Buletinul Ştiinţific al Universităţii “Politehnica” din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICĂŢII

TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 56(70), Fascicola 2, 2011

Design of short irregular LDPC codes based on a constrained Downhill-Simplex method

W. Proß[1], Franz Quint [2], M. Otesteanu [3]

***Abstract*—In this paper we present an optimization procedure for the design of irregular Low-Density Parity-Check (LDPC) codes with short blocklength. For the optimization of the Symbol-Node Degree-Distribution (SNDD) of an irregular LDPC code we adapted the complete DHS-algorithm to the constrained problem. This is in contrast to [1], where the authors only applied a simplified version of the Downhill-Simplex (DHS) method. Furthermore our optimization procedure comprises several rounds including differently initialized simplezes in order to prevent from converging to a local minimum. Compared to simulation-results based on the simplified DHS-method provided in [1] the performance of our designed LDPC code shows a gain up to 0.3dB for the Bit-Error-Ratio (BER) and 0.2dB for the Word-Error-Ratio (WER).**

I. 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 [2]. After LDPC codes had been forgotten for decades, mainly because of their computational burden, they were rediscovered by MacKay and Neal in 1995 [3]. 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).

II. 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 $r = \frac{k}{n}$.

[1] Univ. ’Politehnica’ Timisoara, *wolfgang.pross@hs-karlsruhe.de*
[2] Univ. of Appl. Sciences Karlsruhe, *franz.quint@hs-karlsruhe.de*
[3] Univ. ’Politehnica’ Timisoara, *marius.otesteanu@etc.upt.ro*

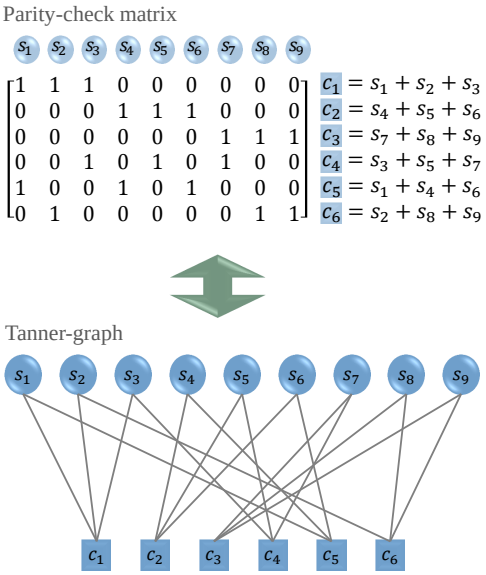

Fig. 1. Tanner-graph

An alternative representation is obtained by use of a Tanner-graph [4]. Such a bipartite graph consists of n symbol-nodes and m check-nodes corresponding to the n columns and m rows of the PCM respectively. The symbol-nodes and check-nodes are connected dependent on the nonzero entries in the PCM. Figure 1 shows an example of a PCM and the appropriate Tanner-graph.

The decoding of LDPC codes is done using the Belief-Propagation (BP) algorithm [2] or an approximation of it (e.g. the Min-Sum (MS) decoder) [5].

A. regular LDPC codes

The PCM of a regular LDPC code always possesses exactly γ nonzero elements in each column and ρ nonzero elements in each row and thus the number of adjacent edges is the same for all symbol-nodes and check-nodes respectively.

B. irregular LDPC codes

In contrast to regular LDPC codes, irregular 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).

$$\Lambda(x) = \sum_{i>=2}^{d_s^{max}} \Lambda_i x^i \tag{1}$$

The degree i determines how many edges are connected to one symbol-node (and thus the column-weight). Λ_i is the fraction of symbol-nodes for that degree i applies. Λ_i multiplied by the total number of symbol-nodes yields in the number of symbol-nodes that share the same number of adjacent edges which is i. d_s^{max} is the maximum degree. The description of the check-node degree-distribution is likewise. The use of a monomial for a pair of degree-distributions (for the symbol-nodes and check-nodes) leads to a regular LDPC code where the coefficients have to be one. $\Lambda(x) = x^3$ for example denotes a LDPC code with three adjacent edges for all symbol-nodes and thus a column-weight of three for all the columns.

C. Design of the symbol-node degree-distribution

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 symbol-nodes and check-nodes respectively). In [6] and [7] the authors showed the possibility of designing good irregular LDPC codes based on DE. In [8] and [9] 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 blocklength 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 [10] it can be seen that the gap between the predicted performance based on DE and the real performance increases inversely proportional to the blocklength. Furthermore the concentration theorem does not hold for short LDPC codes. This can be seen in [11] 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, Eleftheriou and Arnold to consider the Downhill-Simplex (DHS) optimization for the design of short LDPC codes in [1].

III. Downhill-Simplex Optimization

The downhill-simplex (DHS) optimization is a direct search method that involves direct evaluations of the function itself instead of derivations of the function. It is also called Nelder-Mead algorithm, named by the authors that first introduced the optimization method for multidimensional unconstrained nonlinear problems in [12]. It is based on a simplex

$$\mathcal{S} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N, \mathbf{v}_{N+1}\} \tag{2}$$

consisting of $N+1$ vertices when optimizing a minimization problem in a $N-$ dimensional space $\mathbf{R}^N$. During the optimization process the vertices are constantly sorted according to their function evaluations so that

$$f(\mathbf{v}_1) \leq f(\mathbf{v}_2) \leq \cdots \leq f(\mathbf{v}_N) \leq f(\mathbf{v}_{N+1}). \tag{3}$$

$\mathbf{v}_1$ is called the best vertex and $\mathbf{v}_{N+1}$ the worst vertex. While the iterative algorithm operates, it always tries to replace the worst vertex by a better one. The first step when searching for a better vertex is the reflection operation. The worst vertex is thereby reflected on the centroid of the simplex, which is computed without considering the worst vertex according to

$$\bar{\mathbf{v}}' = \frac{1}{N}\sum_{i=1}^{N} \mathbf{v}_i. \tag{4}$$

The reflection is then computed as follows.

Reflection:

$$\mathbf{v}_r = \bar{\mathbf{v}}' + \alpha(\bar{\mathbf{v}}' - \mathbf{v}_{N+1}) \tag{5}$$

where α is usually set to $\alpha = 1$. Depending on the function evaluation $f(\mathbf{v}_r)$ of the reflected vertex $\mathbf{v}_r$ one of the following operations is processed. The usual settings of the parameters can be seen in the brackets to the right of the equations respectively.

Expansion

$$\mathbf{v}_e = \bar{\mathbf{v}}' + \gamma(\bar{\mathbf{v}}' - \mathbf{v}_{N+1}); \qquad (\gamma = 2) \tag{6}$$

OutwardContraction

$$\mathbf{v}_{oc} = \bar{\mathbf{v}}' + \beta(\bar{\mathbf{v}}' - \mathbf{v}_{N+1}); \qquad (\beta = 2) \tag{7}$$

InwardContraction

$$\mathbf{v}_{ic} = \mathbf{v}_{N+1} + \beta(\bar{\mathbf{v}}' - \mathbf{v}_{N+1}); \qquad (\beta = 0.5) \tag{8}$$

Reduction

$$\mathbf{v}_{i_{new}} = \mathbf{v}_1 + \sigma(\mathbf{v}_i - \mathbf{v}_1) \qquad \forall i \setminus 1; (\sigma = 0.5) \tag{9}$$

The whole downhill-simplex algorithm can be seen in algorithm 1. The while loop is processed until a predefined termination criterion is fulfilled. As in the case of creating an initial simplex there are different possibilities to set up a termination criterion. A valid criterion would for example be a specific value r_t for the average distance r_{av} of the vertices to the centroid of the simplex $\bar{\mathbf{v}}$. It is computed as follows.

$$r_{av} = \frac{1}{N+1}\sum_{i=1}^{N+1}\sqrt{\sum_{j=1}^{N}(v_{i,j} - \bar{v}_j)^2} \tag{10}$$

$v_{i,j}$ denotes the value of the vertex v_i in the $j-$th dimension. The centroid $\bar{v}$ of the simplex is computed according to

$$\bar{\mathbf{v}} = \frac{1}{N+1} \sum_{i=1}^{N+1} \mathbf{v}_i. \tag{11}$$

Algorithm 1 Downhill-Simplex algorithm

$\mathcal{S}_{initial} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N, \mathbf{v}_{N+1}\}$ ▷ create initial simplex
while (Termination criteria is not fulfilled) **do**
  SORT VERTICES;
  COMPUTE REFLECTION;
    ▷ $f(\mathbf{v}_r)$ in between worst and 2.worst
  if $f(\mathbf{v}_N) < f(\mathbf{v}_r) < f(\mathbf{v}_{N+1})$ **then**
    COMPUTE OUTWARDCONTRACTION;
    if $f(\mathbf{v}_{oc}) < f(\mathbf{v}_r)$ **then**
      $\mathbf{v}_{N+1} \leftarrow \mathbf{v}_{oc}$
    else
      PERFORM REDUCTION;
    end if
      ▷ $f(\mathbf{v}_r)$ worse than worst or equal
  else if $f(\mathbf{v}_{N+1}) \leq f(\mathbf{v}_r)$ **then**
    COMPUTE INWARDCONTRACTION;
    if $f(\mathbf{v}_{ic}) < f(\mathbf{v}_{N+1})$ **then**
      $\mathbf{v}_{N+1} \leftarrow \mathbf{v}_{ic}$
    else
      PERFORM REDUCTION;
    end if
      ▷ $f(\mathbf{v}_r)$ better than best or equal
  else if $f(\mathbf{v}_r \leq f(\mathbf{v}_1)$ **then**
    COMPUTE EXPANSION;
    if $f(\mathbf{v}_e) < f(\mathbf{v}_r)$ **then**
      $\mathbf{v}_{N+1} \leftarrow \mathbf{v}_e$
    else
      $\mathbf{v}_{N+1} \leftarrow \mathbf{v}_r$
    end if
      ▷ $f(\mathbf{v}_r)$ in between best and 2.worst
  else if $f(\mathbf{v}_1) < f(\mathbf{v}_r) \leq f(\mathbf{v}_N)$ **then**
    $\mathbf{v}_{N+1} \leftarrow \mathbf{v}_r$
  end if
end while

IV. DHS OPTIMIZATION OF THE SNDD

To adapt the polynomial description of the symbol-node degree-distribution (SNDD) from equation (1) to the downhill-simplex (DHS)-optimization environment we use x^{d_j} in equation (12) with d_1 being the lowest degree which is set to $d_1 = 2$. Thus the maximum value for d_j in equation (12) is $d_{max} = d_s^{max} - 1$ which is the dimension $N = d_{max}$ of the problem.

$$\sum_{j=1}^{N} \Lambda_j x^{d_j} \tag{12}$$

A. Constraints

The SNDD-optimization problem requires a constrained optimization-algorithm since

$$\sum_{j=1}^{N} \Lambda_j = 1. \tag{13}$$

As in [1] we compute the $N-$th parameter by

$$\Lambda_N = 1 - \sum_{j=1}^{N-1} \Lambda_j. \tag{14}$$

Thus we have the following two inequality constraints.

CONSTRAINT1

$$0 < \Lambda_j < 1 \qquad \forall j \setminus N \tag{15}$$

CONSTRAINT2

$$0 < \sum_{j=1}^{N-1} \Lambda_j < 1 \tag{16}$$

When optimizing the SNDD based on the DHS algorithm (Algorithm 1) the simplex $\mathcal{S}$ in equation (2) becomes $\mathcal{S} = \{\mathbf{\Lambda}_1, \mathbf{\Lambda}_2, ..., \mathbf{\Lambda}_N, \mathbf{\Lambda}_{N+1}\}$. So each vertex $\mathbf{\Lambda}_i$ consists of N values $\{\Lambda_{i,j}\}_{j=1}^{N}$ referring to the fractions of symbol-nodes having d_j adjacent edges.

In contrast to the authors in [1], that used a reduced version of the DHS algorithm, we established the complete algorithm and adapted it in order to meet the two constraints of equations (15) and (16). Every time a new vertex is computed the first constraint of equation (15) is respected by use of the procedure in Algorithm 2 (as in [1]).

Algorithm 2 Ensure 1.constraint

procedure ENSURECONSTRAINT1(Λ_j)
  while $\Lambda_j \geq 1$ **do**
    $\Lambda_j = \Lambda_j - \delta$ ▷ $\delta = e^{-5}$
  end while
  return Λ_j
end procedure

The procedure in algorithm 3 ensures to respect the second constraint of equation (16).

Algorithm 3 Ensure 2.constraint

procedure ENSURECONSTRAINT2($\mathbf{\Lambda}_a, \mathbf{\Lambda}_b$)
  while $\sum_{j=1}^{N-1} \Lambda_{a,j} \geq 1$ **do**
    $\mathbf{\Lambda}_{a_{new}} = \frac{\mathbf{\Lambda}_a + \mathbf{\Lambda}_b}{2}$
    for all $j \setminus N$ **do**
      ENSURECONSTRAINT1(Λ_j)
    end for
  end while
  return $\mathbf{\Lambda}_{a_{new}}$
end procedure

Depending on the currently processed operation the following assignments are done to the pair of vertices $(\mathbf{\Lambda}_a, \mathbf{\Lambda}_b)$:

$$(\mathbf{\Lambda}_a, \mathbf{\Lambda}_b) = \begin{cases} (\mathbf{\Lambda}_r, \bar{\mathbf{\Lambda}}') & \textbf{for } \textsc{Reflection} \\ (\mathbf{\Lambda}_e, \bar{\mathbf{\Lambda}}') & \textbf{for } \textsc{Expansion} \\ (\mathbf{\Lambda}_{oc}, \bar{\mathbf{\Lambda}}') & \textbf{for } \textsc{OutwardContraction} \\ (\mathbf{\Lambda}_{ic}, \mathbf{\Lambda}_{N+1}) & \textbf{for } \textsc{InwardContraction} \\ (\mathbf{\Lambda}_{i_{new}}, \mathbf{\Lambda}_1) & \textbf{for } \textsc{Reduction} \end{cases} \tag{17}$$

B. Function evaluations

Each time the simplex changes the vertices are sorted according to equation (3). This is done based on the function evaluations for each of the vertices. In the context of SNDD-optimization the function-evaluation is represented by the computation of the Word-Error-Rate (WER). Based on the SNDD of a vertex a Parity-Check-Matrix (PCM) is created, which is done using the Progressive-Edge-Growth algorithm from [1]. Then a simulation of the resulting LDPC code follows. We use the Min-Sum-decoder [5](an approximation of the common Belief-Propagation algorithm [2]) to decode 10^4 codewords. Each of the binary codewords is affected by a Binary-Input Additive-White-Gaussian-Noise Channel (BI-AWGNC). Thereby white gaussian noise is added to each bit of the codeword depending on a $\frac{E_b}{N_0}$-value, which is the SNR per bit and was chosen as in [1]. The WER is then computed by dividing the number of false decoded codewords by the total number of codewords.

C. Optimization process

Unfortunately the minimum to which the DHS-algorithm converges is not necessarily a global minimum. We used the process explained in Algorithm 4 to increase the probability of convergence to the global minimum.

Algorithm 4 Optimization process

```
k = 1
while k < 10 do                                   ▷ 9 repetitions
    create initial simplex;
    apply constrained DHS-algorithm;
    store Λ^k_best;
    k = k + 1;
end while
create initial simplex and integrate {Λ^1_best, ..., Λ^9_best};
apply constrained DHS-algorithm;
return Λ_best
```

The optimization process showed in Algorithm 4 consists of 10 repetitions of the constrained DHS-algorithm. This means that an initial simplex is created for 10 times.

D. Initializing simplex

For the first round of the optimization process (Algorithm 4) the i^{th} vertex $\mathbf{\Lambda_i} = \{\Lambda_{i,1}, ..., \Lambda_{i,N}\}$ of the simplex $\mathcal{S} = \{\mathbf{\Lambda}_1, \mathbf{\Lambda}_2, ..., \mathbf{\Lambda}_N, \mathbf{\Lambda}_{N+1}\}$ is initialized as follows:

$$\Lambda_{i,j} = \begin{cases} \frac{0.5-\frac{1}{N}}{N-1} & ,\forall i \setminus N, \forall j \setminus i \\ 0.5 + \frac{1}{N} & ,j = i \\ random[0, r_{max}] & ,i = N+1 \end{cases} \tag{18}$$

with

$$r_{max} = \begin{cases} 1 - \sum_{l=1}^{j-1} \Lambda_{i,l} & ,\forall l \setminus 1 \\ 1 & ,l = 1 \end{cases} \tag{19}$$

So for the first N vertices all values are exactely the same except for one degree respectively (when $j = i$) to which a bigger value is assigned. For the last of the $N + 1$ vertices all values are created randomly under the restriction of the contraints in equation (15) and (16).

The initializations of the next 8 start-simplezes are done based on the following assignment:

$$\Lambda_{i,j} \leftarrow random[0, r_{max}] \tag{20}$$

The initializing simplex of the last round is then created by integrating the best simplezes obtained from all previous optimization rounds. The remaining vertices are constructed according to equation (20). At the end of the optimization process the very best vertex is returned.

V. Results

Based on the optimization process explained in section IV we designed a SNDD for a rate $\frac{1}{2}$ LDPC code of length $n = 504$. The maximum degree was thereby set to $d_s^{max} = 15$. By use of a following simulation of the resulting LDPC code, the Bit-Error-Ratio (BER) as well as the Word-Error-Ratio (WER) was computed for several $\frac{E_b}{N_0}$-values. The simulation was done based on the All-Zero-Codeword (all bits set to zero), an BI-AWGN channel and the MS-decoder [5]. We thereby ensured that for each computation at least 200 bit-errors occurred. The results can be seen in Figure 2 and Figure 3 for several numbers of processed decoding iterations. For comparison purposes the results of a simulation based on the SNDD of [1] are depicted as well.

It is well seen that if a number of decoding-iterations $i > 50$ is processed, the performance of our LDPC code beats the one from [1] for nearly all $\frac{E_b}{N_0}$-values with up to $0.25dB$ for the BER and up to $0.35dB$ for the WER results. Furthermore it is important to mention that compared to $2(N-1)$ vertices used in [1], we reduced the number of vertices to $N + 1$ and thus decreased the computation time for one round of the DHS-algorithm.

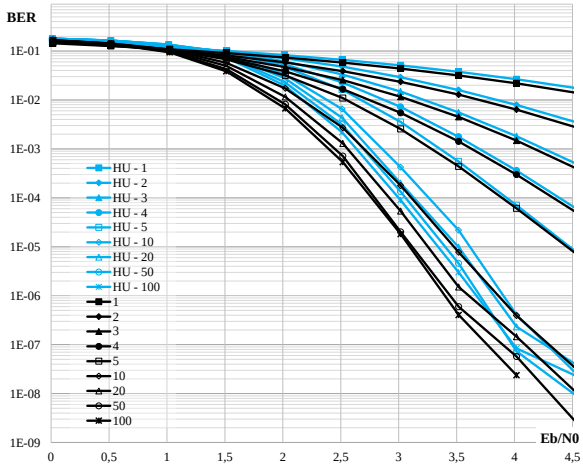

Fig. 2. Two BER-simulations of a irregular $\frac{1}{2}$-rate LDPC code of length $n = 504$. One PCM was constructed based on a SNDD designed by Hu et. al. [1] and the second PCM based on a SNDD designed by our constrained DHS-optimization procedure (Algorithm 4)

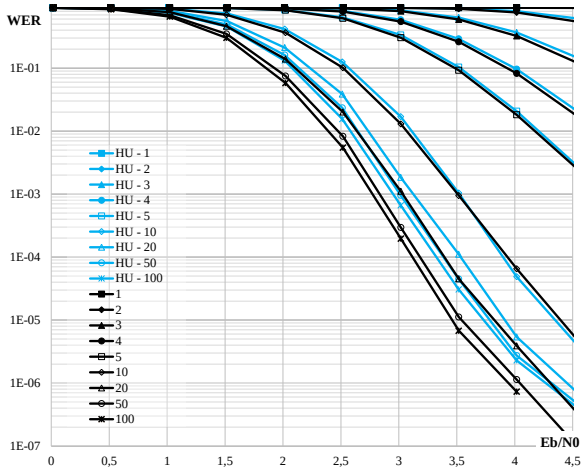

Fig. 3. Two WER-simulations of a irregular $\frac{1}{2}$-rate LDPC code of length $n = 504$. One PCM was constructed based on a SNDD designed by Hu et. al. [1] and the second PCM based on a SNDD designed by our constrained DHS-optimization procedure (Algorithm 4)

VI. Conclusion

The downhill-simplex (DHS) optimization algorithm has been adapted for the design of irregular LDPC codes. In contrast to [1], where the authors used a simple method of the DHS algorithm, we applied the DHS algorithm including all available operations, which are: reflection, expansion, outward contraction, inward contraction and reduction. As in [1] we considered the underlying constraints when optimizing the SNDD of a LDPC code. In addition we processed several optimization rounds based on differently initialized simplezes in order to prevent from converging to a local minimum. The results show a slight improvement of $\sim 0.25dB$ for the BER and $\sim 0.35dB$ for the WER compared to the results based on the simplified method presented in [1]. In the future we want to use our constrained DHS-method to design short LDPC codes for a Markov-Modulated Gaussian Channel (MMGC).

Acknowledgment

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

References

[1] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *Information Theory, IEEE Transactions on*, vol. 51, no. 1, pp. 386–398, 2005.

[2] R. G. Gallager, "Low density parity check codes," *IRE Trans on Information Theory*, vol. 1, pp. 21–28, 1962.

[3] D. MacKay and R. Neal, "Good codes based on very sparse matrices," *Cryptography and Coding*, pp. 100–111, 1995.

[4] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, pp. 533–547, 1981.

[5] X. Y. Hu, E. Eleftheriou, D. M. Arnold, and A. Dholakia, Eds., *Efficient implementations of the sum-product algorithm for decoding LDPC codes*, vol. 2, 2002.

[6] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 619–637, 2001.

[7] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 585–598, 2001.

[8] M. Luby, M. Mitzenmacher, A. Shokrollah, and D. Spielman, "Analysis of low density codes and improved designs using irregular graphs," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 249–258.

[9] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 599–618, 2001.

[10] T. R. Amin, T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low-density parity check codes," in *IEEE Transactions on Information Theory*, 1999.

[11] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "Comparison of constructions of irregular gallager codes," *Communications, IEEE Transactions on*, vol. 47, no. 10, pp. 1449–1454, 1999.

[12] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, p. 308, 1965.