Improved Random Redundant Iterative HDPC Decoding

Ilan Dimnik, and Yair Be'ery, *Senior Member, IEEE*

***Abstract*—An iterative algorithm for soft-input soft-output (SISO) decoding of classical algebraic cyclic block codes is presented below. Inspired by other approaches for high performance belief propagation (BP) decoding, this algorithm requires up to 10 times less computational complexity than other methods that achieve similar performance. By utilizing multiple BP decoders, and using random permutation taken from the permutation group of the code, this algorithm reaches near maximum likelihood performance. A computational complexity comparison of the proposed algorithm versus other methods is presented as well. This includes complexity versus performance analysis, allowing one to trade between the former and the latter, according to ones needs.**

***Index Terms*—Iterative decoding, soft-decision decoding, permutation group, message-passing algorithm.**

I. Introduction

ITERATIVE decoding of classic codes has created much interest in recent years. These codes, which have been known for decades and are being used nowadays in many applications, are sometimes referred to as high density parity-check codes (HDPC) [1], to differentiate them from the low density parity-check codes (LDPC). It is well known that iterative methods for decoding LDPC codes achieve similar performance to Maximum Likelihood (ML) decoding, due to the sparseness of its parity check matrix.

By nature, classical codes have a very dense matrix, with small girth and high number of short cycles. Therefore, performing standard iterative decoding on them usually leads to poor results. Jiang and Narayanan [1], the first to crack the barrier, demonstrated a modification to the regular Sum-Product (SP) decoding algorithm, which produces a better result than the standard SP algorithm. The innovation in their work was the adaptation of the parity check matrices from iteration to iteration, based on the Log Likelihood Ratio (LLR) of the incoming signal. As nice as it may be, the hardware realization of Jiang and Narayanan decoder is a complicated task, due to the Gaussian elimination process involved in every SP iteration.

From an entirely different direction emerged the work of Feldam, Wainwright, and Karger [2], [3], who interpreted the parity check matrix as a set of constraints, and used linear programming methods to decode the obtained codeword. Their work indicated that the linear programming decoder (LPD) acts on a fundamental polytope that is created by the constraints of the parity check matrix. The polytope has vertices that are valid codewords and vertices that are invalid codewords. Linear programming decoding eventually settles on one of the vertices, but not necessarily on a valid codeword, referred to as pseudo codewords (PCW) [4], [5]. The resemblance in performance and the similar decoding behavior of LPD and belief propagation (BP) have led to the utilization of PCW weight and distribution, in order to evaluate and improve decoding performance.

Kelley and Sridhara [4] have demonstrated that enriching the parity check matrix structure, by adding redundant rows, improves its PCW weight distribution and decoding performance. Although this conclusion contradicts ones intuition, which would suggest that adding more rows to the parity check matrix would increase the number of short cycles in the graph, and decrease its decoding performance. Hereafter, a parity check matrix H for an $[n, k]$ linear block code C, with more than $(n-k)$ rows will be referred to as a redundant parity check matrix.

This letter presents a new algorithm for iteratively decoding classical codes. The algorithm was inspired by the publications presented by Halford and Chugg [6], who introduced the Random Redundant Iterative Decoding (RRD) decoder, and by Hehn et al. [7], who introduced the Multiple-Bases Belief-Propagation (MBBP) decoder. Both algorithms use a redundant parity check matrix, although in different ways

While the MBBP decoder defines $n \times n$ parity check matrices derived from the minimum weight codewords of the dual code, the RRD uses the "temporal" redundant parity check matrix, by altering the basic parity check matrix throughout the decoding process. The different structure of the redundant matrices of the decoders affects the decoding computational complexity.

The outline of this paper is as follows. In Section II we present the new decoding algorithm, as well as a geometric interpretation for it. Section III analysis and compares the computational complexity of the decoders. Section IV includes simulation results of some algebraic block codes and Section V contains concluding remarks.

II. Proposed Decoding Algorithm

This section presents the proposed decoding algorithm. We first offer a very brief description of the RRD decoder and the MBBP decoder. All codes discussed in this letter are binary codes. Therefore, we can assume without loss of optimality that the decoding algorithms base their decision on the log-likelihood ratio (LLR) vector, which is provided by the observed channel output sequence.

Paper approved by A. K. Khandani, the Editor for Coding and Information Theory of the IEEE Communications Society. Manuscript received November 28, 2007; revised August 4, 2008 and November 22, 2008.

The authors are with the Tel Aviv University, School of Electrical Engineering, Ramat Aviv 69978, ISRAEL (e-mail: ilandimnik@gmail.com, ybeery@eng.tau.ac.il).

Digital Object Identifier 10.1109/TCOMM.2009.070621

The "temporal" redundancy used by RRD is achieved by using *permutation group* of $\mathcal{C}$, Per($\mathcal{C}$). Per($\mathcal{C}$) is defined in [8] (also referred as automorphism group) as a set of permutations of coordinate places that send $\mathcal{C}$ into itself. The permutation group of many classical block codes are known ([8]) and can be used to generate the matrix redundancy. By randomly choosing elements from Per($\mathcal{C}$), RRD decoder changes the set of constraints that are used temporally. This process can also be seen as changing the polytope on the move, while trying to converge to a solution. In [6], it is also demonstrated that decoding with the permuted soft input vector is equivalent to decoding over the permuted parity check matrix. This principle is to be used in this letter and all permutations referred to below are actually permutations over the input LLR values. RRD also utilizes a damping coefficient α (positive number which less than one) to scale the LLR values. The decoder starts with a small damping coefficient and it is increased from decoding iteration to another until it reaches a valid codeword.

The MBBP decoding algorithm utilizes ℓ parallel iterative decoders, where each of the decoders uses a different parity-check matrix $\boldsymbol{H_\ell}$. MBBP parity check matrix $\boldsymbol{H_\ell}$, for a certain cyclic code $\mathcal{C}$, is created by partitioning the set of codewords of its dual code $\mathcal{C}^\perp$ into sets consisting only of cyclic shifts of one codeword. One of these codewords represent that group and referred as the *cyclic group generator* (CGG). The CGGs with Hamming weight equal to the minimum distance of the dual code $d^\perp$, is to be used to construct $n \times n$ parity check matrices $\boldsymbol{H_\ell}$. The possible number of the $\boldsymbol{H_\ell}$ parity check matrix depends on the size of the CGG and varies with the code.

Each decoder performs at most i iterations, and the hard decision of its output is denoted as $\boldsymbol{c}_\ell$. In case some of the decoders converged to a valid codeword, the resultant codewords are used along with a *least metric selector* (LMS) to choose the most likely codeword. In case none have converged, all output codewords are used with LMS.

A. New decoding algorithm - mRRD

The new decoding algorithm attempts to benefit from both of the above two approaches. It utilizes ℓ iterative decoders in parallel, in which each decoder uses the same parity check matrix (sizes $(n-k)\times n$), but with random permutation. Initial permutation is selected randomly at the beginning of each decoding iteration. The damping coefficient α no longer plays a role in the decoding algorithm as in RRD, and it stays fixed throughout the decoding process and is set to some empirically chosen value. Each ℓ-th decoder makes use of two loops:

1) Outer loop performed I_2 times: If we receive a valid codeword, we perform hard decision on the current LLR, perform inverse permutation Θ^{-1} on the received codeword, and exit. If that is not the case, we alternate the current LLRs using a random permutation $\boldsymbol{\theta}$, and we accumulate the total permutations in Θ.
2) Inner loop performed I_1 times - the basic SP iterative decoding process.

As in MBBP, an LMS is then used to select the most likely codeword, and in case none of the decoders converged, all output codewords are used with LMS. We choose to call this the modified RRD (mRRD) algorithm, due to its close relationship with the RRD decoding algorithm The algorithm is given as follows:

$S := \emptyset$
for $\ell := 1, \ldots, l$ **do**
  $\boldsymbol{w} \leftarrow LLR(\boldsymbol{y})$
  $\boldsymbol{\theta} \leftarrow$ random element of Per($\mathcal{C}$)
  Apply $\boldsymbol{\theta}$ to $\boldsymbol{w}$
  $\Theta \leftarrow \boldsymbol{\theta}$
  for $1 \le i_2 \le I_2$ **do**
    Perform I_1 decoding iterations on TG($\boldsymbol{H}$)
    place soft output in w' such that $w \leftarrow w + w'$
    $\hat{\boldsymbol{c}}_\ell =$ Hard Decision (w)
    if $\hat{\boldsymbol{c}}_\ell \cdot \boldsymbol{H^T} = 0$ **then**
      $S := S \cup \ell$
      Apply Θ^{-1} to $\boldsymbol{w}$ and $\hat{\boldsymbol{c}}_\ell$
      Break out of for loop
    end if
    $\boldsymbol{\theta} \leftarrow$ random element of Per($\mathcal{C}$)
    Apply $\boldsymbol{\theta}$ to $\boldsymbol{w}$
    $\Theta \leftarrow \boldsymbol{\theta} \cdot \Theta$
  end for
end for
if $S = \emptyset$ **then**
  $S := \{1, \ldots, l\}$
end if
$\hat{\boldsymbol{c}} := \arg\max_{s \in S} \sum_{\nu=1}^{n} |y_\nu - \hat{c}_{s,\nu}|^2$

B. Geometric Interpretation of mRRD

Richardson was the first to give a geometric interpretation to the iterative decoding process [9]. Jiang and Narayanan [1] gave a very similar interpretation, in which they presented the decoding process as a gradient descent optimization algorithm. They defined a potential function $J(\boldsymbol{H}, \underline{T})$ as a function of the parity check matrix, which is being used for decoding H, and the LLR values of the received word T. They also stated that the gradient descent updating rule can be written as follows:

$$\underline{T}^{(l+1)} \leftarrow \underline{T}^{(l)} - \alpha \nabla J(\boldsymbol{H}, \underline{T})$$

When each decoder reaches a valid codeword, we can say that the gradient descent algorithm reached an equilibrium point. By using the permutation group to change the parity check matrix, we change the potential $J(\boldsymbol{H}, \underline{T})$ and its gradient $\nabla J(\boldsymbol{H}, \underline{T})$ along the decoding process, thus preventing the decoder from settling on a false equilibrium point (that can also be interpreted as a pseudo codeword (PCW)). The proposed algorithm takes advantage of the fact that the potential $J(\boldsymbol{H}, \underline{T})$ has many local minimas. Performing gradient descent on a random potential increases the probability of reaching a better solution in one (or more) of the decoders, thus achieving better overall performance.

III. Complexity Comparison

Realization complexity of iterative decoders has been widely studied in many publications (for example [10]). Generally speaking, all expressions are given as a function of the number of edges of the bipartite graph induced by the code. Therefore, a comparison between two different decoders,

which use different parity check matrices, is achieved by normalizing the number of edges of their graphs.

RRD and mRRD use the same parity check matrix. Therefore the decoder's complexity is evaluated by averaging the number of sum-product iterations performed until a valid codeword is reached, or until the number of iterations reaches a certain limit in case the decoder doesn't converge to a codeword. A single iteration is defined as the action of sending messages from variable nodes to check nodes, processing these messages at the check nodes, and sending back new messages from the check nodes to the variable nodes.

MBBP parity check matrix $\boldsymbol{H_\ell}$ for a certain code $\mathcal{C}(n,k)$, has a size of $n \times n$. By normalizing MBBP's relative complexity, we can set all decoders on the same scale. Table I displays the relative complexity for some popular codes that have been investigated in this letter. The number of edges in MBBP graph is $d^{\perp}_{min} * n$. The relative complexity is the ratio between the number of edges in MBBP and RRD (or mRRD) graphs.

MBBP's and mRRD's complexity can be compared by multiplying MBBP's average number of sum-product iterations with the relative complexity figure

IV. Simulation Results

This section presents the empirical investigation of mRRD's performance. The results presented are for the BCH[63,45,7] and [24,12,8] extended Golay code. Similar performances have been observed for other codes, such as BCH[31,21,5], BCH[63,36,11], BCH[63,39,9] and BCH[63,45,7]. We present the bit error rate (BER) performance and the computation complexity of various decoders discussed in this letter. Our simulations assume a binary antipodal signaling on an Additive White Gaussian Noise (AWGN) channel. Each iteration the codeword was randomly chosen. To generate the set of the permutation groups, the various codes were taken from [8], [11] and [12].

Fig. 1 displays simulation results for the BCH[63,45,7] code and demonstrates the benefits of the mRRD decoder in terms of performance and complexity:

- The mRRD decoder, which employs three parallel decoders, gives the same performance as an RRD decoder with less than $1/10$ of the complexity at E_b/N_0 = 3dB and $1/3$ of complexity at E_b/N_0 = 7dB.
- The mRRD decoder, which employs twenty parallel decoders, has an advantage of 0.25dB in terms of performance relatively compared to the RRD. Furthermore it requires less computational complexity for $E_b/N_0 <$ 5.5dB.
- The mRRD, with twenty parallel decoders, will cost seven times more SP iterations than mRRD with three parallel decoders, while adding just 0.25dB for the BER.
- The MBBP decoder, which employs three parallel decoders, provides similar BER results compared to mRRD decoder, which employs five parallel decoders. However, MBBP requires 66 SP iteration on the average, compared to 20 iteration for mRRD ($E_b/N_0 >$ 4dB). Considering the relative complexity, MBBP requires ten times more computational power to achieve the same performance.
- MBBP, as presented in [7], does not implement a stopping criterion. We believe that adding a simple stopping criterion to the MBBP decoder may decrease its complexity. Nevertheless, due to the structure of its parity check matrix, even with a stopping criterion, the MBBP will still surpass the RRD/mRRD decoder in its complexity.

Fig. 2 displays simulation results for the extended Golay code. Although the focal area around the ML curve is crowded, one can see that mRRD(20), mRRD with twenty parallel decoders, and mRRD(15), achieve the same performance. They both have an advantage of 0.1dB over MBBP and RRD, which behave similarly in this case. In terms of complexity, Fig. 2b demonstrates that mRRD(15) requires a reduced number of iterations (up to 3 times less) for BER values >1e-6. Fifteen MBBP parallel decoders with 70 decoding iterations, normalized in accordance with Table I as 1800 iterations, exceed the number of iterations required by both RRD (40-165 iterations) and mRRD(15) (30-120 iterations).

V. Conclusions

There is still much mystery as to why iterative decoding achieves such outstanding performance. This letter discloses one aspect of this mystery. Combining several decoders that operate on a dense graph, so that the only difference between them is the random choice of elements from their permutation group, leads to near ML performance. Removing the need to change the decoder's damping coefficient also adds another interesting issue for future research. However, despite these unanswered questions, the presented analysis allows one to opt between complexity and performance, choosing the best conditions for the selected application.

References

[1] J. Jiang and K. R. Narayanan, "Iterative soft decision decoding of Reed-Solomon codes based on adaptive parity check matrices," *IEEE Trans. Inform. Theory*, vol. 52, no. 8, pp. 3746-3756, Jan. 2006.

[2] J. Feldman, "Decoding Error-Correcting Codes via Linear Programming," Ph.D. dissertation, Massachusetts Institute of Technology, 2003. [Online]. Available: http://www.clumbia.edu/ jf2189/pubs.html

[3] J. Feldman, M. J. Wainwright, and D. R. Krager, "Using lineae programming to decode binary liner codes," *IEEE Trans. Inform. Theory*, vol. IT-51, no. 3, pp. 954-972, 2005.

[4] C. A. Kelley and D. Sridhara, "Pseudo codewords of Tanner Graphs," *IEEE Trans. Inform. Theory*, vol. 53, no. 11, pp. 4013-4038, Nov. 2007.

[5] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," *IEEE Trans. Inform. Theory*, submitted for publication. [Online]. Available: http://arxiv.org/abs/cs/0512078

[6] T. R. Halford and K. M. Chugg, "Random redundant iterative soft-in soft-out decoding," *IEEE Trans. Commun.*, vol. 56, no. 4, pp. 513-517, Apr. 2008.

[7] T. Hehn, J. B. Huber, S. Laendner, and O. Milenkovic, "Multiple-bases belief-propagation for decoding of short block codes," in *Proc. IEEE International Symp. Inform. Theory*, Nice, France, June 2007, pp. 311–315.

[8] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, 1978.

[9] T. Richardson, "The geometry of turbo-decoding dynamics," *IEEE Trans. Inform. Theory*, vol. IT-46, no. 1, pp. 9-23, 2000.

TABLE I
MBBP AND RRD/MRRD RELATIVE COMPLEXITY

Code	$d_{min}(\mathcal{C}^{\perp})$	Possible CGGs	Edges in $\boldsymbol{H_\ell}$	Edges in $\boldsymbol{H}$[a]	Relative Complexity
Golay [24,12,8]	8	33	192	112	1.71
BCH[31,16,7]	8	15	248	176	1.4
BCH[63,45,7]	16	3	1008	312	3.23

[a] after performing short cycle reduction as proposed in [6]

[10] M. P. C. Fossorier, M. Mihaljevi´c, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, pp. 673-680, May 1999.
[11] T. R. Halford, "The Extraction and Complexity Limits of Graphical Models for Linear Codes," Ph.D. dissertation, University of Southern California, May 2007.
[12] C. C. Lu and L. R. Welch, "On automorphism groups of binary primitive BCH codes," in *Proc. IEEE International Symp. Inform. Theory*, Trondheim, Norway, 1994, p. 51.

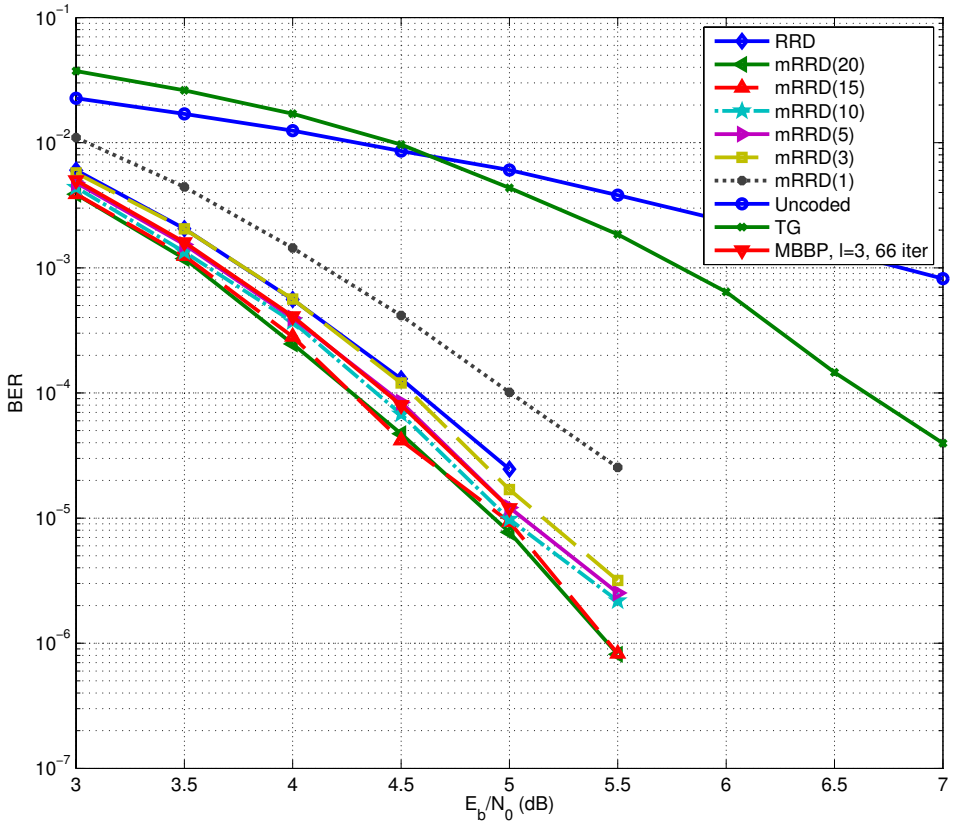
Ilan Dimnik was born in Israel in 1974. He received the B.Sc. (Cum Laude) and M.Sc. (Summa Cum Laude) degrees, both in electrical engineering, from Tel Aviv University, Israel, in 1996 and 2000, respectively.

He is currently an Engineering Manager at Horizon Semiconductors, Herzliya, Israel. His fields of interests include digital communications, error correcting codes and iterative decoding algorithms.

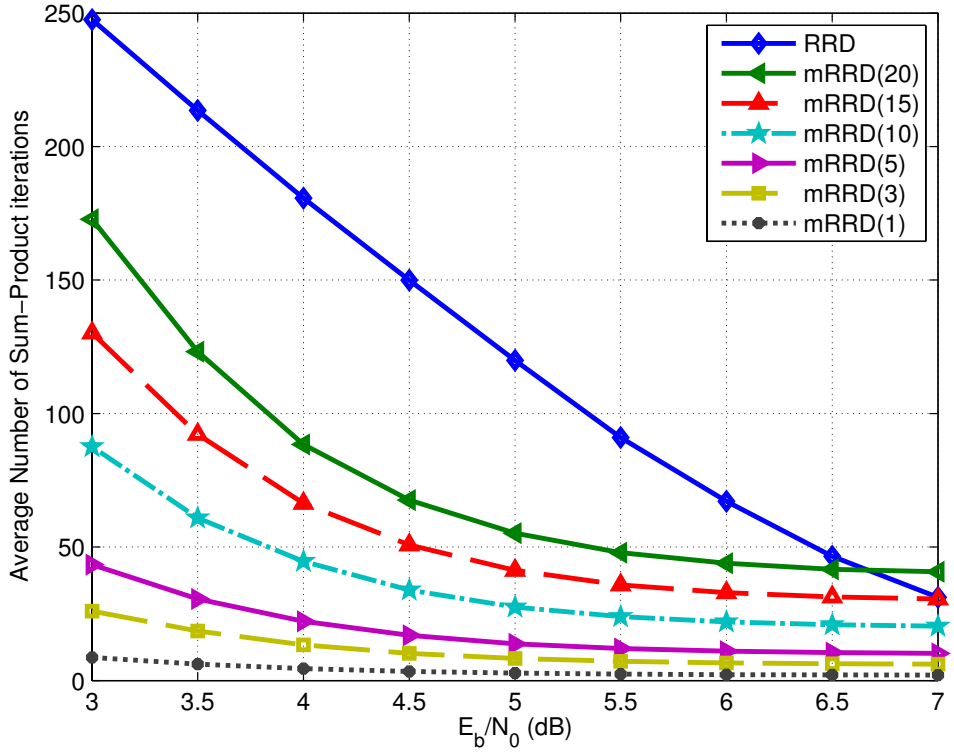
Yair Be'ery was born in Israel in 1956. He received the B.Sc. (Summa Cum Laude), M.Sc. (Summa Cum Laude), and Ph.D. degrees, all in electrical engineering, from Tel Aviv University, Israel, in 1979, 1979, and 1985, respectively.

He is currently a Professor in the Department of Electrical Engineering - Systems, Tel Aviv University, where he has been since 1985. He served as the Chair of the Department during the years 1999-2003. He is the recipient of the 1984 Eliyahu Golomb Award from the Israeli Ministry of Defense, the 1986 Rothschild Fellowship for postdoctoral studies at Rensselaer Polytechnic Institute, Troy, NY, and of the 1992 Electronic Industry Award in Israel.

His research interests include digital communications, error control coding, turbo decoding, combined coding and modulation, VLSI architectures and algorithms for systolic arrays.

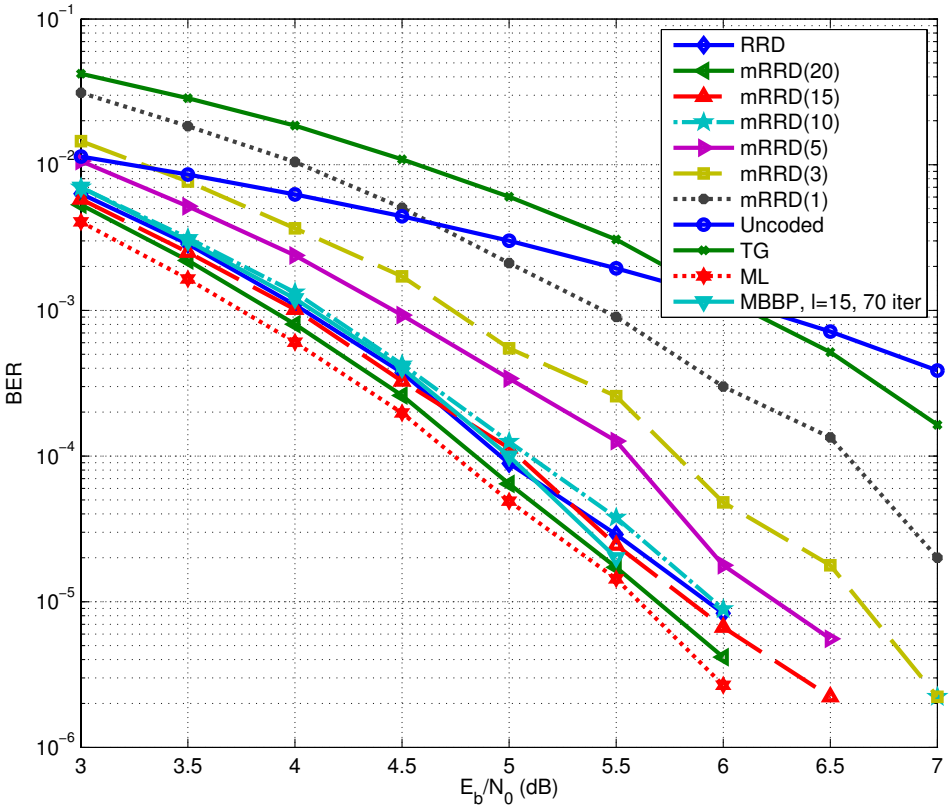


(a) BER performance

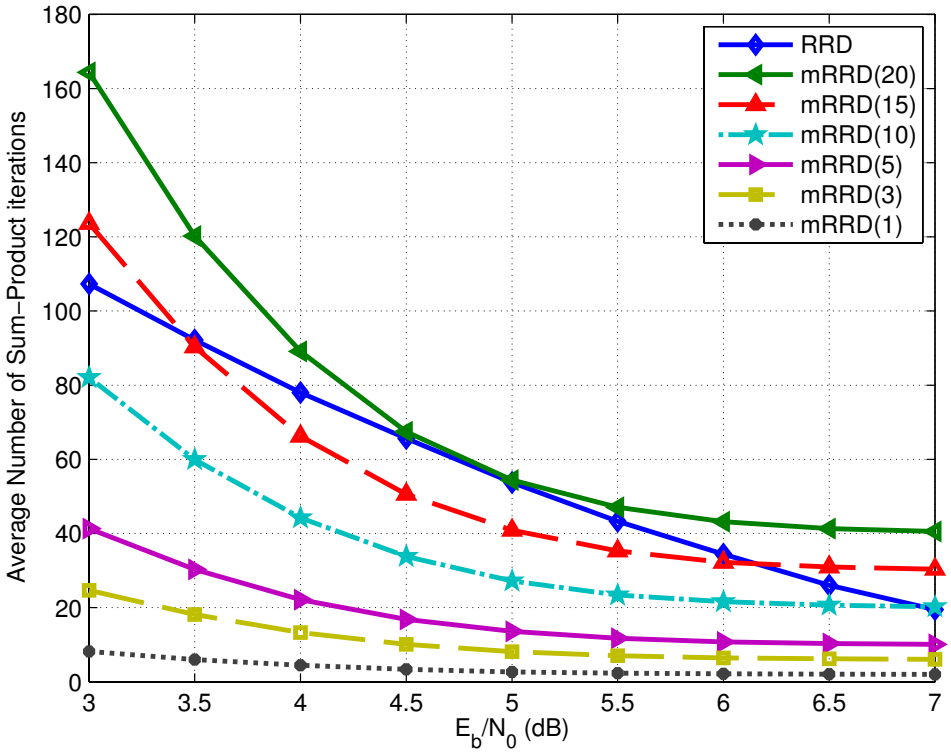


(b) Complexity charts

Fig. 1. Performance and complexity charts for BCH[63, 45, 7]



(a) BER performance



(b) Complexity charts

Fig. 2. Performance and complexity charts for extended Golay [24,12,8]