

# Design of Low-Density Parity-Check Codes with Optimized Complexity-Rate Tradeoff

Benjamin Smith, Masoud Ardakani, Wei Yu, and Frank R. Kschischang

**Abstract**—The optimal complexity-rate tradeoff for error-correcting codes at rates strictly below the Shannon limit is a central question in coding theory. This paper proposes a numerical approach for the joint optimization of rate and decoding complexity for long-block-length irregular low-density parity-check (LDPC) codes. The proposed design methodology is applicable to any binary-input memoryless symmetric channel and any iterative message-passing decoding algorithm with a parallel-update schedule. A key feature of the proposed optimization method is a new complexity measure that incorporates both the number of operations required to carry out a single decoding iteration and the number of iterations required for convergence. This paper shows that the proposed complexity measure can be accurately estimated from a density-evolution and extrinsic-information transfer chart analysis of the code. Under certain mild conditions, the complexity measure is a convex function of the variable edge-degree distribution of the code, allowing an efficient design of complexity-optimized LDPC codes using convex optimization methods. The results presented herein show that when the decoding complexity is constrained, the complexity-optimized codes significantly outperform threshold-optimized codes at long block lengths.

**Index Terms**—Convex optimization, extrinsic-information transfer (EXIT) charts, decoding complexity, low-density parity-check (LDPC) codes.

## I. INTRODUCTION

The conventional design of irregular low-density parity-check (LDPC) codes usually involves finding a code degree distribution that maximizes the decoding threshold (i.e., the largest noise variance for which successful decoding is possible) at a given rate, or equivalently, finding a degree distribution that maximizes the code rate for a given decoding threshold [1]–[3]. However, highly optimized codes obtained from such a design procedure cannot actually be used at their thresholds, even with infinite block length, because decoding over channels very close to the threshold would require an impractical number of iterations for convergence [3], [4]. Thus, in the conventional design, codes optimized for threshold are always used on slightly better channels. This way, convergence

can be ensured in a finite number of steps. Since the channel used for design is different than the actual channel, this approach does not necessarily yield the best codes at finite complexity.

The problem with the threshold optimization is that decoding complexity is not explicitly considered in the code design process. Stated differently, if one wishes to design a practically decodable code, it would be more natural to try to find the highest-rate code for a certain affordable level of complexity on a given channel, or the code with the minimum decoding complexity for a required rate and a given channel condition, or the code that works for the largest channel threshold at a given rate and complexity. Clearly, these design objectives better reflect the requirements of practical communications systems.

A main challenge in incorporating complexity in code design is that characterizing decoding complexity as a function of code parameters may appear, at a first glance, to be a difficult task. This is especially true for iterative decoding systems in which decoding complexity depends not only on the complexity of each iteration step, but also on the number of iterations required for convergence. The main idea of this paper is that by using a uniparametric representation of the decoding trajectory, the required number of iterations in the iterative message-passing decoding of a long-block-length LDPC code under the parallel-update schedule can be accurately *estimated*. This allows one to define and to quantify a measure of decoding complexity, which can then be used to optimize the complexity-rate tradeoff for LDPC codes.

Toward this end, the paper uses a density evolution analysis of the LDPC decoding process and adopts a uniparametric representation of the decoding trajectory in terms of the evolution of message-error rate, in a format similar to an extrinsic-information transfer (EXIT) chart [5]–[7], except that message-error rate is tracked instead of mutual information and that no Gaussian assumption is made. The use of probability-of-error-based EXIT charts (which originated from the work of Gallager [8]; see also [9]) offers two key advantages. First, the uniparametric representation of the decoding trajectory allows one to accurately estimate the number of iterations required to reduce the bit-error rate (BER) from that given by the channel to a desired target. Second, in terms of message error rate, one can show that the decoding trajectory of an irregular LDPC code can be expressed as a linear combination of trajectories of *elementary* codes parameterized by their variable degrees. These two facts allow one to define a measure that relates the decoding complexity of an irregular LDPC code to its degree distribution. The code design problem

Manuscript to be submitted to the *IEEE Transactions on Information Theory* on March 5, 2007. The material in this paper was presented in part at the IEEE international Symposium on Information Theory, Adelaide, Sept. 2005 and at the 43rd Annual Allerton Conference on Communication, Control, and Computing, Allerton, IL, Sept. 2005. Masoud Ardakani is with the Electrical and Computer Engineering Department, University of Alberta. email: ardakani@ece.ualberta.ca. Benjamin Smith, Wei Yu and Frank R. Kschischang are with the Electrical and Computer Engineering Department, University of Toronto, 10 King's College Road, Toronto, Ontario M5S 3G4, Canada. emails: {ben,weiyu,frank}@comm.utoronto.ca. Phone: 416-946-8665. FAX: 416-978-4425. This work was supported by Natural Science and Engineering Research Council (NSERC) of Canada. Kindly address correspondence to Benjamin Smith (ben@comm.utoronto.ca).

then reduces to the shaping of the decoding trajectory for optimal complexity-rate tradeoff. An optimization program that determines the minimum-complexity degree distributions for a fixed code rate and a target channel can then be formulated.

This paper shows that the optimization problem formulated in this way is convex under a mild condition, which facilitates its numerical solution. Numerical code design examples suggest that complexity-optimized codes can significantly outperform threshold-optimized ones (e.g., those from [1]–[3]) at long block lengths. Further, the complexity- and threshold-optimized codes can have quite different degree distributions. In particular, complexity-optimized codes tend to have fewer degree-two variable nodes than threshold-optimized ones. This confirms a common design intuition in a rigorous way.

The methodology proposed in this paper offers one of the few instances of an iterative system in which analytic tools are available not only for predicting the convergence trajectory, but also for the tuning of system parameters for fast convergence. The proposed code design method is applicable to any binary-input memoryless symmetric (BMS) channels and any iterative decoding algorithms with symmetric update rules. These are the conditions required for the validity of density evolution analysis [2].

#### A. Related Work

The performance-complexity tradeoff for error-correcting codes has always been a central issue in coding theory. In recent years, particularly since the discovery of capacity-approaching codes, “performance” has come to mean rate at a given threshold (or threshold at a fixed rate.) For the binary erasure channel (BEC), the rate-complexity issue is addressed by Shokrollahi [10] for LDPC codes, and by Khandekar and McEliece [11] and also by Sason and Urbanke [12] for irregular repeat-accumulate codes. Hsu and Anastopoulos present in [13] a family of codes with bounded graphical complexity that are capacity-achieving on certain noisy channels under maximum likelihood decoding, and also capacity-achieving under iterative decoding on the erasure channel. Decoding the output of an erasure channel is quite different from decoding the output of a noisy channel, as the decoding procedure operates via an edge-deletion process, whose complexity scales linearly with the number of edges in the factor graph of the code. For more general channel models in which decoding complexity also scales with the number of iterations, Richardson et al. [2] proposed a numerical design procedure for irregular LDPC codes involving an ad-hoc measure of the number of iterations. However, their motivation was in providing an efficient technique for finding threshold-optimized codes. A decoding complexity measure similar to ours is presented in [14]; however, its usefulness is limited to capacity-approaching codes over the BEC, and it is in fact a special case of our measure. In a related paper [15], we studied the optimization of decoding complexity for LDPC codes under Gallager’s decoding Algorithm B [8] over the binary symmetric channel. The results of [15] rely on the one-dimensional nature of decoding Algorithm B; this paper addresses the more general case.

#### B. Outline of the Paper

The rest of this paper is organized as follows. In Section II, we briefly review the background and terminology pertaining to LDPC codes. In Section III, we present the new complexity measure for the iterative decoding system, and formulate the complexity optimization methodology. In addition, we give an EXIT-chart interpretation of the methodology, prove the convexity of the complexity measure, and propose efficient solution techniques. Numerically optimized degree distributions and simulation results are presented in Section IV. Conclusions are provided in Section V.

## II. LDPC CODES

#### A. Preliminaries

Recall that every binary linear  $(n, k)$  code can be described as the solution space of a homogeneous system of linear equations in  $F_2$  with  $n$  unknowns (“variables”) and at least  $n - k$  equations (“checks”). Equivalently such a code can be described by a bipartite Tanner graph [16] with  $n$  variable nodes and at least  $n - k$  check nodes, where a variable node and a check node are adjacent if and only if the variable participates (with nonzero coefficient) in the corresponding check. LDPC codes are characterized by the property that each variable node and each check node in the graph has “small” degree. For example, in the  $(j, k)$ -regular ensembles introduced originally by Gallager [8], each variable node has degree  $j$  and each check node has degree  $k$ . In this paper we consider ensembles of irregular LDPC codes having a Tanner graph description determined, as in [1], [2], by a pair

$$\lambda(x) = \sum_{i \geq 2} \lambda_i x^{i-1} \text{ and } \rho(x) = \sum_{j \geq 2} \rho_j x^{j-1}$$

of degree distributions, with  $\lambda(1) = \rho(1) = 1$ . Here  $\lambda_i \geq 0$  gives the probability that a randomly chosen edge in the graph is incident on a variable node of degree  $i$  and  $\rho_j \geq 0$  gives the probability that a randomly chosen edge in the graph is incident on a check node of degree  $j$ . Assuming that the checks are linearly independent, the rate of an LDPC code is related to its degree distribution by

$$R = 1 - \frac{\sum_i \frac{\rho_i}{i}}{\sum_i \frac{\lambda_i}{i}}. \quad (1)$$

Due to their representation via sparse bipartite graphs, LDPC codes are amenable to iterative decoding techniques. Iterative decoding proceeds by successively passing messages between variable nodes and check nodes, where the messages represent ‘beliefs’—typically expressed as log-likelihood ratios (LLRs)—about the values of the variable node connected to a given edge. In a single decoding iteration, messages  $m_{v \rightarrow c}$  are first sent (in parallel) from variable nodes to check nodes. At each check node a check-update computation is performed, generating messages  $m_{c \rightarrow v}$  to be sent (in parallel) from the check nodes to the variable nodes. Finally, at each variable node, a variable-update computation is performed to generate the messages for the next iteration. After sufficiently many iterations have been performed, each variable node can produce a decision about the corresponding variable. In the

rest of this paper, we assume that the check- and variable-node updates are performed according to the sum-product algorithm [17], although our technique is applicable to any symmetric update rules. For the sum-product algorithm, the update rule for LLR message at a variable-node  $v$  is

$$m_{v \rightarrow c} = m_0 + \sum_{h \in n(v) - \{c\}} m_{h \rightarrow v}, \quad (2)$$

and the update rule at a check-node  $c$  is

$$m_{c \rightarrow v} = 2 \tanh^{-1} \left( \prod_{y \in n(c) - \{v\}} \tanh(m_{y \rightarrow c}/2) \right), \quad (3)$$

where  $m_0$  is the channel message in LLR form, and  $n(v)$  represents the nodes connected to node  $v$  by an edge.

In a pair of landmark papers [2], [18], Richardson et al. presented density evolution, a numerical tool to track the density of messages passed during iterative decoding of LDPC codes, under the parallel message-passing schedule defined above. Given a degree distribution pair  $(\lambda(x), \rho(x))$  and a target channel, one can use density evolution to determine the ‘threshold’ of the code under iterative decoding, where the threshold corresponds to the largest value of the noise parameter such that iterative decoding succeeds in the asymptotic case.

In one iteration of density evolution, the algorithm combines the probability density function (pdf) of channel messages, the pdf of messages sent from variable nodes to check nodes in the previous iteration, and the degree distributions of the code, and produces the pdf of the messages that will be sent from variable nodes to check nodes in the succeeding iteration. In other words, one iteration of density evolution computes

$$\mathcal{D}_{v \rightarrow c}^{(i)} = \Phi(\mathcal{D}_{v \rightarrow c}^{(i-1)}, \mathcal{D}_{ch}, \lambda, \rho), \quad (4)$$

where  $\mathcal{D}_{v \rightarrow c}^{(i)}$  represents the pdf of variable-node to check-node messages at iteration  $i$ , and  $\mathcal{D}_{ch}$  represents the pdf of channel messages.

## B. EXIT Charts

EXIT charts, originally introduced by ten Brink in the context of turbo codes [5], provide a graphical description of density evolution. This is accomplished by tracking some statistic of the variable-to-check message density  $\mathcal{D}_{v \rightarrow c}^{(i)}$ . Often the mutual information between the message value and the corresponding variable value is tracked in an EXIT chart. In this paper, as in [8], [9], we track a *different* parameter of the message distribution; namely, the probability that the log-likelihood ratio (LLR) message has the incorrect sign. Roughly speaking, this parameter measures the fraction of messages in ‘error’. Although we do not track mutual information, we nevertheless continue to refer to the parameterized trajectories as ‘EXIT charts.’ The use of EXIT charts is central to the formulation of a threshold-optimization program, and later, a complexity-minimization program. However, to minimize the loss of information associated with a uni-parametric representation of message densities, we use density evolution for

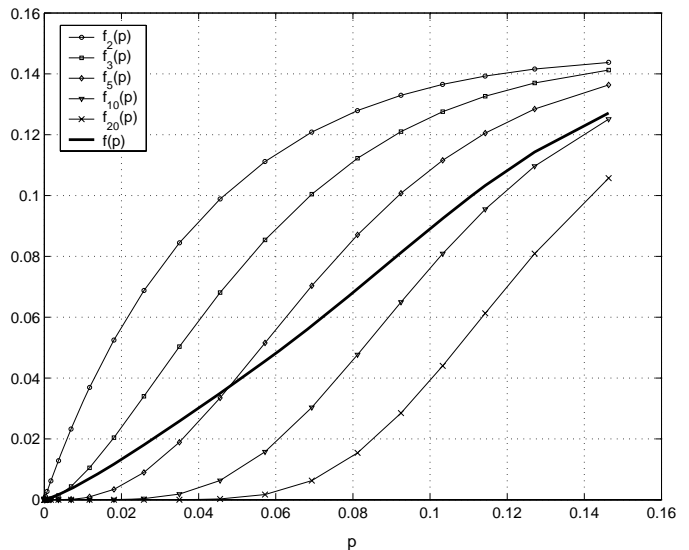


Fig. 1. Elementary EXIT charts,  $\lambda(x) = 0.1x + 0.3x^2 + 0.1x^4 + 0.2x^9 + 0.3x^{19}$ ,  $\rho(x) = x^7$

analysis; the EXIT charts serve only to provide a graphical representation of the convergence trajectory.

For a given degree distribution pair and channel condition, one can visualize the convergence behavior of the decoder by performing  $N$  iterations of density evolution, and plotting the message error rate at iteration  $i$ ,  $i \in \{1, 2, \dots, N\}$  (let us call this  $p^{(i)}$ ) versus the message error rate of iteration  $i - 1$ , i.e.,  $p^{(i-1)}$ . One can think of these EXIT charts as a mapping

$$p^{(i)} = f(p^{(i-1)}), \quad i \in \{1, 2, \dots, N\}. \quad (5)$$

Here  $p^{(i)}$  is computed as  $P_e(\mathcal{D}_{v \rightarrow c}^{(i)})$ , where  $\mathcal{D}_{v \rightarrow c}^{(i)}$  is computed using (4), and  $P_e(\cdot)$  returns the message error rate (the area under the error-tail) of a pdf.

In general,  $f$  is a function of both  $\lambda$  and  $\rho$ . However, in most approaches to the design of LDPC codes,  $\rho$  is fixed and design efforts are generally focused on optimizing  $\lambda$ . In the following, we fix  $\rho$ , and we provide advice for an appropriate choice of  $\rho$  in Section III-D4.

Due to the random construction of LDPC code graphs, one can use the total probability theorem to decompose the overall EXIT chart into a sum of elementary EXIT charts [9], i.e.,

$$f(p) = \sum_i \lambda_i \cdot f_i(p, \lambda), \quad (6)$$

where  $f_i(p, \lambda)$  is the elementary EXIT chart associated with degree- $i$  variable nodes. Fig.1 presents the elementary EXIT charts and the overall EXIT chart for a particular irregular code, using sum-product decoding over the AWGN channel. Note from the preceding equation that  $f_i$  is in general a function of  $\lambda$ .<sup>1</sup> This is because the densities that are passed in iteration  $m$  are influenced by  $\lambda$ . Thus, the output density of degree- $i$  nodes, and hence their associated message error rate, are also functions of  $\lambda$  (and  $\rho$ ). This dependency is denoted explicitly in (6).

<sup>1</sup>The dependency on  $\rho$  is dropped as  $\rho$  is considered fixed.

For decoding algorithms in which the density of messages is described by a single parameter (e.g., belief propagation over the BEC, and Gallager A/B decoding over the BSC), the elementary EXIT charts are independent of  $\lambda$ . This is explained by noting that for a given message error rate, the pdf of variable-to-check messages is independent of  $\lambda$ . Therefore, the overall EXIT chart is a linear combination of elementary EXIT charts (which are independent of  $\lambda$ ), and the code-design problem is reduced to a curve-shaping problem.

In the general case, however, due to the dependency of the elementary EXIT chart  $f_i$  on  $\lambda$ , the code-design problem cannot immediately be cast as a curve-shaping problem. Indeed, the EXIT chart of the code is affected by each  $\lambda_k$  in two ways. The first is as a multiplying factor in the linear combination of (6), and the second is the effect of  $\lambda$  on the elementary EXIT charts  $f_i(p, \lambda)$ . Therefore,

$$\frac{\partial f(p)}{\partial \lambda_k} = f_k(p, \lambda) + \sum_i \lambda_i \frac{\partial f_i(p, \lambda)}{\partial \lambda_k}. \quad (7)$$

In practice, the first term on the right hand side of (7) is much larger than the second term. An intuitive justification for this follows from the accuracy of the Gaussian assumption, for which the EXIT charts are independent of  $\lambda_k$ . Therefore, if  $\lambda$  undergoes a small change, we disregard the dependency of elementary EXIT charts on  $\lambda$ . With this assumption, (6) can be simplified to

$$f(p) = \sum_i \lambda_i f_i(p), \quad (8)$$

and, therefore, the code-design problem becomes that of synthesizing a suitable EXIT chart as a convex combination of a number of pre-computed elementary EXIT charts. However, due to the weak dependence of elementary EXIT charts on  $\lambda$ , these pre-computed elementary EXIT charts must be updated whenever  $\lambda$  undergoes a sufficiently large change.

Finally, for degree distributions of interest, it is assumed the message error rate converges to the target error rate, which guarantees that the elementary EXIT charts are defined over a discrete set of  $p$  such that  $\min(p) \leq p_t$ , where  $p_t$  is the target error rate specified in the code design problem. If needed, by interpolating, one can acquire a continuous version of the elementary EXIT charts over the interval  $[p_t, p_0]$ , where  $p_0$  is the initial message error rate (from the channel messages) and  $p_t$  is the target error rate.

### C. Threshold Optimization

Traditionally, degree distributions have been designed to achieve a desired rate, while maximizing the decoding threshold. Equivalently, given a target channel condition, one can determine the maximum rate code such that successful decoding is possible. Indeed, for a fixed check degree, the latter is obtained as the solution to the following linear optimization

program:

$$\begin{aligned} & \text{maximize} && \sum_i \frac{\lambda_i}{i} \\ & \text{subject to} && \sum_i \lambda_i f_i(p) < p, \quad p \in (0, P_e(\mathcal{D}_{ch})] \\ & && \sum_i \lambda_i = 1 \\ & && \lambda_i \geq 0 \end{aligned} \quad (9)$$

Here the central condition is the first constraint, namely that  $\sum_i \lambda_i f_i(p) < p$ ; this condition ensures an “open” EXIT chart in which the decoder can make progress (decrease  $p$ ) during each iteration. One clearly observes the code-design problem as “curve-shaping.”

Of course, since the elementary EXIT charts depend weakly on  $\lambda(x)$ , these EXIT charts should be re-computed when  $\lambda(x)$  changes by a sufficiently large amount, and the optimization should be repeated with the updated EXIT charts. This iterative procedure is repeated until the change in rate from one iteration to the next is sufficiently small.

As mentioned earlier, threshold optimization does not take decoding complexity explicitly into account. In the remainder of this paper, we will introduce a measure of decoding complexity, and propose an alternative optimization program to design codes with a better tradeoff between complexity and code rate for a fixed channel condition.

## III. COMPLEXITY-OPTIMIZED LDPC CODES

In this section, we formulate an optimization problem that finds the minimum-complexity code as a function of code rate on a fixed channel. A binary-input additive white Gaussian noise (AWGN) channel with a fixed noise variance is assumed (although the proposed optimization technique applies to any binary-input memoryless symmetric channel.)

An extreme point of the complexity-rate tradeoff curve is achieved by threshold-optimized codes. Under the assumption that the decoding scheme permits an arbitrarily large number of iterations of the sum-product algorithm, the threshold-optimized code is the highest-rate code for some target channel condition. In practice, however, the performance of a code is always evaluated for a decoder with a finite decoding complexity. In this case, one would expect the optimal codes to have progressively reduced code rates as decoding complexity becomes more and more constrained. To quantify such a tradeoff, we first propose a measure for decoding complexity, then develop a design methodology to efficiently search for the parameters (i.e., the degree distributions) of codes on the complexity-rate tradeoff curve. The relative BER performance improvements as compared to threshold-optimized codes are shown in Section IV.

### A. Measure of Decoding Complexity

For a parallel message-passing decoder, the decoding complexity of LDPC codes is proportional to the product of the number of decoding iterations and the number of arithmetic operations performed per iteration. The computational effort

(in computing (2) and (3)) per iteration is proportional to the number of edges  $E$  in the Tanner graph, while the number of messages passed per iteration is easily seen to be  $2E$ . Thus, as we argue below, the overall complexity is proportional to the total number of messages passed in the iterative decoding process.

To see that the complexity per iteration scales linearly with  $E$ , let's explicitly compute (for the sum-product algorithm) the number of operations performed in one iteration. The analysis of other update rules is similar. The variable-node update for the sum-product algorithm (2) can be performed as follows. At each variable node of degree  $v$ , we first compute the sum of the  $v$  extrinsic messages and the channel message; this requires  $v$  additions. To compute the output message for some particular edge, we can then subtract the extrinsic message received over the same edge from the computed sum. Thus, computing the  $m$  output messages requires  $2v$  operations. Summing over all variable nodes,  $2E$  operations are required.

Similarly, to perform the check-update (3) at a check node of degree  $d$ , we can first compute  $\tanh(\frac{m}{2})$  for each incoming message  $m$ . Next, using  $d - 1$  operations, we compute the product of these terms. Finally, computing an output requires dividing this quantity by the term associated with the incoming message over the same edge, then taking  $2\tanh^{-1}$  of the resulting value. Thus, computing all the check-node output messages for a check node of degree  $d$  again requires  $O(d)$  operations. Summing over all check nodes,  $O(E)$  operations are needed in total.

The overall complexity for decoding one codeword is therefore proportional to  $N \cdot E$ , where  $N$  is the number of decoding iterations. Since each codeword encodes  $R \cdot n$  information bits, where  $R$  is the code rate and  $n$  is the block length, the decoding complexity per information bit is then  $O(\frac{NE}{Rn})$ . Now, using the fact that  $n = E \sum \frac{\lambda_i}{\rho_i}$  and using the expression (1), which relates  $R$  with  $\lambda_i$  and  $\rho_i$ , we obtain the following expression for complexity  $K$

$$K = N \cdot \frac{E}{Rn} = N \cdot \frac{1}{R \sum \frac{\lambda_i}{\rho_i}} = N \cdot \frac{1 - R}{R \sum \frac{\rho_i}{\lambda_i}}. \quad (10)$$

In the rest of this section, we formulate the optimal complexity-rate tradeoff problem as that of minimizing  $K$  for a fixed  $R$ . Further, we make an assumption that the check degree distribution  $\rho_i$  is fixed to an appropriate value. In this case, minimizing the decoding complexity becomes equivalent to minimizing the number of decoding iterations.

### B. Characterizing the Number of Decoding Iterations

The main idea of this paper is that the total number of decoding iterations can be accurately estimated based on the shape of the EXIT chart  $f(p)$ . Further, this characterization of complexity can be expressed as a differentiable function of code parameters (i.e.,  $\lambda_i$ ), which allows one to use a continuous optimization approach for finding good codes with reduced decoding complexities.

From the definition of EXIT charts, it is clear that computing the decoding trajectory for a given code is equivalent to an iterative function evaluation of  $f(p)$ . Given some initial

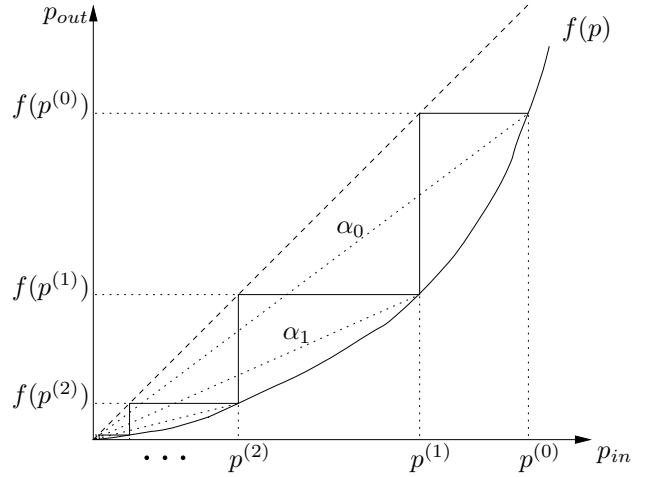


Fig. 2. The Iterative decoding process can be represented by an iterative function evaluation of the EXIT chart, i.e.  $p^{(k)} = f(p^{(k-1)})$ . Further,  $p^{(k)}$  and  $p^{(k-1)}$  are related by the local slope  $\alpha_k = \frac{p^{(k-1)}}{p^{(k)}}$ .

message error rate  $p_0 > 0$  and a target message error rate  $p_t < p_0$ , let  $p^{(0)} = p_0$ . The iterative decoding process is represented by (5), i.e.  $p^{(k)} = f(p^{(k-1)})$  for  $k \in \mathbb{N}$ . Convergence to  $p_t$  implies the existence of some  $j \in \mathbb{N}$  such that  $p^{(j)} = f(p^{(j-1)}) \leq p_t$ . The required number of iterations for convergence,  $N$ , is defined to be the minimum such  $j$ . Note that the number of required iterations is determined by the shape of  $f(p)$  alone.

Note that a sufficient condition for convergence to  $p_t$  is that  $f(p) < p, \forall p \in [p_t, p_0]$ , since this implies that the output of an iteration is strictly less than its input, and that the recursion has no fixed points in the interval of interest. Note that EXIT charts also respect the following additional conditions:  $f(p) > 0$  and  $f(p)$  is an increasing function of  $p$ , i.e.,  $f'(p) > 0$  for  $p \in [p_t, p_0]$ .

The following definition plays an important role in the characterization of the required number of iterations for convergence. Define the *local slope* of  $f(\cdot)$  at  $p^{(k)}$  to be

$$\alpha_k = \frac{f(p^{(k)})}{p^{(k)}}. \quad (11)$$

Since  $p^{(1)} = f(p^{(0)})$ ,  $p^{(2)} = f(p^{(1)})$ ,  $\dots$ ,  $p^{(N)} = f(p^{(N-1)})$ , where  $p^{(N)} \leq p_t$  and  $p^{(N-1)} > p_t$ , we have

$$p^{(N)} = \alpha_{N-1} \alpha_{N-2} \cdots \alpha_1 \alpha_0 p_0. \quad (12)$$

Note that the value of  $\alpha_k$  is equal to the slope of the line passing through the origin and the point  $(p^{(k)}, f(p^{(k)}))$ . This is illustrated in Fig. 2.

We first consider the case of determining the required number of iterations for convergence for an EXIT function that is a straight line passing through the origin, i.e.,  $f(p) = \alpha p$ ,  $\alpha < 1$ . In this simplest case, (12) reduces to  $p^{(N)} = \alpha^N p_0$ . The number of iterations required to go from  $p_0$  to  $p_t$  can then be computed exactly:

$$N = \left\lceil \frac{\ln(p_t) - \ln(p_0)}{\ln(\alpha)} \right\rceil. \quad (13)$$

$f(p)$	Actual	Estimated
$0.4p + 0.45p^2 - 1.05p^3 + 0.2p^4 + 0.2p^5 + 0.4p^6$	16	15.4
$0.7p + 0.2p^2 + 0.40p^3 - 0.4p^6$	60	59.1
$0.5p - 0.45p^2 + 0.5p^4 + 0.4p^6$	21	19.6

TABLE I

ESTIMATES FOR THE NUMBER OF ITERATIONS,  $p_t = 10^{-6}$ ,  $p_0 = 1$ 

The main idea for extending the above formula for nonlinear  $f(p)$  is to momentarily ignore the fact that  $N$  has to be an integer, and to compute the incremental increase in  $N$  as a function of the incremental change in  $p_t$ . The key is to recognize from (12) that the required number of iterations is a function of local slopes only. When  $f(p)$  is nonlinear, the nonuniform local slopes can be used as weighting factors in an accurate estimate of  $N$ .

Fix, now, some  $w \in [p_t, p_0]$ . Assume that  $\alpha(p) = \frac{f(p)}{p}$  is constant for  $p \in [w - \Delta w, w]$ , with  $\Delta w$  being sufficiently small. Returning to the linear case, and considering  $N$  to be a noninteger quantity, the incremental change in  $N$  as a function of an incremental change in  $p_t$  can be obtained by taking the derivative of  $N$  with respect to  $p_t$ :

$$\frac{dN}{dp} = \frac{-1}{p \ln(\alpha(p))}. \quad (14)$$

Therefore, the required number of iterations to progress from  $w$  to  $w - \Delta w$  is  $\Delta N(w) = \frac{-\Delta w}{w \cdot \ln(\alpha(w))}$ . Thus, to estimate the total number of iterations from  $p_0$  to  $p_t$ , we may integrate (14) and obtain

$$N \approx \int_{p_t}^{p_0} \frac{dp}{p \ln\left(\frac{p}{f(p)}\right)}. \quad (15)$$

Despite making various ‘‘continuous approximations’’ in its derivation, this formula provides a surprisingly accurate estimate of the required number of function evaluations, as exemplified in Table I for a set of polynomials that closely approximate the shape of typical EXIT charts. We note that one can generate functions for which the measure is arbitrarily inaccurate, but that for the class of EXIT-chart-like functions, such pathological cases do not arise.

Of course, to obtain  $f(p)$  one must perform density evolution, from which one could directly obtain the required number of iterations to achieve a target error rate. Thus the value of the measure lies not so much in its ability to accurately predict the number of iterations but rather as an objective function for an optimization program that seeks to determine an  $f(p) = \sum_i \lambda_i f_i(p)$  that minimizes  $N$ , subject to a rate constraint. This is facilitated by the fact that the above expression for  $N$  is differentiable with respect to the design parameters  $\lambda_i$ . More importantly, under a mild condition, (15) can be shown to be a convex function of  $\lambda_i$ .

*Theorem 1:* Let  $f(p) = \sum_i \lambda_i f_i(p)$ , where  $f(p) < p$ . The function  $\int_{p_t}^{p_0} \frac{dp}{p \ln\left(\frac{p}{f(p)}\right)}$  is convex in  $\lambda_i$  in the region  $\{\lambda_i \mid f(p) \geq e^{-2}p\}$ .

*Proof:* Convexity of the integrand is sufficient for convexity of the integral. Further, to show convexity of the integrand as a function of  $\lambda_i$ , we need only show convexity along an

arbitrary line in  $\lambda$ -space that intersects its domain [19]. Let  $\lambda_i = t\gamma_i + \psi_i$  for arbitrary  $\gamma_i$  and  $\psi_i$ . The convexity of the integrand as a function of  $t$  (on its domain,  $\{t \mid \sum_i (t\gamma_i + \psi_i)f_i(p) < p\}$ ) can be verified directly by taking its second derivative. The integrand, restricted to the line, is of the form:

$$g(t) = \frac{1/p}{\Phi - \ln(t\Gamma + \Psi)} \quad (16)$$

where  $\Phi = \ln(p)$ ,  $\Gamma = \sum_i \gamma_i f_i(p)$  and  $\Psi = \sum_i \psi_i f_i(p)$ . Using the fact that  $\Phi > \ln(t\Gamma + \Psi)$  on the domain of  $g(t)$ , a direct computation reveals that the second derivative with respect to  $t$  is always positive if  $\Phi - \ln(t\Gamma + \Psi) \leq 2$ , which is equivalent to  $f(p) \geq e^{-2}p$ . ■

Convexity is crucial for fast algorithms for numerical optimization. In Section III-D1, we describe how to determine whether the optimization region of interest lies entirely within the region for which the measure is a convex function of  $\lambda_i$ .

### C. Optimization Problem Formulation

We are now ready to formulate the problem of minimizing the complexity of a code subject to a rate constraint. The optimization variables are the variable degree distribution parameters  $\lambda_i$ . Implicitly, we must have  $\sum_i \lambda_i = 1$  and  $\lambda_i \geq 0$ . It is easy to see from (1) that if the check-degree distribution  $\rho_i$  is assumed to be fixed, the code rate is simply a function of  $\lambda_i$ . More specifically, the rate constraint becomes a linear constraint:

$$\sum_i \lambda_i/i \geq \frac{1}{1-R_0} \sum_j \rho_j/j. \quad (17)$$

Clearly, the above constraint would be met with equality for the minimal complexity code. In this case, the complexity measure  $K$  is then directly proportional to the number of iterations  $N$ .

The idea is now to start with some initial  $\bar{\lambda}$ , compute its associated EXIT functions  $f_i(p)$ , and update  $\lambda$  by solving the following optimization problem:

$$\begin{aligned} & \text{minimize} && \left( \frac{1-R_0}{R_0 \sum_i \rho_i/i} \right) \int_{p_t}^{p_0} \frac{dp}{p \ln\left(\frac{p}{\sum_i \lambda_i f_i(p)}\right)} \\ & \text{subject to} && \sum_i \lambda_i/i \geq \frac{1}{1-R_0} \sum_i \rho_i/i \\ & && \sum_i \lambda_i = 1 \\ & && \lambda_i \geq 0 \\ & && \|\lambda - \bar{\lambda}\|_\infty < \epsilon \end{aligned} \quad (18)$$

Here,  $\rho_i$  is fixed,  $R_0$  is the fixed target rate, and  $\epsilon$  is the maximum permissible change in any component of  $\lambda$ , which is set to a small number to ensure that the elementary EXIT charts  $f_i(p)$  remain accurate.

In practice, the above optimization problem is solved repeatedly, with  $\bar{\lambda}$  updated in each step. The initial  $\bar{\lambda}$  can be set to be the variable degree distribution of the threshold-optimized code for some target channel condition. Such a distribution can be obtained by solving (9). Next, we set  $R_0$  to be the target rate, (which must be lower than the rate of the

threshold-optimized code), and solve the optimization problem to obtain a minimal complexity code. However, because of the  $\epsilon$  constraint, the achievable complexity reduction in one iteration is typically limited. To further reduce complexity, we set  $\bar{\lambda}$  to the most recently computed  $\lambda$ , and repeat the optimization procedure using the updated elementary EXIT charts. This iterative process should be repeated until the complexity reduction from one iteration to the next falls below some threshold value.

#### D. Design Notes

While the preceding optimization program forms the core of our design procedure, several further comments are in order.

1) *Convexity and Optimality*: Each iteration of the design procedure consists of finding the minimum-complexity code whose degree distribution deviates from the initial condition  $\bar{\lambda}$  by at most  $\epsilon$ , by solving the optimization problem (18). This is done for a fixed channel condition and a given target rate  $R_0$ . Note that the constraints of (18) are linear. Thus, when the objective function of (18) is indeed convex (as per Theorem 1), the optimization problem (18) belongs to a class of convex optimization problems for which the globally optimal solution can be efficiently found using standard convex optimization techniques [19]. However, as shown in Theorem 1, convexity holds only in a region  $\{\lambda_i \mid f(p) \geq e^{-2}p\}$ . Thus, to ensure that a numerical solution for (18) indeed yields the minimum-complexity code in a global sense, one must be able to argue that the globally optimum solution always lies in the convex region, in which case, one may then restrict  $\{\lambda_i\}$  to be within the convexity region *a priori*, without affecting global optimality. Note that it is not sufficient to verify that the  $f(p)$  associated with a locally optimal solution lies in the convex region.

Toward this end, we argue that for all cases of interest (i.e., for any useful code), when the convexity condition is violated, it is violated at (and in the vicinity of) the origin of the EXIT chart. To understand the implications of this fact, it is instructive to consider a modified version of the rate-optimization program (9), wherein the constraint

$$\sum_i \lambda_i f_i(p) < e^{-2}p, \quad p \rightarrow 0$$

is added to the existing set of constraints, and in practice the constraint is enforced for  $p \in (0, \epsilon)$  (a reasonable value for  $\epsilon$  is  $10^{-6}$ ). The solution to the modified rate-optimization program (at some fixed channel condition) corresponds to a code of rate  $R_c$ , which is the maximum rate code whose EXIT chart lies outside the convex region (near the origin), since the added constraint requires the resulting code to violate the convexity condition near the origin. It follows that any code of rate greater than  $R_c$  will necessarily have an EXIT chart that does not violate the convexity condition near the origin, since otherwise its existence would violate the optimality of the solution (i.e., the code of rate  $R_c$ ) to the modified rate-optimization program. Therefore, if the target rate  $R_0$  of a complexity-minimized code (for the same channel condition) is greater than  $R_c$ , this strongly suggests that all codes of

rate  $R_0$  have EXIT charts that lie in the convex region, and a globally optimal solution to (18) can be efficiently found via convex programming.

In practice, threshold-optimized codes always lies inside the convex region. Thus, as one allows the target complexity to increase, the minimal complexity code must also eventually be inside the convex region. Indeed, our calculations of  $R_c$  for a range of channel conditions suggest that when the target number of decoding iterations is greater than 65, the equivalent target rate  $R_0$  is always greater than  $R_c$ . Thus, the complexity-optimized code always lies in the convex region.

When the convexity condition is not satisfied, one may always repeat the optimization with different initial conditions and choose the best among the local optimal solutions. In our code design examples, we find that locally optimal codes designed for a target complexity of 50 iterations always lie in the convex region. At a target of 20 iterations, the locally optimal codes may lie outside the convex region, but all local optimal solutions have effectively the same complexity. This suggests that the structure of the optimization problem is such that the optimal solution is not very sensitive to initial conditions.

Finally, we argue that the successive optimization approach where  $f_i(p)$  is updated by  $\lambda_i$  in each step does not affect convexity. First, note that under the assumption that the density of variable-to-check messages is Gaussian, the EXIT charts are independent of  $\lambda(x)$ . Therefore, in this case, a globally optimal solution can be found with respect to the resulting EXIT charts, while ignoring the  $\epsilon$  constraint. Now, the EXIT charts due to a Gaussian assumption are close approximations to the true EXIT charts. Therefore, whenever the EXIT charts obtained from a Gaussian assumption give rise to a convex optimization problem, we argue that convexity holds when the EXIT charts are updated with  $\lambda_i$ . Indeed, iteratively updating the EXIT charts simply serves to eliminate any minor inaccuracies, and seemingly does not affect the convexity of the optimization program.

2) *Setting a Target Message Error Rate*: The proposed design procedure involves tracking the evolution of the average message error rate over all variable nodes. In practice, however, we are not directly interested in achieving a target message-error rate  $p_t$ , but rather a target bit-error rate  $p_b$  over the information bits of the code. In the following, we relate  $p_b$  with  $p_t$  and give guidelines as to how to set  $p_t$  in practice.

First, the message error rates at different variable nodes are different and they depend on the variable degrees. Further, when a decoder completes its allotted quantity of iterations, making a decision at each variable node involves combining the channel message and all  $d$  check-to-variable messages. Note that this is equivalent to the extrinsic message error rate for a degree  $d + 1$  variable node, which is different from that of the outgoing message of a degree  $d$  variable node, (which involves a channel message and  $d - 1$  check-to-variable messages.)

To compute the message error rate of variable nodes of different degrees, it is necessary to compute the node-perspective

(rather than edge-perspective) variable degree distribution:

$$\Lambda(x) = \sum_{i=2}^{d_v} \Lambda_i x^i, \quad (19)$$

where

$$\Lambda_i = \frac{\frac{\lambda_i}{i}}{\sum_{j=2}^{d_v} \frac{\lambda_j}{j}}. \quad (20)$$

Furthermore, in a systematic realization of an LDPC code, one can assign information bits to high-degree variable nodes and parity-check bits to low-degree variable nodes. In the case of threshold-optimized codes, this is without loss in performance [20, Theorem 1]. Thus, we define a node-perspective variable degree distribution over the information bits as follows. Let  $k$  be such that  $\sum_{i=k}^{d_v} \Lambda_i \geq R_0$  and  $\sum_{i=k+1}^{d_v} \Lambda_i < R_0$ , where  $R_0$  is the target rate. The degree distribution  $\Gamma(x)$  over the information bits is:

$$\Gamma(x) = \sum_{i=k}^{d_v} \Gamma_i x^i, \quad (21)$$

where

$$\Gamma_k = 1 - \frac{\sum_{j=k+1}^{d_v} \Gamma_j}{R_0}, \text{ and } \Gamma_j = \frac{\Lambda_j}{R_0} \text{ for } i > k. \quad (22)$$

Finally, for a given code and channel condition, the bit-error rate over the information bits  $p_b$ , and the average message-error rate  $p_t$  can be related as follows:

$$p_b = \text{BER}(p_t) = \sum_{i=k}^{d_v} \Gamma_i f_{i+1}(p_t), \quad (23)$$

The above relation allows one to determine the appropriate  $p_t$  in code design, for a given target  $p_b$ . In each iteration of the optimization procedure,  $p_t$  can be computed based on the target  $p_b$  and the  $\lambda(x)$  in the current iteration. Note that equation (23) can be easily solved numerically, as  $\text{BER}(p_t)$  is a strictly increasing function of  $p_t$ .

3) *Setting the Maximum Change in  $\lambda_i$* : The optimization problem also involves a constant  $\epsilon$ , which constrains the maximum change of any component of  $\lambda(x)$  in each step. Clearly, for uniparametric decoders,  $\epsilon$  can be made arbitrarily large, since the elementary EXIT charts are not functions of  $\lambda$ . In the general case (e.g., for the sum-product algorithm), we found that setting  $\epsilon$  to be on the order of  $10^{-2}$  works well.

4) *Optimizing Check-Degree Distribution*: The proposed design procedure assumes a fixed check-degree distribution  $\rho(x)$ . We now discuss the optimal choice of  $\rho(x)$ . The check-degree distribution affects the decoding complexity (10) in two distinct ways. On the one hand, given a fixed channel condition, the elementary EXIT charts (6) implicitly depend on  $\rho(x)$ , and therefore the number of decoding iterations depends on the check-degree distribution. On the other hand, for some fixed block length and target rate, the number of edges  $E$  is proportional to

$$\bar{d}_c = \left( \sum_i \frac{\rho_i}{i} \right)^{-1}, \quad (24)$$

the average check-degree. Given some fixed average check-degree, there exists convincing evidence that it suffices to consider only check-degree distributions whose degrees are concentrated on two consecutive values [21], [22]. That is to say, if the average check node degree is chosen to be  $n \leq \bar{d}_c < n+1$ , where  $n$  is an integer, then non-zero weights exist only on degree- $n$  and degree- $n+1$  check-nodes, and  $\rho_n$  satisfies

$$\frac{1}{\bar{d}_c} = \frac{\rho_n}{n} + \frac{1-\rho_n}{n+1}. \quad (25)$$

For complexity-minimized codes, it is possible to use a procedure similar to the proposed complexity-optimization program to provide further evidence in favor of concentrated check distributions. Given a target rate  $R_0$  and a corresponding  $\lambda(x)$ , we wish to determine  $\rho(x)$  such that the decoding complexity is minimized. Note that the average check-degree is fixed (due to  $R_0$  and  $\lambda(x)$ ), therefore minimizing the complexity is equivalent to minimizing the number of decoding iterations. It is possible to use an approach analogous to that outlined in Section II-B to define a check-perspective EXIT chart (i.e., one that tracks the message error rate of check-to-variable messages). One can then express the decoding complexity as a function of  $\rho(x)$  and the elementary check-perspective EXIT charts, and a suitable optimization program can be formulated to find the complexity-minimizing check-degree distribution. We have carried out this process and found that in all investigated cases, the resulting check-degree distribution was indeed concentrated on two consecutive degrees.

Finally, in many cases it suffices to consider only regular check degrees, i.e.  $\rho_n = 1$  for some  $n$ , with negligible performance degradation. Note that with regular check degrees, the capacity of the BEC can be achieved [10], and for the Gaussian channel, performance very close to the Shannon limit has also been reported [9]. Under the regular check-degree assumption, the search for the best check-degree distribution reduces to a search over a small number of candidate check degrees. For each of these values, one determines the complexity-minimizing  $\lambda(x)$ , and selects the resulting code with the smallest complexity.

#### IV. CODE-DESIGN RESULTS

We now compare the BER performance of complexity-optimized codes, designed by the methodology outlined in this paper, with the performance of threshold-optimized codes from `LdpcOpt` [23].

The proposed optimization program determines the degree distribution of the minimum complexity code (relative to our measure) for a fixed rate, a target BER, and a *fixed channel condition*. However, the performance of practical codes are always evaluated at a fixed rate and a fixed target decoding complexity (i.e., with a target number of iterations of the sum-product algorithm in a practical decoder) over a range of channel conditions. We argue that the best code for a target decoding complexity is one that achieves the target BER on the worst channel, given a fixed number of decoding iterations. This is an approach analogous to the optimality of threshold-optimized codes when the number of iterations is unbounded.

Roughly speaking, such a code can be understood as the highest threshold code for the target decoding complexity. Toward this end, we design a series of minimum-complexity codes at a fixed rate, but over a series of channel conditions. The code that meets the target complexity exactly is then chosen as the best code.

In practice, the optimization problem is solved numerically with an interior-point method, using Newton's method with equality constraints for the inner iterations [19].

### A. Degree Distributions

We present degree distributions for codes with optimal complexity-rate tradeoff for a range of rates and for two different target numbers of decoding iterations. Target decoders with 50 iterations and 20 iterations are considered, respectively. The degree distributions of the corresponding threshold-optimized codes at the same rates is also presented.

A binary-input AWGN channel with noise variance  $\sigma^2$  and a target probability of bit error  $p_b = 10^{-6}$  are assumed. The sum-product algorithm is used at the decoder.

1) *Rate=0.2 Codes:* We design complexity-optimized codes at a code rate  $R = 0.2$  and over a range of channel variances. The complexity-minimized code  $\mathcal{C}_1$  that requires exactly 50 iterations to reach a BER of  $10^{-6}$  is found to correspond to  $\sigma^2 = 2.28$ . The complexity-minimized code  $\mathcal{C}_2$  that requires exactly 20 iterations is found to correspond to  $\sigma^2 = 1.64$ . In contrast, the threshold-optimized code  $\mathcal{C}_3$  has an asymptotic threshold of  $\sigma^2 = 3.04$ . Clearly, the complexity-optimized codes have lower threshold. For each code, the check degree distribution is fixed to be  $\rho(x) = 0.5x^3 + 0.5x^4$ . The variable degree distributions are as follows:

$$\begin{aligned} \mathcal{C}_1 : \lambda(x) &= 0.18411x + 0.50891x^2 + 0.09001x^{14} \\ &\quad + 0.21697x^{15} \\ \mathcal{C}_2 : \lambda(x) &= 0.00566x + 0.80228x^2 + 0.09878x^{16} \\ &\quad + 0.09328x^{17} \\ \mathcal{C}_3 : \lambda(x) &= 0.33666x + 0.18921x^2 + 0.1035x^4 \\ &\quad + 0.03769x^5 + 0.0162x^6 + 0.14283x^9 \\ &\quad + 0.03915x^{21} + 0.02941x^{25} + 0.0345x^{26} \\ &\quad + 0.02105x^{27} + 0.0113x^{30} + 0.0385x^{39} \end{aligned}$$

2) *Rate=0.5 Codes:* We now design complexity-optimized codes at a code rate  $R = 0.5$  and over a range of channel variances. The complexity-minimized code  $\mathcal{C}_1$  that requires exactly 50 iterations to reach a BER of  $10^{-6}$  is found to correspond to  $\sigma^2 = 0.84$ . The complexity-minimized code  $\mathcal{C}_2$  that requires exactly 20 iterations is found to correspond to  $\sigma^2 = 0.73$ . The threshold-optimized code  $\mathcal{C}_3$  has an asymptotic threshold of  $\sigma^2 = 0.943$ . For each code, the check-degree distribution is  $\rho(x) = x^8$ . Their variable degree distributions

are as follows:

$$\begin{aligned} \mathcal{C}_1 : \lambda(x) &= 0.09302x + 0.44544x^2 + 0.058x^{12} \\ &\quad + 0.24452x^{13} + 0.15902x^{29} \\ \mathcal{C}_2 : \lambda(x) &= 0.00284x + 0.563x^2 + 0.03535x^3 \\ &\quad + 0.10334x^{10} + 0.03791x^{11} + 0.12935x^{19} \\ &\quad + 0.10006x^{22} + 0.02815x^{29} \\ \mathcal{C}_3 : \lambda(x) &= 0.21236x + 0.19853x^2 + 0.00838x^4 \\ &\quad + 0.07469x^5 + 0.01424x^6 + 0.16652x^7 \\ &\quad + 0.00912x^8 + 0.02002x^9 + 0.00025x^{19} \\ &\quad + 0.29589x^{29} \end{aligned}$$

3) *Rate=0.8 Codes:* Finally, we design complexity-optimized codes at a code rate  $R = 0.8$ . The complexity-minimized code  $\mathcal{C}_1$  that requires exactly 50 iterations to reach a BER of  $10^{-6}$  is found to correspond to  $\sigma^2 = 0.367$ . The complexity-minimized code  $\mathcal{C}_2$  that requires exactly 20 iterations is found to correspond to  $\sigma^2 = 0.342$ . The threshold-optimized code  $\mathcal{C}_3$  has an asymptotic threshold of  $\sigma^2 = 0.386$ . For each code, the check-degree distribution is  $\rho(x) = x^{24}$ . Their variable degree distributions are as follows:

$$\begin{aligned} \mathcal{C}_1 : \lambda(x) &= 0.06427x + 0.40013x^2 + 0.32146x^{11} \\ &\quad + 0.04231x^{26} + 0.1718x^{27} \\ \mathcal{C}_2 : \lambda(x) &= 0.00533x + 0.44486x^2 + 0.06509x^3 \\ &\quad + 0.00482x^5 + 0.01549x^6 + 0.1703x^9 \\ &\quad + 0.09494x^{19} + 0.19917x^{24} \\ \mathcal{C}_3 : \lambda(x) &= 0.15211x + 0.21512x^2 + 0.01424x^5 \\ &\quad + 0.16418x^6 + 0.09083x^7 + 0.017x^{13} \\ &\quad + 0.06875x^{14} + 0.27777x^{29} \end{aligned}$$

### B. Simulation Results

We first compare the performance of these codes at long block lengths, for which density evolution provides a close approximation to the true decoding behavior. For each degree distribution, we design a parity-check matrix for a code of length  $n = 10^5$ . We ensure that the matrix does not have any cycles of length 4, but the matrix is otherwise randomly constructed. We compare the performance of the complexity-minimized codes with that of the threshold-optimized codes. The complexity-minimized codes are evaluated for their target number of decoding iterations, while the performance of the threshold-optimized codes are presented for a range of allowable iterations. The BERs, computed over the information bits of the code, as a function of the signal-to-noise (SNR) ratio are plotted.

While it is traditional to discuss the coding gain (in dB) of one coding scheme relative to another, we prefer to evaluate the complexity-reduction afforded by our complexity-minimized codes. Fig. 3 shows the BER curves of  $R = 0.2$  codes. It can be seen that relative to the threshold-optimized code  $\mathcal{C}_3$ ,  $\mathcal{C}_2$  achieves a BER of  $10^{-6}$  with approximately 25% fewer decoding iterations. Indeed, while  $\mathcal{C}_2$  requires 20 decoding iterations,  $\mathcal{C}_3$  would require approximately 27 iterations to reach a BER of  $10^{-6}$ . Similarly, relative to  $\mathcal{C}_3$ ,  $\mathcal{C}_1$  achieves a BER of  $10^{-6}$  with approximately 33% fewer decoding iterations;  $\mathcal{C}_1$  requires 50 decoding iterations, while  $\mathcal{C}_3$  would require approximately 75 iterations to reach a BER of  $10^{-6}$ . The equivalent coding gain at a fixed 20 or 50 iterations at  $10^{-6}$  is approximately 1dB.

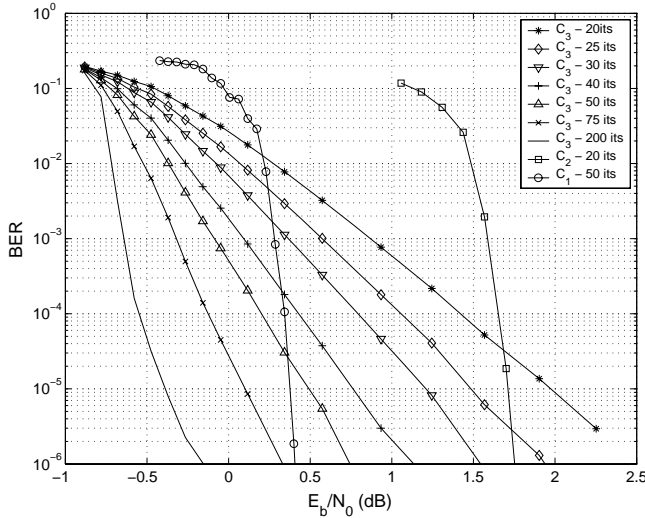


Fig. 3. Performance of long-block-length complexity-optimized codes  $C_1$  (optimized for 50 iterations),  $C_2$  (optimized for 20 iterations), and threshold-optimized code  $C_3$  at various numbers of decoding iterations.  $R = 0.2$ . Block-length  $n = 10^5$ .

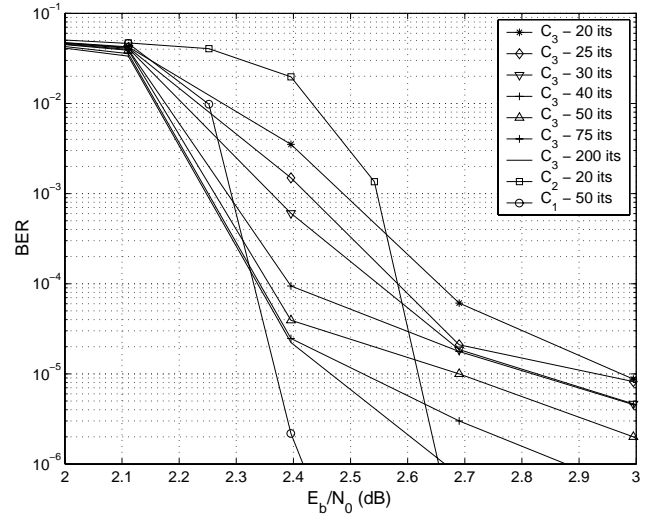


Fig. 5. Performance of long-block-length complexity-optimized codes  $C_1$  (optimized for 50 iterations),  $C_2$  (optimized for 20 iterations), and threshold-optimized code  $C_3$  at various numbers of decoding iterations.  $R = 0.8$ . Block-length  $n = 10^5$ .

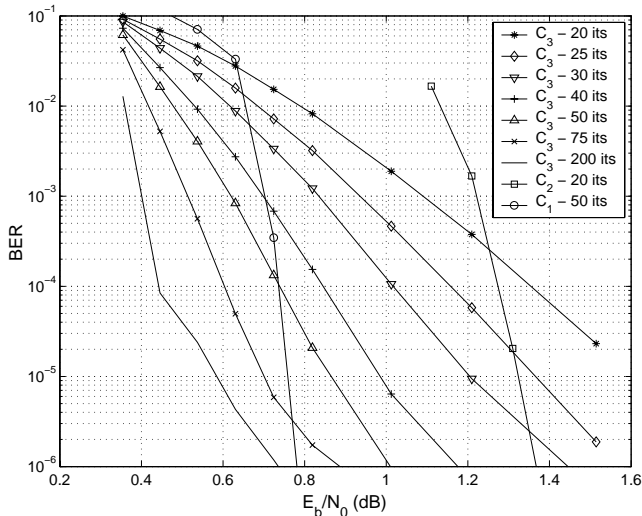


Fig. 4. Performance of long-block-length complexity-optimized codes  $C_1$  (optimized for 50 iterations),  $C_2$  (optimized for 20 iterations), and threshold-optimized code  $C_3$  at various numbers of decoding iterations.  $R = 0.5$ . Block-length  $n = 10^5$ .

Fig. 4 shows the BER curves for codes of rate  $R = 0.5$ . It can be seen that relative to the threshold-optimized code  $C_3$ ,  $C_2$  achieves a BER of  $10^{-6}$  with approximately 33% fewer decoding iterations; while  $C_2$  requires 20 decoding iterations,  $C_3$  requires approximately 30 iterations to reach a BER of  $10^{-6}$ . Similarly, relative to  $C_3$ ,  $C_1$  achieves a BER of  $10^{-6}$  with more than 33% fewer decoding iterations;  $C_1$  requires 50 decoding iterations, while  $C_3$  requires more than 75 iterations to reach a BER of  $10^{-6}$ . The equivalent coding gain is about 0.6dB.

Finally, for codes of rate  $R = 0.8$ , it can be seen from Fig. 5 that relative to the threshold-optimized code  $C_3$ ,  $C_2$  achieves a BER of  $10^{-4}$  with approximately 33% fewer decoding

iterations. Indeed, while  $C_2$  requires 20 decoding iterations,  $C_3$  requires approximately 30 iterations reach a BER of  $10^{-4}$ . Similarly, relative to  $C_3$ ,  $C_1$  achieves a BER of  $10^{-4}$  with more than 33% fewer decoding iterations;  $C_1$  requires 50 decoding iterations, while  $C_3$  requires more than 75 iterations to reach a BER of  $10^{-4}$ . The BER target of  $10^{-4}$  is chosen as a point of comparison here, because the performance of the threshold-optimized code appears to suffer from a severe error floor; the performance improvements diminish as SNR increases. A discussion of the mechanisms by which error floors are introduced is beyond the scope of this paper, but it is important to note that the complexity-minimized codes do not display an error floor for error rates greater than  $10^{-6}$ .

1) *EXIT Chart Interpretation of the Results:* To further explain the observed results, Fig. 6 shows the EXIT charts of two codes of rate 0.5,  $C_1$  and  $C_3$  for  $\sigma^2 = 0.84$ . The behavior of the EXIT charts near the origin on a log-scale is plotted in Fig. 7. One interesting observation is that the complexity-optimized code has a more closed EXIT chart in the earlier iterations and a more open EXIT chart at the later iterations. It is shown in [4] that, for a wide class of decoding algorithms, when the EXIT chart of two codes  $f_{C_A}(p)$  and  $f_{C_B}(p)$  satisfy  $f_{C_A}(p) \leq f_{C_B}(p)$ ,  $\forall p \in (0, p_0]$ , then  $C_B$  has a higher rate than  $C_A$ . In this example, since both codes have the same rate, one EXIT chart cannot dominate the other one everywhere, but the optimization program carefully trades the area underneath the EXIT chart for the complexity. Furthermore, for the complexity-minimized codes, the delayed appearance of the waterfall region is due to the fact that their EXIT charts are initially less open than the EXIT chart of the threshold-optimized code. On the other hand, the openness of the complexity-minimized EXIT charts near the origin, translates to the steeper waterfalls in the above figures. These effects can be attributed to the decreased fraction of degree-two nodes in the complexity-minimized codes. Furthermore, it

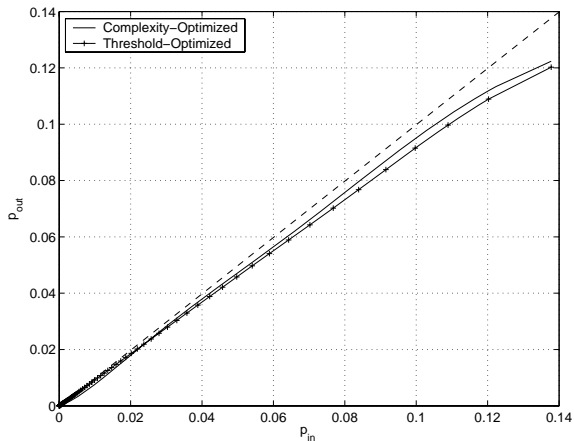


Fig. 6. EXIT charts for complexity-optimized and threshold-optimized rate 0.5 codes on an AWGN channel in linear scale

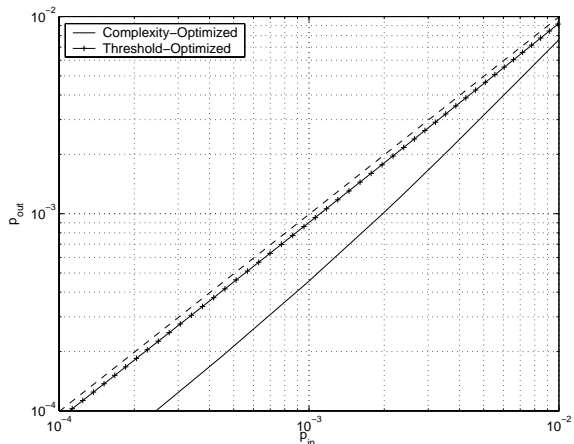


Fig. 7. EXIT charts near origin for complexity-optimized and threshold-optimized rate 0.5 codes on an AWGN channel in log scale

is well known that the minimum distance [24] and the size of the smallest stopping set [25] are strongly dependent on the fraction of degree-two nodes. Therefore, it is not surprising that the complexity-minimized codes do not appear to suffer from error floors, even when the number of decoding iterations is small.

2) *Short Block Length Performance:* Finally, we address the performance of our codes at short block lengths for which density evolution is not necessarily accurate. For short block lengths, it is well known that the performance of LDPC codes is sensitive to the structure of the parity-check matrix. While it is sufficient to randomly construct the parity-check matrices when  $n$  is large, specific algorithms need to be used to design graphs for short block length codes. Most prominent among these is the progressive-edge-growth algorithm [26], which we use in the following to construct codes of length  $n = 1000$ . Once again, we compare the performance of the complexity-minimized codes to that of the threshold-optimized codes.

The BER curves for the  $R = 0.2, 0.5$  and  $0.8$  cases are plotted in Figs. 8, 9 and 10, respectively. It is observed that the relative performance of short-block-length complexity-

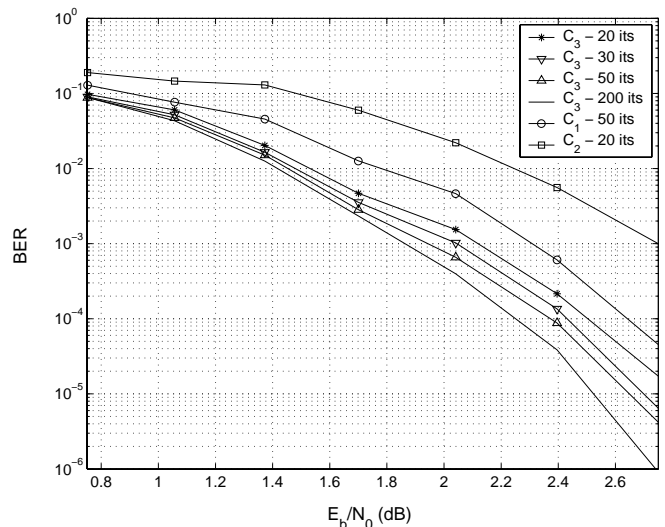


Fig. 8. Performance of short-block-length complexity-optimized codes  $C_1$  (optimized for 50 iterations),  $C_2$  (optimized for 20 iterations), and threshold-optimized code  $C_3$  at various numbers of decoding iterations.  $R = 0.2$ . Block-length  $n = 1000$ .

optimized codes as compared to threshold-optimized codes is code-rate dependent. While at  $R = 0.2$ , the short-block-length complexity-minimized codes seem to be inferior to that of the threshold-optimized codes, the performances of short-block-length complexity-optimized codes at  $R = 0.5$  and  $R = 0.8$  appear comparable to that of threshold optimized codes. (At  $R = 0.5$ , the complexity-optimized code is slightly inferior to the threshold-optimized code for a target complexity of 20 iterations, but is comparable at 50 iterations.) We note that a well-developed theory for short-block-length LDPC codes is still lacking. Most existing methods for designing irregular LDPC codes rely on the accuracy of density evolution, which provides a true description of the decoding process only when  $n \rightarrow \infty$ . Therefore, it is not clear that degree sequences that perform well for long block lengths will also perform well at short block lengths. In particular, the design procedure introduced in this paper relies on the notion of shaping the EXIT charts to achieve some desired decoding trajectory. However, predicting how the shape of such EXIT charts varies as the block length decreases is still an open problem.

## V. CONCLUDING REMARKS

This paper presents a methodology for the design of LDPC codes for optimized complexity-rate tradeoff. Our methodology is based on a new measure of complexity, which accurately models the required number of iterations for convergence in the iterative decoding process, and a novel formulation of the complexity-minimization problem as a convex optimization problem. From the code design examples, it is apparent that our complexity-minimized codes provide, at long block lengths, a superior performance-complexity tradeoff as compared to threshold-optimized codes. The effectiveness of our design methodology stems from the accuracy of our measure of complexity and the convexity of the optimization

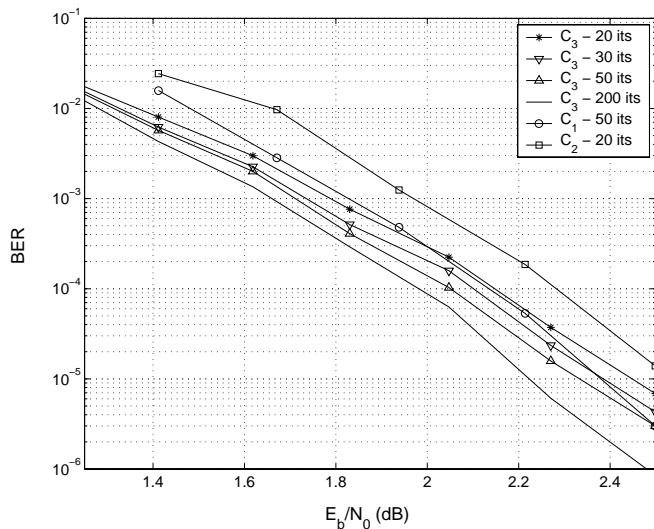


Fig. 9. Performance of short-block-length complexity-optimized codes  $\tilde{C}_1$  (optimized for 50 iterations),  $C_2$  (optimized for 20 iterations), and threshold-optimized code  $C_3$  at various numbers of decoding iterations.  $R = 0.5$ . Block-length  $n = 1000$ .

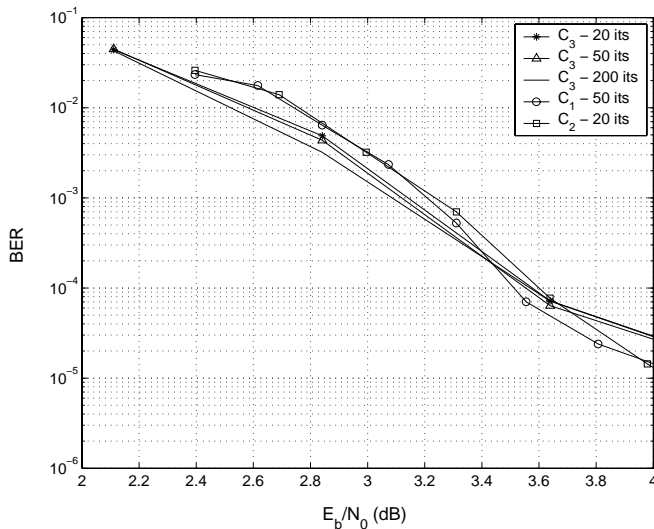


Fig. 10. Performance of short-block-length complexity-optimized codes  $\tilde{C}_1$  (optimized for 50 iterations),  $C_2$  (optimized for 20 iterations), and threshold-optimized code  $C_3$  at various numbers of decoding iterations.  $R = 0.8$ . Block-length  $n = 1000$ .

program, but it also rests upon the accuracy of density evolution analysis. When the block lengths of the codes are such that density evolution provides an accurate prediction of decoding trajectory, our methodology is capable of essentially finding minimum-complexity codes.

In the short-block-length case, Amraoui, et al. [27], [28] recently derived an equation that accurately predicts the performance of LDPC codes at finite block lengths. However, while eliminating the assumption of  $n \rightarrow \infty$ , their analysis nevertheless assumes an unconstrained number of decoding iterations. The accuracy of their method lies in predicting the frequency of ‘bad’ channel events and their contribution to the waterfall region. However, when the number of iterations

is constrained, an additional error mechanism is introduced. With a constrained decoding complexity, in addition to the errors due to atypical channel events, an error due to a premature termination of the decoding process can also occur. It would be interesting to generalize their analysis to the case of a constrained decoder complexity. Such a procedure, if successful, may permit one to design complexity-minimized codes with superior performance even at short block lengths.

## REFERENCES

- [1] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, “Improved low-density parity-check codes using irregular graphs,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.
- [2] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.
- [3] S.-Y. Chung, G. D. Forney Jr., T. J. Richardson, and R. L. Urbanke, “On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit,” *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.
- [4] M. Ardakani, T. H. Chan, and F. R. Kschischang, “EXIT-Chart properties of the highest-rate LDPC code with desired convergence behavior,” *IEEE Commun. Lett.*, vol. 9, no. 1, pp. 52–54, Jan. 2005.
- [5] S. ten Brink, “Convergence behavior of iteratively decoded parallel concatenated codes,” *IEEE Trans. Commun.*, vol. 49, pp. 1727–1737, Oct. 2001.
- [6] S. ten Brink and G. Kramer, “Design of repeat-accumulate codes for iterative detection and decoding,” *IEEE Trans. Signal Process.*, vol. 51, pp. 2764–2772, Nov. 2003.
- [7] S. ten Brink, G. Kramer, and A. Ashikhmin, “Design of low-density parity-check codes for modulation and detection,” *IEEE Trans. Commun.*, vol. 52, pp. 670–678, Apr. 2004.
- [8] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
- [9] M. Ardakani and F. R. Kschischang, “A more accurate one-dimensional analysis and design of LDPC codes,” *IEEE Trans. Commun.*, vol. 52, no. 12, pp. 2106–2114, Dec. 2004.
- [10] A. Shokrollahi, “New sequences of linear time erasure codes approaching the channel capacity,” in *Proc. the International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, no. 1719, Lecture Notes in Computer Science, 1999, pp. 65–67.
- [11] A. Khandekar and R. J. McEliece, “On the complexity of reliable communication on the erasure channel,” in *Proc. IEEE International Symposium on Information Theory*, 2001, p. 1.
- [12] I. Sason and R. Urbanke, “Complexity versus performance of capacity-achieving irregular repeat-accumulate codes on the binary erasure channel,” *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1247–1256, Jun. 2004.
- [13] C.-H. Hsu and A. Anastopoulos, “Capacity-achieving codes with bounded graphical complexity on noisy channels,” in *Proc. Allerton Conf. Commun., Control, Comp.*, Monticello, IL, Sep. 2005.
- [14] X. Ma and E.-h. Yang, “Low-density parity-check codes with fast decoding convergence speed,” in *Proc. IEEE International Symposium on Information Theory*, Chicago, USA, Jun. 2004, p. 103.
- [15] W. Yu, M. Ardakani, B. Smith, and F. R. Kschischang, “Complexity-optimized LDPC codes for Gallager decoding algorithm B,” in *Proc. IEEE International Symposium on Information Theory*, Adelaide, Australia, Sep. 2005.
- [16] R. M. Tanner, “A recursive approach to low complexity codes,” *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.
- [17] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
- [18] T. J. Richardson and R. L. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.
- [19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.
- [20] T. Richardson and R. Urbanke, “Efficient encoding of low-density parity-check codes,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.
- [21] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, “Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

- [22] L. Bazzi, T. J. Richardson, and R. L. Urbanke, "Exact thresholds and optimal codes for the binary-symmetric channel and Gallager's decoding algorithm A," *IEEE Trans. Inf. Theory*, vol. 50, no. 9, pp. 2010–2021, Sep. 2004.
- [23] [Online]. Available: <http://lthcwww.epfl.ch/research/ldpcopt/>
- [24] C. Di, T. J. Richardson, and R. L. Urbanke, "Weight distribution of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 11, pp. 4839–4855, Nov. 2006.
- [25] A. Orłitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 929–953, Mar. 2005.
- [26] E.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, pp. 386–398, Jan. 2005.
- [27] A. Amraoui, A. Montanari, T. Richardson, and R. Urbanke, "Finite-length scaling for iteratively decoded LDPC ensembles," *submitted to IEEE Trans. on Information Theory*, Jun. 2004.
- [28] A. Amraoui, A. Montanari, and R. Urbanke, "Finite-length scaling of irregular LDPC ensembles," *submitted to IEEE Trans. on Information Theory*, Dec. 2005.