symbols

$u_i, i > \phi, i \in \mathcal{F}$. However, performance of the SC algorithm can be enhanced by considering several possible paths $\hat{u}_0^{n-1}$ in the code tree.

This approach is used in the successive cancellation list (SCL) decoding algorithm introduced by Tal and Vardy [5]. Interestingly, the same idea was described earlier by Dumer and Shabunov in their work [6] devoted to the decoding of Reed-Muller codes. Below we revise the min-sum version of the SCL algorithm.

The SCL decoder successively extends L vectors $\hat{u}_0^{\phi}$ of equal length trying to maximize their score

$$M(\hat{u}_0^{\phi}, y_0^{n-1}) = \sum_{i=0}^{\phi} \tau(\hat{u}_i, S_m^{(i)}(\hat{u}_0^{i-1}, y_0^{n-1})), \tag{6}$$

where

$$\tau(u, S) = \begin{cases} 0, \text{ if } (-1)^u = \operatorname{sgn}(S) \\ -|S|, \text{ otherwise} \end{cases} \tag{7}$$

is the penalty function. $S_m^{(i)}(\hat{u}_0^{i-1}, y_0^{n-1})$ denotes the log-likelihood ratio given by

$$\begin{aligned} S_{\lambda}^{(2\phi)}(\hat{u}_0^{2\phi-1}, y_0^{n-1}) &= \operatorname{sgn}(a)\operatorname{sgn}(b)\min(|a|, |b|), \\ S_{\lambda}^{(2\phi+1)}(\hat{u}_0^{2\phi}, y_0^{n-1}) &= (-1)^{\hat{u}_{2\phi}} a + b, \end{aligned} \tag{8}$$

where $n = 2^{m-\lambda}$, $a = S_{\lambda-1}^{\phi}(\hat{u}_{0,e}^{2\phi-1} + \hat{u}_{0,o}^{2\phi-1}, y_0^{n/2-1})$, $b = S_{\lambda-1}^{(\phi)}(\hat{u}_{0,o}^{2\phi-1}, y_{n/2-1}^{n-1})$, and $S_0^{(i)} = \log \mathcal{W}(y_i|0) - \log \mathcal{W}(y_i|1)$. Here $\hat{u}_{0,o}^{\phi}$ and $\hat{u}_{0,e}^{\phi}$ denote subvectors of $\hat{u}_0^{\phi}$ consisting of elements on odd end even indices respectively. The list size L defines the complexity $O(Ln \log n)$ of the decoding.

B. Dynamic frozen symbols

It was shown in [7], that any $(n = 2^m, k, d)$ code with check matrix H can be represented as a polar code with dynamic frozen symbols. Let denote $A_m = A^{\otimes m}$, where $A = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$. Since $A_m^{-1} = A_m$, it is possible to choose matrices V and W, such that $H = VA_m^T$ and $c_0^{n-1}H^T = u_0^{n-1}WV^T = 0$. Hence, some symbols $u_i, i \in \mathcal{F}$ can be set to predefined linear functions of previous symbols, i.e.

$$u_{j_i} = \sum_{s<j_i} V_{i,s}u_s, 0 \le i < n-k, \tag{9}$$

where $V \in \mathbb{F}_2^{n-k \times n}$ is a constraint matrix and j_i is the maximal index of the non-zero element of the row $V_{i,-}$.

Encoding can be done by evaluation of $c_0^{n-1} = u_0^{n-1}WA_m$, where $W \in \mathbb{F}_2^{n \times n}, WV^T = 0$ is a precoding matrix.

Note, that for $\mathrm{eBCH}(2^m, k, d > \delta)$ code the number of dynamic frozen symbols of weight t equals to the total number of elements of weight t in cyclotomic classes, containing $\alpha, \alpha^2, \ldots, \alpha^{\delta-1}$ (see Theorem 2 [7]). This theorem provides an empiric observation that representation of a eBCH code as a polar code leads to relatively good set $\mathcal{F}$, therefore successive cancellation decoding may be applied to decode eBCH code.

III. Proposed approach

Observe, that automorphism group of Reed-Muller codes contains general affine group [8]. In other words, it can be shown that any permutation of symbols in a codeword $\pi(x) = Mx + b$, where x is a binary representation of a bit index (coordinate), and M is non-singular matrix, defines an automorphism of a Reed-Muller code.

It was proposed in [6] to initialize the SCL decoder with several permutations of a codeword to further enhance decoding performance. We propose to apply this idea to the decoding of eBCH codes.

Let consider a $\mathrm{eBCH}(n, k, d)$ code $\mathcal{C}$ and its check matrix H defined as it was shown in equation (1). Consider a permutation taken from automorphism group of the corresponding Reed-Muller supercode $\mathcal{R}$, $\mathcal{C} \subseteq \mathcal{R}$. This permutation applied to H defines an equivalent code $\mathcal{C}'$, which may not be equal to $\mathcal{C}$, although $\mathcal{C}' \subseteq \mathcal{R}$. Since $\mathcal{C}'$ is also included in $\mathcal{R}$, representations of $\mathcal{C}$ and $\mathcal{C}'$ as polar codes will have similar sets of frozen symbols $\mathcal{F}$. For instance, Theorem 2 is still valid for equivalent codes obtained that way, and performance of successive cancellation decoding doesn't differ much.

The algorithm Decode illustrates the proposed method. Main parameters of the algorithm are list size L, and the set of permutations $\mathcal{P} = \{\pi_0, \ldots, \pi_l\}$ of size l. Note that each permutation of $\mathcal{P}$ also defines permutation on the check matrix H of the considered eBCH code. Hence, permutation may share different sets of frozen symbols $\mathcal{F}$ and different constraint matrices V.

In line 1 the constraint matrix $V^{(i)}$ and set of frozen symbols $\mathcal{F}^{(i)}$ are evaluated for each permutation π_i as it was described in section II-B. Permuted input vector of log-likelihood ratios $\mathbf{S}_0 = (S_0^{(0)}, \ldots, S_0^{(\mu n-1)})$ is used to initialize paths with indices $i = 0, \ldots, l-1$ in line 4. In line 5 list $\mathcal{U}$ is filled with these paths.

The list $\mathcal{U}$ consists of triplets $\langle M, \hat{u}_0^{\phi-1}, i\rangle$, which define paths considered by the decoder on iteration ϕ. Here M denotes a path score, $\hat{u}_0^{\phi-1}$ is a vector of bits estimated on previous phases, and i is a permutation index.

The main decoding loop starts with line 6. In line 9 values $S_m^{(i)}(\hat{u}_0^{i-1}, y_0^{n-1})$ are computed according to (6)–(8), and set of continuations $\tilde{\mathcal{U}}$ is constructed.

Function *GetBestPaths*($\mathcal{U}, L$) returns L continuations with highest scores from the set $\mathcal{U}$. Selected continuations are passed to the next decoding iteration.

The decoder terminates when phase $\phi = n$ is reached, and path $\hat{u}_0^{n-1} \in \mathcal{U}$ with the highest score is returned.

The complexity of the proposed algorithm is determined by SCL decoding and equals to $O(Ln \log n)$.

IV. Choosing the set of permutations

Decoding performance of the algorithm described in previous section highly depends on the initial set of permutations $\mathcal{P}$ passed to the decoder. Unfortunately, we have no analytic way to construct $\mathcal{P}$, although one may use greedy approach to form $\mathcal{P}$ by choosing permutations one by one.

Procedure Decode($L, \mathcal{P}, \mathbf{S}_0$)

input : list size L of the decoder,
set of permutations $\mathcal{P} = \{\pi_0, \dots, \pi_l\}$,
log-likelihood ratios $\mathbf{S}_0 = (S_0^{(0)}, \dots, S_0^{(\mu n-1)})$
output: estimated bits $\hat{u}_0^{n-1}$

for $i \leftarrow 0$ **to** $l-1$ **do**
  $(V^{(i)}, \mathcal{F}^{(i)}) \leftarrow$ `GetConstraintMatrix`(π_i)
$\mathcal{U} \leftarrow \varnothing$
for $i \leftarrow 0$ **to** $l-1$ **do**
  `InitPath`(`Permute`$(\mathbf{S}_0, \pi_i), i)$
  $\mathcal{U} \leftarrow \mathcal{U} \cup \{\langle 0, \epsilon, i\rangle\}$
  // (score, path, permutation index)
for $\phi \leftarrow 0$ **to** $n-1$ **do**
  $\tilde{\mathcal{U}} \leftarrow \varnothing$ // list to be sorted
  foreach $(M, \hat{u}_0^{\phi-1}, i) \in \mathcal{U}$ **do**
    $S \leftarrow$ `CalcS`$(\hat{u}_0^{\phi-1})$ // penalty
    if $\phi \in \mathcal{F}^{(i)}$ **then**
      $b \leftarrow$ `GetFrozenBit`$(\hat{u}_0^{\phi-1}, V^{(i)})$
      $\tilde{\mathcal{U}} \leftarrow \tilde{\mathcal{U}} \cup \left\{\langle M + \tau(b, S), (\hat{u}_0^{\phi-1}, b), i\rangle\right\}$
    else
      $\tilde{\mathcal{U}} \leftarrow \tilde{\mathcal{U}} \cup \left\{\langle M + \tau(0, S), (\hat{u}_0^{\phi-1}, 0), i\rangle\right\}$
      $\tilde{\mathcal{U}} \leftarrow \tilde{\mathcal{U}} \cup \left\{\langle M + \tau(1, S), (\hat{u}_0^{\phi-1}, 1), i\rangle\right\}$
  $\mathcal{U} \leftarrow$ `GetBestPaths`$(\tilde{\mathcal{U}}, L)$
$\langle M, \hat{u}_0^{n-1}, i\rangle \leftarrow$ `GetBestPaths`$(\mathcal{U}, 1)$
return `PermuteInverse`$(\hat{u}_0^{n-1}, \pi_i)$

The automorphism group of Reed-Muller codes is too large to test all possible permutations. Recall, that we consider permutations $\pi(x) = Mx + b$, where M is non-singular.

Observe, that some permutations are equivalent in terms of successive cancellation decoding performance. That is, some permutations may only permute the order of computation of (8) and not change values of penalties computed at each phase of the list decoding algorithm. These permutations were described by Bardet et al. in [9] and have form $\pi(x) = Tx+b$, where T is non-singular lower triangular matrix.

For the purpose of searching good set of permutations for the list decoding algorithm we propose to restrict the set of possible permutations to the set of permutations $\pi(x) = Mx$, which are not equivalent in terms of successive cancellation decoding. Two permutations π_1, π_2 we call equivalent if $\pi_1(x) = T\pi_2(x)$, where T is non-singular lower triangular matrix.

This number is still too large to brute force every possible permutation, therefore we propose to pick permutations at random. Each permutation is generated in the following way. Let start with the zero permutation matrix M of size m. First row of M is generated at random out of $2^m - 1$ possible non-zero binary vectors. Second row of M is generated such that it doesn't end in the same column as the first row. Third row shouldn't end in the same columns as first and second rows and so on. The total number of possible permutations therefore is reduced to $\prod_{i=1}^{m} 2^i - 1$.

We start with a single permutation, which corresponds to the standard bit ordering of the check matrix H. That is, the binary representation of the second row in (1) corresponds to the vector $(0, 1, 2, \dots, 2^m - 1)$.

Other permutations are chosen iteratively: for a fixed signal to noise ratio $N_{\max}$ permutation matrices are generated at random, than the permutation leading to smallest error probability of the proposed list decoding algorithm is added to $\mathcal{P}$. Updated set $\mathcal{P}$ is used in searching for the next permutation.

V. Numeric results

Performance of the proposed algorithm was measured depending on the list size L and number of initial permutations l. Figures 1, 2 were obtained by simulation of transmitting 10^6 codewords of eBCH$(256, 155, 28)$ code through additive Gaussian (AWGN) channel with binary phase-shift keying (BPSK) modulation. Permutations for the considered code were obtained by simulation with $L = 64, N_{\max} = 1000$ and $E_b/N_0 = 5.5$ dB. Curves entitled "hard" and "uncoded" illustrate FER of bounded distance hard decision decoding and transmission of uncoded data respectively.

Figure 1 illustrates the dependence of error probability on the list size L. It can be seen, that the proposed algorithm outperforms bounded distance decoder even on the list size $L = 64$ approximately by 0.125 dB. It can be seen, that performance gain grows almost linearly with increase of the list size: performance curve shifts to the left by ≈ 0.25 dB when L doubles.

Figure 2 illustrates the dependence of error probability on the size of permutation set $\mathcal{P}$. It can be seen, that the most significant gain is obtained by employing two permutations, further increase of the set $\mathcal{P}$ provides significantly smaller gain. Moreover, large set $\mathcal{P}$ may lead even to performance degradation, since all permutations are processed within a single list, and in average each permutation "occupies" some fraction of available paths, which is limited.

Summing up, permutation techniques can provide substantial performance gain, although this gain is determined mostly by the list size L. Performance gain grows linearly, while L increases exponentially.

VI. Conclusion

To conclude, the proposed soft decision decoding algorithm for eBCH codes provides performance gain compared to bounded distance decoding. The proposed algorithm is based on the representation of eBCH codes as polar codes with dynamic frozen symbols, and uses SCL decoding with list size L to jointly decode several permutations of an input noisy vector. Decoding complexity is defined by the list size L and equals to $O(Ln \log n)$. Appropriate choice of permutations can sufficiently improve decoding performance, and can be done using greedy algorithm.

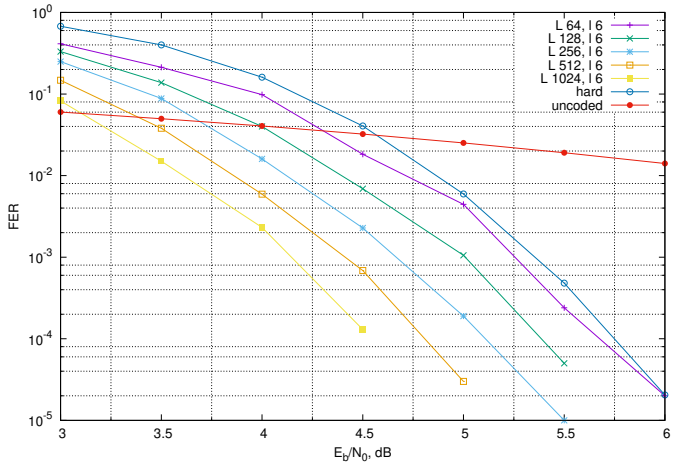

Figure 1: Performance of the SCL permutation decoding with different list size L

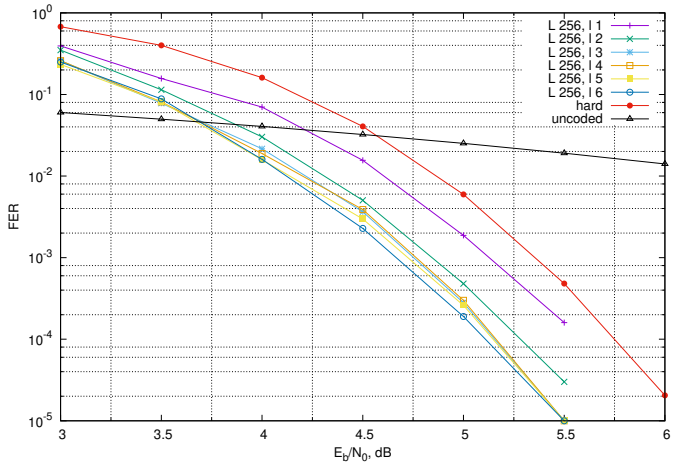

Figure 2: Performance of the SCL permutation decoding with different number of permutations l

Acknowledgments

This work was supported by the Ministry of Science and Higher Education of Russian Federation, project (Goszadanie) no. 2019-0898.

References

[1] Y. Wu, "Fast chase decoding algorithms and architectures for reedsolomon codes", *IEEE Transactions on Information Theory*, vol. 58, pp. 109–129, 1 2012.

[2] N. Kamiya, "On algebraic soft-decision decoding algorithms for BCH codes", *IEEE Transactions on Information Theory*, vol. 47, pp. 45–58, 1 2001, ISSN: 1557-9654.

[3] C. Choi and J. Jeong, "Fast and scalable soft decision decoding of linear block codes", *IEEE Communications Letters*, vol. 23, pp. 1753–1756, 10 2019.

[4] T. L. Tapp, A. A. Luna, X.-A. Wang, and S. B. Wicker, "Extended hamming and bch soft decision decoders for mobile data applications", *IEEE Transactions on Communications*, vol. 47, pp. 333–337, 3 1999.

[5] I. Tal and A. Vardy, "List Decoding of Polar Codes", *IEEE Transactions on Information Theory*, vol. 61, pp. 2213–2226, 5 2015.

[6] I. Dumer and K. Shabunov, "Soft-decision decoding of Reed-Muller codes: Recursive lists", *IEEE Transactions on Information Theory*, vol. 52, pp. 1260–1266, 3 2006.

[7] P. Trifonov and V. Miloslavskaya, "Polar Subcodes", *IEEE Journal on Selected Areas in Communications*, vol. 34, pp. 254–266, 2 2016, ISSN: 1558-0008.

[8] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North Holland, 1983, p. 782, ISBN: 9780444851932.

[9] M. Bardet, V. Dragoi, A. Otmani, and J. Tillich, "Algebraic properties of polar codes from a new polynomial formalism", *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 230–234, 2016.