Design of Short Irregular LDPC Codes for a Markov-modulated Gaussian Channel

W. Proß[1,2], M. Otesteanu[1] and F. Quint[2]

[1]Faculty of Electronics and Telecommunications, Politehnica University of Timisoara, Timisoara, Romania

[2]Faculty of Electrical Engineering and Information Technology, University of Applied Sciences, Karlsruhe, Germany

Keywords: Irregular LDPC Codes, Density-evolution, Downhill-simplex, Markov-modulated Gaussian Channel.

Abstract: This paper deals with the design of short irregular Low-Density Parity-Check (LDPC) codes. An optimization method for the underlying symbol-node degree-distribution (SNDD) of an irregular LDPC code is introduced, which is based on the Downhill-Simplex (DHS) algorithm. In order to compare our method with the optimization described in (Hu et al., 2005), which is based on a simplified version of the DHS algorithm, we first designed a rate 0.5 irregular LDPC code of length $n = 504$ for an Additive White Gaussian Noise Channel (AWGNC). The proposed optimization method was then used to design an irregular LDPC code for a Markov-modulated Gaussian Channel (MMGC). The decoding performance of the resulting LDPC code is then compared to the design based on the Density-Evolution (DE) method.

1 INTRODUCTION

The importance of channel coding has increased rapidly together with the still vast growing market in the field of digital signal processing. One channel code that is more and more significant is the Low-Density Parity-Check (LDPC) code. The principle of this linear block code has already been published in 1962 by Robert Gallager (Gallager, 1962). After LDPC codes had been forgotten for decades, mainly because of their computational burden, they were rediscovered by MacKay and Neal in 1995 (MacKay and Neal, 1995). Since then lots of design techniques have been developed, yielding in LDPC codes optimized with respect to different design criteria (e.g. low error-floor, performance close to capacity, hardware implementation). A commonly used tool for the design of a class of LDPC codes called irregular LDPC codes is Density-Evolution (DE) (Richardson et al., 2001)(Luby et al., 2001). In (Eckford, 2004) the author derived Density-Evolution for Markov-modulated channels and based on that designed an irregular LDPC code for a Markov-modulated Gaussian Channel (MMGC).

In section 2 we briefly explain irregular LDPC codes and their design. In the next section we introduce the DHS-based design that we propose and then show results in section 4.

2 IRREGULAR LDPC CODES

Low-Density Parity-Check (LDPC) codes are based on a sparse Parity-Check Matrix (PCM). The n columns of a PCM stand for the n symbols of a LDPC codeword and each row represents one of $m = n - k$ unique parity-check equations with k being the number of information symbols. The code rate is then $r = \frac{k}{n}$. An alternative representation is obtained by use of a Tanner-graph (Tanner, 1981). Such a bipartite graph consists of n symbol-nodes (SN) and m check-nodes (CN) corresponding to the n columns and m rows of the PCM respectively. The SNs and CNs are connected dependent on the nonzero entries in the PCM. Considering an AWGNC the decoding of LDPC codes is done using the Belief-Propagation (BP) algorithm (Gallager, 1962) or an approximation of it (e.g. the Min-Sum (MS) decoder) (Hu et al., 2002). The LDPC code of interest in this work is the irregular LDPC code that, in contrast to regular LDPC codes, exhibit several row weights and column weights. They are described by use of polynomials. The following polynomial is used to specify the symbol-node degree-distribution (SNDD).

$$\lambda(x) = \sum_{j=1}^{N} \lambda_j x^{d_j} \quad (1)$$

λ_j is the fraction of SNs that have $j+1$ connected

Proß W., Otesteanu M. and Quint F..
Design of Short Irregular LDPC Codes for a Markov-modulated Gaussian Channel.
DOI: 10.5220/0004017000310034
In *Proceedings of the International Conference on Signal Processing and Multimedia Applications and Wireless Information Networks and Systems* (SIGMAP-2012), pages 31-34
ISBN: 978-989-8565-25-9

edges and d_N is the maximum number of adjacent edges. The description of the check-node degree-distribution is likewise.

Density-Evolution (DE) is a powerful tool to analyze the asymptotic performance of a LDPC code ensemble described by a pair of degree-distributions (for the SNs and CNs respectively). In (Richardson et al., 2001) and (Luby et al., 2001) the authors showed the possibility of designing good irregular LDPC codes based on DE. In (Luby et al., 1998) and (Richardson and Urbanke, 2001) a concentration theorem is proved that states, that the performance of an ensemble of LDPC codes decoded with a BP-decoder is concentrated around the average performance of the ensemble. The analysis of LDPC codes using DE is based on the concentration theorem and on the assumption of a cycle-free code. It is well known that the shorter the LDPC code the more cycles occur. Furthermore for short LDPC codes the length of the cycles is short with respect to the decoding iterations required in average, which leads to an harmful impact on the decoding performance. In (Amin et al., 1999) it can be seen that the gap between the predicted performance based on DE and the real performance increases inversely proportional to the block-length. Furthermore the concentration theorem does not hold for short LDPC codes. This can be seen in (MacKay et al., 1999) where a significant variation of the decoding performance over an ensemble of LDPC codes is shown. Thus DE is not an appropriate tool for the design of short LDPC codes. That is the reason for Hu et al. to consider the Downhill-Simplex (DHS) optimization (Nelder and Mead, 1965) for the design of short LDPC codes in (Hu et al., 2005).

3 DHS-BASED DESIGN

We adopt the concept of designing short LDPC codes by means of the DHS algorithm and the Progressive-Edge-Growth (PEG) algorithm, but in contrast to (Hu et al., 2005) we establish the whole DHS-algorithm instead of using a simplified version. The DHS optimization is based on a simplex $\mathcal{S} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \cdots, \boldsymbol{\lambda}_N, \boldsymbol{\lambda}_{N+1}\}$ with each vertex $\boldsymbol{\lambda}_i$ representing a unique SNDD and thus consists of N values $\{\lambda_{i,j}\}_{j=1}^{N}$. During the optimization process the vertices are constantly sorted according to their function evaluations so that $f(\boldsymbol{\lambda}_1) \le f(\boldsymbol{\lambda}_2) \le \cdots \le f(\boldsymbol{\lambda}_N) \le f(\boldsymbol{\lambda}_{N+1})$. In the context of SNDD-optimization the function evaluation is represented by the computation of the Word-Error-Ratio (WER). For the AWGNC we used the Min-Sum-decoder (Hu et al., 2002) and for the MMGC we decoded by means of the Estimation-Decoder (ED) as described in (Proß et al., 2010). The DHS algorithm always tries to replace $\boldsymbol{\lambda}_{N+1}$ (the worst vertex) by a better one. This is done based on an operation called *Reflection* which is computed by

$$\boldsymbol{\lambda}_r = \bar{\boldsymbol{\lambda}}' + \alpha(\bar{\boldsymbol{\lambda}}' - \boldsymbol{\lambda}_{N+1}) \tag{2}$$

with $\alpha = 1$ and $\bar{\boldsymbol{\lambda}}'$ being the centroid of the simplex (computed without considering $\boldsymbol{\lambda}_{N+1}$) on which the worst vertex is reflected. It is calculated according to

$$\bar{\boldsymbol{\lambda}}' = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\lambda}_i. \tag{3}$$

Depending on the WER of the reflected vertex $\boldsymbol{\lambda}_r$ one of the following four operations is processed:

Inward Contraction:

$$\boldsymbol{\lambda}_{ic} = \boldsymbol{\lambda}_{N+1} + \beta(\bar{\boldsymbol{\lambda}}' - \boldsymbol{\lambda}_{N+1}); \qquad (\beta = 0.5) \tag{4}$$

Outward Contraction:

$$\boldsymbol{\lambda}_{oc} = \bar{\boldsymbol{\lambda}}' + \beta(\bar{\boldsymbol{\lambda}}' - \boldsymbol{\lambda}_{N+1}); \qquad (\beta = 0.5) \tag{5}$$

Reduction:

$$\boldsymbol{\lambda}_{i_{new}} = \boldsymbol{\lambda}_1 + \sigma(\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_1) \qquad \forall i \setminus 1; (\sigma = 0.5) \tag{6}$$

The computation of the *Expansion* operation is based on equation 5 with $\beta = 2$. The whole algorithm can be seen in Algorithm 1.

r_{av} is the average distance of the vertices to the centroid $\bar{\boldsymbol{\lambda}}$ of the simplex and is computed by

$$r_{av} = \frac{1}{N+1}\sum_{i=1}^{N+1}\sqrt{\sum_{j=1}^{N}(\lambda_{i,j} - \bar{\lambda}_j)^2}. \tag{7}$$

$\bar{\boldsymbol{\lambda}}$ is calculated as in equation 3 except that the sum includes $\boldsymbol{\lambda}_{N+1}$. Two constraints have to be considered when optimizing a SNDD:

$$0 < \lambda_j < 1 \qquad \forall j \setminus N \tag{8}$$

$$0 < \sum_{j=1}^{N-1}\lambda_j < 1 \tag{9}$$

We respect the constraints in the same way as in (Hu et al., 2005). In order to reduce the probability of converging to a local minimum, we repeat the DHS optimization ten times. The threshold r_{thres} in Algorithm 1 is set to $1e^{-4}$ for the first nine rounds. In the last round we then include the resulting SNDDs of the nine previous rounds when creating the initial simplex and set $r_{thres} = 1e^{-10}$. For the first round the i^{th} vertex $\boldsymbol{\lambda}_i = \{\lambda_{i,1}, ..., \lambda_{i,N}\}$ of the simplex $\mathcal{S} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_N, \boldsymbol{\lambda}_{N+1}\}$ is initialized as follows:

$$\lambda_{i,j} = \begin{cases} \frac{0.5 - \frac{1}{N}}{N-1} & , \forall i \setminus N, \forall j \setminus i \\ 0.5 + \frac{1}{N} & , j = i \\ random[0, r_j^{max}] & , i = N+1 \end{cases} \tag{10}$$

with

$$r_j^{max} = 1 - \sum_{l=1}^{j-1} \lambda_{i,l} \tag{11}$$

The initializations of the next eight start-simplezes are done randomly.

4 RESULTS

We designed two short irregular LDPC codes using the optimization method described in section 3. In order to compare our design with the one described in (Hu et al., 2005) we designed one irregular LDPC code with the same code rate $r = 0.5$ and blocklength $n = 504$. We thereby used the only channel-model that the authors in (Hu et al., 2005) designed for,

Algorithm 1: DHS optimization of the SNDD.

$\mathcal{S}_{initial} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_N, \boldsymbol{\lambda}_{N+1}\}$ ▷ create initial simplex
while $(r_{av} > r_{thres})$ **do**
  SORT VERTICES;
  COMPUTE REFLECTION;
    ▷ $f(\boldsymbol{\lambda}_r)$ in between worst and 2.worst
  if $f(\boldsymbol{\lambda}_N) < f(\boldsymbol{\lambda}_r) < f(\boldsymbol{\lambda}_{N+1})$ **then**
    COMPUTE OUTWARDCONTRACTION;
    if $f(\boldsymbol{\lambda}_{oc}) < f(\boldsymbol{\lambda}_r)$ **then**
      $\boldsymbol{\lambda}_{N+1} \leftarrow \boldsymbol{\lambda}_{oc}$
    else
      PERFORM REDUCTION;
    end if
    ▷ $f(\boldsymbol{\lambda}_r)$ worse than worst or equal
  else if $f(\boldsymbol{\lambda}_{N+1}) \leq f(\boldsymbol{\lambda}_r)$ **then**
    COMPUTE INWARDCONTRACTION;
    if $f(\boldsymbol{\lambda}_{ic}) < f(\boldsymbol{\lambda}_{N+1})$ **then**
      $\boldsymbol{\lambda}_{N+1} \leftarrow \boldsymbol{\lambda}_{ic}$
    else
      PERFORM REDUCTION;
    end if
    ▷ $f(\boldsymbol{\lambda}_r)$ better than best or equal
  else if $f(\boldsymbol{\lambda}_r \leq f(\boldsymbol{\lambda}_1)$ **then**
    COMPUTE EXPANSION;
    if $f(\boldsymbol{\lambda}_e) < f(\boldsymbol{\lambda}_r)$ **then**
      $\boldsymbol{\lambda}_{N+1} \leftarrow \boldsymbol{\lambda}_e$
    else
      $\boldsymbol{\lambda}_{N+1} \leftarrow \boldsymbol{\lambda}_r$
    end if
    ▷ $f(\boldsymbol{\lambda}_r)$ in between best and 2.worst
  else if $f(\boldsymbol{\lambda}_1) < f(\boldsymbol{\lambda}_r) \leq f(\boldsymbol{\lambda}_N)$ **then**
    $\boldsymbol{\lambda}_{N+1} \leftarrow \boldsymbol{\lambda}_r$
  end if
end while

which is the AWGNC. The second design was done for comparison purposes with the DE-based design in (Eckford, 2004), where the author designed a rate $r = 0.608$ irregular LDPC code for a MMGC. Thus we designed a LDPC code with the same code rate for the MMGC as well and chose the blocklength to be $n = 576$. For both designs the maximum symbol-node degree was set to $d_N = 15$. Based on the designed SNDD a PCM was constructed using the PEG algorithm and based on a following simulation the Bit-Error-Ratio (BER) and the Word-Error-Ratio (WER) were computed for different values of $\frac{E_b}{N_0}$.

The resulting SNDD for the AWGNC was $\lambda(x) = 0.42958x^2 + 0.40154x^3 + 0.00017x^4 + 0.07714x^5 + 0.0001x^6 + 0.00362x^7 + 0.00085x^8 + 0.06449x^9 + 0.00028x^{10} + 0.00029x^{11} + 0.00347x^{12} + 0.01379x^{13} + 0.00031x^{14} + 0.00438x^{15}$. The simulation-results can be seen in Figure 1. For comparison purposes the results based on the SNDD of (Hu et al., 2005) are depicted as well. All curves in Figure 1 are based on 100 decoding iterations with the Min-Sum decoder (Hu et al., 2002). It is well seen that the performance of our LDPC code beats the one from (Hu et al., 2005) with up to $0.25dB$ for the BER and up to $0.35dB$ for the WER.

The design results for the MMGC can be seen in Figure 2. For each of the three simulations a MMGC with $P_{good} = 0.6$ and $P_{bad} = 0.3$ and an ED-decoder as described in (Proß et al., 2010) was used. The first simulation is based on the design results in (Eckford, 2004) that are obtained by using DE. The second simulation evaluates the decoding performance by using our design method described in section 3. The resulting SNDD was $\lambda(x) = 0.35489x^2 + 0.24392x^3 + 0.18349x^4 + 0.12821x^5 + 0.04756x^6 + 0.00259x^7 + 0.00221x^8 + 0.00227x^9 + 0.00231x^{10} + 0.00239x^{11} + 0.00661x^{12} + 0.00279x^{13} + 0.01083x^{14} + 0.00995x^{15}$. We additionally added the results of a simulation with a regular LDPC code having a SNDD of $\lambda(x) = x^3$. The results in Figure 2 show that the irregular LDPC code designed with our DHS-method beats the regular LDPC code as well as the irregular LDPC code designed with DE in terms of BER and WER.

5 CONCLUSIONS

In this paper an optimization method for the symbol-node degree-distribution of irregular LDPC codes is introduced, that is based on the Downhill-Simplex algorithm. We proved that the decoding performance increases when designing short irregular LDPC codes with our design method instead of the simplified DHS version in (Hu et al., 2005). Furthermore an irregular

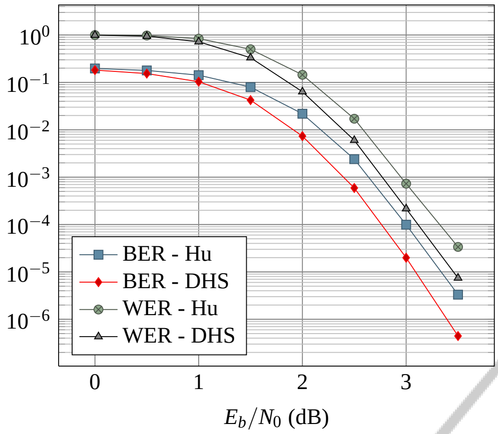

Figure 1: Two $r = 0.5$ LDPC codes with $n = 504$ and an AWGNC. DHS refers to the Downhill-Simplex based design proposed in this paper and Hu refers to the design results obtained in (Hu et al., 2005).

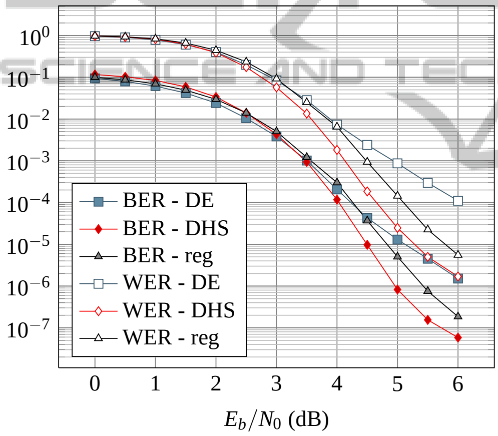

Figure 2: Three rate 0.608 LDPC codes of length $n = 576$ with a MMGC and Estimation-Decoding. DE stands for the Density-Evolution design, DHS for the Downhill-Simplex based design described in section 3 and reg for a regular LDPC code.

LDPC code of length $n = 576$ is designed with the proposed algorithm for a Markov-modulated Gaussian Channel. The results of a following simulation reveal a superior decoding performance of the LDPC code designed with our method compared to the design by means of Density-Evolution.

ACKNOWLEDGEMENTS

This work is part of the project MERSES and has been supported by the European Union and the German state Baden-Württemberg.

REFERENCES

Amin, T. R., Richardson, T., Shokrollahi, A., and Urbanke, R. (1999). Design of provably good low-density parity check codes. In *IEEE Transactions on Information Theory*.

Eckford, A. W. (2004). *Low-density parity-check codes for Gilbert-Elliott and Markov-modulated channels*. PhD thesis, University of Toronto.

Gallager, R. G. (1962). Low density parity check codes. *IRE Trans on Information Theory*, 1:21–28.

Hu, X. Y., Eleftheriou, E., and Arnold, D. M. (2005). Regular and irregular progressive edge-growth tanner graphs. *Information Theory, IEEE Transactions on*, 51(1):386–398.

Hu, X. Y., Eleftheriou, E., Arnold, D. M., and Dholakia, A., editors (2002). *Efficient implementations of the sum-product algorithm for decoding LDPC codes*, volume 2.

Luby, M., Mitzenmacher, M., Shokrollah, A., and Spielman, D. (1998). Analysis of low density codes and improved designs using irregular graphs. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 249–258.

Luby, M. G., Mitzenmacher, M., Shokrollahi, M. A., and Spielman, D. A. (2001). Improved low-density parity-check codes using irregular graphs. *Information Theory, IEEE Transactions on*, 47(2):585–598.

MacKay, D. and Neal, R. (1995). Good codes based on very sparse matrices. *Cryptography and Coding*, pages 100–111.

MacKay, D. J. C., Wilson, S. T., and Davey, M. C. (1999). Comparison of constructions of irregular gallager codes. *Communications, IEEE Transactions on*, 47(10):1449–1454.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308.

Proß, W., Quint, F., and Otesteanu, M. (2010). Estimation-decoding of short blocklength ldpc codes on a markov-modulated gaussian channel. In *3rd IEEE International Conference on Signal Processing Systems (ICSPS2011) Proceedings*, pages 383–387.

Richardson, T. J., Shokrollahi, M. A., and Urbanke, R. L. (2001). Design of capacity-approaching irregular low-density parity-check codes. *Information Theory, IEEE Transactions on*, 47(2):619–637.

Richardson, T. J. and Urbanke, R. L. (2001). The capacity of low-density parity-check codes under message-passing decoding. *Information Theory, IEEE Transactions on*, 47(2):599–618.

Tanner, R. M. (1981). A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27:533–547.