On Unequal Error Protection LDPC Codes Based on Plotkin-type Constructions *

Vidya Kumar and Olgica Milenkovic

Electrical and Computer Engineering Department,

University of Colorado, Boulder

e-mail: (kumarv, milenkov)@colorado.edu

January 24, 2006

Abstract

We introduce a new family of unequal error protection (UEP)codes, based on low-density parity-check (LDPC) component codes and Plotkin-type constructions. The codes are decoded iteratively in multiple stages, and the order of decoding determines the level of error protection. The level of UEP among the code bits is also influenced by the choice of the LDPC component-codes, and by some new reliability features incorporated into the decoding process. The proposed scheme offers a very good trade-off between code performance on one side and encoding/decoding and storage complexity on the other side. The novel approach to UEP also allows for finding simple approximations for the achievable degrees of UEP, which can be used to govern practical code design implementations.

1 Introduction

Low-density parity-check (LDPC) codes are iteratively decodable codes on graphs that exhibit capacity approaching performance over binary erasure and additive white Gaussian noise (AWGN) channels [2]. The near-optimal performance of LDPC codes over these standard channels suggests that they may also offer

*The material in this paper was presented in part at the Global Telecommunications Conference, Dallas, Texas, December 2004. The work was supported by the Colorado Center for Information Storage in Boulder.

good error-correcting performance when used over channels with non-uniform characteristics. The goal of this paper is to explore one possible LDPC code design and decoding scheme for such channels. More specifically, we propose to investigate the application of iterative decoding methods for channels that require unequal error protection (UEP). UEP of data is a necessary system requirement for many practical applications, including joint coding for parallel channels with non-uniform noise statistics, robust transmission of still images/videos over wireless channels, and storage on holographic memories. The code design problem for non-uniform channels reduces to constructing codewords, which, when transmitted over a uniform channel, have the property that different parts experience different effective noise powers. Such a situation usually arises when the codewords can be divided into frames, according to the importance of the information they convey. For storage applications, UEP schemes can be used to simplify the data processing architecture of the system, although at the cost of a certain performance loss.

We propose to design UEP-LDPC codes based on the algebraic Plotkin construction, and to decode the codeword frames in multiple stages. For this technique, one starts with a set of random-like and/or structured LDPC codes as the basic components, and then creates a code of long length in an iterative fashion. The component-codes are decoded *individually*, and in multiple rounds. The underlying soft-decision decoding algorithm operates iteratively on two different levels: one, the level of the Plotkin construction itself, and another, the level of iterative decoding of the component LDPC codes. The results of the individual decoding processes are combined in a manner that guarantees UEP characteristics for the component-codes. The decoding algorithm also incorporates a reliability feature – a threshold function that describes how the outputs of the component decoders ought to be combined. The UEP features of this scheme can be vaguely seen as arising both from the irregular structure of the parity-check matrix, where different blocks of variable nodes have different degrees, and as an inherent characteristic of the decoding scheme employed for the component-codes. A decoding method with only the first of the described features was also investigated by Dumer and Shabunov [3, 4] for soft-decision decoding of Reed-Muller (RM) codes [10], although without any reference to UEP.

The UEP techniques described in this work are conceptually fundamentally different from previously known UEP schemes [15], [12], [16]. In [15], the effect of applying the direct sum or Plotkin construction on the UEP performance of the resultant code was investigated. The resulting codes' *separation vector*, a vector which quantifies the UEP properties of a code, was described in terms of separation vectors of the component codes. A different approach to UEP was pursued in [12], [16], where codes with such properties

were designed by using irregular LDPC codes. The technique proposed in this work is based on a specialized decoding algorithm that exploits the special "nested" structure of the code and is therefore different from both the approach in [15] and the schemes in [12], [16]. Furthermore, as will be demonstrated subsequently, there exist simple approximation methods that can be used to estimate the different protection levels of Plotkin-type LDPC codes. These estimates can be used to govern the practical code-design process.

The outline of the paper is as follows: in Section II we introduce the Plotkin-type construction, relevant terminology and definitions, as well as the multi-stage, multi-round decoding algorithm. We also propose a novel method for constructing UEP codes based on LDPC component-codes. In Section III we describe how the Plotkin-type construction can lead to an UEP scheme, and analyze the UEP levels offered by the new decoding scheme. Additionally, we propose several modification of the basic decoding algorithm that further improve the performance of the coding scheme of interest. Simulation results and conclusions are presented in Section IV.

Throughout the paper, all derivations and simulations are performed for BPSK-modulated codewords transmitted over an AWGN channel, with noise samples generated according to the distribution $\mathcal{N}(0,\sigma_{ch}^2)$. The proposed UEP techniques and the corresponding analysis can be extended in a straightforward manner so as to apply to other classes of memoryless channels.

2 Plotkin-Type Constructions and Multi-Stage Decoding

The Plotkin construction, also known as the $|u|u+v|$ construction, is a code design method which increases the length of the code by combining component-codes in an additive manner [10]. For the sake of completeness, we will briefly review the most relevant characteristics of this and related construction strategies.

Definition 1: Assume that two codes, C_1 and C_2, with parameters $[n,k_1,d_1]$ and $[n,k_2,d_2]$ are given. The Plotkin code C with components C_1 and C_2 is defined as:

$$C = \{|\boldsymbol{u}|\boldsymbol{u}+\boldsymbol{v}|, \boldsymbol{u} \in C_1, \boldsymbol{v} \in C_2\}, \tag{1}$$

where $||$ denotes word-concatenation. The parameters of the code C are $[2n,(k_1+k_2),d=\min(2d_1,d_2)]$. The codes C_1 and C_2 will be referred to as the *component-codes*. Note that the first half of the codeword contains k_1 information bits, while the second half contains k_2 information bits.

The Plotkin construction can be applied recursively up to a prescribed depth m. For example, if each of

the component-codewords $\mathbf{u}$ and $\mathbf{v}$ is itself of Plotkin-type, constructed from codes with components $(\mathbf{u}_1, \mathbf{v}_1)$ and $(\mathbf{u}_2, \mathbf{v}_2)$, then

$$\mathbf{u} = |\mathbf{u}_1|\mathbf{u}_1 + \mathbf{v}_1|, \quad \mathbf{v} = |\mathbf{u}_2|\mathbf{u}_2 + \mathbf{v}_2|.$$

The components $(\mathbf{u}_1, \mathbf{v}_1)$ and $(\mathbf{u}_2, \mathbf{v}_2)$ can themselves be taken from a Plotkin-type code, and such a "nested construction" can be extended up to depth m. For a depth m Plotkin-type code, the longest component-codes specifying C (i.e. $\mathbf{u}$ and $\mathbf{v}$) will be called *depth-one component-codes*, while those defining $\mathbf{u}$ and $\mathbf{v}$ are adequately named *depth-two component-codes*. This terminology extends in an obvious manner to components at any given depth. Note that there are 2^l depth-l component-codes, for $1 \leq l \leq m$. The generator and parity-check matrix of a Plotkin-type code of depth $m = 1$ are of the form:

$$G^{(1)} = \begin{bmatrix} G_1 & G_1 \\ \mathbf{0} & G_2 \end{bmatrix}, H^{(1)} = \begin{bmatrix} H_1 & \mathbf{0} \\ H_2 & H_2 \end{bmatrix}, \tag{2}$$

where G_i and H_i represent the generator and parity-check matrix of the codes C_i, $i = 1, 2$, respectively.

Plotkin-type codes usually have very good error-correcting properties. For example, the well-known class of Reed-Muller codes [10] can be devised in this manner.

There exist many other code construction techniques closely related to the Plotkin method. Examples include the (a, b, x) construction [10], and extensions thereof, described bellow.

Definition 2: Let C_1 be a code with parameters $[n, k_1, d_1]$, and let C_2 be a code with parameters $[n, k_2, d_2]$. A code C defined according to

$$C = \{|\mathbf{a} + \mathbf{x}|\mathbf{b} + \mathbf{x}|\mathbf{a} + \mathbf{b} + \mathbf{x}|, \mathbf{a}, \mathbf{b} \in C_1, \mathbf{x} \in C_2\}, \tag{3}$$

is called an (a, b, x)-type code. The length of C is $3n$, while its rate is $(2R_1 + R_2)/3$, with $R_1 = k_1/n$, $R_2 = k_2/n$. The generator and parity check matrix of an (a, b, x)-type of code are of the form:

$$G_{a,b,x} = \begin{bmatrix} G_2 & G_2 & G_2 \\ G_1 & \mathbf{0} & G_1 \\ \mathbf{0} & G_1 & G_1 \end{bmatrix}, \text{ and } H_{a,b,x} = \begin{bmatrix} H_2 & H_2 & H_2 \\ H_1 & \mathbf{0} & H_1 \\ \mathbf{0} & H_1 & H_1 \end{bmatrix},$$

where G_i and H_i represent the generator and parity check matrix of the codes C_i, $i = 1, 2$, respectively.

Let us briefly describe some of the properties of Plotkin-type codes of depth $m = 1$ based on LDPC component-codes, under standard encoding and message-passing decoding.

First, the component-codes C_1 and C_2 of a Plotkin-type code can be chosen arbitrarily from the class of linear codes; hence, LDPC codes are good candidates for this technique. For most LDPC codes, the encoding complexity scales as a square function of the length of the code. But Plotkin-type of encoding is performed in several stages, using short information sequences. This implies that for a depth m construction of resulting length N, the component-codes are of length $N/2^m$, and can be encoded much faster individually than jointly. Additionally, LDPC component-codes can be decoded using the sum-product algorithm. There are several additional advantages for using LDPC component-codes in the described setting: firstly, the component-codes can be very long, and yet efficiently decoded producing soft outputs; secondly, one can choose some component-codes to be structured while others to be random-like. In this way, the performance of the coding scheme can be traded for storage and encoding complexity and vise versa. This is why, we will focus exclusively on Plotkin-type codes with LDPC component-codes.

The next observation is that in order to ensure large error resilience for a given set of variable nodes, it is desirable to have these nodes involved in as many check-equations as possible. Hence, one *may* expect the first half of the variables in a Plotkin-type code C to have a higher level of error protection than the second half. But such requirement can be seen not to be sufficient in general. Good irregular codes usually have a degree distribution optimized with respect to the overall codes' performance. Large degree nodes in the code graph have a high level of error-protection during the first several iterations of belief propagation, but due to the non-separable nature of iterative decoding, these nodes also tend to improve the decisions of the low-degree nodes in the final stages of decoding. Hence, for good irregular codes there should be a very small difference in the error-protection levels of the nodes (see Figure 11 for an illustration of this phenomena). Furthermore, in an arbitrary irregular LDPC code many low-degree variables may share a check with a variable that has large degree; such a configuration tends to cause unreliable information to propagate to reliable variables. This observation would suggest the need to design a code structure where nodes of high degree share many checks with other nodes of high degree, and very few with low-degree nodes. A code design approach satisfying such constraints is clearly very hard to devise. This is why an alternative solution, like the one pursued in this paper, is desirable.

It is also to be noted that $H^{(1)}$ has a block-row with two identical copies of H_2 placed side by side. Hence, any column in H_2 with $c > 1$ non-zero entries introduces at least $\binom{c}{2}$ four-cycles in the Tanner graph of the

code. Consequently, the *complete* code graph of a Plotkin-type LDPC code necessarily has a large number of cycles of length four. This implies that decoding Plotkin-type LDPC codes in a *global* iterative fashion will produces very poor results, regardless of the quality of the component-codes. On the other hand, as will be shown in the next section, Plotkin-type codes can be decoded in multiple stages: each stage involves one component only, and one can repeat the different decoding stages multiple times. If the component-codes have Tanner graphs without short cycles, then such a decoding method does not lead to a poor performance of the general scheme. It is also worth pointing out that the multi-stage decoding process may not *fully* exploit the codes error-correcting potential, but it can be analytically shown to achieve a very good quality of UEP differentiation. The multi-stage (MS) decoding algorithm, and its multi-round (MR-MS) version will be described next.

2.1 Multi-stage (MS) Decoding

Plokin-type codes are inherently suitable for MS decoding. The MS decoding process can be most easily described in terms of the binary tree representation of the component-codes hierarchy. The tree is constructed by starting from a root node, representing a codeword $\mathbf{z} \in C$. Two branches leave the root node: one, the left branch, leads to the **u** component of **z**, and is labeled 0; the other, the right branch, leads to the **v** component, and is labeled 1. The same procedure is repeated recursively, starting with the components **u** and **v** as root nodes. Hence, each component-code at depth m of the Plotkin construction has an equivalent binary representation $\{a_i\}_{i=1}^{m}$. The binary representations are assumed to be ordered in the *standard* lexicographical order. A vector **a** in such an ordered list is said to be ranked higher than a vector **b**, if it is positioned anywhere below **b**. Decoding is performed only on depth-m components, such that a component of higher rank is always decoded before one of lower rank. For the sake of simplicity, we will describe the decoding process only for depth $m = 1$ codes. The generalization to larger values of m is straightforward.

Let the transmitted codeword be $\mathbf{z}=|\mathbf{u}|\mathbf{u}+\mathbf{v}| = \{z_i\}_{i=1}^{n} \in \{-1,1\}^n$, and the received vector be $\mathbf{y} = \{y_i\}_{i=1}^{n} \in \mathcal{R}^n$. Assume that all the transmitted bits are equally likely. For the described setting, we define the *log-likelihood* $\mathcal{L}_i^y$ and the *spread* $\mathcal{S}_i^y$ (following the same terminology as in [3]) of the i-th symbol of the word **y** as

$$\mathcal{L}_i^y = \log(p(z_i = 1/y_i)/p(z_i = -1/y_i)) = 2y_i/\sigma_{ch}^2,$$
$$\mathcal{S}_i^y = p(z_i = 1/y_i) - p(z_i = -1/y_i) = \tanh(\mathcal{L}_i^y).$$

Let **y'**, **y"** denote the left and the right half of the received vector **y**. The log-likelihoods and spreads of the

i-th coordinate of the two halves of **y** are $\mathcal{L}_i^{y'}$, $\mathcal{L}_i^{y''}$, $\mathcal{S}_i^{y'}$ and $\mathcal{S}_i^{y''}$, respectively. It can be shown that $\mathcal{L}_i^{y'}$ and $\mathcal{L}_i^{y''}$ are both distributed according to the Gaussian distribution $\mathcal{N}(2/\sigma_{ch}^2, 4/\sigma_{ch}^2)$. For *noise-free* transmission, one has:

$$\mathbf{v} = \mathbf{y'} + \mathbf{y''} \quad \mod 2. \tag{4}$$

For $\sigma_{ch} \neq 0$, the above formula can recover only an estimate of **v**, and the available information for finding the estimate consists of the likelihoods and the spreads of the various coordinates (bits) of the received vector **z**. By invoking the well-known duality principle [5], it is straightforward to see that the spread and the log-likelihood of the *i*-th coordinate of the word **v** can be determined as:

$$S_i^v = \mathcal{S}_i^{y'} \mathcal{S}_i^{y''}, \; L_i^v = \tanh^{-1}(S_i^v). \tag{5}$$

Since **u** is used in constructing both halves of the codeword **z**, one needs to distinguish between the two incidences of **u**. These will be denoted by **u'** and **u"**, while the respective log-likelihoods of their *i*-th coordinate will be denoted by $L_i^{u'}$ and $L_i^{u''}$. Clearly, $L_i^{u'} = \mathcal{L}_i^{y'}$.

The MS decoding process can now be summarized as follows.

- ***Step 1 – Computation of the log-likelihoods of v:*** From the received vector **y**, the log-likelihood $\mathcal{L}^{\mathbf{y}}$ and the spread $\mathcal{S}^{\mathbf{y}}$ are evaluated. Based on $\mathcal{S}_i^{y'}$ and $\mathcal{S}_i^{y''}$, the spread and log-likelihood of **v** are computed according to (5).

- ***Step 2 – Decoding of v:*** The log-likelihoods of **v** evaluated in Step 1 are used for soft-input, soft-output decoding. The output of the decoder for codeword **v** is denoted by $\hat{\mathbf{v}}$.

- ***Step 3 – Computation of the log-likelihoods of u' and u":*** As already shown, $L_i^{u'} = \mathcal{L}_i^{y'}$, while $L_i^{u''}$ can be determined based on the soft output $\hat{\mathbf{v}}$. This is accomplished by observing that if $\hat{v}_i = -1$ then,

$$L_i^{u''} = \log\left(\frac{p(u_i + v_i = -1/y_i)}{p(u_i + v_i = 1/y_i)}\right) = -\mathcal{L}_i^{y''}, \tag{6}$$

  while if $\hat{v}_i = 1$, then $L_i^{u''} = \mathcal{L}_i^{y''}$. In summary, $L_i^{u''} = \hat{v}_i \mathcal{L}_i^{y''}$.

- ***Step 4 – Decoding of u:*** The values of $L_i^{u'}$ and $L_i^{u''}$ represent the log-likelihoods of the *i*-th coordinate of two repeated observations of **u**, namely |**u'**|**u"**|. Based on these observations, the log-likelihoods of

the coordinates of $\mathbf{u}$ can be obtained according to:

$$L_i^u = L_i^{u'} + L_i^{u''}. \tag{7}$$

Finally, $L^{\mathbf{u}}$ is used to initialize the decoder for $\mathbf{u}$, which subsequently produces the estimate $\hat{\mathbf{u}}$.

A pictorial description of the algorithm is provided in Figure 1.

Example 1 – Decoding Depth $m = 2$ ***Plotkin-Type Codes****:* For a depth two Plotkin-type code, the channel output sequence can be represented in the form $\mathbf{y} = |\mathbf{y}^1|\mathbf{y}^2|\mathbf{y}^3|\mathbf{y}^4| = |\mathbf{y'}|\mathbf{y''}|$, where, $\mathbf{y'} = |\mathbf{y}^1|\mathbf{y}^2|$ and $\mathbf{y''} = |\mathbf{y}^3|\mathbf{y}^4|$. The log-likelihoods and spreads of $\mathbf{v}$ can be computed according to (5), as described for the MS algorithm above. These results are then used to compute the spread of $\mathbf{v}_2$, and this component-codeword is decoded first. The log-likelihoods and spreads of $\mathbf{u}_2$ are calculated according to (6) and (7), using information provided by the decoder output for $\mathbf{v}_2$. Using the decoder output for $(\mathbf{u}_2, \mathbf{v}_2)$, the log-likelihoods and spreads of $\mathbf{u}$ are evaluated according to (6) and (7); a similar procedure is performed in order to find estimates for $\mathbf{v}_1$ and $\mathbf{u}_1$, thereby leading to an estimate of the complete codeword.

2.2 Multi-Round, Multi-Stage (MR-MS) Decoding

Based on the description of the MS algorithm, one can observe that once the complete codeword is decoded using a soft-output decoder, new (and more accurate) estimates for the log-likelihoods of the component-codes are available. This suggest that one should perform the MS decoding algorithm in an iterative (recursive) manner. For example, after one round of MS decoding, the log-likelihood values at the output of the iterative decoder for $\mathbf{u}$, rather than the original channel information, can be used to find an improved estimate for $\mathbf{v}$. Such an iterative scheme is meaningful since the $\mathbf{u}$ component after decoding usually has a significantly reduced number of errors, compared to the same component before decoding. In other words, at each round of decoding, better estimates of $\mathbf{u}$ are used to improve the estimate of $\mathbf{v}$. We will refer to this repeated application of MS decoding as to the *MR process*. A description of the MR-MS algorithm for a Plotkin-type construction of depth $m = 1$ is summarized below. As noted before, a straightforward extension is possible for decoders operating on codes of depth $m > 1$ as well.

- ***Step 1 – MS Decoding of u and v:*** Based on the channel log-likelihoods $|\mathcal{L}^{\mathbf{y'}}|\mathcal{L}^{\mathbf{y''}}|$, the MS decoding algorithm is executed to obtain soft-output estimates for the $\mathbf{u}$ and $\mathbf{v}$ component, termed $\mathbf{u}_{\{1\}}, \mathbf{v}_{\{1\}}$.

- ***Step 2 – Iterating the MS algorithm:*** The round-counting index is set to $r = 2$. For a j-round decoder, the following instructions are followed:

  (1.) The log-likelihood values $L^{\hat{\mathbf{u}}_{\{r-1\}}}$ of the output of the decoder for $\mathbf{u}_{\{r-1\}}$ are computed.

  (2.) The MS decoding data is initialized to $|L^{\hat{\mathbf{u}}_{\{r-1\}}}|\mathcal{L}^{\mathbf{y}"}|$; complete MS decoding is performed to obtain new estimates for the component-codes, namely $\mathbf{u}_{\{r\}}$ and $\mathbf{v}_{\{r\}}$. The counting index is increased, $r \rightarrow r+1$.

  (3.) Steps **(1.)** and **(2.)** are repeated, unless $r = j$. For $r = j$, the decoding process is terminated. The final estimates of $\mathbf{u}$ and $\mathbf{v}$ are of the form $\mathbf{u}_{\{r\}}$ and $\mathbf{v}_{\{r\}}$.

3 Plotkin-type UEP Schemes: Intuition and Analysis

In this section, we will describe why Plotkin-type constructions with MS or MR-MS decoding provide for UEP of the transmitted component-codewords. We will also perform an analysis that shows the approximate levels of error protection achieved by the different components. The analysis will be presented for a depth-one construction only, but it can be extended to higher depths as well. In order to simplify the analysis, we will define the notion of an *equivalent channel*. Although not all distributions of the noise present in the component-codes at various stages of the decoding algorithm are Gaussian, we will only use the first two moments of the variables involved, and assume that for all equivalent channels considered the Signal-to-Noise Ratio (SNR) is defined as for a Gaussian channel. Recall that the SNR for a Gaussian channel variable with distribution $\mathcal{N}(m,\sigma^2)$ is defined as m^2/σ^2. These ideas are formalized by the following set of *simplifying* assumptions:

Assumption 1: For a Plotkin-type construction, the means and variances of the noise "embedded" in the $\mathbf{u}$ and $\mathbf{v}$ components during and after decoding will be used to specify an *equivalent Gaussian variable*[1]. An AWGN channel corresponding to such a random variable will be referred to as the *equivalent* AWGN (or simply, equivalent) channel. Furthermore, if the mean and variance of two variables X, Y, say, are m_x, σ_x^2, and m_y, σ_y^2, we say that the variables experience an equivalent channel SNR if $m_x^2/\sigma_x^2 = m_y^2/\sigma_y^2$. This implies that for a variable X experiencing the same channel as Y, and for which $m_x = 1$, one would have $\sigma_{x,equiv}^2 = \sigma_y^2/m_y^2$.

Assumption 2: The log-likelihood variables of different component-codes at any decoding stage are assumed to be independent. A similar assumption was also used for conducting the analysis in [3].

[1]This is, in essence, equivalent to the Gaussian approximation method described in [2]

Assumption 3: For the proofs of Lemmas 2-5 only, it will be assumed that no errors are present at any previous stage of the MS or MR-MS decoding process. Alternatively, any component-code is assumed to be decoded based on perfect estimates provided by higher-ranked components (a valid assumption for high SNRs). This assumption was also used in [4], for the purpose of determining the effective minimum distance of RM codes under a special form of MS decoding.

3.1 Analysis of MS Decoding: The Perfect Decoder Model

Let us start by characterizing the equivalent channel for the component **v**. Based on (5), it is straightforward to show that the noise variance of the equivalent channel experienced by **v** is $\sigma_v^2 = \sigma_{ch}^4$, provided that $\sigma_{ch} \to \infty$. The same result was proved in [3], based on the fact that $E[S^{\mathbf{v}}] \sim \sigma_{ch}^{-4}$. The following Lemma describes the equivalent channel of **v** for the case $\sigma_{ch} \to 0$.

Lemma 1: For $\sigma_{ch} \to 0$, the variance of the equivalent channel experienced by the **v** component-codeword, σ_v^2, equals σ_{ch}^2.

Proof: Based on the analysis in [4] it can be seen that

$$E[\mathcal{S}^{\mathbf{y}}] \sim 1 - \sqrt{\pi/2}\ \ \sigma_{ch}\ \exp(-1/2\sigma_{ch}^2). \tag{8}$$

Therefore, $E[S^{\mathbf{v}}] = E[\mathcal{S}^{\mathbf{y'}}]\ E[\mathcal{S}^{\mathbf{y''}}] \sim \left(1 - \sqrt{\pi/2}\ \sigma_{ch}\ \exp(-1/2\sigma_{ch}^2)\right)^2$. Ignoring higher order terms in the expansion of the square leads to the following approximation:

$$E[S^{\mathbf{v}}] \sim 1 - 2\sqrt{\pi/2}\ \ \sigma_{ch}\ \exp(-1/2\sigma_{ch}^2). \tag{9}$$

Since Formula (8) holds for any Gaussian variable, σ_v^2 can be obtained from the following equation

$$1 - \sqrt{\pi/2}\ \ \sigma_v\ \exp(-1/2\sigma_v^2) = 1 - 2\sqrt{\pi/2}\ \ \sigma_{ch}\ \exp(-1/2\sigma_{ch}^2). \tag{10}$$

Consequently, $2\,\sigma_{ch}\ \exp(-1/2\sigma_{ch}^2) \sim \sigma_v\ \exp(-1/2\sigma_v^2)$. By taking logarithms of both sides of the last asymptotic equality, and by neglecting lower order terms in the expansion (in this case $\log(\sigma_v)$), one obtains:

$$\begin{aligned} \sigma_v &= \sigma_{ch} / \sqrt{1 - 2\sigma_{ch}^2 \log(2\sigma_{ch})} \\ &= \sigma_{ch}\left(1 + \sigma_{ch}^2 \log(2\sigma_{ch})\right) \quad \text{(since } 1/\sqrt{1-x} \approx 1 + x/2 \text{ for } x \to 0\text{)}. \end{aligned}$$

Repeating the above bootstrapping process shows that $\sigma_v = \sigma_{ch}(1+\sigma_{ch}^2 \log 2) \approx \sigma_{ch}$. ■

For both $\sigma_{ch} \to \infty$ and $\sigma_{ch} \to 0$, by invoking Assumption 3 and (7), it holds that $L_i^u = \mathcal{L}_i^{y'} + \mathcal{L}_i^{y''}$. Hence, L_i^u is distributed according to $\mathcal{N}(4/\sigma_{ch}^2, 8/\sigma_{ch}^2)$, as both $\mathcal{L}_i^{y'}$ and $\mathcal{L}_i^{y''}$ are Gaussian $\mathcal{N}(2/\sigma_{ch}^2, 4/\sigma_{ch}^2)$ variables. This shows that **u** experiences an equivalent channel with noise variance $\sigma_{ch}^2/2$. The last result now clearly demonstrates how a UEP gradation is achieved by a Plotkin-type code construction and MS decoding. The vector **v** is decoded first, by using "indirect" information, and therefore experiences an equivalent channel degraded compared to the original one. On the other hand, **u** can be estimated from two different parts of the codeword, and therefore has a higher level of error-protection. For high-depth Plotkin constructions there exists a fine gradation in the error protection quality. Additional variations in error protection levels can be also achieved by suitably choosing the component-codes. It is worthwhile to point out that with this scheme a gradation of UEP levels can also be achieved for different parts of the *information sequence*. More details about UEP properties of the information symbols are provided in Section 3.3.

The next lemmas describe the equivalent noise-variances experienced by component-codes at different depths. The results are derived based on the binary tree representation of the Plotkin construction and Assumption 3. The two cases $\sigma_{ch} \to \infty$ and $\sigma_{ch} \to 0$ are treated separately.

Lemma 2: Let a component-code C_{comp} of a Plotkin-type construction of depth m have the binary representation $\{a_i\}_{i=1}^{l}$, $l \leq m$. The equivalent noise variance $\sigma_{comp,\,l}^2$ experienced by C_{comp}, for $\sigma_{ch} \to \infty$, is obtained from the recursion:

$$\sigma_{comp,\,i}^2 = \begin{cases} \sigma_{comp,\,i-1}^2/2, & \text{if } a_i = 0, \\ \sigma_{comp,\,i-1}^4, & \text{if } a_i = 1, \end{cases} \tag{11}$$

where $\sigma_{comp,\,0}^2 = \sigma_{ch}^2$.

Proof: For large values of σ_{ch}, **v** experiences an equivalent channel with noise variance σ_{ch}^4, whereas **u** experiences an equivalent channel with noise variance $\sigma_{ch}^2/2$. Every right branch followed (corresponding to a bit $a_i = 1$) in the binary tree results in squaring the noise variance; similarly, every left branch followed (corresponding to a bit $a_i = 0$) leads to halving the noise variance. This proves the claimed result. ■

Lemma 3: Let a component-code C_{comp} of a Plotkin-type construction of depth m have the binary representation $\{a_i\}_{i=1}^{l}$, $l \leq m$. The equivalent noise variance $\sigma_{comp,\,l}^2$ experienced by C_{comp}, for $\sigma_{ch} \to 0$, is obtained

from the recursion:

$$\sigma^2_{comp,i} = \begin{cases} \sigma^2_{comp,i-1}/2, & \text{if } a_i = 0, \\ \sigma^2_{comp,i-1}, & \text{if } a_i = 1, \end{cases} \tag{12}$$

where $\sigma^2_{comp,0} = \sigma^2_{ch}$.

Proof: The proof follows along the same lines as the proof of Lemma 2. ■

Remark 1: Note that the results of Lemma 2 and Lemma 3 show that the operations performed on the equivalent noise variance differ only when a bit $a_i = 1$ in encountered in the binary representation of a component-code.

The above Lemmas can be generalized for the case of MR-MS decoding as demonstrated below.

Lemma 4: Let a component-code C_{comp} of a Plotkin-type construction of depth m have the binary representation $\{a_i\}_{i=1}^{l}$, $l \leq m$. The equivalent noise variance $\sigma^2_{comp,\,l}$ experienced by C_{comp}, for $\sigma_{ch} \to \infty$, and for j rounds of MR-MS decoding, is given by (11), where

$$\sigma^2_{comp,1} = \sigma^2_{ch}/(j+1), \text{ for } a_1 = 0, \qquad \sigma^2_{comp,1} = \sigma^4_{ch}/j, \text{ for } a_1 = 1. \tag{13}$$

Proof: During the first round of MR-MS decoding, both **u** and **u+v** experience channel noise with variance σ^2_{ch}. At the beginning of the second round of MR-MS decoding, **u** experiences an equivalent channel with noise variance $\sigma^2_{ch}/2$. The log-likelihoods of **u+v** are fixed to the channel estimates throughout the whole decoding process. In other words, the second half of the codeword always experiences a channel with noise variance σ^2_{ch}. This implies that $E[S^v] \sim (\sigma_{ch}/\sqrt{2})^{-2}\,\sigma^{-2}_{ch}$. Therefore, during the second round of decoding, **v** experiences a channel with equivalent noise variance $\sigma^2_v = \sigma^4_{ch}/2$. The log-likelihoods of the first half of the codeword are now distributed according to $\mathcal{N}(8/\sigma^2_{ch}, 16/\sigma^2_{ch})$, while the log-likelihoods of the second half of the codeword are distributed according to $\mathcal{N}(4/\sigma^2_{ch}, 8/\sigma^2_{ch})$. Based on (7) and Assumption 3, one can show that the log-likelihoods of **u** are distributed according to $\mathcal{N}(12/\sigma^2_{ch}, 24/\sigma^2_{ch})$. This is equivalent to the statement that **u** experiences a channel with noise variance $\sigma^2_{ch}/3$. Repeating the argument iteratively, up to j-th round of decoding, shows that **u** and **v** experience equivalent noise variances $\sigma^2_{ch}/(j+1)$ and σ^4_{ch}/j, respectively. These variances are used as the initial values for the recurrence described in Lemma 2. ■

Corollary 1: The equivalent noise variance experienced by any component-code at a fixed stage of MS decoding, decreases with the number of rounds j of the MR process.

Proof: As already shown, the component-codes at depth one experience equivalent noise variances $\sigma_{ch}^2/(j+1)$ and σ_{ch}^4/j after j rounds of MR-MS decoding. These variances clearly decrease when j increases. Hence, components at higher depths also experiences channels with equivalent noise variances decreasing with j. This is due to the fact that the initial conditions for the recursion of Lemma 4 are determined by the depth-one component-codes noise variances only. ■

Lemma 5: Let a component-code C_{comp} of a Plotkin-type construction of depth m have the binary representation $\{a_i\}_{i=1}^{l}$, $l \leq m$. The equivalent noise variance $\sigma_{comp,l}^2$ experienced by C_{comp}, for $\sigma_{ch} \to 0$, and for j rounds of MR-MS decoding is given by (12), where

$$\begin{aligned} \sigma_{comp,1}^2 &= \sigma_{ch}^2/(j+1), \quad \text{for} \quad a_1 = 0, \\ \sigma_{comp,1}^2 &= \sigma_{ch}^2, \qquad\quad \text{for} \quad a_1 = 1. \end{aligned} \tag{14}$$

Proof: In the first round of decoding, both **u** and **u+v** experience an equivalent noise variance σ_{ch}^2. In the second round of decoding, the first half of the codeword experiences a channel with equivalent noise variance $\sigma_{ch}^2/2$. As in Lemma 4, the log-likelihood values of **u+v** remain fixed, implying that the second half of the codeword always experiences an equivalent noise variance σ_{ch}^2. Hence, for $\sigma_{ch} \to 0$, according to (8), one has:

$$\begin{aligned} E[S^{\mathbf{v}}] &\sim \left(1 - \sqrt{\pi/2}\ (\sigma_{ch}/\sqrt{2})\ \exp(-1/\sigma_{ch}^2)\right)\left(1 - \sqrt{\pi/2}\ \sigma_{ch}\ \exp(-1/2\sigma_{ch}^2)\right) \\ &\sim 1 - \sqrt{\pi/2}\,(\sigma_{ch}/\sqrt{2})\ \exp(-1/\sigma_{ch}^2) - \sqrt{\pi/2}\ \sigma_{ch}\ \exp(-1/2\sigma_{ch}^2). \end{aligned} \tag{15}$$

Applying the bootstrapping technique as described in Lemma 1, one can prove that based on (15), the equivalent noise variance observed by **v** is asymptotically σ_{ch}^2. It can also be shown that **v** experiences the same noise variance σ_{ch}^2 independently from the number of rounds of MR-MS decoding. On the other hand, the noise variances experienced by **u** can be derived in the same way, and *are* of the same form, as described in Lemma 4. ■

Example 2 – Effective Noise Variances: Based on Lemma 2-5, one can show that a depth $m = 2$ and $j = 1$-round MR-MS decoding algorithm results in equivalent noise variances of the four component-codes equal to

$$\begin{aligned} &\sigma_{ch}^2/4, \sigma_{ch}^4/4, \sigma_{ch}^4/2, \ \text{ and } \ \sigma_{ch}^8; \quad \text{for} \quad \sigma_{ch} \to \infty; \\ &\sigma_{ch}^2/4, \sigma_{ch}^2/2, \sigma_{ch}^2/2, \ \text{ and } \ \sigma_{ch}^2; \quad \text{for} \quad \sigma_{ch} \to 0; \end{aligned} \tag{16}$$

3.2 Analysis of MS Decoding: Incorporating Decoder Error Probabilities

The results of the previous subsection do not provide sufficiently accurate approximations for the exact forms of the equivalent noise variances, due to the fact that they are based on the assumption that no decoding errors occur in any of the component-codes (Assumption 3). This assumption is, of course, not valid for low-to-medium channel SNR values. For these SNR values, the properties of the component-codes, as well as the characteristics of the decoding algorithm, have a strong influence on the equivalent noise variance. In this section, we will show how to find more accurate approximations for the special case of Plotkin-type constructions with iteratively decoded LDPC component-codes. For this purpose, we will use Gaussian Approximation (GA) technique for density evolution presented in [2]. Since $\mathbf{v}$ is the *first* component to be decoded, we can use the expressions for the equivalent noise variances of $\mathbf{v}$ derived in the previous subsection. For large values of σ_{ch}, $\sigma_v^2 = \sigma_{ch}^4$, while for small values of σ_{ch}, $\sigma_v^2 = \sigma_{ch}^2$.

Assume that the densities of the messages passed during sum-product decoding are Gaussian variables. Let $\lambda(x)$ and $\rho(x)$ be the degree distributions of the variable and the check nodes, respectively. For an AWGN channel with noise variance σ_{ch}^2, the message variables can be approximated by Gaussian variables distributed according to $\mathcal{N}(t_l, 2t_l)$. Here, the mean, t_l, at the l^{th} iteration of sum-product decoding is given by [2]:

$$t_l = \sum_{j=2}^{d_r} \rho_j \phi^{-1}\left(1 - \left[1 - \sum_{i=2}^{d_l} \lambda_i \phi(s + (i-1)t_{l-1})\right]^{j-1}\right), \tag{17}$$

where d_l and d_r are the maximum degree of the variable and check nodes, respectively, and $s = 2/\sigma_{ch}^2$. The function ϕ is defined as:

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{\mathcal{R}} \tanh\frac{u}{2} \exp\left(-\frac{(u-x)^2}{4x}\right) du & \text{if } x > 0 \\ 1 & \text{if } x = 0. \end{cases}$$

The equations above imply that for a left-regular code with variable degree i, after l iterations of sum-product decoding, one can approximate the log-likelihoods of the variable nodes by Gaussian variables, with distribution $\mathcal{N}(s + it_l, 2(s + it_l))$. Consequently, the probability of error after l iterations [2] of sum-product decoding can be approximated by:

$$p_e = Q\left(\sqrt{\frac{s + it_l}{2}}\right). \tag{18}$$

When $\lambda_2 = 0$, and $\sigma_{ch} \to 0$, $t_l \sim d'(i-1)^l$, where d' is a positive constant that depends on the degree distribution of the code and s only [2].

Lemma 6: The equivalent noise variance of the channel experienced by $\hat{\mathbf{v}}$, the output of the sum-product decoder for $\mathbf{v}$, is $\sigma^{*2}_{\hat{\mathbf{v}}} = 2/(s+it_l)$. Here, $s = 2/\sigma^4_{ch}$, for $\sigma_{ch} \to \infty$, and $s = 2/\sigma^2_{ch}$, for $\sigma_{ch} \to 0$.

Proof: The proof is a straightforward consequence of the observation that the log-likelihoods of variable nodes after l iterations of sum-product decoding, $\hat{L}^v_i(l)$, can be approximated by a Gaussian $\mathcal{N}(m_{\hat{v}} = s + it_l, \sigma^2_{\hat{v}} = 2(s+it_l))$ variable, where $s = 2/\sigma^2_v$, and σ^2_v depends on σ^2_{ch} as described in previous subsection. ■

Theorem 1: The mean and the variance of the variable $L^{u''}_i$ are approximately equal to $2(1-2p_e)/\sigma^2_{ch}$ and $4/\sigma^2_{ch} + 16p_e(1-p_e)/\sigma^4_{ch}$, respectively. Consequently, the mean and the variance of the variable L^u_i are $4(1-p_e)/\sigma^2_{ch}$ and $8/\sigma^2_{ch} + 16p_e(1-p_e)/\sigma^4_{ch}$, respectively. Here, p_e denotes the bit error probability of $\mathbf{v}$, and is given by (18).

Proof: Assume without loss of generality that the all-ones codeword was transmitted. An error occurs in $\mathbf{v}$ when $\hat{v}_i$ is decoded to -1, so that from (18) it can be shown that:

$$p\{\hat{v}_i = -1\} = p_e, \quad p\{\hat{v}_i = 1\} = 1 - p_e,$$
$$E[\hat{v}_i] = (-1)p_e + 1(1-p_e) = 1 - 2p_e, \quad E[\hat{v}_i{}^2] = (-1)^2(p_e) + (1)^2(1-p_e) = 1.$$

Note that $\mathcal{L}^{y''}_i$ is distributed according to $\mathcal{N}(2/\sigma^2_{ch}, 4/\sigma^2_{ch})$. Therefore, based on Assumption 2 and the fact that $L^{u''}_i = \hat{v}_i \, \mathcal{L}^{y''}_i$, one has:

$$E[L^{u''}_i] = (1-2p_e)\, 2/\sigma^2_{ch}, \; E[(L^{u''}_i)^2] = 4/\sigma^2_{ch} + 4/\sigma^4_{ch}. \tag{19}$$

This implies

$$\sigma^2(L^{u''}_i) = 4/\sigma^2_{ch} + 16p_e(1-p_e)/\sigma^4_{ch}. \tag{20}$$

Based on Assumption 1 it therefore follows that the equivalent noise variance experienced by the component **u"** is $\sigma^2_{u''} = \left(\sigma^2_{ch} + 4p_e(1-p_e)\right)/(1-2p_e)^2$. Observe that since $(1-2p_e)^2 < 1$, it holds that $\sigma^2_{\mathbf{u"}} \geq \sigma^2_{ch}$. Since $L^{\mathbf{u'}}$ is distributed according to $\mathcal{N}(2/\sigma^2_{ch}, 4/\sigma^2_{ch})$, (7) implies

$$E[L^{\mathbf{u}}] = 4(1-p_e)/\sigma^2_{ch}, \quad \sigma^2(L^u_i) = 8/\sigma^2_{ch} + 16p_e(1-p_e)/\sigma^4_{ch}.$$

Hence, ***u*** experiences a channel with equivalent noise variance

$$\sigma_u^2 = \frac{\sigma_{ch}^2 + 2p_e(1-p_e)}{2(1-p_e)^2} \geq \frac{\sigma_{ch}^2}{2}. \tag{21}$$

For $p_e = 0$, the result reduces to the expression in Lemma 2. For $p_e > 0$, it follows that the UEP gradation level between **u** and **v** is actually smaller than shown in Lemma 2, under Assumption 3. ■

3.3 Improved Reliability-Based Decoding

For a Plotkin-type construction of high depth, the first few codes to be decoded will experience a significantly degraded equivalent channel. The reliability values of their code bits are used in all subsequent decoding steps, and may be detrimental to the overall performance of the decoding scheme. It is therefore desirable to modify the decoding process in such a way as to include only very reliable estimates from previous decoding stages into the re-evaluation process of the log-likelihoods of subsequently decoded components. Hence, instead of always summing up the likelihoods of both halves of the vector |**u'**|**u"**|, one can choose to perform this step only for sufficiently reliable **v**-based estimates. This can be accomplished by using a threshold t in the following way. If $\left|\hat{L}_i^v\right| > t$, where $\hat{L}_i^v$ denotes the log-likelihoods of the $i-th$ bit of $\hat{\mathbf{v}}$, then $L_i^u = L_i^{u'} + L_i^{u''}$, otherwise $L_i^u = L_i^{u'}$. The threshold may also be taken to depend on σ_{ch}^2 itself, and be adapted according to the channel quality at hand. We will refer to the above described iterative decoding methods as the threshold multi-stage (TMS) and adaptive threshold multi-stage (ATMS) algorithms.

Analysis of the ATMS Decoding Algorithm: For the ATMS algorithm, the log-likelihood values of the bits of **u** are given by:

$$L_i^u = \begin{cases} L_i^{u'} + L_i^{u''}, & \text{for } \left|\hat{L}_i^v\right| > t(\sigma_{ch}) \\ L_i^{u'}, & \text{otherwise.} \end{cases}$$

This is equivalent to $L_i^u = L_i^{u'} + L_i^{new}$, where

$$L_i^{new} = \begin{cases} \mathcal{L}_i^{y''}, & \text{for } \hat{v}_i = 1,\ \left|\hat{L}_i^v\right| > t(\sigma_{ch}) \\ -\mathcal{L}_i^{y''}, & \text{for } \hat{v}_i = -1, \left|\hat{L}_i^v\right| > t(\sigma_{ch}), \\ 0, & \text{otherwise.} \end{cases}$$

The conditions $\hat{v}_i = 1$ and $\left|\hat{L}_i^v\right| > t(\sigma_{ch})$, are equivalent to the condition $\hat{L}_i^v > t(\sigma_{ch})$, and similarly, $\hat{v}_i = -1$ and $\left|\hat{L}_i^v\right| > t(\sigma_{ch})$, are equivalent to $\hat{L}_i^v < -t(\sigma_{ch})$. Therefore,

$$L_i^{new} = \begin{cases} \mathcal{L}_i^{y''}, & \text{for } \hat{L}_i^v > t(\sigma_{ch}) \\ -\mathcal{L}_i^{y''}, & \text{for } \hat{L}_i^v < -t(\sigma_{ch}), \\ 0, & \text{otherwise.} \end{cases}$$

Under Assumption 2, it follows that:

$$E[L_i^{new}] = E[\mathcal{L}_i^{y''} | \hat{L}_i^v > t(\sigma_{ch})] p\{\hat{L}_i^v > t(\sigma_{ch})\} + E[-\mathcal{L}_i^{y''} | \hat{L}_i^v < -t(\sigma_{ch})] p\{\hat{L}_i^v < -t(\sigma_{ch})\} = \frac{2}{\sigma_{ch}^2}(p_1 - p_2),$$

where $p_1 = Q\left(\frac{t(\sigma_{ch}) - m_{\hat{v}}}{\sigma_{\hat{v}}}\right)$, and $p_2 = 1 - Q\left(\frac{-t(\sigma_{ch}) - m_{\hat{v}}}{\sigma_{\hat{v}}}\right)$. Similarly, one has

$$E[(L_i^{new})^2] = (4/\sigma_{ch}^2 + 4/\sigma_{ch}^4)(p_1 + p_2),$$
$$\sigma^2(L_i^{new}) = 4/\sigma_{ch}^2(p_1 + p_2) + 4/\sigma_{ch}^4(p_1 + p_2 - (p_1 - p_2)^2).$$

When combined with the fact that $L^{\mathbf{u}'}$ is distributed according to $\mathcal{N}(2/\sigma_{ch}^2, 4/\sigma_{ch}^2)$, the last equation implies that under TMS decoding one has:

$$E[L^{\mathbf{u}}] = 2(1 + p_1 - p_2)/\sigma_{ch}^2,$$
$$\sigma^2(L^{\mathbf{u}}) = 4(1 + p_1 + p_2)/\sigma_{ch}^2 + 4(p_1 + p_2 - (p_1 - p_2)^2)/\sigma_{ch}^4.$$

A good choice for the threshold can now be obtained by maximizing the *equivalent* SNR experienced by $\mathbf{u}$ (SNR_u as shown below), with respect to $t(\sigma_{ch})$:

$$SNR_u = \frac{(1+p_1-p_2)^2}{\sigma_{ch}^2(1+p_1+p_2)+(p_1+p_2-(p_1-p_2)^2)}. \tag{22}$$

3.4 Properties of UEP-Plotkin Schemes

Before presenting the simulation results, we would like to briefly describe various advantages offered by the Plotkin-type UEP scheme.

The $k_1 + k_2$ information bits of a Plotkin-type codeword are divided in such a way that the first half of the codeword contains k_1 information bits, while the second half of the codeword contains k_2 information bits.

From the analysis provided in the previous sections, it can be seen that the first half of the codeword is better protected than the second half of the codeword. Consequently, one may expect that the first k_1 information bits have better error protection than the remaining k_2 information bits. Hence, a gradation of UEP levels also exists with respect to the information bits. This gradation can be further improved by an adequate choice of the generator matrices of the component codes [1]. It was shown in [1] that in many cases the *optimal encoding matrix* (for which information bits have the highest degree of protection) is systematic. Therefore, one can use systematic generator matrices for all the component codes, along with the aforementioned decoding algorithms, in order to improve the quality of error protection for the information bits. Finally, by using the Plotkin-type UEP scheme one can also obtain a *relatively large number of information bits* that have good error protection. Consequently, the average SNR required to achieve a given BER for the technique proposed in the paper is much smaller than the one needed for the UEP scheme proposed in [16].

4 Simulation Results

In this section, we present the UEP BER performance of Plotkin-type LDPC codes under MS, TMS, ATMS and MR-MS decoding. The component-codes used in this comparative study include random-like codes listed in [9], and two classes of structured codes. For all schemes investigated, half of the component-codes are chosen to be random-like, and half of the component-codes are chosen to have a mathematical structure. The first class of structured LDPC codes is based on *Sidon sets*, described by the authors in [6], while the second class is a generalization of array codes [11]. Both families of structured LDPC codes are quasi-cyclic, with parity-check matrices composed of powers of a basic, circulant permutation matrix **P**, as shown below.

$$H_S = \begin{bmatrix} P^{i_1} & P^{i_2} & P^{i_3} & \dots & P^{i_s} \\ P^{i_s} & P^{i_1} & P^{i_2} & \dots & P^{i_{s-1}} \\ P^{i_{s-1}} & P^{i_s} & P^{i_1} & \dots & P^{i_{s-2}} \end{bmatrix}, \quad H_P = \begin{bmatrix} P^{a_0 \cdot b_0} & P^{a_0 \cdot b_1} & \dots & P^{a_0 \cdot b_{f-1}} \\ P^{a_1 \cdot b_0} & P^{a_1 \cdot b_1} & \dots & P^{a_1 \cdot b_{f-1}} \\ \dots & \dots & \dots & \dots \\ P^{a_{r-1} \cdot b_0} & P^{a_{r-1} \cdot b_1} & \dots & P^{a_{r-1} \cdot b_{f-1}} \end{bmatrix}. \tag{23}$$

Code Construction 1 – H_S: A set $\mathcal{A} = \{a_1, ..., a_s\}$ is called a *Sidon set* of order g if all the sums

$$a_{i_1} + a_{i_2} + ... + a_{i_r}, 1 \leq i_1 < i_2 ... < i_r \leq s, \tag{24}$$

for $1 \leq r \leq g$, are distinct. One simple construction for Sidon sets $\mathcal{A}$ is based on the formula,

$$\mathcal{A} = \{0 \leq h \leq s^t - 1 | \beta^h + \beta \in GF(s)\}, \tag{25}$$

where β is a primitive element of $GF(s^t)$, for some prime s. Let $\{i_j\}_1^s$ denote the exponents of $\mathbf{P}$ in the parity-check matrix H_S. If $\{i_j\}$ is a subset of a Sidon set $\mathcal{A}$, where $order(P) = s^t - 1$, then the code describes by H_S has girth at least six. The simulation results shown in Figure 2-7, 9 are obtained for a length 3120 component-code based on the Sidon set $\{23, 72, 244, 313, 565\}$, for which $s = 5$ and $t = 4$. The other component-code is random-like, and of the same length, taken from [9].

Code Construction 2 – H_P: A generalized array code has a parity-check matrix H_P of the form shown in (23). Here, $a_0, a_1, \ldots, a_{r-1}$ and $b_0, b_1, \ldots, b_{f-1}$ are integer sequences avoiding solutions to specific sets of linear equations, as described in [11]. For $order(P) = q = 311$, a code graph specified by H_P, with $\{a_i\}_0^2 = \{0, 1, 3\}$ and $\{b_i\}_0^5 = \{0, 3, 7, 18, 31, 50\}$, has girth ten [11]. The length and dimension of such a code are 1866 and 933, respectively. A code graph with variable-degree four and girth eight can be obtained by choosing $q = 887$, with $\{a_i\}_0^3 = \{0, 1, 2, 3\}$ and $\{a_i\}_0^8 = \{0, 3, 5, 17, 30, 49, 102, 131, 226\}$. The length and dimension of the resulting code are 7983 and 4435, respectively. The simulation results involving these codes are shown in Figure 8.

The results presented in Figure 2-9 demonstrate several important properties of the MS, TMS, ATMS and MR-MS decoding algorithms. First, it can be observed that the BERs' vary to a great extent with the choice of the component-codes. This can be explained by the following arguments. When compared to the original channel, $\mathbf{v}$ experiences a relatively degraded channel. Hence, by choosing $\mathbf{v}$ to belong to a code with a powerful error-correcting capability, one can ensure less degradation in the overall code performance. Note also from Figure 2 that C^* has a better performance than C^{**} (the descriptions of these codes are given in the figure). This is due to the fact that the rate of the random-like code which is used for the $\mathbf{v}$ component of C^{**} is very high, and hence its error-correcting potential is quite limited. Figure 3 shows two degrees of UEP offered by a depth-one Plotkin-type construction, under MS, TMS ($t = 0.86$) and ATMS, $[SNR, t(\sigma_{ch})] = [(0, 1.1), (0.5, 1), (1, 1), (1.5, 0.95), (2, 0.9), (2.5, 0.9), (3, 0.9), (3.5, 0.9)]$, decoding, with 15 iterations of sum-product at each level. The BER improvement for codes at depth one is not very significant under TMS and ATMS, as compared to MS decoding. But as one increases m, a considerable improvement in BERs is observed for TMS and ATMS decoding. As shown in Figure 4, for $m = 2$, there exists a substan-

tial enhancement in the performance of TMS-decoded (with $t = 0.55$) and ATMS-decoded, $[SNR, t(\sigma_{ch})] = [(0, 1.83), (0.5, 1.47), (1, 1.25), (1.5, 1.25), (2, 1.1), (2.5, 0.97), (3, 0.86), (3.5, 0.86)]$, compared to MS-decoded Plotkin-type codes. The thresholds used are obtained from (22) and through several trial-and-error rounds of simulation. The performance of the best protected part of the code under TMS decoding shows a 1-dB gain, while ATMS shows a 1.5-dB gain in SNR, when compared to MS decoding. Figures 5, 6 and 7 show the performance of the above described codes under MR-MS decoding. Fixed thresholds were set to incorporate the reliability information. Furthermore, since the reliability of the decoded bits of $\mathbf{v}$ increases with the number of rounds of MR-MS decoding, the thresholds of the MR-MS decoder are taken to be a decreasing sequence (from one round to the next one). It is clear from the simulation results that as the number of rounds of MR-MS decoder increases from one to three, the performance of the component-codes experiences significant improvements. The simulation results indicate a performance gain of nearly 1-dB for the best protected part of the codeword, when 2-round MR-MS decoding is used instead of MS decoding. Also, it is observed by extensive simulations that the largest improvement in the performance of MR-MS decoding is achieved when switching from $j = 1$ to $j = 2$. The performance gain obtained for $j > 3$ is not significant when compared to the increase in decoding complexity. As a side remark, the apparent emergence of an error floor for several BER curves is due to the *small number* of iterations (15 in this case) performed. By increasing the number of iterations to 30, the error-floor vanishes. It is also worth pointing out that in the Plotkin UEP scheme, even the worst protected component codes show significant improvement with respect to uncoded signalling. This is illustrated in Figures 3 and 6.

Based on the simulation results described above, one can also check the validity of the approximation results described in Section 3. Lemma 2 and 3 indicate that for both the cases of $\sigma_{ch} \to 0$ and $\sigma_{ch} \to \infty$, one can expect a 3dB performance gain for the first half of the codeword under MS decoding, as compared to the performance of the component code associated with this part under standard sum-product decoding. From Figure 2, one can see an improvement of 0.8dB at $SNR = 3$dB ($\sigma_{ch} \approx 0.66$) for the performance of the first half of the code word under MS decoding, when compared to the performance of the component code under sum-product decoding. Since the two described BER curves are diverging, for very high SNRs one can expect to achieve a 3dB gain as predicted by analysis. But, for the case of $\sigma_{ch} \to \infty$ there is clearly no 3dB gain between these two curves. This validates the observation made in Section 3.2 which states that for low SNRs the assumption of no decoding error in certain decoding stages is not valid. Hence, one needs to use the improved approximation given in Theorem 1. For $SNR = 0$dB, applying Theorem 1 with $p_e = 0.18$ gives

a gain in performance of 0.0046dB, which is almost identical to the one observed by simulation. Also, by Lemma 3 there is no loss in the performance of the second part of the codeword for large SNRs, and this can also be observed from the simulation results.

Figure 8 gives an illustration of the influence of the choice of code structure and design on achievable levels of UEP. The most important fact to be noted for all these results is that *the best protected component-code* has a significantly better performance than the best known LDPC codes of comparable length, rate and decoding complexity.

The performance of one (a,b,x)-UEP code construction is shown in Figure 9. The component **x**, when decoded using the MS algorithm, experiences an equivalent channel with very large noise variance. Since all other components of the codeword are dependent on **x**, all three component-codewords have large error rates and no fine error-characteristic tuning is possible.

Figure 10 shows a comparison of the BER-performance of the scheme introduced in this paper and the LDPC-UEP technique described in [16]. The component codewords **u** and **v** used for the Plotkin-type construction are taken from $[271,144]$ and $[271,188]$ codes, respectively. The overall length and rate of the code are 542 and 0.58, respectively. The simulations are performed for a 3-round MS-MR algorithm with 10-iterations per component code. Another UEP code $\mathcal{C}_{cdf}$ was designed using Netto's cyclic difference families, as outlined in [16]. The parameters of the construction are: $v = 79$ points, block-size $c = 3$, and with $|C_1| = |C_2| = |C_3| = 3$, $|C_4| = |C_5| = |C_6| = |C_7| = 1$. The performance of $\mathcal{C}_{cdf}$ is shown for a total number of 60 iterations of sum-product decoding. The length of the code $\mathcal{C}_{cdf}$ is 553, while the rate is 0.57. The simulations show a significant improvement in the performance of the Plotkin-type codes as compared to the irregular codes of [16]. One should also note that, in the Plotkin-type of construction the number of bits that are best protected is 271 as compared to 237 for $\mathcal{C}_{cdf}$, while the number of least protected bits in the Plotkin-type construction is 271 as compared to 316 least protected bits in $\mathcal{C}_{cdf}$. The average SNR per codebit required to attain a given BER for a code of rate 0.57 offering four levels of protection in the Plotkin scheme is nearly identical to that required for a code of rate 0.43 obtained by the method in [16].

4.1 Irregular LDPC Codes and UEP

So far, several approaches to UEP coding based on the use of irregular LDPC codes were proposed, including the work described in [12] and [13]. It was tacitly assumed that UEP can be achieved by using non-uniform degree distributions of variable nodes. Clearly, in such a setting, nodes of large degree are assumed to be

better protected than nodes of small degree. We would like to point out that such findings are valid for the AWGN channel only under certain restrictions, including:

- A very small number of performed iterations of belief propagation;
- A two-level UEP scheme where only a small fraction of coded (or information) bits is guaranteed additional protection;
- A very large number of required levels of UEP, of which only the best protected few components tend to show significant improvements.

The reasons behind this phenomena can be deduced directly from the seminal paper on irregular code degree optimization [8, p. 586]. There, the well-known "wave effect" of LDPC iterative decoding is described. This effect is a consequence of the fact that variable nodes of high degree get corrected first, consequently aiding in the process of error-correction of variable nodes of slightly smaller degree, all up to the level of the nodes of smallest degree. After a sufficiently large number of iterations, the wave effect leads to all variable nodes exhibiting the same level of error protection. This is illustrated in Figure 11, where the results for five different Progressive Edge Growth (PEG) [9] coding schemes are plotted. In the first four plots (viewed from the right), the dashed curves correspond to the BER of a fraction of 50% variable nodes of largest degree, while the solid curves correspond to the BER of a fraction of 50% variable nodes of smallest degree. When read from the right, the codes have parameters $[600, 300]$, $[1000, 500]$, $[600, 300]$, and $[1000, 500]$, and they correspond to codes with non-optimized and optimized degree distributions for a given rate of $1/2$, respectively. The optimizing degree distributions were obtained from [14], with maximum degree 15, while the non-optimized degree distribution was chosen so as to have only two possible degrees appearing with the same frequency, namely 3 and 15. As can be seen from the figure, no UEP gradation can be detected for these codes after 50 iterations of iterative decoding. It is also worth pointing out that for the first code, the variable nodes of higher degree actually exhibit a slightly worse performance than the variable nodes of lower degree. On the other hand, if one repeats the same experiments for a very small number of iterations, say five, the findings are slightly different. For the degree-optimized length $n = 600$ irregular LDPC code, at 2.5 dB one observes a BER of $9.24 \cdot 10^{-3}$ and $1.82 \cdot 10^{-3}$ for the lower and higher degree nodes. Similarly, at 2.75 dB, one obtains the following BER values: $4.72 \cdot 10^{-3}$ and $1.55 \cdot 10^{-3}$. In both cases, the slight increase in UEP gradation comes at the cost of an overall performance loss of almost two orders of magnitude of the BER. Furthermore,

observe that for 15 iterations of iterative decoding, and at 2.5 dB the BER of the two components exhibit negligible performance variations of $1.24 \cdot 10^{-4}$ and $8.15 \cdot 10^{-5}$. The above described findings are consistent with the ideas behind the wave effect, since only five iterations do not suffice to achieve a significant "spread" of the wave.

The UEP properties of small fractions of variable nodes were observed in [12] and [13], but we would like to point out that for many applications of interest, one is actually concerned with providing protection levels for a large portion of variable nodes. Finally, the fifth set of curves in Figure 11 corresponds to a 4-level variable node partition in a degree-optimized irregular LDPC code of length $n = 1600$. As one can see, after 30 iterations of decoding only the variables of highest degree of protection show a slightly improved performance, and only for high SNR values. This finding can be attributed to the fact that the number of iterations required to achieve the wave effect is dependent on the code length.

5 Conclusions

We analyzed both analytically and through extensive computer simulations, the properties of a novel, recursive, Plotkin-type construction of UEP-LDPC codes. The proposed technique is structured around a specialized multi-stage, multi-round iterative decoding algorithm, which provides for a high degree of error protection differentiation. Furthermore, the proposed UEP-LDPC scheme was shown to offer the possibilities of significant trade-offs between code performance, storage and computational complexity. This makes it a good potential candidate for practical applications where UEP of data is sought.

Acknowledgment: The authors are grateful to Stefan Laendner for his help in generating the code performance simulation results. Furthermore, the authors express their gratitude to Dr. Truong for handling the manuscript and to the anonymous reviewers for their useful comments which largely improved the presentation of the results in the paper.

References

[1] H. Chen, "Optimal Encoding, Trellis Structure and Normalized Weight of Linear Block Codes," *PhD Thesis Dissertation*, University of Michigan, 1999.

[2] S. Chung, T. Richardson, and R. Urbanke, “Analysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using a Gaussian Approximation,” *IEEE Trans. on Inform. Theory*, Vol. 47, No. 2, pp. 657-670, Feb. 2001.

[3] I. Dumer and K.Shabunov, “Recursive decoding of Reed-Muller codes,” *Proceedings of IEEE International Symposium on Information Theory*, ISIT’2000, p. 63, June 2000.

[4] I. Dumer and R. Krichevskiy, “Soft-Decision Majority Decoding of Reed-Muller Codes,” *IEEE Trans. on Inform. Theory*, Vol. 46, No. 1, pp. 258-265, Jan. 2000.

[5] G. D. Forney, Jr., “Codes on Graphs: Normal Realizations,” *IEEE Trans. on Inform. Theory*, Vol. 47, No, 2, pp. 520-549, Feb. 2001.

[6] V. Kumar, O. Milenkovic, and B. Vasic, “Structured LDPC codes over $GF(2^m)$ and Companion Matrix Based Decoding,” *Proceedings of the International Symposium of Information Theory,* ISIT’2004, p. 271, June 2004.

[7] V. Kumar and O. Milenkovic, “On Unequal Error Protection LDPC Codes Based on Plotkin-type Constructions,” *Proceedings of the Global Telecommunications Conference*, Vol. 1, pp. 493-497, Dec. 2004.

[8] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman, “Improved Low-Density Parity-Check Codes Using Irregular Graphs,” *IEEE Trans. on Inform. Theory*, Vol. 47, No. 2, pp. 585-599, Feb. 2001.

[9] Web resource: http://www.inference.phy.cam.ac.uk/mackay/CodeFiles.html

[10] F.J. MacWilliams and N.J. Sloane, *The Theory of Error-Correcting Codes*, North Holland Publishing Company, 1977.

[11] O. Milenkovic, D. Leyba, D. Bennett, and N. Kashyap, “New Partition-Regular Sequences and Array Codes of Large Girth,” *Proceedings of the 42nd Annual Allerton Conference on Communication, Control and Computing*, pp. 240-249, Sept. 2004.

[12] H. Pishro-Nik, N. Rahnavard, and F. Fekri, “Nonuniform Error Protection Using Low-Density Parity-Check Codes,” *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing*, pp. 311-320, Oct. 2002.

[13] N. Rahnavard, and F. Fekri, " Unequal Error Protection Using Low-Density Parity-Check Codes," *Proceedings of the International Symposium of Information theory*, ISIT'2004, pp.449, Jun. 2004.

[14] Web resource: http://lthcwww.epfl.ch/research/ldpcopt/

[15] W.J. Van Gils, "Linear Unequal Error Protection Codes from Shorter Codes," *IEEE Trans. on Inform. Theory*, Vol. 30, No. 3, pp. 544-546, May 1984.

[16] B. Vasic, A. Cvetkovic, S. Sankaranarayanan, and M. Marcellin, "Adaptive Error Protection Low-Density Parity-Check Codes For Joint Source-Channel Coding Schemes," *Proceedings of the International Symposium on Information Theory*, ISIT'2003, p. 267, July 2003.

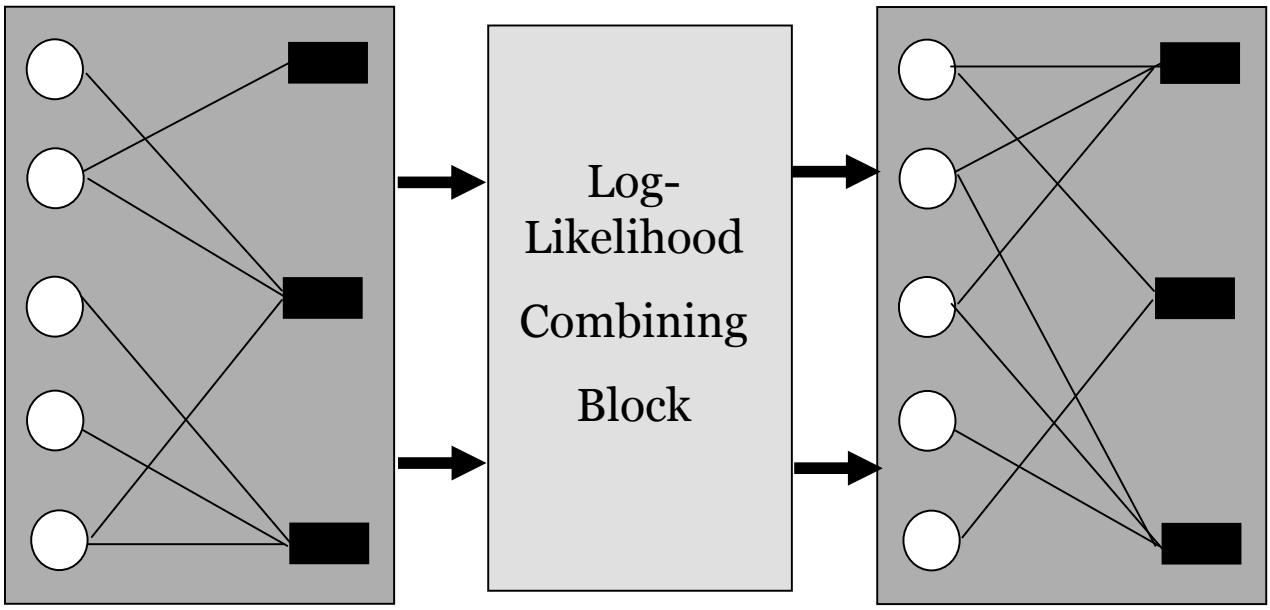

Figure 1: Graphical representation of MS decoding for a depth-one Plotkin-type construction.

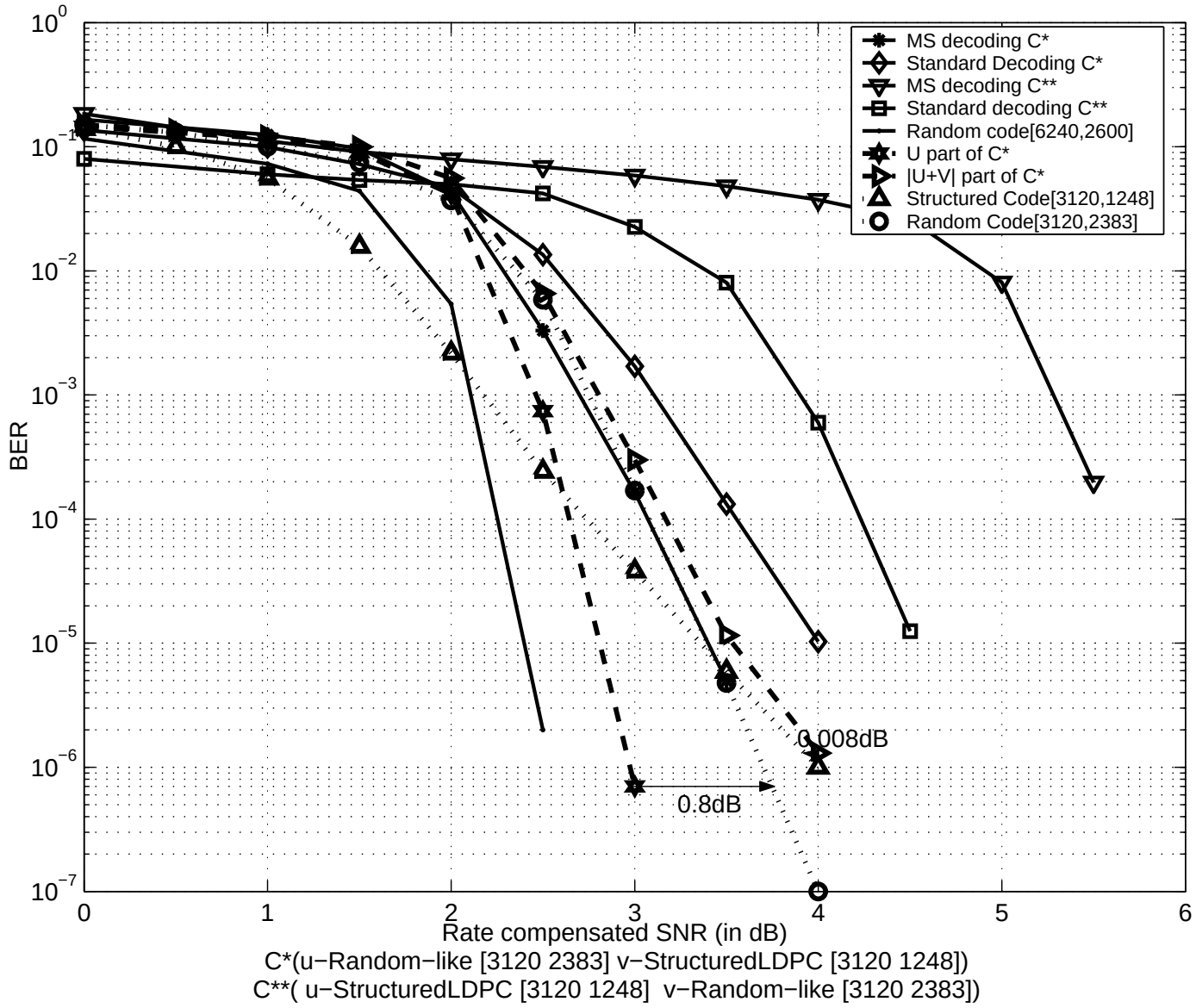

Figure 2: BER versus E_b/N_0 (dB) for different choices of the code components.

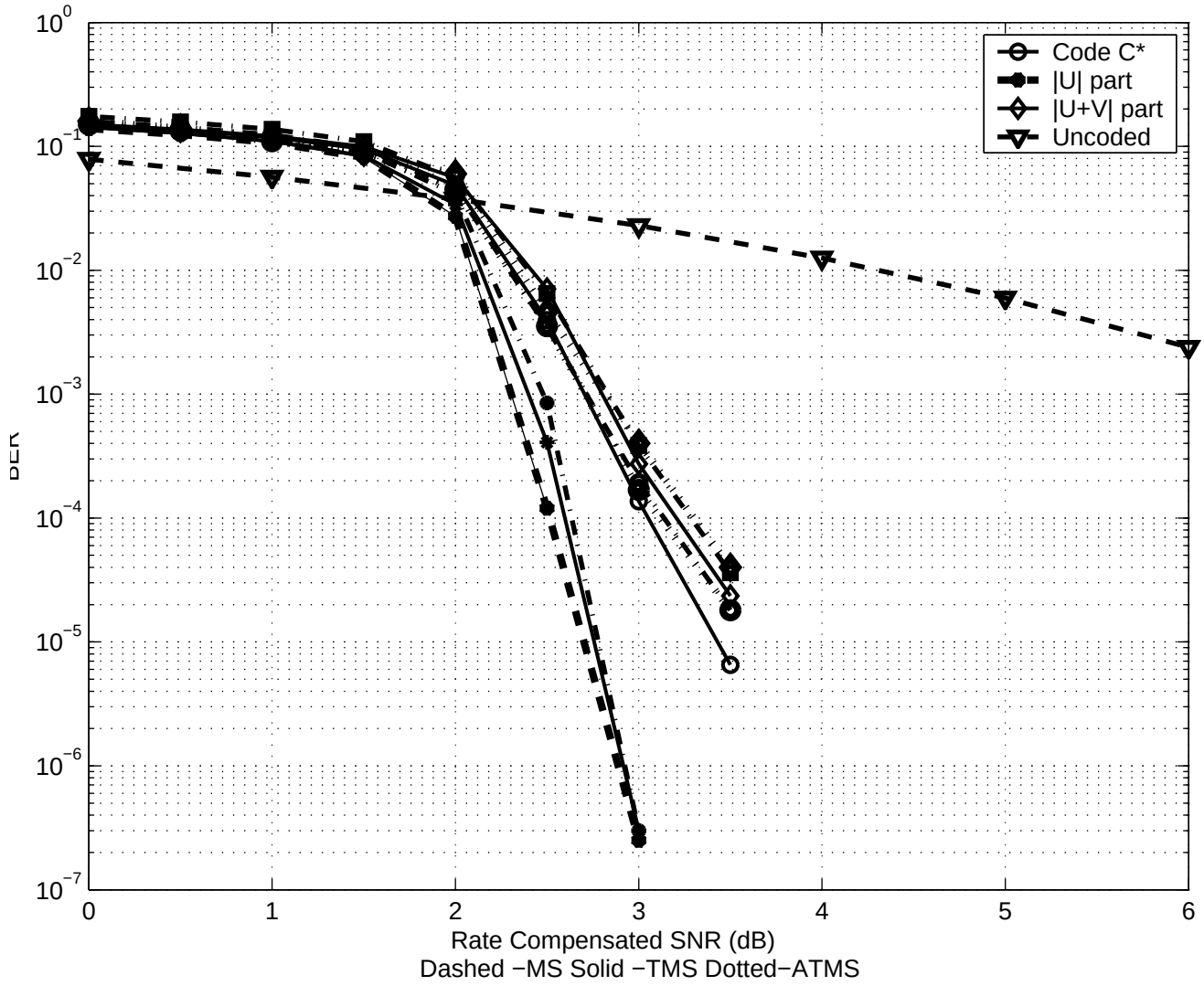

Figure 3: Performance of the $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ UEP scheme under MS, TMS, and ATMS decoding.

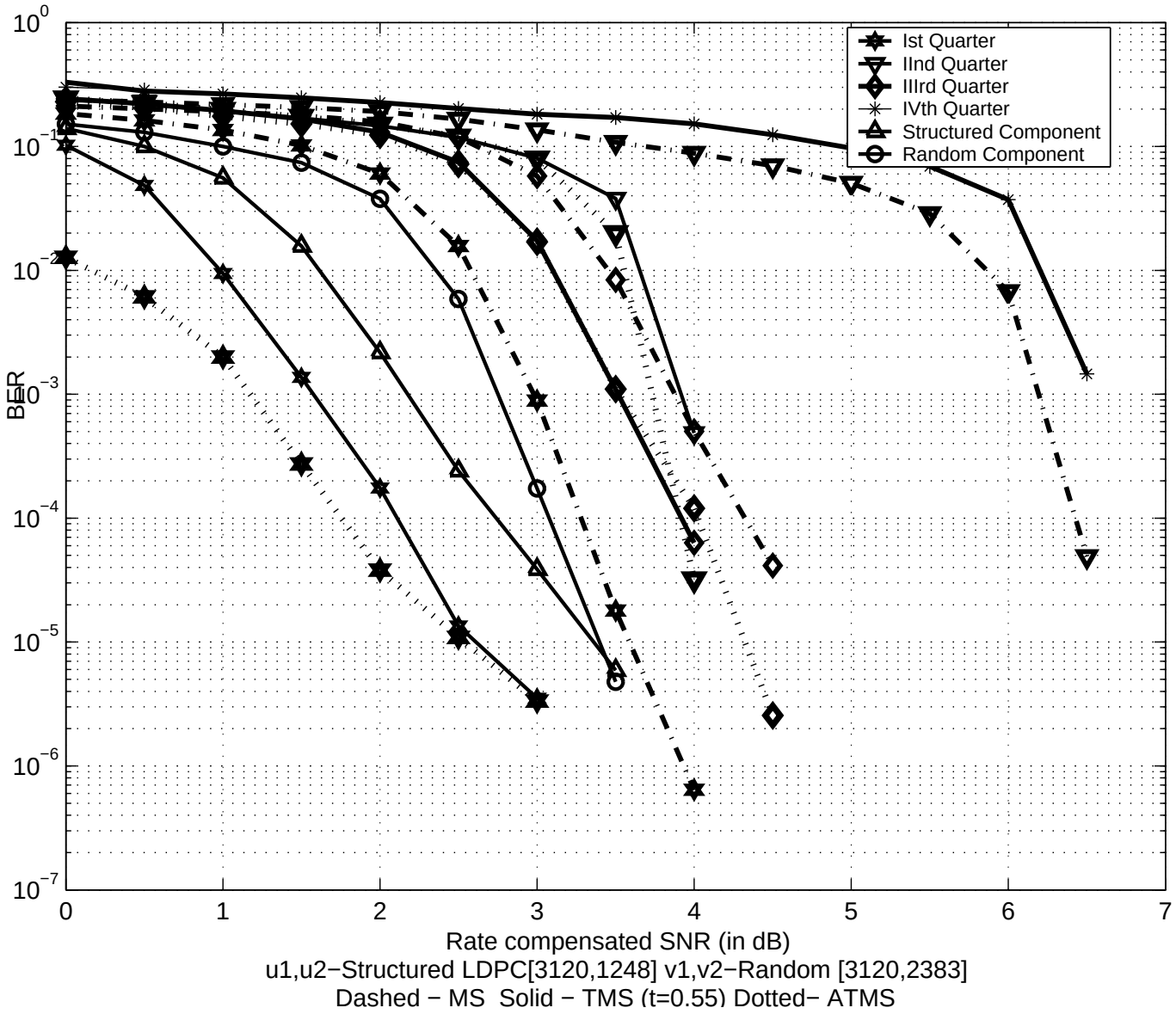

Figure 4: Performance of the 4-level Plotkin-type UEP scheme under MS, TMS, and ATMS decoding.

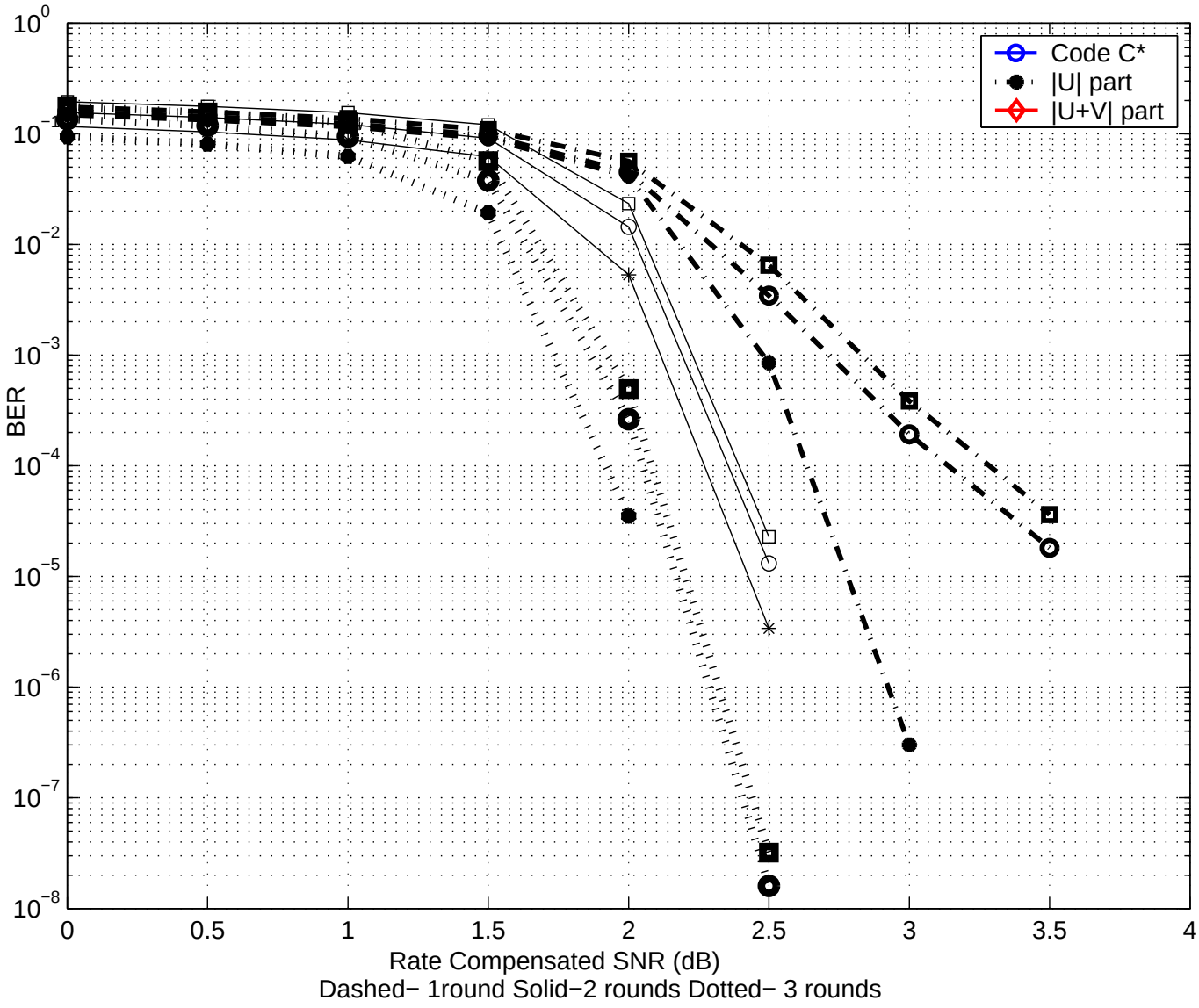

Figure 5: Performance of the $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ UEP scheme under MR-MS decoding.

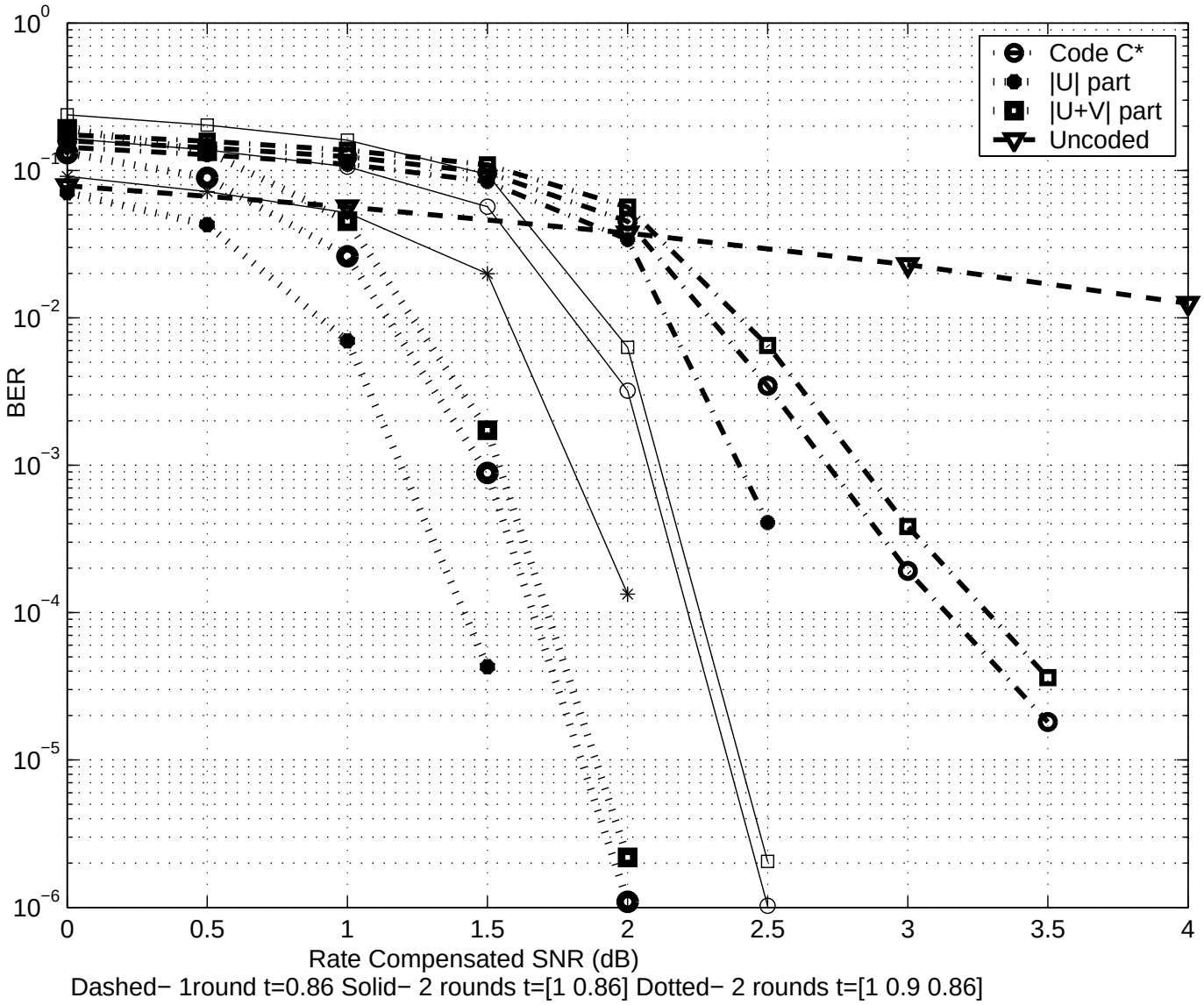

Figure 6: Performance of the $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ UEP scheme under MR-MS decoding with thresholding.

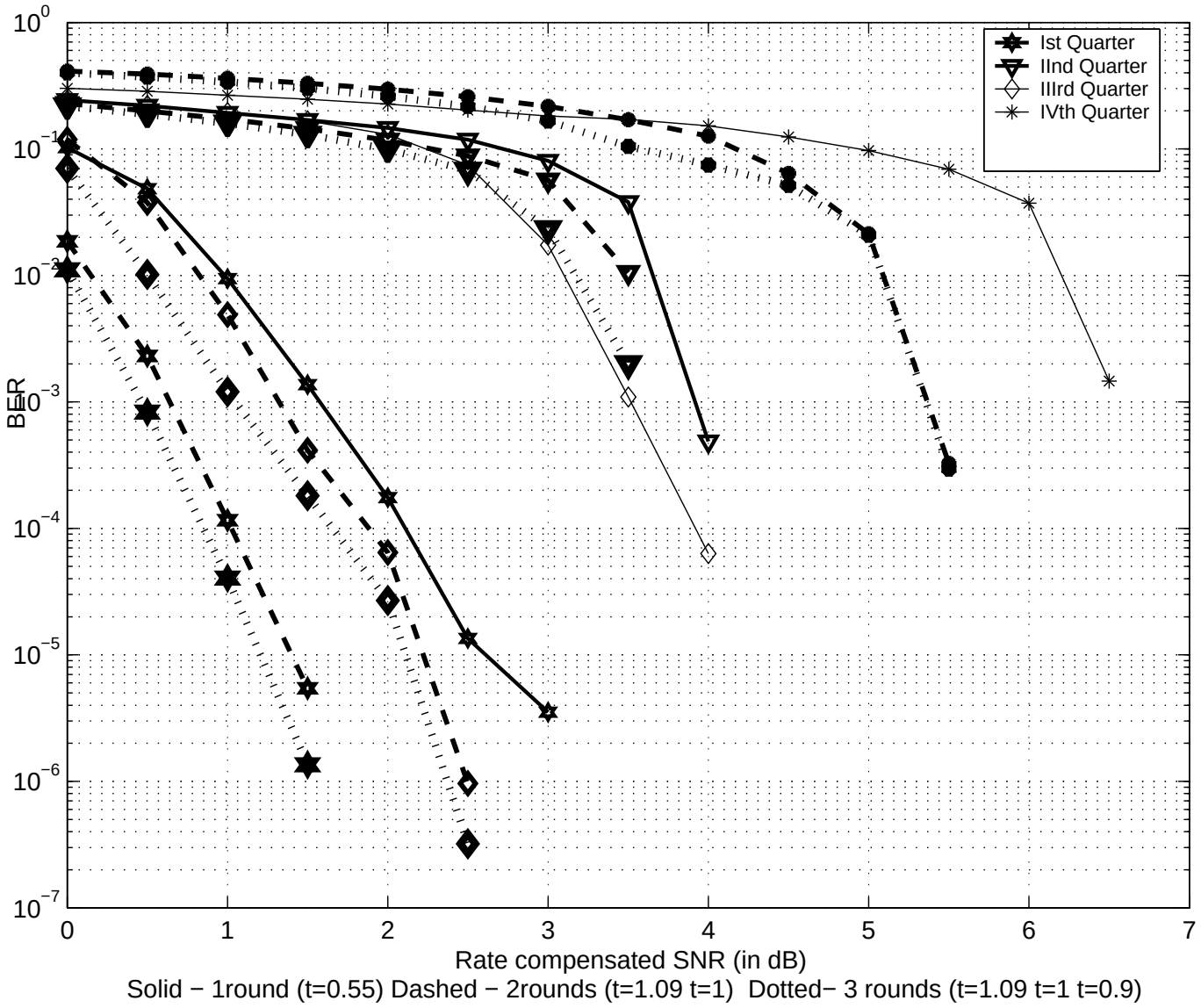

Figure 7: Performance of the $|\mathbf{u}_1|\mathbf{u}_1+\mathbf{v}_1|\mathbf{u}_1+\mathbf{u}_2|\mathbf{u}_1+\mathbf{v}_1+\mathbf{u}_2+\mathbf{v}_2|$ UEP scheme under MR-MS decoding with thresholding.

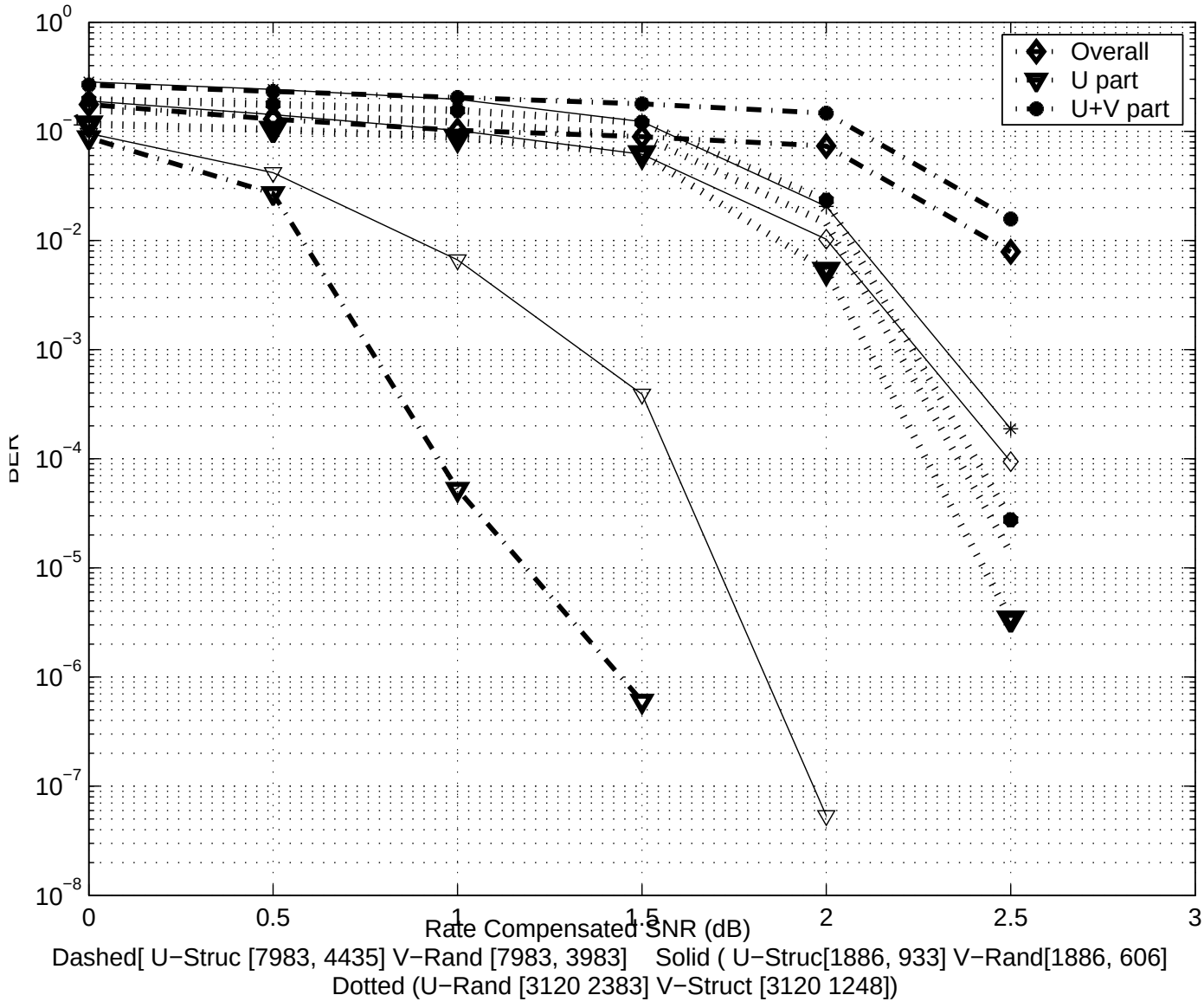

Figure 8: UEP of the $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$ scheme for different code components and with two-rounds of MR-MS decoding with parameters $t = 1$ and $t = 0.86$.

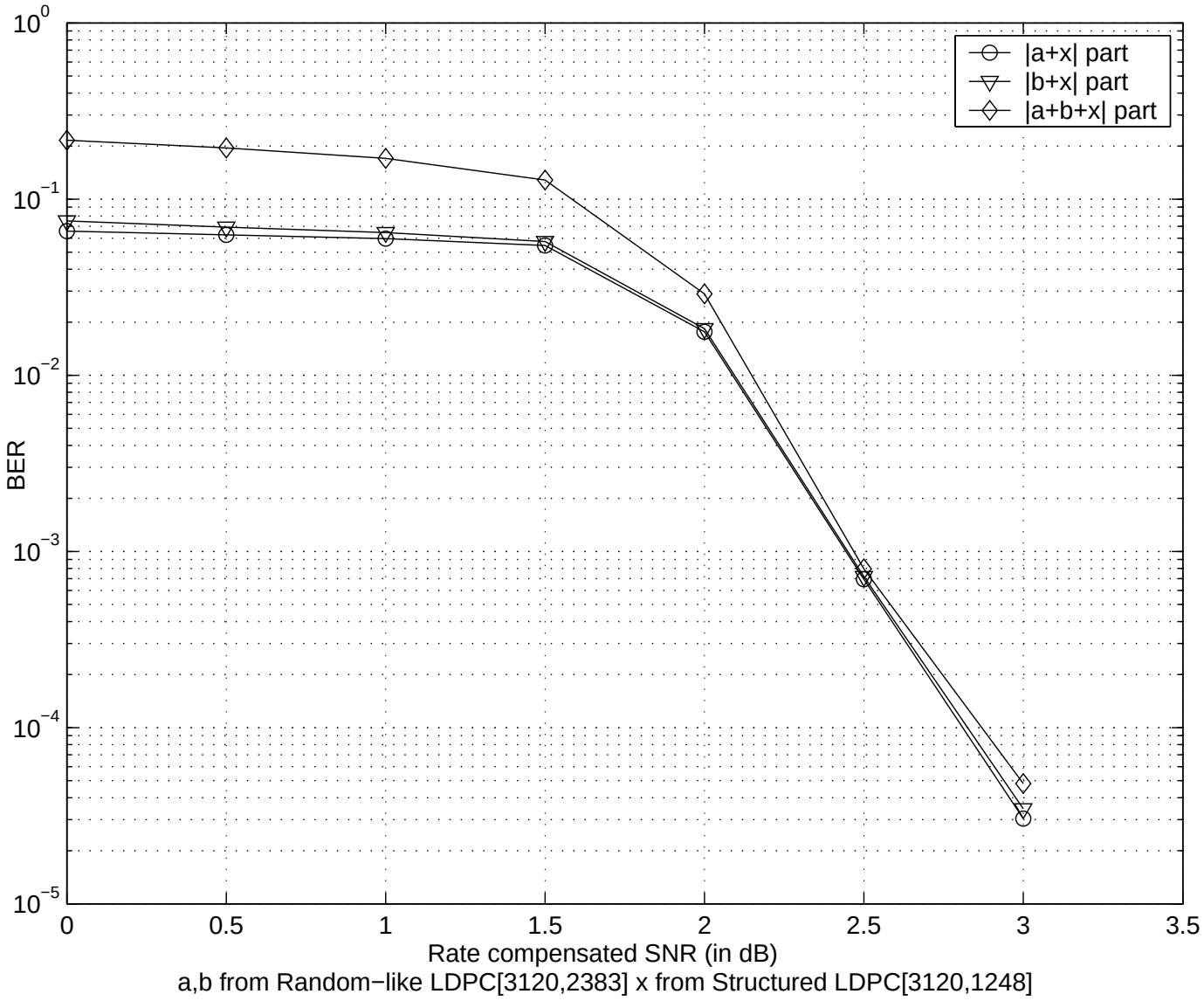

Figure 9: UEP of the $|\mathbf{a}+\mathbf{x}|\mathbf{b}+\mathbf{x}|\mathbf{a}+\mathbf{b}+\mathbf{x}|$ scheme under MS decoding.

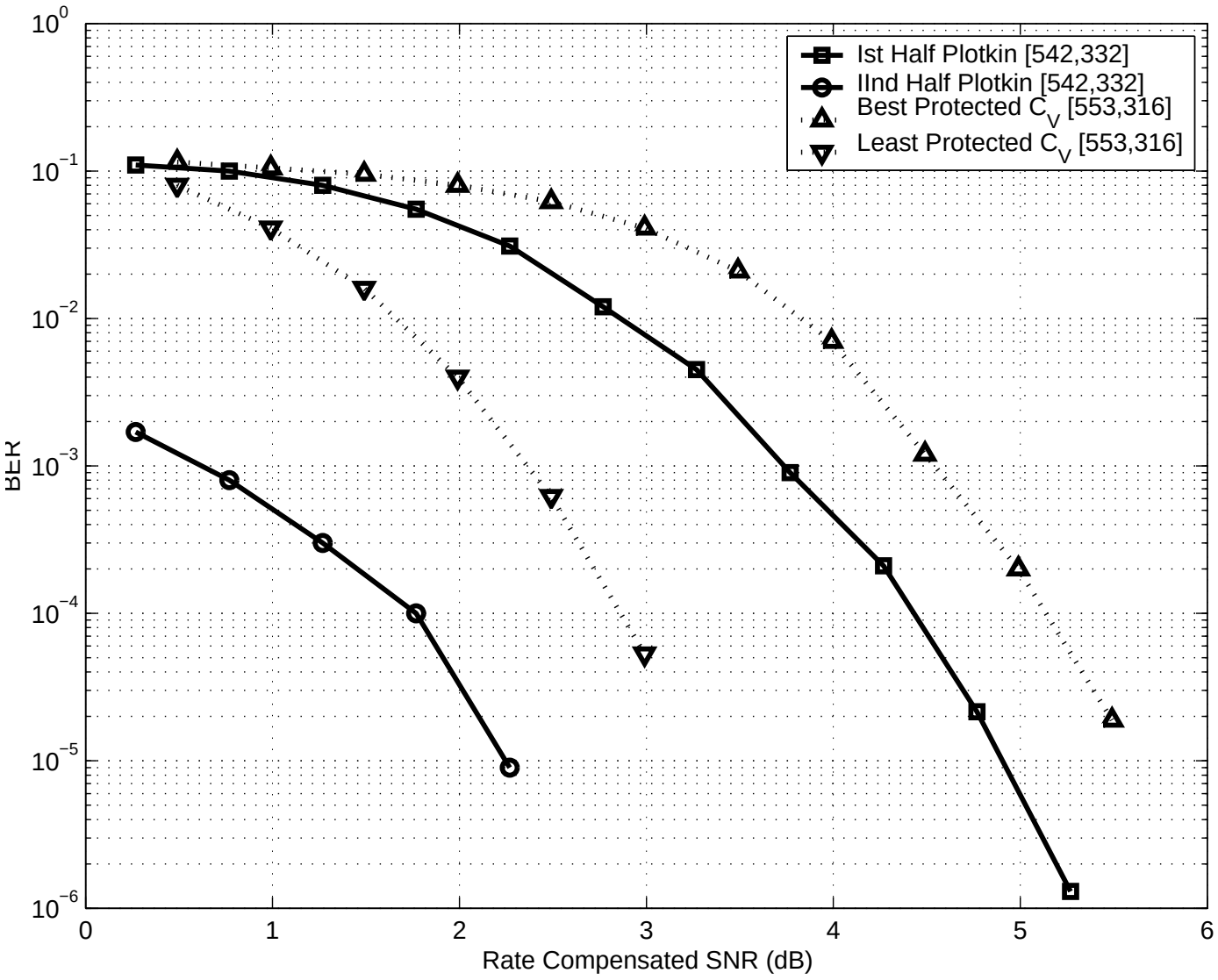

Figure 10: Comparison of the performance of an MR-MS decoded Plotkin-type code (with thresholding) and the performance of the irregular $\mathcal{C}_{cdf}$ code under standard belief propagation.

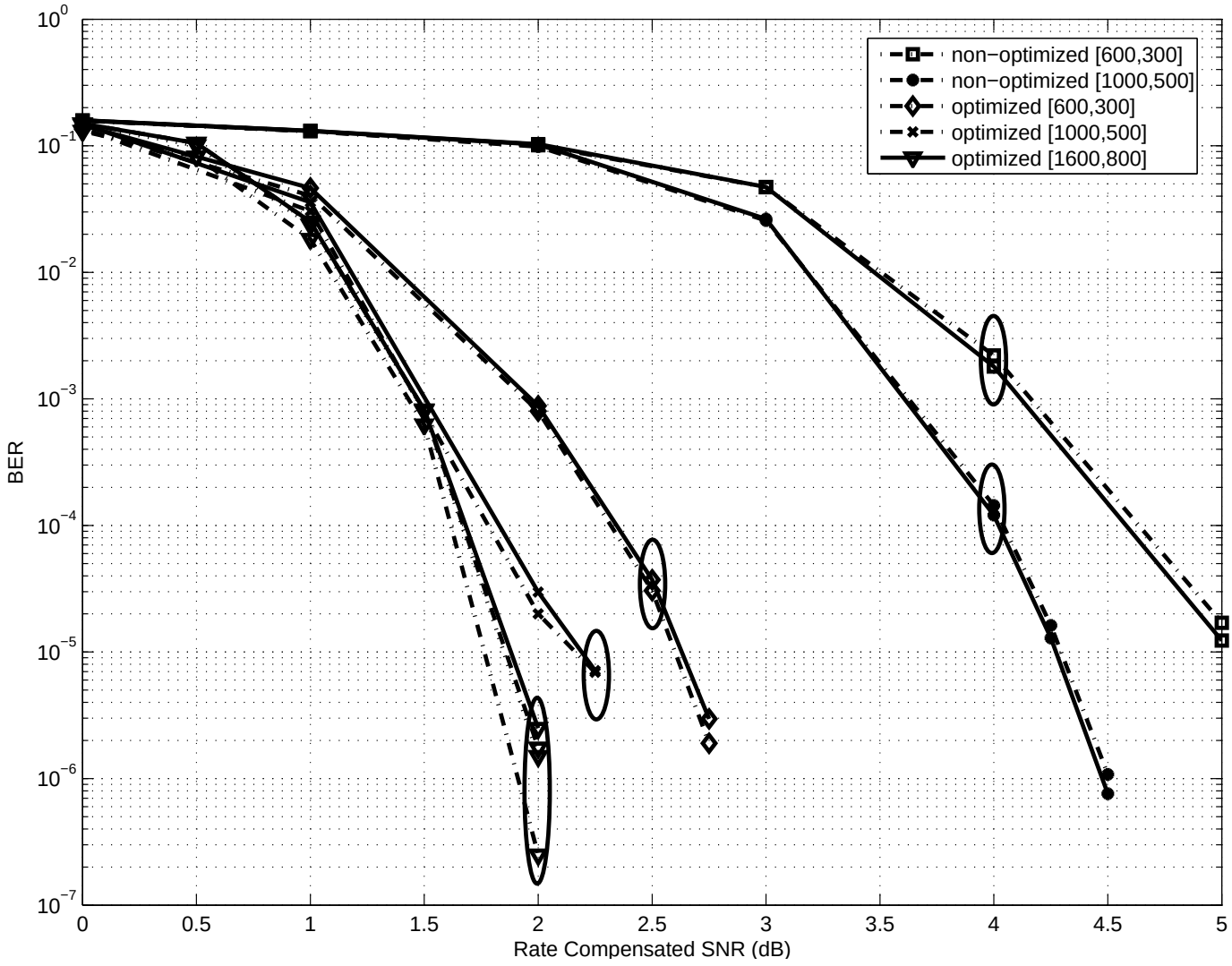

Figure 11: Protection levels in irregular codes with optimized and non-optimized degree distributions