Check-Irregular LDPC Codes for Unequal Error Protection under Iterative Decoding

Lucile SASSATELLI[1,2], Werner HENKEL[1], David DECLERCQ[2]

[1] International University Bremen, Campus Ring 1 28759 Bremen, Germany, w.henkel@iu-bremen.de
[2] ETIS ENSEA/UCP/CNRS UMR-8051, 6 avenue du Ponceau 95014 Cergy, France,
sassatelli,declercq@ensea.fr

Abstract

In many optimization techniques of LDPC codes, the check node irregularity is usually considered as being fixed because the profile must contain only two consecutive degrees in order to maximize the overall convergence speed of the code. We show in this paper that it is possible to construct bit-regular codes with more than two different degrees at check nodes side to speed up the convergence of some parts of the codeword, even if we may decrease a little the global convergence. Then, we take advantage of the check irregularity to create unequal error protection (UEP) codes, and present a very flexible method to achieve it. This method is based on pruning a mother code, thereby creating different subcodes with different UEP properties, but decoded with the same decoder.

1 Introduction

This paper deals with unequal error protection (UEP) LDPC Codes, achieved by irregularity on the check node profile, the bit node profile is set to be regular. A UEP coding scheme could be useful in the transmission of multi-media content (voice, fixed image, or video) whose characteristics have heterogeneous sensibility to errors. The code stream of source-encoded blocks is hierarchically structured and contains typically:

- headers to describe the type and parameters of compression,
- control data for code-stream synchronization, position, or indexing,
- compressed data delivered from the source coder, e.g., speech encoder coefficients, image texture, or motion vectors.

For such data streams, errors on headers or control data generally lead to a source-decoder failure since true reconstruction parameters of the compression are missing. In contrast, errors on high frequency coefficients of a DCT may be much less disturbing. Consequently, uniform protection for such a code stream would be sub-optimal. This highlights the interest of realizing unequal error protection by modifying the structure of a code. Well-known coding methods adapted to heterogeneous sensitivity of data often focus on average performance over the whole codeword. UEP could be achieved by puncturing or pruning (see [1]) convolutional codes to adapt the code rate without changing the decoder. In our work, we make use of LDPC codes [2] as a UEP coding scheme, because more and more standards consider these codes thanks to their very good performance. We concentrated on heterogeneously protecting some bits inside a codeword by suitable selection of irregularities of the code. The linear problem of hierarchical optimization of variable node profile, assuming the check node profile to be fixed, has been studied in [3]. We chose to focus on the dual case of bit-regular check-irregular LDPC codes. To do so, we will start by presenting asymptotical study of new codes ensemble minutely represented, i.e., a generalized density evolution [4], and we will show in particular that a carefully chosen cost function could lead to families of UEP LDPC codes which are bit-regular. We discuss the optimization procedure and the advantage of our approach in Section 3. Moreover, in real systems, flexibility in terms of types of data that can be protected is important, i.e., the parameters of the protection should be adaptable dynamically with minimum changes in the encoder and/or the decoder. In order to achieve this flexibility of our UEP codes, we propose a practical and flexible method based on the pruning of a mother code. This procedure is presented in Section 4. Finally the results are presented and a conclusion given.

This work is part of the FP6 / IST project M-Pipe and is co-funded by the European Commission.

2 Detailed Representation of Irregular LDPC Codes

A very useful parameterization for our work is the *detailed representation* of irregular LDPC codes presented by Kasai et al. in [4]. They constructed new families of LDPC codes which are sub-ensembles of conventional irregular LDPC code ensembles. The detailed representation they adopted allows to design optimal codes more accurately by restricting the

possible choices for the interleaver.

2.1 The Parameterization

These new codes ensembles are introduced by specifying the fraction of edges connecting sets of variable nodes and check nodes with given degree at the same time. Let B and D be two sets or irregularity degrees of variable and check nodes, respectively. A function $\pi : B \times D \rightarrow [0,1]$ is said to be the joint degree distribution of (B,D) if $\sum_{b\in B}\sum_{d\in D}\pi(b,d) = 1$. This function describes the connections between the different degrees of the code, and is called the detailed representation of the code.
They also defined a marginal degree distributions of variable and check blocks with respect to π, which are the usual polynomials for conventional representation:

$$\hat{\lambda}(x) = \sum_{b\in B}\hat{\lambda}_b x^{b-1} \quad , \quad \hat{\rho}(x) = \sum_{d\in D}\hat{\rho}_d x^{d-1}$$

with

$$\hat{\lambda_b} = \sum_{d\in D}\pi(b,d) \quad , \quad \hat{\rho_d} = \sum_{b\in B}\pi(b,d)\ .$$

For $\pi(b,d)$, we define two fractions

$$\lambda(b,d) = \frac{\pi(b,d)}{\hat{\rho_d}} \quad , \quad \rho(b,d) = \frac{\pi(b,d)}{\hat{\lambda_b}}\ .$$

It can be verified that $\rho(b,d)$ equals the fraction of edges connecting nodes of degree b and d among all edges of degree b. This detailed representation can be used, for example, to describe the methods for different Poisson constructions explored by MacKay et al. in [5].

2.2 Density Evolution with Detailed Representation

In [4], the authors also present a generalized density evolution for the newly proposed code ensembles. This generalized density evolution can treat density evolution for conventional code ensembles as a special case. From Theorem 3 in [4], we can derive a detailed evolution of the mutual information of messages on the edges under Gaussian approximation. The description is "detailed" in the sense that we distinguish the mutual information of messages coming from bit nodes or checkodes from different degrees. Let $s = 2/\sigma^2$ where σ^2 is the noise variance of the additive white Gaussian noise (AWGN) channel. The function J is defined by

$$J(m) = 1 - \mathbb{E}_x(\log_2(1+e^{-x})), \quad x \sim N(m,2m)\ .$$

Let $x_{cv}^{(l)}(d)$ and $x_{vc}^{(l)}(b)$ be the mutual information between the input of the channel and the messages from check nodes of degree d to any bit node at the lth iteration, and from bit nodes of degree b to any check node, respectively.

$$x_{cv}^{(l)}(d) = 1 - J\left((d-1)J^{-1}\left(1-\sum_{b\in B}\lambda(b,d)x_{vc}^{(l)}(b)\right)\right) \quad (1)$$

$$x_{vc}^{(l)}(b) = J\left(s+(b-1)J^{-1}\left(\sum_{d\in D}\rho(b,d)x_{cv}^{(l-1)}(d)\right)\right) \quad (2)$$

From Equation (1), we observe that the smaller d is, the greater is the mutual information of messages coming out of check nodes of degree d, i.e., the faster is the local convergence. In contrast, we see on Equation (2) that the mutual information of messages coming out of bit nodes of degree b is larger when b is larger. This is what we are going to exploit to optimize the local convergence speeds of UEP LDPC codes.

3 Achieving UEP with Check-Irregular LDPC Codes

3.1 A Cost Function for UEP

Let us define a sensitivity class by a set of information bits in the codeword that will have the same protection, i.e., approximately the same error probability at a given number of iterations. In practice, the sensitivity classes are defined by the source encoder. *B* and *D* can either be the sets of the degrees over the whole graph, and then Equation (1) describes the usual Gaussian approximation of density evolution, or the sets of the degrees inside the kth sensitivity class called C_k. A check node will belong to a class C_k if it is linked to at least one bit node of this class. Consequently, a check node can belong to several sensitivity classes. The average mutual information of messages coming out of the check nodes of class C_k to the bit nodes of this class can be expressed as

$$x_{cv}^{(l)^{(C_k)}} = \sum_{b\in C_k}\lambda_b^{(C_k)}\sum_{d\in C_k}\rho^{(C_k)}(b,d)x_{cv}^{(l)}(d) \quad (3)$$

with $\rho^{(C_k)}(b,d) = \frac{\pi(b,d)}{\lambda_b^{(C_k)}}$ and $\lambda_b^{(C_k)} = \sum_{d\in C_k}\pi(b,d)$, then $\sum_{d\in C_k}\rho^{(C_k)}(b,d) = 1$. We denote by $x_{cv}^{(l-1)} = \sum_{d\in\text{graph}}\rho_d x_{cv}^{(l-1)}(d)$ the average over the whole graph at the $(l-1)$th iteration of the mutual information of messages from check nodes to bit nodes, whereas $x_{cv}^{(l)^{(C_k)}}$ (Equation (3)) is the average mutual information only on the part of the codeword corresponding to C_k, i.e., on edges that belong to C_k. We can express the UEP criterion we have chosen by

$$x_{cv}^{(l)^{(C_k)}} - x_{cv}^{(l-1)}\ . \quad (4)$$

If the difference (4) is positive and large, it means that the quality of messages arriving to the bit nodes of C_k is much better than the average quality of the messages in the whole graph. Note that this difference (4) is necessarily negative for the least protected classes. In

our particular case of regularity over bit nodes ($\lambda(x) = x^4$), this criterion can be deduced from Equation (3)

$$x_{cv}^{(l)^{(C_k)}} - x_{cv}^{(l-1)} = \tag{5}$$

$$1 - \sum_{d \in C_k} \rho^{(C_k)}(d) J((d-1)J^{-1}(1 - x_{vc}^{(l)})) - x_{cv}^{(l-1)}$$

Equation (5) can be lower bounded by

$$1 - J\left((\sum_{d \in C_k} \rho^{(C_k)}(d)d - 1) J^{-1}(1 - x_{vc}^{(l)}) \right) - x_{cv}^{(l-1)}$$

$$\leq x_{cv}^{(l)^{(C_k)}} - x_{cv}^{(l-1)} \tag{6}$$

We observe that the lower bound depends on the average check connection degree of the class C_k:

$$\overline{\rho}^{(C_k)} = \sum_{d=d_{min}^{(C_k)}}^{d_{max}^{(C_k)}} \rho^{(C_k)}(d)d$$

To maximize the difference (4), we have to minimize $\overline{\rho}^{(C_k)}$. The most protected classes will have the lowest average check degrees. $\overline{\rho}^{(C_k)}$ will be our cost function, that depends also on the minimum connectivity degree $d_{min}^{(C_k)}$ of check nodes of the class. Our algorithm presented in Section 4 is directly derived from the bound given in Equation (6). We sequentially decrease $d_{min}^{(C_k)}$ to be able to decrease $\overline{\rho}^{(C_k)}$, and minimize $\overline{\rho}^{(C_k)}$ at each step.

3.2 Discussion on Breaking the Concentration of Check Degrees

Chung et al. have shown in [6] that a concentrated $\rho(x)$ check degree $\rho(x) = \rho x^d + (1-\rho)x^{d-1}$ maximizes the speed of convergence of the overall code. This also means that concentrated check degrees minimize the convergence threshold of the code, expressed in E_b/N_0 for an AWGN channel. To achieve UEP properties by irregularity in the check profile, we have to increase the range of the check degrees, and then break the concentration. Since not obeying concentration increases the gap to capacity of the overall code, we must define a tolerance on this gap before starting the optimization process. The global UEP code will converge slower, but its most protected classes faster than the ones of the concentrated code. The first solution to limit the degradation of the overall convergence threshold is to limit the range of the check irregularity around $\overline{\rho}^{(C_k)}$ in the optimization. Or we could check after the optimization process wether the non-concentrated code has a threshold that is not too far from the optimum (concentrated code) threshold. We have also verified by simulations that for short block lengths, the UEP designed codes have similar global performance as the concentrated code.

4 A Practical Means to Achieve UEP: Pruning a Mother Code

The problem in optimizing the check irregularity using Equation (6) lies in the fact that the optimization is non-linear due to the dependence between the irregularities of the different sensitivity classes. In [3], the authors have proposed a hierarchical procedure to overcome this problem. In this section, we present a sequential procedure based on "code pruning" which has both the advantage of providing UEP codes based on Equation (6) and also leads to a practical and flexible scheme for various UEP configurations.

4.1 General Presentation of Pruning

To achieve irregularity on check nodes, we chose to prune a regular mother code in order to build a subcode with UEP behavior. Pruning is a well-known method for convolutional codes [1], but not so much for LDPC codes, for which it has been applied to reduce the influence of stopping sets [7]. Pruning away some bits of the codeword means to consider them deterministic, i.e., fixing the pruned bits, e.g., to zero. Consequently, we do not transmit these bits that disappear from the graph of the code since their messages are equal to infinity. Besides, since the edges connected to the pruned bits disappear, the girth (minimum cycle length) of the subcode can only be increased. Thus, the columns of the parity matrix that correspond to these bits are removed. Let $\mathbf{H_m}$ and $\mathbf{G_m}$ denote the parity-check and generator matrices, respectively, of the mother code of dimension K_0 and length N_0. To construct a subcode of dimension K_1, we prune away $K_0 - K_1$ columns of $\mathbf{H_m}$, and obtain the parity-check matrix $\mathbf{H_s}$ of the subcode. The subcode will have a length of $N_1 = N_0 - (K_0 - K_1)$. The next section deals with our sequential pruning procedure.

4.2 The Sequential Pruning Procedure

Due to the chosen coding scheme, K_1 and the mother code are fixed at the beginning of the optimization, therefore the code rate is fixed to $R_1 = \frac{K_1}{N_0 - K_0 + K_1}$. The N_c sensitivity classes to be optimized are defined by the proportions $\alpha(k)$ for $k \leq N_c - 1$. The number of info bits in the class C_k is $\alpha(k) \cdot R_1 \cdot N_1$ if $k = 1, ..., N_c - 1$, and $\sum_{k=1}^{N_c-1} \alpha(k) = 1$. The last class (the N_cth class) contains the redundancy. The amount of redundancy is the same in the mother code and in the subcode, and equals $(1 - R_1) \cdot N_1 = (1 - R_0) \cdot N_0$. The optimization focuses on the two important quantities in the bound (6) : $\overline{\rho}^{(C_k)}$ and $d_{min}^{(C_k)}$, and is composed of two main stages. For a given class C_k:

- We choose the $(\alpha_k K_1)$ most protected bit nodes.
- For a given $d_{min}^{(C_k)}$, we try to put a maximum number of degrees of the check nodes linked to these bit nodes near to $d_{min}^{(C_k)}$ in order to decrease $\overline{\rho}^{C_k}$.

- The following constraints need to be fulfilled.
  - any pruned bit must not be linked with a check node of degree lower or equal to the concentration constraint
  - avoid unvoluntary pruning (a column of $\mathbf{H}$ can become independent from all the others and then does not define a code anymore)
  - proportion constraint
  - code rate constraint
  - convergence constraint
  - stability constraint

  If these previous constraints are fulfilled:
- We decrease $d_{min}^{(C_k)}$ by one if the tolerance that we fixed regarding the concentration is not yet reached, and start over again.

4.3 General Preprocess

In this section, our goal is to find the generator matrix $\mathbf{G_s}$ of the subcode obtained by pruning the mother code, given $\mathbf{H_s}$ which is $\mathbf{H_m}$ without the pruned columns. Consequently, the generator matrix $\mathbf{G_s}$ must fix the pruned bits to zero. A great advantage of the pruning method is that it enables us to use the mother decoder to decode any subcode created by pruning. However, we assume that the encoder of the subcode can be different from the encoder of the mother code. We observe that for any matrix $\mathbf{P}$ of size $K_1 \times K_0$, we still have $\mathbf{H_m}{\mathbf{G_m}}^T\mathbf{P}^T = \mathbf{0}_{(N_0-K_0)\times K_1}$. Thus, we look for a matrix $\mathbf{P}$ that is be used at the transmitter side as $\mathbf{G_s} = \mathbf{P}\mathbf{G_m}$ and such that forall binary vector $\mathbf{i}$ of size K_1, $\mathbf{G_s}\mathbf{i} = \mathbf{c}$ with elements of $\mathbf{c}$ corresponding to the pruned bits must be equal to zero. Consequently, the matrix $\mathbf{P}$ will be called preprocessing matrix, since it can be considered as a kind of precode (see Fig.(1)). But we now explicit all the conditions that must be fulfilled by $\mathbf{P}$, and how this matrix can be computed. In

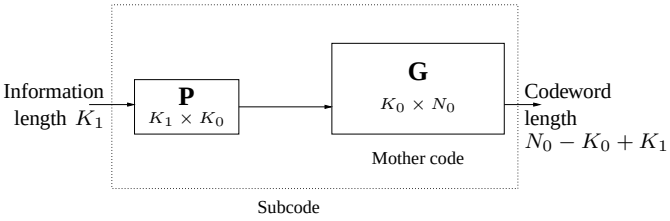

Fig. 1. Scheme of the subcode encoder

the following, $\mathbf{H_{mprun}}$ will denote $\mathbf{H_m}$ whose pruned columns are replaced by zero columns. Then we have

$$\mathbf{H_{mprun}}.{\mathbf{G_m}}^T.\mathbf{P}^T = \mathbf{0}_{(N_0-K_0)\times K_1} \quad (7)$$

We can reformulate $\mathbf{H_{mprun}}$ using a permutation matrix Π_1 on its columns:

$$\mathbf{H_{mprun}} = \left[\mathbf{H_1}|\mathbf{0}_{(N_0-K_0)\times(K_0-K_1)}\right]\Pi_1$$

In order to reach the target code rate $R_1 = \frac{K_1}{N_0-K_0+K_1}$ and to have $\mathbf{P}$ of full rank, $\mathbf{H_1}$ must be of full rank. This implies to add a constraint at the end of the optimization. We then assume that $\mathbf{H_1}$ is of full rank in the following, which ensures that there exist a matrix $\mathbf{Q}$ and a permutation Π_2 such that:

$$\left[\mathbf{H_1}|\mathbf{0}_{(N_0-K_0)\times(K_0-K_1)}\right] = \quad (8)$$
$$\mathbf{Q}\left[\mathbf{I}_{(N_0-K_0)}|\mathbf{R}_{(N_0-K_0)\times K_1}|\mathbf{0}_{(N_0-K_0)\times(K_0-K_1)}\right]\Pi_2$$

where $\mathbf{I}_{(N_0-K_0)}$ denotes the identity matrix of size N_0-K_0 and $\mathbf{R}_{(N_0-K_0)\times K_1}$ any matrix of size $(N_0-K_0)\times K_1$. Using Equation (7), we have

$$\mathbf{Q}\cdot\left[\mathbf{I}_{(N_0-K_0)}|\mathbf{R}_{(N_0-K_0)\times K_1}|\mathbf{0}_{(N_0-K_0)\times(K_0-K_1)}\right]\cdot \quad (9)$$
$$\cdot\Pi_2\cdot\Pi_1\cdot{\mathbf{G_m}}^T\cdot\mathbf{P}^T$$
$$= \mathbf{0}_{(N_0-K_0)\times K_1}$$

Let $\mathbf{A} = \Pi_2\Pi_1{\mathbf{G_m}}^T$ and $\mathbf{i}$ be the information word of length K_1 of the subcode. The information part of $\mathbf{A}\mathbf{P}^T\mathbf{i}^T$ should be associated to $\mathbf{R}_{(N_0-K_0)\times K_1}$ while the part of $\mathbf{A}\mathbf{P}^T\mathbf{i}^T$ corresponding to pruned columns should be associated to all-zero columns. $\mathbf{P}$ is therefore computed such that

$$\mathbf{A}(\mathbf{N_0}-\mathbf{K_0}+\mathbf{1}:\mathbf{N_0},:)\mathbf{P}^T = \begin{bmatrix}\mathbf{I}_{K_1}\\ \mathbf{0}_{(K_0-K_1)\times K_1}\end{bmatrix} \quad (10)$$

We can show that if $\mathbf{H_1}$ is full rank then $\mathbf{A}(\mathbf{N_0}-\mathbf{K_0}+\mathbf{1}:\mathbf{N_0},:)$ is invertible. With this procedure, we obtain a preprocessing matrix $\mathbf{P}$ which is full rank. This method allows to achieve different UEP configurations using different subcodes, only by changing the preprocessing matrix at the transmitter and knowing the indices of the pruned columns of the mother code at the receiver. Our scheme thus offers some flexibility. However, storing the preprocessing matrices can represent a drawback since they are block code matrices in front of LDPC encoder. In the next section, we present a completely flexible particular case of the pruning procedure.

4.4 A Flexible Particular Pruning Case

What has been previously explained allows to explore all the reachable UEP configurations given the number of pruned columns, but may be too complex for a real system. An alternative is to restrict ourselves in choosing the pruned columns and the information columns of the subcode only among the information columns of the mother code. Then $\mathbf{H_s}$ and $\mathbf{G_s}$ are obtained by removing columns in $\mathbf{H_m}$, and the corresponding ones in $\mathbf{G_m}$ which are columns of the identity part of $\mathbf{G_m}$. The corresponding rows of $\mathbf{G_m}$ are also removed. Then $\mathbf{H_s}$ and $\mathbf{G_s}$ are of size $M_0 \times N_0 - (K_0 - K_1)$ and $K_1 \times N_0 - (K_0 - K_1)$, respectively. They are both of full rank and the code rate of the subcode is the target rate:

$$R_1 = 1 - \frac{\text{rank}(\mathbf{H_s})}{N-(K_0-K_1)} = \frac{K_1}{N_0-(K_0-K_1)}$$

Despite the restriction, this is a very flexible approach since we only need to know the indices of the pruned

columns of the mother code at the transmitter and at the receiver, to encode and decode, so almost no memory is required to reach different UEP configurations with the same mother code. Thus, since the preprocessing matrix $\mathbf{P}$ has not a neglectible size, if we accept a slight loss in optimization, we would prefer the flexible case for real-world applications.

5 Results

We present here the results we obtained using the general pruning case, that says the optimization ran over all the columns of the parity matrix. In the simulations, we did not use the preprocessing matrix since we worked with the all-zero codeword. We started from a regular $(3, 6)$ LDPC mother code of length $N_0 = 2000$ and code rate $R_0 = 1/2$. The subcode has a length of $N_1 = 1500$ and code rate $R_1 = 1/3$. The optimization is done for $N_c = 3$ classes with $\alpha(1) = 0.1$, $\alpha(2) = 0.9$. We compare the performances of optimized non-concentrated (degrees of checks between 2 and 6) code and almost concentrated (degrees of checks between 4 and 6) codes. The decoding is done by using only the pruned parity-check matrix of the mother code.

TABLE I

COMPARISON OF DEGREE DISTRIBUTIONS $\rho^{(C_k)}(j)$ FOR THE DIFFERENT CLASSES OF THE CONCENTRATED CODE.

Check profile of the almost concentrated code					
j	2	3	4	5	6
C_1	0	0	9.03e-01	9.61e-02	0
C_2	0	0	6.66e-01	3.33e-01	0
C_3	0	0	3.55e-01	4.86e-01	1.58e-01

TABLE II

COMPARISON OF DEGREE DISTRIBUTIONS $\rho^{(C_k)}(j)$ FOR THE DIFFERENT CLASSES OF THE UNCONCENTRATED CODE.

Check profile of the non-concentrated code					
j	2	3	4	5	6
C_1	5.66e-01	2.60e-01	1.53e-01	2.00e-02	0
C_2	6.44e-02	1.40e-01	3.76e-01	3.60e-01	5.77e-02
C_3	1.33e-03	8.66e-03	1.60e-01	4.81e-01	3.48e-01

Figure (2) shows the EXIT curves defined in equation (5) for each class of almost concentrated and non-concentrated check irregularity codes. The more the first class is protected, the more the less protected ones are degraded: the best protected class has a faster convergence for the non-concentrated code than the corresponding one in the concentrated code. The intermediate classes are quite equivalent whereas the last class of the non-concentrated code has a slower convergence than the corresponding one in the concentrated one. Figure(3) represents bit error rates of the UEP almost concentrated and non-concentrated codes after 30 decoding iterations. We observe that the check irregularity is a mean to achieve UEP at low number of iterations (accelerating the convergence), but also at a high number of iterations since the differences between classes are still visible after 30 decoding iterations when looking at the bit error rates. In particular, we still have better performance at the 30th iteration for the first class of the non-concentrated code than for the concentrated one, whereas the last class is worse in the non-concentrated code than in the almost concentrated one. We can see that stretching the check degrees allows a stronger difference in the error protection, without degradation of total average bit error probability at this code length, compared with the concentrated code.

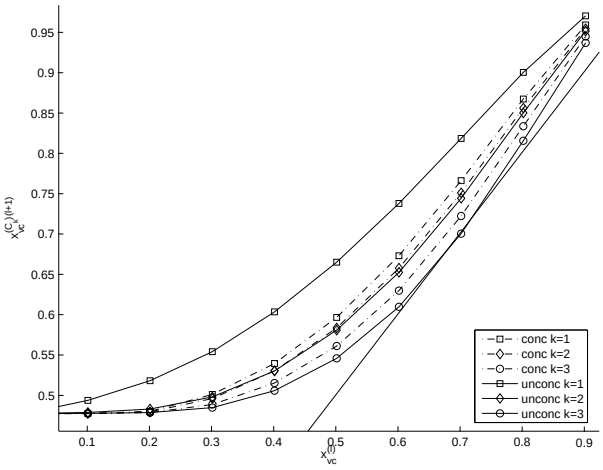

Fig. 2. EXIT curves of classes of almost concentrated and non-concentrated check irregularity codes at $E_b/N_0 = 1.5$dB.

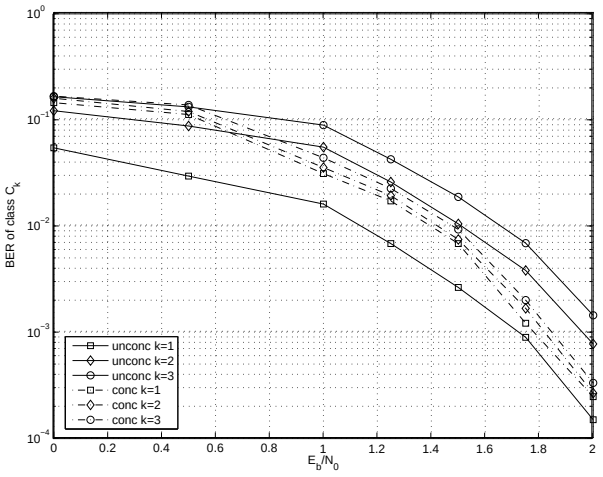

Fig. 3. Bit error rates of classes of almost concentrated and non-concentrated check irregularity codes after 30 iterations.

6 Conclusion

In this paper, we have proposed a method to optimize the UEP properties of a bit-regular code by optimizing its check-irregularity. The so-called detailed representation of LDPC codes allowed us to define a cost function which improves the local convergence of the messages, thereby creating UEP behavior. We implemented the

cost function by a highly flexible pruning method, that allows to have different UEP configurations with a same mother code. The next step of this work would be to combine bit and check irregularities to provide the best possible UEP with LDPC codes under iterative decoding.

References

[1] C.-H. Wang and C.-C. Chao. Path-Compatible Pruned Convolutionnal (PCPC) Codes: A New Scheme for Unequal Error Protection. In *ISIT1998*, Cambridge, MA, USA, 1998.

[2] R.G. Gallager. Low-Density Parity-Check Codes. *IRE Trans. on Inform. Theory*, pages 21–28, 1962.

[3] C. Poulliat, D. Declercq, and I. Fijalkow. Optimization of LDPC Codes for UEP Channels. In *ISIT 2004*, Chicago, USA, June 2004.

[4] K. Kasai, T. Shibuya, and K. Sakaniwa. Detailedly Represented Irregular Low-Density Parity-Check Codes. *IEICE Trans. Fundamentals*, E86-A(10):2435–2443, October 2003.

[5] D.J.C. MacKay, S.T. Wilson, and M.C. Davey. Comparison of Constructions of Irregular Low-Density Parity-Check Codes. *IEEE Trans. on Communications*, 47(10):1449–1453, October 1999.

[6] S.Y. Chung, T. Richardson, and R. Urbanke. Analysis of Sum-Product Decoding Low-Density Parity-Check Codes using a Gaussian Approximation. *IEEE Trans. on Inform. Theory*, 47(2):657–670, February 2001.

[7] T. Tian, C. Jones, J.D. Villasenor, and R.D. Wesel. Construction of Irregular LDPC Codes with Low Error Floors. In *ICC2003*, Anchorage, Alaska, USA, 2003.

[8] J.C. Chen, A. Dholakia, E. Eleftheriou, M.P.C Fossorier, and X-Y Hu. Reduced-Complexity Decoding of LDPC Codes. *IEEE Trans. on Communications*, 53(8):1288–1299, August 2005.

[9] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes. *IEEE Transactions on Communications*, 47(2):619–637, February 2001.

[10] I.M. Boyarinov and G.L. Katsman. Linear Unequal Error Protection Codes. *IEEE Trans. on Inform. Theory*, 27(2):168–175, March 1981.