On Design of Optimized Low-Density Parity-Check Codes Starting From Random Constructions

Fred Daneshgaran
ECE Department
California State University
Los Angeles, USA
e-mail: fdanesh@calstatela.edu

Massimiliano Laddomada
EE Department
Texas A&M University-Texarkana
Texarkana, USA
e-mail: mladdomada@tamut.edu

Marina Mondin
DELEN
Politecnico di Torino
Turin, ITALY
e-mail: mondin@polito.it

***Abstract*—In this paper we present a novel two step design technique for Low Density Parity Check (LDPC) codes, which, among the others, have been exploited for performance enhancement of the second generation of Digital Video Broadcasting-Satellite (DVB-S2). In the first step we develop an efficient algorithm for construction of quasi-random LDPC codes via minimization of a cost function related to the distribution of the length of cycles in the Tanner graph of the code. The cost function aims at constructing high girth bipartite graphs with reduced number of cycles of low length. In the second optimization step we aim at improving the asymptotic performance of the code via edge perturbation. The design philosophy is to avoid asymptotically weak LDPCs that have low minimum distance values and could potentially perform badly under iterative soft decoding at moderate to high Signal to Noise Ratio (SNR) values. Subsequently, we present sample results of our LDPC design strategy, present their simulated performance over an AWGN channel and make comparisons to some of the construction methods presented in the literature.**

***Keywords*-Block codes; iterative decoding; LDPC; low density parity check codes; minimum distance; near-codeword.**

I. Introduction

Low-density parity-check codes were discovered by Gallager [1] in 1962 but did not receive much attention at the time essentially for complexity reasons. In [2], Tanner resurrected LDPCs generalizing them through the introduction of the so-called Tanner graph. Only recently in 1999, Mackay [3] demonstrated that LDPCs when optimally decoded are capable of achieving information rates up to the Shannon limit.

To date, there are three main classes of LDPC constructions that can be summarized as follows:

- **Density evolution optimized LDPCs:** these LDPCs are designed by exploiting some analytic properties of the probability density functions of the log-likelihood ratios (LLRs) associated with both the bit and check nodes in the Tanner graph during iterative decoding [4].
- **Combinatorial and Algebraically designed LDPCs ([5]-[12]):** the approach is aimed at designing well structured LDPC codes based on cyclic difference families, affine configurations, Margulis-Ramanujan algebraic designs and pseudorandom constructions.
- **Graph-theoretically designed LDPCs ([13]-[15]):** this class of codes make use of graph-theoretical properties to design good LDPCs.
- **Randomly designed LDPCs ([16]-[20]):** this class of codes exploits random techniques to design parity-check matrices of good LDPCs.

In this paper we present an effective algorithm for design of ”optimized” LDPCs. The design is accomplished in two steps. Given a specific set of values (k, n) for the LDPC code, in the first step the goal is to design a related Tanner graph having a pre-specified distribution of both bit and check node degrees. In connection with this latter point, we note that we have not taken into account *optimal* degree distributions, even though we are conscious of the fact that optimal degree sequences have been proposed aimed at designing capacity-achieving degree sequences for the Binary Erasure Channel (BEC). We invite the interested reader to review [21] for a systematic study of capacity-achieving sequences of LDPCs for the BEC. The objective function for the first step in our design process is related to the cycle-distribution of the LDPC. In the second step, we consider the problems related to trapping sets/near-codewords/stopping sets by an indirect method. In particular, given the very high complexity of explicitly enumerating and determining these sets which are problematic for iterative decoding for any practically sized LDPC, we shall use a specific instance of a recursive algorithm proposed in [22] to optimize the LDPCs.

II. The LDPC Design Algorithm

In this section we present the basic rationales behind the proposed quasi-random algorithm for LDPC design. It is known that the belief-propagation decoder used for decoding LDPCs provides optimum decoding over cycle-free Tanner graphs $T(H)$. Hence, an obvious design goal is to try to minimize the influence of the cycles on the iterative decoder. This in turn suggests that a good way of designing LDPCs may be to try to maximize the smallest length cycle over all the cycles in $T(H)$ (i.e., to try to maximize the girth), while simultaneously attempting to minimize the multiplicities of the shortest length cycles in any bit node $v_{1,i} \in V_1$, $\forall i = 1, \dots, n$, or equivalently, the multiplicities of the cycles in any check nodes $v_{2,j} \in V_2$, $\forall j = 1, \dots, m$ for increasing length of the cycles above the minimum value. Another design issue that should be considered for the LDPC is the minimum distance of the code. In [22], based on experimental results we have found that the error-floor performance of a *randomly-designed* LDPC code is often constrained by its minimum distance. Note that it is known that structured codes present error-floor performance that are constrained by near-codewords/stopping sets, or more generally trapping sets [23], [24]. Unfortunately, it is not a simple task to design a LDPC code by considering both trapping sets and the minimum distance of the code. This is essentially due to the combinatorial complexity of the problem for any practically sized LDPC. In the approach proposed in this paper, we attack the problem of achieving both goals in two successive steps. In the first step, we design random LDPCs of a given size (k, n) and given bit node distribution $d_{v,i},\ \forall i = 1, \ldots, n$, whereby $d_{v,i}$ is the degree of the i-th bit node in $T(H)$, by inserting one edge in $T(H)$ at a time on a best effort basis via iterative minimization of a suitable cost function strictly related to the cycle multiplicities for any given bit node. The design is accomplished with a constraint of guaranteeing regular check node degrees. This is because experimental evidence in the literature suggests that certain check node distributions lead to LDPCs with relatively good performance in Additive White Gaussian Noise (AWGN) channels. However, the algorithm proposed here is general in that any suitable check node degree distribution can be imposed during the design.

In a second step, we optimize the edge positions using an instance of the algorithm proposed in [22], whose aim is to maximize the minimum distance of a specific LDPC code. Efficient minimum distance estimation is needed in this process and is conducted through a variant of the error-impulse method proposed by Berrou [25].

We consider the parity-check matrix H of a generic LDPC as a probabilistic space $\Omega = G(n_V, M)$ whereby n_V is the number $n = |V_1|$ of bit nodes of the code plus the number $m = |V_2|$ of the check nodes (i.e., in the case of a Tanner graph $T(H)$ associated to a LDPC code[1] $n_V = n + m$), and M is the overall number of edges in $T(H)$. For simplicity, in what follows we consider regular LDPC codes in which the number of ones in any column of H (i.e., the number of bit node edges) is equal to d_v.

Consider an algorithm to insert one edge at a time in order to create a matrix H corresponding to a regular Tanner graph $T(H)$ containing d_v edges for every bit node. Enumerate the edges with i, where $i = 1, \ldots, M = n \cdot d_v = m \cdot d_c$. Denote by X_l the variable representing the number of cycles of length l in $T(H)$, so that X_l is a random variable over the space Ω.

Let $X_{l,i}$ be the random variable representing the number of length-l cycles resulting from the addition of the i-th edge in $T(H)$. A suitable algorithm for the design of random instance of LDPC codes should try to at least minimize the number of length-4 cycle (i.e., the random variable $X_{4,i}$) for any edge $i = 1, \ldots, M$. A specific instance of this algorithm is proposed in the following.

[1]Note that we consider matrix H with m rows and n columns. This simply means that the actual rate of the code is $R \geq m/n$. In the case in which the m rows are linearly independent then $m = n - k$ and the LDPC has full rate k/n.

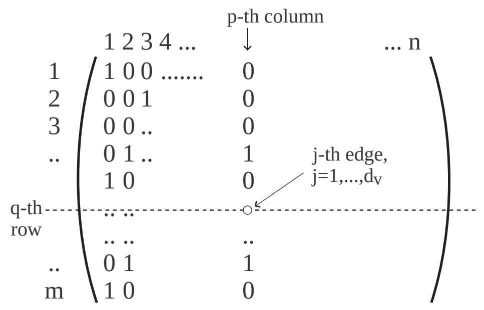

Fig. 1. Edge-by-edge growth of a sample LDPC parity-check matrix.

1) Set the bit node degree distribution $d_v(j),\ j = 1, \ldots, n$.
2) Set the number i of inserted edges to zero.
3) For any j from 1 to n perform the following tasks:
   - For any p from 1 to $d_v(j)$:
     a) generate a random set P of N_m uniformly distributed candidate positions whereby $P = \{p_k,\ k = 1, \ldots, N_m:\ p_k \in [1, \ldots, m]\}$,
     b) choose the position p_m where to insert the i-th edge by minimizing the cost function,

$$p_m = \min_{p_k \in P} X_{4,i}(p_k) \tag{1}$$

     whereby, $X_{4,i}(p_k)$ is the number of length-4 cycles over all the partial graphs $T(H)$, when the i-th edge is inserted at the j-th bit node in the p_k-th row of the parity check matrix.
     c) Increase the inserted edge counter $i \leftarrow i + 1$.
   - End (for p).
4) End (for j).

This algorithm is clearly suboptimal. An optimum algorithm would require the minimization of $X_{4,j}(p_k)$ for any inserted edge j by re-positioning all the previous inserted edges for $i = 1, \ldots, j - 1$. Clearly, this approach is impractical due to very high complexity. Our proposed algorithm on the other hand evaluates the position in which to insert the i-th edge without consideration about the positions of the previous inserted edges. The next theorem assures the necessary condition to decouple the effects of the selection of the i-th edge from that of the $(i + 1)$-th edge, on the random variable $X_{4,i+1}$. This in turn suggests that on the average and for the block length tending to infinity, we can add one edge at a time without consideration about the past (i.e., a greedy edge insertion paradigm).

Theorem I: Consider the random construction of a Tanner graph $T(H)$ with vertex sets $V_1,\ V_2$, with $|V_1| = n,\ |V_2| = m = \frac{n \cdot d_v}{d_c}$, as proposed based on the greedy algorithm above, with d_v the average bit node degree and d_c the average check node degree. Then, for any $i = 1, \ldots, M - 1$, we have:

$$\lim_{n \to \infty} E[X_{4,i+1} | X_{4,i}] = E[X_{4,i}]$$

Proof: Consider step $(i+1)$ (i.e., we have already inserted i edges and we wish to add the $(i+1)$-th edge). We conjecture that the $(i+1)$-th edge does not introduce, on the average and for $n \to \infty$, any additional length-4 cycles in comparison to the average number of length-4 cycles present at the end of the previous step.

Suppose we add the $(i+1)$-th edge in $T_i(H)$[2] corresponding to the j-th edge in the p-th column, with $j = 1, \ldots, d_v$ (in Fig. 1 we have $d_v = 3$). With this setup, the p-th column is identified by $p = \lceil \frac{(i+1)}{d_v} \rceil$.

Consider the evaluation of the number of length-4 cycles in $T_{i+1}(H)$ after the insertion of the $(i+1)$-th edge conditioned on the number of length-4 cycles present at the end of the i-th iteration (i.e., the conditioning $X_{4,i+1}|X_{4,i}$ on the probabilistic space $\Omega = G(n_V, M)$). By observing that the insertion of a new edge can only increase the number of length-4 cycles by at most a constant amount, the following relation holds $X_{4,i+1}|X_{4,i} = X_{4,i} + c,\ \forall i = 1, \ldots, M-1$. Hence, we have the inequality:

$$X_{4,i} \leq X_{4,i+1}|X_{4,i} \leq X_{4,i} + c,\ \forall i = 1, \ldots, M-1 \quad (2)$$

Of interest is the probability $P\left(X_{4,i+1} - X_{4,i} = 0 | X_{4,i} = \chi\right)$ (i.e., the probability that the generic $(i+1)$-th edge does not add any length-4 cycles conditioned on the fact that the number of length-4 cycles in the previous stage was χ).

To find this probability, consider the scheme of Fig. 1 in which a pivot point identifies at most $(d_c - 1)(d_v - 1)$ intersection points, each point corresponding to the intersection of a row and column whereby there exists a '1' in the q-th row and p-th column. Note that the number of ones in any row is less than or equal to d_c, while the number of ones in any column is less than or equal to d_v.

The probability that no cycle of length-4 is generated by insertion of a new edge at position (q,p) in H can be lower-bounded by the probability that all the $(d_c - 1)(d_v - 1)$ intersection points identified above, will have zeros. Using counting arguments, the probability of having a zero at any given intersection is:

$$a = \min\left(\frac{n-d_c}{n}, \frac{m-d_v}{m}\right),$$

from which we obtain the lower-bound:

$$P\left(X_{4,i+1} - X_{4,i} = 0 | X_{4,i} = \chi\right) \geq a^{(d_c-1)(d_v-1)}.$$

Above is the probability that during the insertion of the j-th edge in the p-th column, we do not have d_c ones per row and d_v ones per column. From this lower-bound we get the upper-bounded:

$$P\left(X_{4,i+1} - X_{4,i} \geq 1 | X_{4,i} = \chi\right) \leq 1 - a^{(d_c-1)(d_v-1)}$$

From these results we can write:

$$\begin{aligned} E[X_{4,i+1}|X_{4,i}] \quad \leq \quad & (j-1) \cdot \min\left(d_c - 1, p - 1\right) \cdot \\ & \cdot \left[1 - a^{(d_c-1)(d_v-1)}\right] + \chi \qquad (3) \end{aligned}$$

[2]The subscript i in $T_i(H)$ is used to signify the fact that the Tanner graph contains the first i edges.

where $(j-1) \cdot \min\left(d_c - 1, p - 1\right)$ is the maximum number of length-4 cycles that can be introduced. Suppose $\frac{n-d_c}{n} < \frac{m-d_v}{m} \implies a = \frac{n-d_c}{n} = 1 - \frac{d_c}{n}$, from which one obtains $a^{(d_c-1)(d_v-1)} \approx 1 - (d_c - 1)(d_v - 1)\frac{d_c}{n}$, and in turn, $1 - a^{(d_c-1)(d_v-1)} \approx (d_c - 1)(d_v - 1)\frac{d_c}{n}$. Hence, (3) can be written as follows:

$$E[X_{4,i+1}|X_{4,i}] \leq \chi + (d_v - 1)^2 (d_c - 1)\frac{d_c}{n} \approx \chi + \frac{d_c^2 d_v^2}{n} \quad (4)$$

In the limit, for block size tending to infinity, one obtains the result:

$$\lim_{n \to \infty} E[X_{4,i+1}|X_{4,i}] = \chi = E[X_{4,i}] \quad (5)$$

□

The length-4 cycles are not the only one problematic factor limiting the LDPC performance. In our LDPC design, we have used a more general cost function taking into account longer length cycles as well. The cost function adopted for LDPC design takes on the general form:

$$f(T_i(H)) = \sum_{l=4}^{l_{max}} X_{l,i} \cdot L^{-l} \quad (6)$$

where i is the index of the edge to be inserted, and l_{max} is the greatest cycle length taken into account during the LDPC design. In the previous equation, $T_i(H)$ is used to signify the fact that the cost function is evaluated on the partially constructed Tanner graph after insertion of the i-th edge. A good tradeoff between code performance and complexity burden can be obtained by considering $l_{max} = 8$. The constant L is a weighting factor useful for making the smallest length cycles more undesirable than longer length cycles. Extensive tests suggests that the value $L = 10$ may be a good compromise value for LDPC design.

It is clear that the proposed algorithm aims at minimizing the cycle distribution at each bit node in $T(H)$. However, as noted in the introduction, with this design philosophy, while the randomly designed LDPC may have good performance especially for low E_b/N_o ratios, the random approach does not assure either a good minimum distance, or a good cycle clustering. For tackling this issue, we used a second optimization step aiming at code optimization in the error-floor.

III. Optimization of the Randomly Designed LDPCs

As noted above, the second step in our LDPC design algorithm is an optimization step whereby we aim at improving the performance of the code in the error-floor region. Implicit in such a strategy is a need for an efficient algorithm to estimate the minimum distance of a given LDPC. The performance of LDPC codes may be limited by pseudo-codewords. There are however cases when the LDPC decoder behaves as a Maximum-Likelihood (ML) decoder. This is specially so for randomly constructed LDPC codes operating at medium to high SNR values. In such cases, estimation of the minimum distance d_m can be useful to assess asymptotic performance. In short, apart from theoretical interest in designing LDPCs with large d_m, the technique is useful in eliminating LDPC codes that are poor by virtue of having a low d_m. However, we emphasize that if a code has a high d_m, simulations would still be needed to assess real performance.

An effective algorithm for estimating the minimum distance of turbo codes, called the Error Impulse (EI) was proposed in [25]. Although the rationale behind the method hypothesized a ML decoder, the algorithm has shown reasonably good performance with the sub-optimal iterative decoding algorithms used for turbo-codes. In order to keep the presentation concise, we refer the interested reader to [25] for details on the theoretical rationale beyond the algorithm. Suffice it to say that the algorithm exploits the capabilities of a ML soft decoder to prevent EI input patterns. With a reasoning based on the Euclidean space geometry, in [25] it was shown that under the hypothesis that the all-zero codeword is transmitted, the minimum distance of a linear code is equal to the minimum error impulse strength A_i over all the codeword bits $i = 1, \dots, n$ which is able to make the iterative decoder diverge from the detected all-zero vector. A drawback of this algorithm for LDPCs is that it could converge toward a wrong minimum distance because of the sub-optimality of the iterative decoder and presence of decoding cycles.

In what follows, we shall briefly review the algorithm proposed in [22] with the modifications suited to what is needed in this paper. Unlike the EI technique whereby one can only obtain information about the minimum distance of the code, the proposed algorithm evaluates the minimum distance by enumerating the codewords, along with their multiplicities of the input error events to which the sum-product decoder for LDPCs converge when perturbed by an error pattern constituted by at most two EIs located in two different codeword positions. The idea is to perturb the all-zero transmitted codeword with a noise pattern able to make the iterative decoder diverge from the all-zero decoded word toward a codeword $\widehat{c}$ that with high probability, would be the minimum distance codeword (i.e., the codeword with the smallest Hamming distance from the all-zero codeword). Then, knowing the strength and position i (with $i = 1, \dots, n$) where the impulse is applied in the span of the codeword that makes the decoder diverge from the all-zero codeword, in the proposed algorithm we apply a second impulse spanning all the codeword bit positions j ($\forall i \neq j$ and $j = 1, \dots, n$) in order to list all the detected codewords close to $\widehat{c}$ and different from the all-zero codeword.

We consider transmission of an i.i.d. source bit sequence of length k, encoded with a LDPC of size (k, n), rate $R_o = k/n$, and with BPSK-modulation transmitted over an AWGN channel. In this way, the i-th received sample y_i at the output of the matched filter at the receiver is $y_i = (2c_i - 1) + n_i$, whereby c_i is the $i-$th transmitted codeword bit, and n_i is Gaussian noise with variance $\sigma_n^2 = \frac{1}{2R_oE_b/N_o}$.

The algorithm proposed here for estimation of the minimum distance of a generic LDPC code specified by its parity-check matrix H is as follows:

1) Assume that the minimum distance of the LDPC is in the range $[d_l, d_u]$, where d_l and d_u are two positive integers.
2) Set $A_m = d_u + \frac{1}{2}$, the maximum strength of one EI.
3) Consider the i-th bit node on which an EI has to be applied:
   a) set $A = d_l - \frac{1}{2}$;
   b) set flag=true;
   c) while ((flag is true) and $(A \leq A_m - 1)$)
      - $A = A + 1$ (grow the impulse strength one unit at a time until the LDPC decoder fails to decode the all-zero vector);
      - set y_j to -1, $\forall j = 1, \dots, n$ (all-zero transmitted codeword);
      - set an impulse of strength A at the i-th bit node, i.e., $y_i = y_i + A$;
      - each bit node v_i is assigned an a-posteriori Log-Likelihood Ratio (LLR) evaluated as $L_{v,i} = \frac{2}{\sigma_n^2} y_i$;
         – for any iteration of the LDPC decoder from 1 to a maximum number of iterations N_{it} perform the following tasks:
           If the Hamming weight $d_m^i = w_H(\widehat{c}^i)$ of the codeword $\widehat{c}^i$ obtained by the decoder is different from zero and $\widehat{c}^i$ has not been yet encountered during the search, then store d_m^i and update the multiplicities of the codewords with Hamming weights d_m^i;
         – if the decoded codeword $\widehat{c}$ is different from the all-zero vector then set the flag to false;
   d) end of while;
   e) For $j = 1$ to n, and $j \neq i$ (apply a second EI at position j)
      - set y_t to -1, $\forall t = 1, \dots, n$ (all-zero transmitted codeword);
      - set an impulse of strength A at the $i-$-th bit node, i.e., $y_i = y_i + A$;
      - set an impulse of strength A_m at the $j-$th bit node, i.e., $y_j = y_j + A_m$;
         – for any iteration of the LDPC decoder from 1 to a maximum number of iterations N_{it} perform the following tasks:
           If the Hamming weight $d_m^{i,j} = w_H(\widehat{c}^{i,j})$ of the codeword $\widehat{c}^{i,j}$ obtained by the decoder is different from zero and $\widehat{c}^{i,j}$ has not yet been encountered during the search, then store $d_m^{i,j}$, update the multiplicities of the codewords with Hamming weights $d_m^{i,j}$;
   f) End of For $j = 1$ to n

Some considerations are in order. The algorithm above is used on a randomly designed LDPC one bit node at the time. It allows one to obtain a set of codeword weights along with the respective multiplicities due to EIs located in the positions i and j (if any) in the span of the codeword length.

In the LDPC optimization we have conducted, the algorithm outputs a cost function resembling the Frame Error Rate (FER)

TABLE I
PARAMETERS OF THE LDPCS USED FOR SIMULATION.

(k,n)	(252, 504) L_1	(504, 1008) L_2	(252, 504) L_3	(504, 1008) L_4
d_v	3	3	3	3
d_c	6	6	6	6
C_4	0	0	0	0
C_6	1	0	0	0
C_8	1250	509	474	10
N_{it}	80	80	80	80

upper-bound:

$$F = \sum_{d=d_{min}}^{d_{max}} \mu_d e^{-R_c \frac{E_b}{N_o} d} \tag{7}$$

whereby μ_d is the multiplicity of all the codewords with Hamming weight d provided by the algorithm during the i-th iteration. In case no low-weight codeword distances are found, the algorithm generates $d = \infty$. The latter is used to signify the fact that the LDPC decoder, when perturbed by an EI in the generic bit node position i, corrects this error pattern providing the all-zeros codeword. Another way to look at this algorithm is to consider it as a way of identifying weak bit nodes in the code, i.e., the ones for which the application of an EI makes the decoder converge toward a codeword different from the all-zero codeword. Once the weak bit-nodes are identified, the edges at these weak locations are perturbed randomly while maintaining the node degrees. During perturbation, the associated set of cost functions based on (7) are recomputed and the configuration with least cost is selected.

IV. SIMULATION RESULTS

In this section, we report on some of these results and comparisons to other constructions reported in the literature.

Using the notation:

$$\lambda(x) = \sum_{i=1}^{dv_{max}} \lambda_i x^{i-1}, \ \rho(x) = \sum_{i=1}^{dc_{max}} \rho_i x^{i-1} \tag{8}$$

where, λ_i is the percentage of bit nodes of degree i, and ρ_i is the percentage of check nodes of degree i, the degree distributions for our designed LDPC codes labelled L_1, L_2, L_3, L_4 in Table (I) are as follows:

$$\begin{aligned} L_1: \ \lambda(x) &= 0.0019 + 0.498x + 0.496x^3 + 0.0039x^4, \\ \rho(x) &= 0.0079x^4 + 0.98x^5 + 0.011x^6; \\ L_2: \ \lambda(x) &= 0.001 + 0.499x + 0.5x^3, \\ \rho(x) &= 0.0019x^4 + 0.9981x^5; \\ L_3: \ \lambda(x) &= x^2, \ \rho(x) = x^5; \\ L_4: \ \lambda(x) &= x^2, \ \rho(x) = x^5. \end{aligned}$$

LDPCs labelled L_3 and L_4 are regular LDPCs, while LDPCs labelled L_1, L_2 are considered as systematic (n,k) codes. Each codeword $\mathbf{c}$ is composed of a systematic part $\mathbf{u}$, and a parity part $\mathbf{p_u}$ forming $\mathbf{c} = [\mathbf{u}, \mathbf{p_u}]$. With this setup and given the matrix $H^{n-k,n}$ of the LDPC code, it is possible to decompose $H^{n-k,n}$ as $H^{n-k,n} = (H^u, H^{p_u})$, where H^u is an $(n-k)\times(k)$ matrix specifying the source bits participating in each check equation, and H^{p_u} is a $(n-k)\times(n-k)$ matrix of the form:

$$H^{p_u} = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 1 & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 1 & 1 \end{pmatrix}. \tag{9}$$

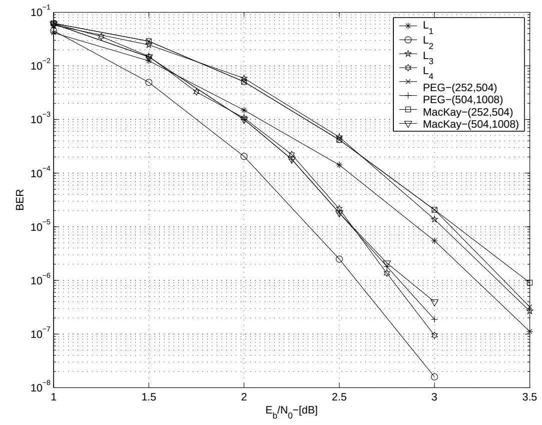

Fig. 2. BER performance of codes L_1, L_2, L_3 whose details are shown in Table (I).

These LDPC codes have been designed in two steps. In the first step we designed a random bipartite graph with the algorithm proposed in section II whose objective is to insert one edge at the time by choosing the edge positions in such a way as to minimize the cost function in (6) evaluated with $l_{max} = 8$. In the second step we optimized the randomly constructed LDPCs by applying the algorithm proposed in section III. In particular, for any edge at any bit node, we evaluated a new candidate position in the same column in such a way as to minimize the cost function in (7). Between potential edge positions yielding the same value of the cost function in (7), we have chosen the one minimizing our first cost function on the cycle distribution in (6).

For comparison purposes, the LDPCs whose performance are shown in Figs. 3 and 4 are respectively, the Ramanujan LDPC with parameters $q = 13$, $p = 5$, $d_v = 3$, $d_c = 6$ and block length $n = 2184$ [23], and the Margulis code with parameters $p = 11$, girth 8 and block length $n = 2640$. As pointed out in [23], the Ramanujan $q = 13$, $p = 5$ code is useless as it is evident from the BER/FER performance shown in Fig. 3 since the error-floor is heavily affected by codewords with weight 14 which feature prominently in the BER/FER performance. The Margulis code with block length $n = 2640$ on the contrary, is a good algebraic code for FER values higher than $5 \cdot 10^{-6} \div 10^{-5}$. Below this FER threshold, this code has performance affected by near-codewords instead of low-weight codewords, leading to the observed error-floor.

We optimized both algebraic codes via edge perturbation by repositioning every edge in any bit node of the respective parity check matrices, by choosing a new position in such a

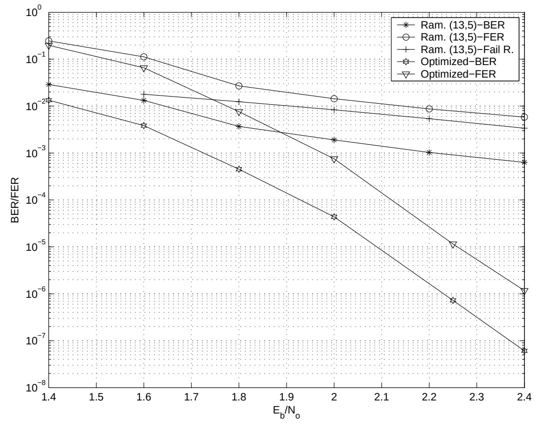

Fig. 3. Performance of the optimized Ramanujan $(13, 5)$ LDPC code.

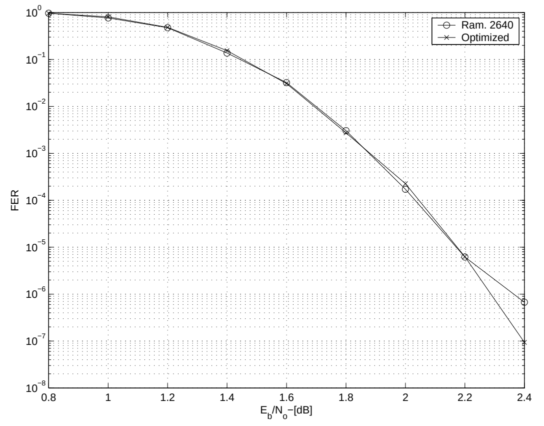

Fig. 4. Performance of the optimized Margulis 2640, $p = 11$ LDPC code.

way as to minimize the cost function in (7). Between potential edge positions yielding the same value of the cost function in (7), we have chosen the one minimizing our first cost function on the cycle distribution in (6).

V. Conclusions

In this paper we have presented a two phase design process for LDPC codes with the aim of satisfying several requirements simultaneously: 1) constructing LDPCs with quasi-random structure that have proven to perform well with sub-optimum iterative soft decoding algorithms, 2) constructing tanner graphs for the code with large girth and in general with desirable cycle distribution, and 3) optimizing the designed codes in the error-floor region via edge perturbation coupled by an efficient minimum distance estimation algorithm.

References

[1] R.G. Gallager, "Low-density parity-check codes", *IRE Transactions on Information Theory*, pp. 21 - 28, Jan. 1962.

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

[3] D.J.C. Mackay, "Good error-correcting codes based on very sparse matrices", *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399 - 431, March 1999.

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

[5] B. Vasic and O. Milenkovic, "Combinatorial constructions of low-density parity-check codes for iterative decoding," *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 1156- 1176, June 2004.

[6] S.J. Johnson and S.R. Weller, "Construction of low-density parity-check codes from Kirkman triple systems", *IEEE Global Telecommunication Conference*, vol. 2, pp. 970 - 974, Nov. 2001.

[7] J. Rosenthal and P.O. Vontobel, "Constructions of LDPC codes using Ramanujan graphs and ideas from Margulis", *Proc. of 38th Allerton Conference on Communication, Control and Computing*, pp. 248 - 257, October 2000.

[8] G. A. Margulis, "Explicit constructions of graphs without short cycles and low-density codes", *Combinatorica*, vol. 2, pp. 71-78, 1982.

[9] R. Echard and Shih-Chun Chang, "The π-rotation low-density parity check codes", *IEEE Global Telecommunications Conference*, vol. 2, pp.980 - 984, Nov. 2001.

[10] A. Prabhakar and K. Narayanan, "Pseudorandom construction of low-density parity-check codes using linear congruential sequences", *IEEE Transactions on Communications*, vol. 50, no. 9, pp. 1389-1396, Sep. 2002.

[11] B. Ammar, B. Honary, Y. Kou, J. Xu, and S. Lin, "Construction of low-density parity-check codes based on balanced incomplete block designs", *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 1257 - 1268, June 2004.

[12] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, "Near-Shannon-limit quasi-cyclic low-density parity-check codes", *IEEE Transactions on Communications*, vol. 52, no. 7, pp. 1038 - 1042, July 2004.

[13] I. Djurdjevic, S. Lin, and K. Abdel-Ghaffar, "Graph-theoretic construction of low-density parity-check codes", *Communications Letters, IEEE*, vol. 7, no. 4, pp. 171 - 173, April 2003.

[14] Hu. Xiao-Yu, E. Eleftheriou, and D.M. Arnold, "Progressive edge-growth Tanner graphs", *IEEE Global Telecommunications Conference*, vol. 2, pp. 995 - 1001, Nov. 2001.

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

[16] S.A.M. Baghdady, Y.A. Fahmy, and M.M.S. El-Soudani, "An incremental redundancy short length LDPC codes based on random construction techniques"; *IEEE 17th International Conference on Telecommunications (ICT)* pp. 18-22, 2010.

[17] S. Shebl, N. El-Fishawy, A.A. Elazm, and F.A. El-Samie,"A random construction of LDPC codes using a sub-optimal search algorithm"; *National Radio Science Conference*, pp. 1-10, 2009.

[18] C. Liang, W. Songyan, Y. Shijun, W. Ziyu, Z. Wenjun, and G. Yunfeng,"Semi-random Construction of Rate-Compatible Low-Density Parity-Check Codes"; *Proceedings of the Third International Conference on Wireless and Mobile Communications*, March 2007.

[19] Z. Huang, L. Shen, and Z. Ye,"A new random construction for low-density parity-check codes"; *Proceedings of International Conference on Communications, Circuits and Systems*, vol. 1, pp. 139-142, 2005.

[20] L. Dengsheng, L. Qiang, and L. Shaoqian,"Semi-random construction of quasi-cyclic LDPC codes"; *Proceedings of International Conference on Communications, Circuits and Systems*, vol. 1, pp. 9-13, 2005.

[21] P. Oswald and A. Shokrollahi, "Capacity-achieving sequences for the erasure channel", *IEEE Transactions on Information Theory*, vol. 48, no. 12, pp. 3017 - 3028, Dec. 2002.

[22] F. Daneshgaran, M. Laddomada, and M. Mondin, "An algorithm for the computation of the minimum distance of LDPC codes", *ETT-European Transactions on Telecommunications*, vol. 17, no. 1, pp. 57-62, Jan. 2006.

[23] D.J.C. MacKay and M.S. Postol, "Weakness of Margulis and Ramanujan-Margulis low-density parity-check codes", *Electronic Notes in Theoretical Computer Science*, vol. 74, Pub. by Elsevier Science B. V., 2003.

[24] T. Richardson, "Error floors of LDPC codes", *In Proc. of 41st Annual Allerton Conf. on Comm., Control, and Comp.*, October 2003.

[25] C. Berrou, S. Vaton, M. Jézéquel, and C. Douillard, "Computing the minimum distance of linear codes by the error impulse method"; *IEEE Globecom 2002*, pp. 1017-1020, Taipei, Taiwan, 2002.