

# AN INTRODUCTION TO THE ANALYSIS OF ITERATIVE CODING SYSTEMS

TOM RICHARDSON \* AND RÜDIGER URBANKE †

**Abstract.** This paper is a tutorial on recent advances in the analysis of iterative coding systems as exemplified by low-density parity-check codes and turbo codes.

The theory described herein is composed of various pieces. The main components are concentration of system performance over the ensemble of codes and inputs, the existence of threshold phenomena in decoding performance, and the computational and/or analytical determination of thresholds and its implications in system design.

We present and motivate the fundamental ideas and indicate some technical aspects but proofs and many technical details have been omitted in deference to accessibility to the concepts. Low-density parity-check codes and parallel concatenated codes serve as contrasting examples and as vehicles for the development.

**Key words.** turbo codes, low-density parity-check codes, belief propagation, iterative decoding, stability condition, threshold, output-symmetric channels, Azuma's inequality, support tree

**AMS(MOS) subject classifications.** 94B05

**1. Introduction.** This paper is a tutorial on recent advances in the analysis and design of iterative coding systems as exemplified by low-density parity-check (LDPC) codes and turbo codes. We will outline a mathematical framework within which both of the above mentioned coding systems may be analyzed. Certain aspects of the theory have important practical implications while other aspects are more academic. Provable statements concerning the asymptotic performance of these iterative coding systems can be made. Here, asymptotic refers to the length of the code. For codes of short length the theory we shall present does not yield accurate predictions of the performance. Nevertheless, the ordering of coding systems which is implied by the asymptotic case tends to hold even for fairly short lengths. The *bit error probability* is the natural measure of performance in the theory of iterative coding systems but the theory also offers insights and guidance for various other criteria, e.g., the block error probability.

We will here briefly describe an example and formulate some claims that represent what we consider to be the apex of the theory: Let us consider the class of (3,6)-regular LDPC codes (see Section 2.1.1 for a definition) of length  $n$  for use over an additive white Gaussian noise channel, i.e., we transmit a codeword consisting of  $n$  bits  $x_i \in \{\pm 1\}$  and receive  $n$  values  $y_i = x_i + z_i$  where the  $z_i$  are i.i.d. zero mean Gaussian random

---

\*Bell Labs, Lucent Technologies, Murray Hill, NJ 07974, USA, email: [tjr@lucent.com](mailto:tjr@lucent.com)

†Swiss Federal Institute of Technology – Lausanne, LTHC-DSC, CH-1015 Lausanne, email: [Rudiger.Urbanke@epfl.ch](mailto:Rudiger.Urbanke@epfl.ch)

variables with variance  $\sigma^2$ . We choose the particular code, as determined by its associated particular graph, at random (see Section 2). We transmit one codeword over the channel and decode using the sum-product algorithm for  $\ell$  iterations. The following statements are consequences of the theory:

1. The expected bit error rate approaches some number  $\epsilon = \epsilon(\ell)$  as  $n$  tends to infinity and this number is computable by a deterministic algorithm.
2. For any  $\delta > 0$ , the probability that the actual fraction of bit errors lies outside the range  $(\epsilon - \delta, \epsilon + \delta)$  converges to zero exponentially fast in  $n$ .
3. There exists a maximum channel parameter  $\sigma^*$  (in this case  $\sigma^* \simeq 0.88$ ), the *threshold*, such that  $\lim_{\ell \rightarrow \infty} \epsilon(\ell) = 0$  if  $\sigma < \sigma^*$  and  $\lim_{\ell \rightarrow \infty} \epsilon(\ell) > 0$  if  $\sigma > \sigma^*$ .

Each of these statements generalize to some extent to a wide variety of codes, channels, and decoders. The existence of  $\epsilon(\ell)$  in statement 1 is very general, holding for all cases of interest. In general there may not be any efficient algorithm to compute this number  $\epsilon(\ell)$  but, fortunately, for the case of most interest, namely the sum-product algorithm, an efficient algorithm is known. Statement 2 holds in essentially all cases of interest and depends only on the asymptotics of the structure of the graphs which define the codes. Statement 3 depends on both the decoding algorithm used and the class of channels considered (AWGN). It depends on the fact that the channels are ordered by *physical degradation* (see section 7) and that the asymptotic decoding performance respects this ordering.

Although the threshold, as introduced above in 3, is an asymptotic parameter of the codes, it has proven to have tremendous practical significance. It is only a slight exaggeration to assert that, comparing coding systems with the same rates, the system with the higher threshold will perform better for nearly all  $n$ . The larger  $n$  is, the more valid the assertion is. Even though the assertion is not entirely true for small  $n$ , one can still significantly improve designs by looking for system parameters that exhibit larger thresholds.

To design for large threshold one needs to be able to determine it or at least accurately estimate it. Therefore an important facet of the theory of these coding systems deals with the calculation of the threshold.

In his Ph.D. thesis of 1961, Gallager [13] invented both LDPC codes and iterative decoding. With the notable exceptions of Zyablov and Pinsker [45] and Tanner [40] (see also Surlas [39]), iterative coding systems were all but forgotten until the introduction of turbo codes by Berrou, Glavieux and Thitimajshima [4] (see also [19]). In the wake of the discovery of turbo codes LDPC codes were rediscovered by MacKay and Neal [24], completing the cycle which had started some thirty years earlier.

For some simple cases Gallager was able to determine thresholds for the systems he considered. In the work of Luby *et. al.* [22] the authors used threshold calculations for the binary erasure channel (BEC) and the binary

symmetric channel (BSC) under hard decision decoding to *optimize* the parameters of *irregular* LDPC codes (Gallager had considered only regular LDPC codes) with respect to the threshold. By this approach they showed that very significant improvements in performance were possible. Indeed, they explicitly exhibited a sequence of LDPC code ensembles which, under iterative decoding, are capable of achieving the (Shannon) capacity of the BEC. To date, the BEC is the only non-trivial channel for which capacity achieving iterative coding schemes are explicitly known.

Another important aspect of the work presented in [22] is the method of analysis. The approach differed from that taken by Gallager in certain key respects. The approach of Gallager allowed statements to be made concerning the asymptotics of certain special *constructions* of his codes. The approach of Luby *et. al.*, allowed similar and, in some ways, stronger statements (such as the ones given in our example above) to be made concerning the asymptotics of *random ensembles*, i.e., randomly chosen codes from a given class. This placed the emphasis clearly on the threshold and away from particular constructions and opened the door to irregular codes for which constructions are generally very difficult to find.

In [30] the approach taken in [22] was generalized to cover a very broad class of channels and decoders. Also in [30], an algorithm was introduced for determining thresholds of LDPC codes for the same broad class of channels and the most powerful and important iterative decoder, the sum-product algorithm, also called belief propagation, which is the name for a generalization of the algorithm as independently developed in the AI community by Pearl [28].<sup>1</sup> In [29] the full power of these results was revealed by producing classes of LDPC codes that perform extremely close to the best possible as determined by the Shannon capacity formula. For the additive white Gaussian noise channel (AWGNC) the best code of rate one-half presented there has a threshold within 0.06dB of capacity, and simulation results demonstrate a LDPC code of length  $10^6$  which achieves a bit error probability of  $10^{-6}$ , less than 0.13dB away from capacity. Recent improvements have demonstrated thresholds within 0.012dB of capacity [5]. These results strongly indicate that LDPC codes can approach Shannon capacity. As pointed out above, only in the case of the BEC has such a result actually been proved [23]. Resolving the question for more general channels remains one of the most challenging open problems in the field.

In the case of turbo codes the same general theory applies, although there are some additional technical problems to be overcome. In the setting of turbo codes belief propagation corresponds to the use of the BCJR algorithm for the decoding of the component codes together with an exchange of *extrinsic information*. This is generally known as “turbo decoding”.<sup>2</sup>

---

<sup>1</sup>The recognition of the sum-product algorithm as an instance of belief propagation was made by Frey and Kschischang [12] and also by McEliece, Rodemich, and Cheng [26].

<sup>2</sup>The original incarnation of turbo decoding in [4] was not belief propagation, Robert-

Thus, one can similarly explore turbo codes and generalizations of turbo codes from the perspective of threshold behavior. It is possible to describe an algorithm for computing thresholds for turbo codes but such an algorithm appears computationally infeasible except in the simplest cases. Fortunately, it is possible to determine thresholds to any desired degree of accuracy using Monte-Carlo methods. The putative deterministic algorithm used to compute thresholds can be mimicked by random sampling. One can prove that certain ergodic properties of the computation guarantee convergence of the Monte-Carlo approach (assuming true randomness) to the answer that would have been determined by the exact algorithm. Moreover, all of the information used to optimize LDPC codes is available from the Monte-Carlo approach and, therefore, it is possible to optimize thresholds for various extensions of turbo codes. Generalizations of turbo codes appear to exhibit thresholds approaching Shannon capacity. The work described here appears in [31].

In a very precise sense, determining the threshold by the methods indicated above corresponds to modeling how decoding would proceed on an infinitely long code. One determines not merely the threshold, but the statistics of the entire decoding process. Decoding proceeds in discrete time by passing messages along edges in a graph. In the infinite limit one considers the distribution of the messages (pick an edge uniformly at random, what message is it carrying?) These distributions are parameterized in a suitable fashion and the algorithms mentioned above iteratively update the distributions in correspondence with iterations of the decoding process. The sequence of distributions so obtained and the method used to obtain them are referred to as *density evolution*. Clearly, density evolution is key to understanding the decoding behavior of such systems and the study of density evolution is a fundamental outgrowth of the general theory.

Our purpose in this paper is to provide the reader with a vehicle for quickly grasping the key features, assumptions, and results of the general theory. The paper consists of further details and examples intended to be sufficient to equip the reader with a practical overview of the current state of knowledge as embodied in this theory. The paper is loosely organized along the lines of assumptions and conclusions: Each section is devoted to stating assumptions required for some part of the theory, usually some examples, to stating what conclusions can be drawn, and indicating what mathematical techniques are used to draw the conclusions. We have ordered material from most general to most specific. Rather than attempting to present the most general, all-encompassing, form of the theory, we have tried to make the main ideas as accessible as possible.

**2. Graphical Representations of Codes.** The mathematical framework described in this paper applies to various code constructions based on graphs. To keep notation to a minimum, we will restrict our

---

son [33] refined the algorithm into a form equivalent to belief propagation.

attention in this paper to the standard parallel concatenated code with two component codes and to LDPC codes. For a more detailed treatment of turbo codes which includes also serially concatenated codes and generalized turbo codes we refer the reader to [31]. The theory developed here also extends to much more general graphical representations such as those developed by, e.g., N. Wiberg and H.-A. Loeliger and R. Kötter [44], Kschischang, Frey, Loeliger [21], Kschischang and Frey [20], or Forney [11]. LDPC codes as well as turbo codes can be represented using quite simple graphs and the decoders of interest operate directly and locally on these graphs. (In the case of turbo codes one should bear in mind *windowed* decoding. For a description of windowed decoding see Section 5.2. The theory extends to standard, i.e., non-windowed, turbo decoding by taking limits but the analogy between LDPC codes and turbo codes is closer in the case of windowed decoding.)

The theory addresses *ensembles of codes* and their associated *ensembles of graphs*. For block codes the idea of looking at *ensembles* of codes rather than individual codes is as old as coding theory itself and originated with Shannon. In the setting of turbo codes this idea was introduced by Benedetto and Montorsi [3] and it was motivated by the desire to bound the maximum likelihood performance of turbo codes.<sup>3</sup> The ensembles are characterized by certain fixed parameters which are independent of the length of the code. The graphs associated to LDPC codes, for example, are parameterized by their degree sequences  $(\lambda, \rho)$  (see below for details). The graphs associated to turbo codes are parameterized by the polynomials determining the constituent codes, their interconnection structure, i.e., parallel vrs. serial, and puncturing patterns. Given the size of the code  $n$ , these fixed parameters determine the number of the various node types in the graph. The ensemble of codes is defined in terms of the possible edges which complete the specification of the graph. In both of the above cases, the graphs are bipartite: One set of nodes, the *variable nodes*, corresponds to variables (bits) and the other set, the *check nodes*, corresponds to the linear constraints defining the code.

In general, the fixed parameters and the length  $n$  determine the nodes of the graph and a collection of permissible edges. One then considers the set of all permissible edge assignments and places on them a suitable probability distribution. The theory addresses properties of the ensemble as  $n$  gets large.

**2.1. Ensembles of Codes and Graphs.** We shall now present more detailed definitions for code ensembles of LDPC codes and parallel concatenated codes. These examples, although important, are not exhaustive. Note that the local connectivity of the graphs remains bounded indepen-

---

<sup>3</sup>A considerable amount of additional research has been done in the direction of bounding the maximum likelihood performance of a code from information on its spectrum, see e.g. [10, 42, 34, 35, 9].

dent of the size of the graph. This is the critical property supporting the asymptotic analysis, i.e., the concentration theorem.

**2.1.1. LDPC Codes.** As described above, low-density parity-check codes are well represented by bipartite graphs in which the variable nodes corresponds to elements of the codeword and the check nodes correspond to the set of parity-check constraints satisfied by codewords of the code. *Regular* low-density parity-check codes are those for which all nodes of the same type have the same degree. Thus, a  $(3,6)$ -regular low-density parity-check code has a graphical representation in which all variable nodes have degree three and all check nodes have degree six. The bipartite graph determining such a code is shown in Fig. 1.

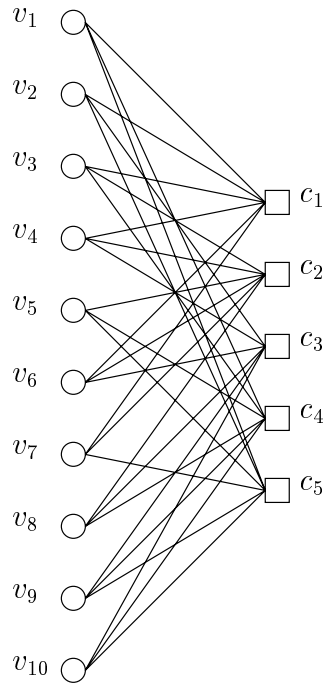


FIG. 1. A  $(3,6)$ -regular code of length 10 and rate one-half. There are 10 variable nodes and 5 check nodes. For each check node  $c_i$  the sum (over  $GF(2)$ ) of all adjacent variable nodes is equal to zero.

For an *irregular* low-density parity-check code the degrees of each set of nodes are chosen according to some distribution. Thus, an irregular low-density parity-check code might have a graphical representation in which half the variable nodes have degree three and half have degree four, while half the constraint nodes have degree six and half have degree eight. For a given length and a given degree sequence (finite distribution) we define an *ensemble* of codes by choosing the edges, i.e., the connections between vari-

able and check nodes, randomly. More precisely, we enumerate the edges emanating from the variable nodes in some arbitrary order and proceed in the same way with the edges emanating from the check nodes. Assume that the number of edges is  $E$ . Then a code (a particular instance of this ensemble) can be identified with a permutation on  $E$  letters. Note that all elements in this ensemble are equiprobable. In practice the edges are never chosen entirely randomly since, e.g., certain potentially unfortunate events, such as double edges and very short loops, in the graph construction can be easily avoided.

Hence, for a given length  $n$  the ensemble of codes will be determined once the various fractions of variable and check node degrees have been specified. Although this specification could be done in various ways the following notation introduced in [22] leads to particularly elegant statements of many of the most fundamental results. Let  $d_l$  and  $d_r$  denote the maximum variable node and check node degrees, respectively, and let  $\lambda(x) := \sum_{i=1}^{d_l} \lambda_i x^{i-1}$  and  $\rho(x) := \sum_{i=1}^{d_r} \rho_i x^{i-1}$  denote polynomials with non-negative coefficients such that  $\lambda(1) = \rho(1) = 1$ . More precisely, let the coefficients,  $\lambda_i$  ( $\rho_i$ ) represent the fraction of *edges* emanating from variable (check) nodes of degree  $i$ . Then, clearly, this *degree sequence pair*  $(\lambda, \rho)$  completely specifies the distribution of the node degrees. The alert reader may have noticed several curious points about this notation. First, we do not specify the fraction of *nodes* of various degrees but rather the fraction of *edges* that emanate from nodes of various degrees. Clearly, it is easy to convert back and forth between this *edge perspective* and a *node perspective*. E.g., assume that half the variable nodes have degree three and half have degree four and that there is a total of  $n$  nodes. Since every degree three node has three edges emanating from it, whereas every degree four nodes has four edges emanating to it we see that there are in total  $1/2 \cdot 3n$  edges which emanate from degree three nodes and that there are in total  $1/2 \cdot 4n$  edges which emanate from degree four nodes. Therefore  $\lambda_3 = \frac{1/2 \cdot 3}{1/2 \cdot 3 + 1/2 \cdot 4} = 3/7$  and  $\lambda_4 = \frac{1/2 \cdot 4}{1/2 \cdot 3 + 1/2 \cdot 4} = 4/7$  so that in this case  $\lambda(x) = 3/7x^2 + 4/7x^3$ . Second, the fraction of edges which emanate from a degree  $i$  node is the coefficient of  $x^{i-1}$  rather than  $x^i$  as one might expect at first. The ultimate justification for this choice comes from the fact that, as we will see later, simple quantities like  $\lambda'(0)$  or  $\rho'(1)$  take on an operational meaning. A particular striking example of the elegance of this notation is given by the *stability condition* which we will discuss in Section 8.3. This condition takes on the form  $\lambda'(0)\rho'(1) < g(\sigma)$  where  $g$  is a function of the channel parameter  $\sigma$  only.

**2.1.2. Turbo Codes.** For every integer  $n$  we define an ensemble of standard parallel concatenated codes in the following manner. We first fix the two rational functions  $G_1(D) = \frac{p_1(D)}{q_1(D)}$  and  $G_2(D) = \frac{p_2(D)}{q_2(D)}$  which describe the recursive convolutional encoding functions. Although the general case does not pose any technical difficulties, in order to simplify notation,

we will assume that all codewords of a convolutional encoder start in the zero state but are not terminated. For  $x \in \{\pm 1\}^n$  let  $\gamma_i(x)$ ,  $i = 1, 2$ , denote the corresponding encoding functions. Then for fixed component codes and a given permutation  $\pi$  on  $n$  letters the unpunctured codewords of a standard parallel concatenated code have the form  $(x, \gamma_1(x), \gamma_2(\pi(x)))$ . Therefore, for fixed component codes and a fixed puncturing pattern there is a one-to-one correspondence between permutations on  $n$  letters and codes in the ensemble. We will assume a uniform probability distribution on the set of such permutations. This is the same ensemble considered in [3] but the present focus is on the analysis of the performance of turbo codes under iterative decoding rather than under maximum likelihood decoding.

The graphical representation of the code contains variable nodes, as in the LDPC code case, and check nodes, which, in this case, represent a large number of linear constraints on the bits associated to the variable nodes. A check node represents the linear constraints imposed by an entire constituent code. Equivalently, it represents the trellis which in turn represents the constituent code. Hence, for standard parallel concatenated codes with two component codes, there are only two check nodes.

### 3. Decoding: Symmetries of the Channel and the Decoder.

In this paper we will limit ourselves to the case of binary codes and transmission over memoryless channels since in this setting all fundamental ideas can be represented with a minimum of notational overhead. The generalization of the theory to larger alphabets [7] or channels with memory [14, 16, 15, 18, 25] is quite straightforward and does not require any new fundamental concepts. As usual for this case, we will assume antipodal signalling, i.e., the channel input alphabet is equal to  $\{\pm 1\}$ .

The decoding algorithms of interest operate directly on the graph, described in Section 2.1, that represents the code. The algorithms are localized and distributed: edges carry messages between nodes and nodes process the incoming messages received via their adjacent edges in order to determine the outgoing messages. The algorithms proceed in discrete steps, each step consisting of a cycle of information passing followed by processing. Generally speaking, computation is memoryless so that, in a given step, processing depends only on the most recent information received from neighboring nodes. It is possible to analyze decoders with memory but we will not consider such decoders here. Given the above setup we distinguish between two types of processing which may occur according to the dependency of the outgoing information. When the outgoing information along an edge depends only on information which has come in along *other* edges, then we say that the algorithm is a *message-passing* algorithm. The sum-product algorithm is the most important example of such an algorithm. The most important example of a non message-passing algorithm is the *flipping* algorithm. This is a very low complexity hard-decision decoder of LDPC codes in which bits at the variable nodes are ‘flipped’ in a given

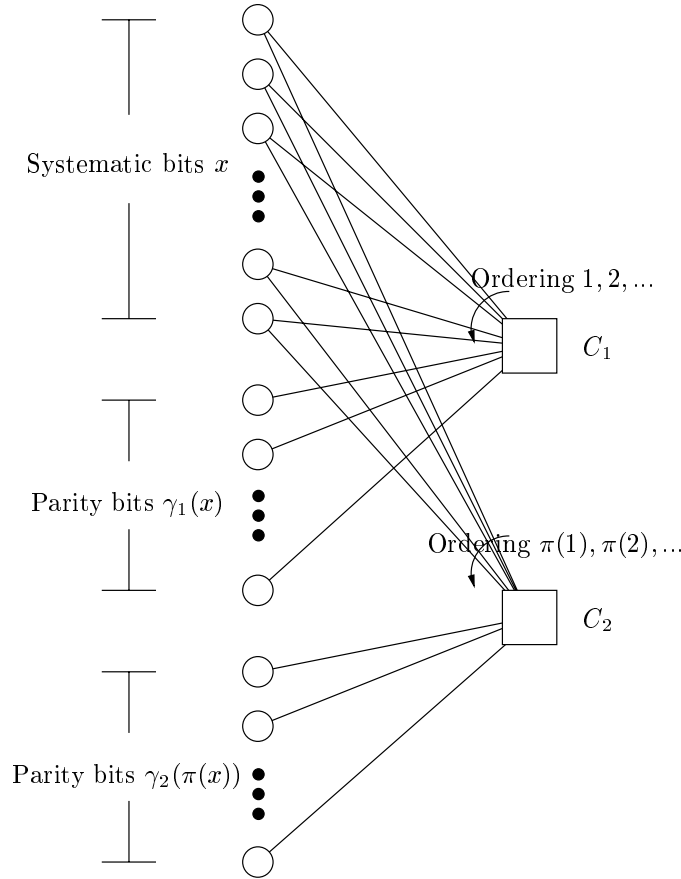


FIG. 2. A graphical representation of a standard parallel concatenated code analogous to the bipartite graph of a LDPC code.

round depending on the number of *unsatisfied* and *satisfied* constraints they are connected to. Here, we say that a constraint is satisfied if and only if the modulo two sum of its neighbor bits is 0. We note that the techniques used to analyze these two types of decoders are quite distinct and the nature of statements which can be made tend to differ significantly. (Nevertheless, certain aspects of the theory outlined here, in particular the concentration results, carry over to many variants of flipping.) The statements usually made for the flipping algorithm are more reminiscent of traditional coding theory in that they assert that for all error patterns of weight less than a given threshold the decoder will decode correctly [45, 38]. For message-passing decoders the overall focus is on the resulting bit error probability. A large fraction of high weight error patterns might be correctable but occasionally even low weight error patterns may lead to errors. Currently

the best coding schemes are based on message-passing decoders but the flipping algorithm is often of interest because of its inherent extremely low complexity.

It is helpful to think of the messages (and the received values) in the following way. Each message which traverses an edge  $(v, c)$ , in either direction, represents an estimate of the particular bit associated to  $v$ . More precisely, it contains an estimate of its *sign* and, possibly, some estimate of its *reliability*. To be concrete, consider a discrete case in which the received alphabet, the alphabet of estimates provided to the decoder from the channel, is  $\mathcal{O} := \{-q_o, -(q_o-1), \dots, -1, 0, 1, \dots, (q_o-1), q_o\}$  and where the message alphabet is  $\mathcal{M} := \{-q, -(q-1), \dots, -1, 0, 1, \dots, (q-1), q\}$  where we assume  $q \geq q_o$ . The sign of the message indicates whether the transmitted bit is assumed to be  $-1$  or  $+1$ , and the absolute value of the message is a measure of the reliability of this estimate. In particular, the value  $0$  represents an erasure. In the continuous case we may assume that  $\mathcal{O} = \mathcal{M} = \mathbb{R}$ . Again, the sign of the message indicates whether the transmitted bit is assumed to be  $-1$  or  $+1$ , and the absolute value of the message is a measure of the reliability of this estimate. Of course, it is not necessary that  $0$  be included in either alphabet, nor is it necessary that the messages actually have this symmetric form, but they should be representable as such.

**3.1. Restriction to the All-One Codeword.** For all algorithms of interest in the current context computation is performed at the nodes based on information passed along the edges. Thus, for every occurring variable degree  $d$  we have a map  $\Psi_{\text{left}}^d : \mathcal{O} \times \mathcal{M}^d \rightarrow \mathcal{M}^d$  to represent the computation done at a variable node of degree  $d$ . Similarly, for every occurring check node of degree  $d$  we have a map  $\Psi_{\text{right}}^d : \mathcal{O} \times \mathcal{M}^d \rightarrow \mathcal{M}^d$  to represent the computation done at a check node of degree  $d$ . (In some cases, i.e., turbo codes, outgoing messages to variable nodes of degree one have no effect on subsequent messages and are therefore usually not actually computed.) Although it is not necessary, one will usually have symmetry under permutation of the input messages so that if  $\Psi_{\text{left}}^d(u_0, u_1, \dots, u_d) = (w_1, \dots, w_d)$  then  $\Psi_{\text{left}}^d(u_0, u_{\pi(1)}, \dots, u_{\pi(d)}) = (w_{\pi(1)}, \dots, w_{\pi(d)})$  for any permutation  $\pi$  on  $d$  letters. In the case of LDPC codes we also have such symmetry for  $\Psi_{\text{right}}$  but for turbo codes, because of trellis termination, we do not.<sup>4</sup>

Under the message-passing paradigm the message maps have the further property that the  $i$ -th outgoing message does not depend on the  $i$ -th incoming message. In this case, and under the permutation symmetry assumption, we have functions  $M_{\text{left}}^d : \mathcal{O} \times \mathcal{M}^{d-1} \rightarrow \mathcal{M}$  such that

$$\Psi_{\text{left}}^d(u_0, u_1, \dots, u_d) = (M_{\text{left}}^d(u_0, u_2, \dots, u_d), M_{\text{left}}^d(u_0, u_1, u_3, \dots, u_d), \dots, M_{\text{left}}^d(u_0, u_1, \dots, u_{d-1})).$$

<sup>4</sup>Asymptotically, however, some permutation symmetry is present in the turbo code case but the permutations must be restricted to, e.g., the information  $(x)$  bits.

Similarly, under permutation symmetry at the check nodes, we shall have a map  $M_{\text{right}}^d : \mathcal{O} \times \mathcal{M}^d \rightarrow \mathcal{M}$ .

Our subsequent analysis and notation will be greatly simplified by assuming the following symmetry conditions on the channel and the decoding algorithm.

- **Channel symmetry:**  $p(Y = q|X = 1) = p(Y = -q|X = -1)$ .
- **Check node symmetry:** If  $\Psi_{\text{right}}^d(u_1, \dots, u_d) = (w_1, \dots, w_d)$  then  $\Psi_{\text{right}}^d(b_1 u_1, \dots, b_d u_d) = (b_1 w_1, \dots, b_d w_d)$  for any  $\pm 1$  sequence  $(b_1, \dots, b_d)$  satisfying the constraint associated to the check node.
- **Variable node symmetry:**

$$\Psi_{\text{left}}^d(-u_0, -u_1, \dots, -u_d) = -\Psi_{\text{left}}^d(u_0, u_1, \dots, u_d).$$

We claim that under these conditions the (bit or block) error probability is independent of the transmitted codeword. For a proof in the message-passing case see [30]. The reader may recognize the channel symmetry condition as the condition that the channel be *output-symmetric*. Note that *linear* codes are known to be capable of achieving capacity for binary-input memoryless output-symmetric channels.<sup>5</sup> This is reassuring, since our search for low complexity coding schemes which can approach capacity takes place in the class of linear codes.

The great simplification which accrues from the symmetry assumption is that one need only determine the probability of error for a single codeword, the all-one codeword.<sup>6</sup> In this case, messages are ‘correct’ when their sign is positive and ‘incorrect’ when their sign is negative. When we consider the distribution of the messages we invoke the all-one codeword assumption so that the probability of an incorrect message is just the probability mass of the distribution supported on negative values. In other words, we are really interested in tracking the distribution of messages *relative to the transmitted codeword*. Under the symmetry assumptions this takes a particularly simple and appealing form.

In the sequel we will assume that the symmetry conditions are fulfilled and that the all-one codeword was transmitted. We will now present some examples of channels and decoders satisfying the above conditions

**EXAMPLE 1** (BSC, LDPC code, Gallager A). *The channel provides only hard information  $\mathcal{O} = \{-1, +1\}$  and the messages are also binary  $\mathcal{M} = \{-1, +1\}$ . The decoding is message passing with  $M_{\text{left}}^d(u_0, u_1, \dots, u_{d-1}) = -u_0$  if  $u_1 = u_2 = \dots = u_{d-1} = -u_0$ , and, otherwise,  $M_{\text{left}}^d(u_0, u_1, \dots, u_{d-1}) = u_0$ . The check node rule is given by  $M_{\text{right}}^d(u_1, \dots, u_{d-1}) = u_1 u_2 \dots u_{d-1}$ .  $\square$*

For this decoder thresholds and optimal codes can be analytically determined in many cases, see [2].

---

<sup>5</sup>The key for the proof of this statement lies in the observation that in this case an input distribution of  $[1/2, 1/2]$  is optimal.

<sup>6</sup>This should be compared to the concept of *geometrically uniform* codes in the setting of maximum likelihood decoding.

EXAMPLE 2 (BEC, LDPC codes, Belief Propagation). *The channel either correctly relays the transmitted bit or this bit is erased. These events happen with probability  $(1 - \delta)$  and  $\delta$ , respectively. Thus, assuming that the output of the channel is represented as log-likelihood ratios we have  $\mathcal{O} = \{-\infty, 0, \infty\}$ . Because of the special structure of this channel we have  $\mathcal{M} = \mathcal{O}$ , where the messages are again represented as log-likelihood ratios. The decoding is message passing with  $M_{\text{left}}^d(u_0, u_1, \dots, u_{d-1}) = \sum_{i=0}^{d-1} u_i$ . Note that this rule is well defined since, by construction, the sum can never contain  $+\infty$  and  $-\infty$  together. The check node rule is given by*

$$M_{\text{right}}^d(u_1, \dots, u_{d-1}) = \prod_{i=1}^{d-1} u_i,$$

where  $0 \cdot \infty = 0$ .  $\square$

EXAMPLE 3 (BSC, LDPC codes, Belief Propagation). *The channel provides only hard information but associated to these values is the reliability magnitude given by  $\log((1 - \epsilon)/\epsilon)$  where  $\epsilon$  is the cross-over probability of the BSC. Thus, we have  $\mathcal{O} = \{-\log((1 - \epsilon)/\epsilon), \log((1 - \epsilon)/\epsilon)\}$ . Messages, like the received values, are log-likelihoods, hence  $\mathcal{M} = \mathbb{R}$ . The decoding is message passing with  $M_{\text{left}}^d(u_0, u_1, \dots, u_{d-1}) = \sum_{i=0}^{d-1} u_i$ . The check node rule is given by  $M_{\text{right}}^d(u_1, \dots, u_{d-1}) = 2 \tanh^{-1}(\prod_{i=1}^{d-1} \tanh(u_i/2))$ . The decoding algorithm described in this example can be applied in general provided the received alphabet is mapped into the associated log-likelihood representation.*

$\square$

EXAMPLE 4 (AWGNC, Turbo Codes, Belief Propagation). *We assume the channel provides the log-likelihood estimate of the bit based on the observation  $y = x + z$  where  $x \in \{\pm 1\}$  is the transmitted bit and  $z$  is zero mean Gaussian with variance  $\sigma^2$ . The log-likelihood estimate is given by  $2\sigma^{-2}y$ . Messages are log-likelihoods so  $\mathcal{O} = \mathcal{M} = \mathbb{R}$ . The decoding is message passing with  $M_{\text{left}}^d(u_0, u_1, \dots, u_{d-1}) = \sum_{i=0}^{d-1} u_i$ . The check node rule is given by APP decoding of the associated constituent code, which corresponds to performing the BCJR algorithm on the trellis and outputting the extrinsic information.*

$\square$

The final example is a flipping type algorithm. Various flipping algorithms have been considered in the literature. Some, such as list-based flipping, do not directly fit into our general description. This is because, in these algorithms, the decision on whether to flip a bit or not is based on a global criterion. Nevertheless, such an algorithm can be mimicked to any degree of accuracy by properly chosen local algorithms so that the analysis can still be applied to such algorithms by taking limits. We present here the flipping algorithm in a somewhat unusual form.

EXAMPLE 5 (BSC, LDPC codes, Flipping). *We assume a  $(d_v, d_c)$ -regular LDPC code and transmission over a BSC. Hence, the channel provides only hard information  $\mathcal{O} = \{-1, +1\}$  and the messages are also binary*

$\mathcal{M} = \{-1, +1\}$ . We are given a threshold  $t$  and

$$\Psi_{left}^{d_v}(u_0, u_1, \dots, u_{d_v}) = (u_0, u_0, \dots, u_0)$$

if  $|\{u_i : u_i = u_0\}| > t$  and

$$\Psi_{left}^{d_v}(u_0, u_1, \dots, u_{d_v}) = (-u_0, -u_0, \dots, -u_0)$$

otherwise. For the check nodes we have

$$\Psi_{right}^{d_c}(u_1, \dots, u_{d_c}) = (u_1 p, u_2 p, \dots, u_{d_c} p)$$

where  $p = \prod_{i=1}^{d_c} u_i$ . □

Note that the check node message map satisfies the message-passing paradigm but the variable node message map does not. In general  $t$  will depend on the iteration number and on the degree of the node. To approximate list based flipping algorithms we might also allow  $t$  to depend on some exogenous random variable.

**4. Decoding: Localization and Concentration.** As we have seen, the coding systems we consider can be represented as graphs on which the decoders directly operate. More precisely, decoding consists of *local* computations at the nodes in the graph together with exchange of information along the edges of the graph. Time is measured discretely so that if decoding proceeds through  $\ell$  time steps then any random variable associated to a node  $v$  in the graph can influence computations at only those nodes whose distance in the graph from the associated node is  $\ell$ , i.e., an influenced node must be able to reach to  $v$  in  $\ell$  steps along the graph. Because of this, and because of the boundedness of the degree of the graph the range of influence of local properties (either of the graph or of received data) remains bounded independent of the size of the graph given a bounded number of iterations.

In this paper we focus mainly on message-passing algorithms although, as pointed out previously, there are decoding algorithms of interest, e.g., the flipping algorithm, which do not fit directly into this framework. Our main reason for this restriction is that the class of message-passing decoders contains the locally-optimal decoder, namely belief-propagation, as well as many other very good and low-complexity decoders. Further, the class of message-passing decoders can be analyzed in a unified manner whereas other decoders, like the flipping algorithm, require quite distinct methods for their analysis. Nevertheless, it is worth pointing out that the concentration theorem, which we will discuss in this section, in its most general form applies to *all* types of local algorithms and not only to message-passing algorithms.

What is the role of the concentration theorem in the context of iterative coding systems? Consider, e.g., the bit error probability for a particular

code and decoding algorithm. Clearly, this is a quantity of great practical significance. Choosing a code from an ensemble amounts to choosing one of the possible random instances for the graph. E.g., in the case of turbo codes a considerable amount of attention has been paid to the issue of choosing a good interleaver. For short lengths different randomly chosen instances of the interleaver may exhibit significantly different behavior. It has been observed, however, that as the codes get longer the variation among interleavers becomes less significant (although it is easy to find bad interleavers). This is a direct consequence of the concentration theorem which asserts that, as  $n$  increases, certain quantities, such as the average decoded bit error probability, *concentrate*: In the probabilistic sense, different instances of the graph and different channel realizations will behave similarly with respect to average quantities. It should be understood that the rate at which this concentration occurs is not well predicted by the current theory: The theory predicts exponential convergence in  $n$  but the actual exponent might differ significantly from the bounds which one derives. To make an analogy, the concentration theorem plays a similar role for iterative coding systems that the asymptotic equipartition theorem plays in much of information theory.

The method of analysis has extremely broad application since it requires very few assumptions. The mathematical technique which leads to the concentration theorem is now virtually standard among probabilistic techniques used to analyze combinatorial problems. In the next section we shall present the technique as applied to graph coloring. For an extended introduction to the iterative decoding application we refer the reader to [32]. The migration of the technique into coding theory was initiated in the work of Luby *et. al.* [22]. They analyzed LDPC codes in an essentially combinatorial setting: transmission over the binary symmetric channel (BSC) or the binary erasure channel (BEC) with hard decision message-passing decoding. In [30] the basic technique, as applied to LDPC codes, was significantly generalized to encompass essentially any message-passing decoding algorithm, including belief propagation, and any memoryless channel.

The general concentration theorem itself consists of two distinct parts. The first part shows convergence of the expected performance to the asymptotic value, the second part shows concentration of the performance around the mean as a function of length. This concentration is further separated into concentration over input noise and concentration over the random ensemble of graphs. In practice it is the concentration over the random ensemble of graphs that is of most interest. Ideally one would like to find the graph that offers the best performance averaged over the channel noise statistics. Simulation performance curves convey precisely this information.

The key asymptotic behavior required for the first part is that, as  $n$  increases, the graphs become more and more locally tree-like, i.e., loop-free. Although it is readily apparent that this holds in the case of LDPC

codes it is not so apparent for the case of turbo codes. In fact, as initially defined, the graphs associated to turbo codes do not exhibit this tree-like asymptotic. Nevertheless, the theory applies to this case because dependencies among variables at the check nodes, i.e., in the trellis, decay with ‘distance’ along the trellis. In the case of windowed decoding this is precise: edges sufficiently well separate participate in non-overlapping disjoint windows and, assuming incoming messages are independent, their outgoing messages will be independent. Since we fix the number of iterations and let  $n$  grow, the expected fraction of nodes whose decoding neighborhood is not tree-like decays as  $n^{-1}$ . Hence, the expected bit error rate converges to that for a random tree.

The key asymptotic behavior required for the second part is that graphs which are strongly similar, i.e., have most edges in common, will perform similarly on the same input. This can be seen most easily as follows. Consider a LDPC code and its associated graph. Imagine altering the graph by swapping the connections of two edges, i.e., replace edges  $(v_1, c_1), (v_2, c_2)$  with  $(v_1, c_2), (v_2, c_1)$ . Now compare results when decoding a particular input. Since decoding proceeds locally and we consider only a finite number of iterations, the effect of the swap is bounded in the graph, limited only to those nodes and variables that are sufficiently close to  $v_1, c_1, v_2$ , and  $c_2$ . The larger the graph, the smaller will be the fraction of the affected portion. If we look at, e.g., the number of errors in the decoding, then the difference between the two graphs is bounded by a constant independent of  $n$ . In the following section we describe the mathematical technique which exploits this property.

**4.1. Doob Martingales and Azuma’s Inequality.** Although we are most interested in applying the mathematical techniques described in this section to iterative decoding, they are most conveniently presented via their application to graph coloring. It was in this application, due to Shamir and Spencer [36], that the method first made its impact in combinatorics.

Consider a random graph on  $n$  vertices in which each possible edge, i.e., vertex pair, is independently admitted with probability  $p$ . Consider the problem of coloring the vertices of a randomly constructed such graph so that no two vertices connected by an edge have the same color. The number of colors required is called the *chromatic number* of the graph  $H$  and is denoted by  $\chi(H)$ . What Shamir and Spencer showed, in effect, is that the chromatic number  $\chi(H)$ , viewed as a random variable, is tightly concentrated around its expected value  $\mathbb{E}[\chi(H)]$ . Perhaps most striking, this result was obtained without the determination of  $\mathbb{E}[\chi(H)]$ .

The argument used is now known as a *vertex exposure* martingale argument. Let  $H$  denote a graph constructed randomly as above. Pick an arbitrary ordering of the  $n$  vertices. Imagine that the graph  $H$  is unknown but that vertices will be *revealed* one at a time. Each time a vertex is revealed all edges connected to previously revealed vertices are revealed, i.e.,

when vertex  $i$  is revealed the presence or absence of the edge  $(j, i)$  for  $j < i$  is revealed. Let  $I_{j,i}$  denote a random variable which indicates whether the edge  $(j, i)$  is present or not. Let  $X_0(H)$  denote the expected chromatic number of  $H$  when nothing has been revealed, i.e.,  $X_0(H) = \mathbb{E}[\chi(H)]$ . Let  $X_i(H)$  denote the expected chromatic number of  $H$  conditioned on the information obtained after revealing the first  $i$  vertices of  $H$ , i.e.,  $X_i(H) = \mathbb{E}[\chi(H) | \{I_{j,k}\}_{1 \leq j < k \leq i}]$ . Thus, in particular,  $X_n(H) = \chi(H)$ . Now, recall that  $H$  is a random variable and therefore  $X_i := X_i(H)$  is a random variable as well (with the randomness residing in the set of random variables  $\{I_{j,k}\}_{1 \leq j < k \leq i}$ ). Note that

$$\begin{aligned} & \mathbb{E}[X_{i+1} | \{I_{j,k}\}_{1 \leq j < k \leq i}] \\ &= \mathbb{E}[\mathbb{E}[\chi(H) | \{I_{j,k}\}_{1 \leq j < k \leq i+1}] | \{I_{j,k}\}_{1 \leq j < k \leq i}] \\ &= \mathbb{E}[\chi(H) | \{I_{j,k}\}_{1 \leq j < k \leq i}] \\ &= X_i \end{aligned}$$

so that the sequence  $\mathbb{E}[\chi(H)] = X_0, X_1, \dots, X_n = \chi(H)$  forms a martingale. More precisely, a martingale constructed this way, as sequence of conditional expectations where the information conditioned on is increasing, is usually called a *Doob's Martingale* [27, p. 90]. If we take the graph  $H$  and modify the edges involving the  $i$ -th vertex in an arbitrary way to obtain a graph  $G$ , then  $|\chi(H) - \chi(G)| \leq 1$ : given a coloring of  $H$  we need at most one more color to color  $G$  and vice versa. It follows that  $|X_i(H) - X_{i-1}(H)| \leq 1$  since  $X_{i-1}(H)$  is an average of all possible values for  $X_i(H)$ , hence,  $|X_i - X_{i-1}| \leq 1$ .

In the case of a random walk,  $X_i = \sum_{j=1}^i Z_j$  where the  $Z_j$  are independent random variables with finite expectation, we can use the Chernoff bound to give an exponential bound on the probability that  $X_i$  will deviate from its mean by more than a fraction  $\epsilon$ . Azuma's inequality provides a similar bound for the case of dependent random variables assuming that the sequence of these random variables forms a martingale.

**THEOREM 4.1** (Azuma's Inequality). *Let  $X_0, X_1, \dots$  be a martingale sequence such that for each  $k \geq 1$ ,*

$$|X_k - X_{k-1}| \leq \alpha_k,$$

where the constant  $\alpha_k$  may depend on  $k$ . Then, for all  $l \geq 1$  and any  $\lambda > 0$

$$\Pr\{|X_l - X_0| \geq \lambda\} \leq 2e^{-\frac{\lambda^2}{2 \sum_{k=1}^l \alpha_k^2}}.$$

The sequence  $X_0, \dots, X_n$  constructed above satisfies the conditions required by Azuma's inequality with  $\alpha_k = 1$ . Since  $X_0(H) = \mathbb{E}[\chi(H)]$  and  $X_n(H) = \chi(H)$ , we obtain

$$\Pr[|\chi(H) - \mathbb{E}[\chi(H)]| > \lambda \sqrt{n-1}] \leq 2e^{-\lambda^2/2}.$$

This is Shamir and Spencer's result [36]. For further applications see [27, 1].

**4.1.1. Application to Iterative Decoding.** The application of the technique described above to iterative decoding involves some additional complexities but follows essentially the same lines of argument. For both LDPC codes and turbo codes, once the nodes are fixed, the ensemble of graphs is in one-to-one correspondence with the set of permutations on  $m$  letters for some  $m$ . A Doob's martingale is formed: conceptually, we reveal the permutation one element at a time. In the case of LDPC codes this corresponds to revealing one edge in the graph. In the case of turbo codes this corresponds to revealing a pair of edges in the graph. As the permutation is revealed we look at the expected number of decoding errors. The martingale may be extended by revealing received values after the graph is fully revealed. In this way one can prove concentration not only of average performance but also concentration of performance on individual inputs.

When revealing the graph, the bound required for Azuma's inequality is derived using the swapping argument indicated earlier. Different possibilities for the revealed edges are compared by identifying the continuations of the revelation process through swapping. In this way the impact of the revealed edge can be shown to be bounded. When revealing the received values the argument is easier. Since we consider a finite number of iterations, the effect of any received value is restricted to a finite portion of the graph independent of  $n$ .

Let  $Z_\ell[n]$  be the random variable denoting the fraction of incorrect messages passed in the  $\ell$ -th iteration of a randomly chosen code and set of received values for a code of length  $n$ . For either LDPC codes or turbo codes one obtains a theorem of the following form.

**THEOREM 4.2.** *For any  $\ell$  there exists a constant  $\beta$  such that the probability that  $|Z_\ell[n] - \mathbb{E}[Z_\ell[n]]| > \epsilon$  is less than  $e^{-\beta\epsilon^2 n}$ . Furthermore,  $Z_\ell[\infty] := \lim_{n \rightarrow \infty} \mathbb{E}[Z_\ell[n]]$  exists and there exists a constant  $\gamma$  such that  $|\mathbb{E}[Z_\ell[n]] - Z_\ell[\infty]| < \frac{\gamma}{n}$ .*

**5. The Support Tree of Message-Passing Decoders.** In anticipation of the analysis of message-passing algorithms, we shall describe in some detail the dependency structure of messages in the graphs of interest under the message-passing paradigm. The extension to non-message-passing is straightforward and we shall not consider it in detail.

Consider a message passed along an edge during a message-passing decoding. Depending on the iteration number this message will, in general, be a function of some subset of the received values and some sub-graph of the graph defining the code. The subset of received values on which the message depends are those associated to variable nodes of the sub-graph. Thus, we can view the message simply as a function of the sub-graph. Note that the sub-graph depends only on the iteration number and not on the particular message-passing algorithm used. (In the case of turbo codes one should bear in mind windowed decoding of width  $W = 2w + 1$ .) Since the

message traverses the edge in a particular direction the sub-graph depends on this direction. If the iteration number is  $\ell$  then the sub-graph is called the  $\ell$ -directed neighborhood of the (directed) edge.

By proceeding backwards in time, one can ‘unvolve’ the directed neighborhood of the edge in terms of iteration number. Described in more detail below, the uninvolved graph can be conveniently pictured as a tree and we refer to such a tree as a support tree. In general the support tree is not actually a tree because certain elements of the tree, i.e., nodes or edges, may be replicated.

If there are no repetitions in the support tree, then we say that the corresponding  $\ell$ -directed neighborhood of the edge is a *tree*. In all cases of interest, as  $n$  increases the probability that the directed neighborhood is a tree approaches one at a rate of  $O(1/n)$ . The underlying reason for this property is the boundedness of the degrees of the nodes in the graph. To see this: imagine picking a node and revealing its neighbors up to distance  $\ell$  in order of increasing distance, then, at any point there will be at most a constant number of edge placements which will create a loop. The total number of possible edge placements grows linearly in  $n$  and the total number of edges to be revealed is bounded by a constant independent of  $n$ . Since the connections are chosen randomly it follows that the probability of creating a loop decays like  $1/n$ .

**5.1. Support Trees of LDPC Codes.** For simplicity we consider a  $(d_v, d_c)$ -regular LDPC code. Consider a randomly chosen edge  $e = (v, c)$  from the graph and consider the message passed along  $e$  from  $v$  to  $c$  in the  $\ell$ -th iteration under message-passing decoding. This message is a function of the received value at  $v$  and the messages arriving at  $v$  in the  $(\ell - 1)$ -th iteration along its *other* edges. These other edges are connected to some collection of constraint nodes. For each such constraint node and connecting edge the message sent to  $v$  along that edge in the  $(\ell - 1)$ -th iteration is a function of the messages sent to this constraint node along its *other* edges. Continuing in this recursive fashion we can determine the support tree of the original message under consideration by uninvolving the graph, tracing back the dependency of the message on previous messages. Fig. 3 gives a pictorial representation of the support tree for a  $(3, 6)$  LDPC code over one and a half iterations.<sup>7</sup> Given the appropriate labelling of the nodes, this picture represents the sub-graph of the original graph on which the message passed from  $v$  to  $c$  depends.

If we write the directed neighborhood in the form of a support tree rooted at  $e$ , as in Figure 3, then, under message passing, information flows up the tree from the leaves to produce the message carried along  $e$ . Messages are passed up the tree: a message from a node is a function only of messages which come from below and, in the case of variable nodes, the received value

---

<sup>7</sup>Although depicted as a tree, some of the nodes and edges in the uninvolved picture may be identical to other nodes and edges.

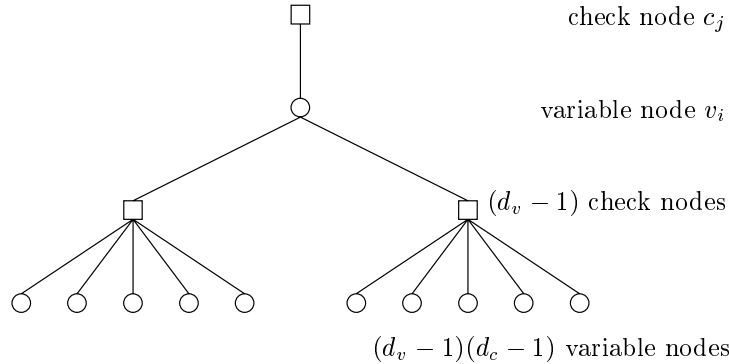


FIG. 3. The directed neighborhood of depth 2 of the edge  $e = (v_i, c_j)$  for an LDPC code.

associated to the node.

In general, for fixed  $n$  the directed neighborhoods of an edge will contain repetitions (nodes and/or edges). As pointed out above, for any fixed  $\ell$  and, assuming a randomly chosen edge  $e$ , the probability that some repetition does occur tends to zero as  $n$  increases at the rate of  $n^{-1}$ . Hence, asymptotically the behavior of the decoder after  $\ell$  iterations is equivalent to the behavior of the messages on a true tree.

In the case of regular LDPC codes the support trees of different edges look the same. In the case of irregular LDPC codes the support trees vary because of the different possible degrees of the nodes. In this case one considers the distribution on trees obtained by picking an edge from the graph uniformly at random, and, in effect, averages over this distribution. How can this averaging be accomplished? Consider choosing uniformly at random an edge  $e = (v, c)$  and developing its directed neighborhoods for increasing  $\ell$ . The degree of the variable node  $v$  is distributed according to  $\lambda$ , i.e., the node  $v$  has  $i - 1$  children with probability  $\lambda_i$ . Each of the check nodes connected to the variable node  $v$  has a degree which is distributed according to  $\rho$ , i.e., each such check node has  $i - 1$  children with probability  $\rho_i$ , and so on.

**5.2. Support Trees of Turbo Codes under Windowed Decoding .** A standard technique for the implementation of Viterbi decoders is to limit the trace-back to a fixed window size [41, p. 258]. Similarly, the Bahl algorithm can be modified to work within a fixed window size and it is well-known that for a large enough window size the resulting loss in performance is negligible. Hence, to decode the  $i$ -th systematic bit we apply a window of length  $W = 2w + 1$  symmetrically around the  $i$ -th bit. We locally construct the trellis of length  $W$  and initialize the Bahl algorithm by assigning the end states a uniform probability. We now run the forward, backward and combining iteration as for the standard Bahl algorithm and



as  $w$  tends to infinity to the standard turbo decoding except that each appearance of the trellis can be assumed to have *independent* realizations of the received values, and, for any fixed number of iterations, the performance of standard turbo decoding is indistinguishable from this in the infinite code length limit.

Therefore, in the analysis of the asymptotic performance of turbo decoding one may assume that the window sizes are in fact infinite and that there are no repetitions in the support tree.

**6. Density Evolution.** In this section we consider the evolution of message distributions as the messages are passed up a support tree. As we have indicated, this amounts to determining the distribution of messages when decoding an infinitely long code. Only for certain message-passing algorithms do we presently know efficient algorithms to determine these distributions. Fortunately, the cases of greatest interest are among them.

**6.1. Density Evolution for LDPC Codes.** Let us imagine how we can determine the distribution of messages passed by a message-passing decoder on an infinitely large  $(d_v, d_c)$ -regular LDPC code.

Let  $M_{\text{left}}^{d_v} : \mathcal{O} \times \mathcal{M}^{d_v-1} \rightarrow \mathcal{M}$  denote the left (variable node) message map and let  $M_{\text{right}}^{d_c} : \mathcal{M}^{d_c-1} \rightarrow \mathcal{M}$  denote the right (check node) message map where, we recall,  $\mathcal{O}$  denotes the alphabet of the received values and  $\mathcal{M}$  denotes the message alphabet. For simplicity we shall assume that the maps are independent of the iteration number and that the initial messages sent out by the variables nodes are equal to their received values.

The received values given to the decoder have distribution  $P_0$ , hence this is the distribution of the messages initially sent out from the variable nodes to the check nodes. Now, choose an edge at random and consider the message sent along it from its neighboring check node  $c$ . That message is  $M_{\text{right}}^{d_c}(u_1, \dots, u_{d_c-1})$  where  $(u_1, \dots, u_{d_c-1})$  are the messages which came in along the other edges connected to  $c$ . The probability of a repetition among  $(u_1, \dots, u_{d_c-1})$  is zero, so the messages  $u_i$  are i.i.d. with distribution  $P_0$ . Let  $R_1$  denote the distribution of  $M_{\text{right}}^{d_c}(u_1, \dots, u_{d_c-1})$ . Now consider the distribution of messages sent in the next round from variable nodes to check nodes. Again, pick an edge at random and let  $v$  be its incident variable node. Then the message sent along this edge is given by  $M_{\text{left}}^{d_v}(u_0, u_1, \dots, u_{d_v-1})$  where  $(u_1, \dots, u_{d_v-1})$  are the messages which came in along the other edges connected to  $v$  and  $u_0$  is the received value for  $v$ . Let us consider the check nodes  $c_1, \dots, c_{d_v-1}$  from which the messages  $(u_1, \dots, u_{d_v-1})$  originate. With probability one they are disjoint, i.e., there are no repetitions. Let us consider for each of them their other  $d_c - 1$  variable node neighbors. With probability one there are no repetitions among all of these neighbors. Thus,  $(u_1, \dots, u_{d_v-1})$  are i.i.d. with distribution  $R_1$ . Let  $P_1$  denote the distribution of  $M_{\text{left}}^{d_v}(u_0, u_1, \dots, u_{d_v-1})$  where  $(u_1, \dots, u_{d_v-1})$  are i.i.d. with distribution  $R_1$  and  $u_0$  is independent with distribution  $P_0$ . More generally, let us recursively define distributions

$P_\ell, R_\ell, \ell = 1, 2, \dots$  by setting  $R_\ell$  to be the distribution of  $M_{\text{right}}^{d_v}(u_1, \dots, u_{d_c-1})$  where  $u_1, \dots, u_{d_c-1}$  are i.i.d. with distribution  $P_{\ell-1}$ , and by setting  $P_\ell$  to be the distribution of  $M_{\text{left}}^{d_v}(u_0, u_1, \dots, u_{d_v-1})$  where  $u_1, \dots, u_{d_v-1}$  are i.i.d. with distribution  $R_\ell$ , and  $u_0$  is independent with distribution  $P_0$ . Then  $P_\ell$  is the distribution of messages sent from variable nodes to check nodes in iteration  $\ell$  and  $R_\ell$  is the distribution of messages sent from check nodes to variable nodes in iteration  $\ell$ .

Now, fix the number of decoding iterations to be  $L$  and consider ensembles of finite graphs. For each  $n$  there will be distributions for the messages passed in the graph, call them  $R_\ell^{(n)}$  and  $P_\ell^{(n)}$ . Not surprisingly the distributions  $R_\ell^{(n)}, P_\ell^{(n)}$  converge (in an appropriate sense) to  $R_\ell, P_\ell$  for  $\ell < L$  as  $n$  tends to infinity. The concentration results from the preceding section further imply that if one picks at random a code from the ensemble and a set of received values distributed according to  $P_0$  then the distribution of the messages in the particular decoding of that particular received sequence will be ‘close to’<sup>8</sup>  $R_\ell, P_\ell$  for  $\ell < L$  with high probability as  $n$  tends to infinity.

The sequence of distributions  $R_\ell, P_\ell$  and their determination is collectively referred to as *density evolution*. The generalization to irregular LDPC codes is straightforward. In this case, when we pick an edge at random the distribution of the degree of its neighboring constraint node  $c$  is given by  $\rho$ . Thus, if  $R_\ell[k]$  denotes the return message distribution assuming  $c$  has degree  $k$ , then  $R_\ell$  is given by  $R_\ell = \sum_k \rho_k R_\ell[k]$ . Similarly, when we pick an edge at random its neighboring variable node  $v$  has degree distributed according to  $\lambda$  and, if  $P_\ell[k]$  denote the return message distribution assuming  $c$  has degree  $k$ , then  $P_\ell$  is given by  $P_\ell = \sum_k \lambda_k P_\ell[k]$ .

EXAMPLE 6 (BSC, Gallager A). Recall Gallager’s decoding algorithm A described in Example 1. For this decoder the message density can be represented by a single real variable  $x$ , namely the probability of error. Under the all-one codeword assumption we identify  $x$  with the probability of sending a minus-one. Let  $\epsilon$  denote the cross-over probability of the BSC. If  $x$  is the probability of a minus-one message entering a variable node of degree  $d$ , then the probability that a minus-one is passed up the tree from that node is given by  $\epsilon(1 - (1 - x)^{d-1}) + (1 - \epsilon)x^{d-1}$ . If  $x$  is the probability of a minus-one message entering a check node of degree  $d$ , then the probability that a minus-one is passed up the tree from that node is given by  $\frac{1}{2}(1 - (1 - 2x)^{d-1})$ . Let  $x_\ell$  denote the probability of a minus-one being passed up from a variable node to a check node in the  $\ell$ -th iteration where  $x_0 = \epsilon$ . Then, given an LDPC code with degree sequence pair  $(\lambda, \rho)$ , we obtain for  $\ell \geq 1$

$$x_\ell = \epsilon(1 - \lambda(\frac{1 + \rho(1 - 2x_{\ell-1})}{2})) + (1 - \epsilon)\lambda(\frac{1 - \rho(1 - 2x_{\ell-1})}{2}).$$

---

<sup>8</sup>The appropriate metric will depend on, e.g., the message alphabets.

**6.2. Density Evolution for LDPC Belief Propagation.** In this section we consider the important special case of density evolution for LDPC codes and belief propagation decoding. We start with the simplest example, namely belief propagation for the BEC.

EXAMPLE 7 (BEC, Belief Propagation Decoding). *Consider the belief propagation decoder for a BEC described in Example 2. In this case the message density can be represented by a single real variable  $x$ , the probability of erasure. Let  $\delta$  denote the probability of erasure in the channel. If  $x$  is the probability of an erasure for messages entering a variable node of degree  $d$ , then the probability that an erasure is passed up the tree from that node is given by  $\delta x^{d-1}$ . If  $x$  is the probability of an erasure for messages entering a check node of degree  $d$ , then the probability that an erasure is passed up the tree from that node is given by  $1 - (1 - x)^{d-1}$ . Let  $x_\ell$  denote the probability of an erasure being passed up from a variable node to a check node in the  $\ell$ -th iteration where  $x_0 = \delta$ . Then given a LDPC code with degree sequence  $(\lambda, \rho)$  we obtain  $x_\ell = \delta\lambda(1 - \rho(1 - x_{\ell-1}))$  for  $\ell \geq 1$ .*

The above example is uncharacteristically simple due to the special nature of the channel. At any point in the decoding algorithm there are only two possible states for a given bit: it can either be known (with infinite confidence) or its log-likelihood ratio is equal to 0. It is for that reason that the progress of the system can be characterized by a single real variable, namely the fraction of erasure messages.

The general scenario is more complicated. Recall that for belief propagation we may represent input and output messages as log-likelihood ratios,

$$\log \frac{p(y|x = 1)}{p(y|x = -1)},$$

where  $y$  represents all the observations, including the received value in the case of variable nodes, conveyed to the node at that time. In general, message distributions are probability distributions on the two-point compactification of  $\mathbb{R}$ , which we denote by  $\bar{\mathbb{R}}$  (we require  $\pm\infty$ .) To simplify the presentation we shall generally assume that distributions are smooth and can therefore be represented by their densities.<sup>9</sup>

At a check node, i.e., internal to the computation at check nodes, it is actually more convenient, see [13, p. 46], to represent the input and output messages as a tuple  $(s, r)$  where  $s \in \mathbb{F}_2 := \{\pm 1\}$  indicates the associated hard decision and where  $r \in \mathbb{R}^+$  is equal to

$$\left| \log |p(x = 1|y) - p(x = -1|y)| \right|.$$

Here  $\mathbb{F}_2$  denotes the two-element group with elements  $\{\pm 1\}$  and group

---

<sup>9</sup>A couple of exceptions will be made for certain discrete measures. In particular, we shall require the distribution  $\Delta_\infty$ , the ‘delta function at infinity.’ We shall also require the distribution  $\Delta_0$  the ‘delta function at zero,’ corresponding to an erasure.

operation equal to real multiplication. Evidently, there is a straightforward transformation from one representation to the other.

Assume now that the received message from the channel and the incoming messages at a variable node are given in log-likelihood form. The outgoing message along an edge  $e$  is then simply the sum of the received message plus all incoming messages, excluding the message incoming along edge  $e$ . Therefore, assuming independence (tree assumption), the *density* of the outgoing message is the convolution of the densities of the messages participating in this sum.

An equivalent statement holds for the check nodes. Assume that the incoming messages to a check node are all in the form  $(s, r)$ . The outgoing message (again in  $(s, r)$  representation) along an edge  $e$  is then simply the sum of all incoming messages, excluding the message incoming along edge  $e$  (here “addition” is performed component-wise, i.e.,  $(s_1, r_1) + (s_2, r_2) = (s_1 * s_2, r_1 + r_2)$ , where the first component is in  $\mathbb{F}_2 = \{\pm 1\}$  and “addition” corresponds to real multiplication, and the second component is an element of  $\mathbb{R}$  with regular addition.) Thus, as in the previous case, the density of the outgoing message, over  $\mathbb{F}_2 \times \bar{\mathbb{R}}^+$ , can be written as the convolution of the corresponding densities of the incoming messages. Here,  $\bar{\mathbb{R}}^+$  refers to  $[0, \infty]$  the compactification of  $\mathbb{R}^+$ . To complete the description we need only specify the change of variables necessary to convert a density from one representation to the other and to incorporate the degree sequences.

To obtain the distribution of messages from check nodes back to variable nodes, we must perform a change of measure. To this end we define the operator  $\gamma$  which takes the space of distributions on  $\bar{\mathbb{R}}^+$  into itself. For densities on  $\mathbb{R}^+$  we define  $\gamma$  as

$$(6.1) \quad \forall y \geq 0: \quad \gamma(f)(y) := f(\ln \coth y/2) \operatorname{csch}(y).$$

The operator  $\gamma$  represents a change of variables  $y \rightarrow \ln \coth y/2$ , i.e.,  $f(y) dy = \gamma(f)(y) dy$ , since  $\frac{d}{dy} \ln \coth y/2 = -\operatorname{csch}(y)$ .

For any distribution  $f$  on  $\bar{\mathbb{R}}$  let  $f^-$  denote  $1_{[-\infty, 0]} f$  and let  $f^+$  denote  $1_{[0, \infty]} f$ , where  $1_A$  denotes the characteristic function of  $A$  so that  $f = f^+ + f^-$ .<sup>10</sup>

Given a distribution  $f$  over  $\bar{\mathbb{R}}$  of log-likelihoods  $l$ , we can represent it as a distribution over  $\mathbb{F}_2 \times \bar{\mathbb{R}}^+$  by representing  $l$  as the pair  $(s, r)$ . We therefore define  $\Gamma$ , the change of variable operator, by

$$\Gamma(f)(1, r) = \gamma(f^+)(r) \text{ and } \Gamma(f)(-1, r) = \gamma(\mathcal{R}f^-)(r).$$

where  $\mathcal{R}f^-$  denotes the reflection of  $f$  about 0, i.e.,  $\mathcal{R}f^-(x) = f^-(-x)$ .

Now, let  $g$  be a distribution over  $\mathbb{F}_2 \times \bar{\mathbb{R}}^+$ . Let  $g^+$  and  $g^-$  be defined by  $g^+(r) = \delta_{\{1\}}(s)g(s, r)$  and  $g^-(r) = \delta_{\{-1\}}(s)g(s, r)$ . Then the inverse of

<sup>10</sup>In the general case, where a point mass at 0 is allowed, the mass at 0 should be split equally between  $f^-$  and  $f^+$ .

$\Gamma$  is given by

$$\Gamma^{-1}(g)(l) = \gamma(g^+)(l) + \mathcal{R}\gamma(g^-)(l).$$

We have now explicitly constructed the change of measure operator for both directions.

Let  $P_0$  be the distribution of the received log-likelihoods and  $P_\ell$  denote the distribution of log-likelihoods sent from the variable nodes to the check nodes in the  $\ell$ -th iteration. Let  $R_\ell$  denote the distribution of log-likelihoods sent from the check nodes to the variable nodes in the  $\ell$ -th iteration. We may initialize with  $R_0 = \Delta_0$ .

The distribution of messages passed from a variable node of degree  $d_v$  to a check node in the  $\ell$ -th iteration is given by the convolution  $P_\ell[d_v] := P_0 \otimes R_{\ell-1}^{\otimes d_v - 1}$ , where, here and in the sequel,  $\otimes$  denotes convolution and  $R^{\otimes m}$  denotes  $R$  convolved with itself  $m$  times.

Suppose that we have a graph with left and right edge degree distributions given by  $\lambda(x) = \sum_{i=1}^{d_\ell} \lambda_i x^{i-1}$ , and  $\rho(x) = \sum_{i=1}^{d_r} \rho_i x^{i-1}$ , respectively. We follow [22] and use the following convention: for a polynomial  $q(x) = \sum_i q_i x^{i-1}$  with non-negative real coefficients  $q_i$  and  $q(1) = 1$  we denote by  $q(f)$  the distribution  $\sum_i q_i f^{\otimes(i-1)}$ .

**THEOREM 6.1 (Density Evolution).** *Let  $P_0$  denote the initial message distribution, under the assumption that the all-one codeword was transmitted, of a low-density parity-check code specified by edge degree distributions  $\sum_i \lambda_i x^{i-1}$  and  $\sum_i \rho_i x^{i-1}$ . If  $R_\ell$  denotes the density of the messages passed from the check nodes to the variable nodes at round  $\ell$  of the belief-propagation, with  $R_0 := \Delta_0$ , then we have*

$$R_\ell = \Gamma^{-1} \rho(\Gamma(P_0 \otimes \lambda(R_{\ell-1}))).$$

Note that the above indicated computations consist of two main components: convolutions and change of measures. It should therefore not be surprising that density evolution can efficiently be computed by means of Fourier transform based methods.

**6.3. Density Evolution for Turbo Code Belief Propagation.**

Although one can consider density evolution for other decoders of turbo codes, we will describe the process only for the belief propagation decoder as applied to standard parallel concatenated codes.

We consider the distribution of messages as they are passed up the support tree as described in Section 5.2. One can view the window  $w$  as fixed but it is appropriate to consider the limit as  $w \rightarrow \infty$  since the resulting distributions converge in this limit and we thereby remove the ancillary variable  $w$ .

Imagine a very long trellis and suppose that the systematic variables have distribution  $P_\ell$  and the parity-check variables have distribution  $P_0$ . In

the forward-backward decoding algorithm one computes one-sided conditional densities for states in the trellis. Usually denoted by  $\alpha$  and  $\beta$ , these vectors are recursively computed from the two terminal ends of the trellis and represent conditional distributions of the states in the trellis conditioned on the trellis section to the ‘left’ and the trellis section to the ‘right’ respectively. In our current view, the inputs to the trellis are random variables and hence so are the  $\alpha$  and  $\beta$  vectors. Under very general hypotheses one can show that the distributions of the  $\alpha$  and  $\beta$  vectors converge to a unique steady state distribution (depending on  $P_\ell$  and  $P_0$ ) regardless of the trellis termination. (In the case of periodic puncturing of the parity bits one has convergence to a periodic sequence of distributions.) These steady state distributions thereby induce steady state distributions for the extrinsic information which will be returned from the trellis. (In the case of periodic puncturing of the parity bits one has a periodic sequence of extrinsic distributions but if the interleaver ignores this additional structure then the distributions can be mixed into a single one for purposes of the asymptotic analysis.) Thus, in the limit of infinite length, one has a well-defined map  $\Theta$  which takes pairs of input distributions, one for the systematic bits and one for the parity bits, into the output distribution of the extrinsic information. Let  $R_\ell$  denote the distribution of extrinsic information returned to the systematic variable nodes at the  $\ell$ -th (half) iteration. Then  $R_\ell = \Theta(P_{\ell-1}, P_0)$ .

In principle the steady state distributions of  $\alpha$  and  $\beta$  can be directly computed. Unfortunately, the computational complexity of such an approach renders practically infeasible. For memory four, for example, the vector  $\alpha$  is fifteen (16-1) dimensional. Even just representing an arbitrary probability distribution in fifteen dimensional space with sufficient accuracy seems hopeless. Of course, the distributions which arise are not arbitrary but there seems to be no obvious way to exploit the constraints which exist. Thus, in practice one estimates the distribution  $R_\ell$  by using Monte-Carlo simulation of a very long trellis. Certain structural conditions on the form of  $R_\ell$  contribute to make such an approach highly practical (see Section 8.1.)

The update of distributions at the variable nodes is identical to that for LDPC codes. In the case of parallel concatenated codes it is particularly simple. Assuming a log-likelihood representation for the messages, the distribution  $P_\ell$  returned from the systematic check nodes is simply given by  $P_\ell = R_\ell \otimes P_0$ . Thus, one has

$$P_\ell = P_0 \otimes \Theta(P_{\ell-1}, P_0).$$

**7. Monotonicity and Thresholds.** Assume we are given a class of channels fulfilling the required symmetry condition and that this class is parameterized by  $\alpha$ . This parameter may be real valued as is the case for, e.g., the cross-over probability  $\epsilon$  for the BSC and the standard deviation  $\sigma$  for the AWGNC, or it may take values in a different domain. For a fixed

parameter  $\alpha$  we can use the above algorithm to determine if, for a given code, the fraction of incorrect messages tends to zero with an increasing number of (loop free) iterations.

In many cases the parameter  $\alpha$  actually reflects a natural ordering of the channels – the capacity decreases with increasing parameter  $\alpha$ . It is therefore natural to ask in such cases whether convergence to zero of the probability of error for a parameter  $\alpha'$  automatically implies convergence to zero of the probability of error for every parameter  $\alpha$  such that  $\alpha \leq \alpha'$ . More generally, we might want to define a partial ordering of channels with respect to a given code and decoder. In the case of the belief propagation decoder quite general results are possible.

Let a channel  $W$  be represented by its transition probability  $p_W(y|x)$ . We say that a channel  $W'$  is *physically degraded* with respect to  $W$  if  $p_{W'}(y'|x) = p_Q(y'|y)p_W(y|x)$  for some auxiliary channel  $Q$ , see [6].

Very often a family of channels is ordered by physical degradation, i.e., if  $\alpha \leq \alpha'$ , then the channel with parameter  $\alpha'$  is physically degraded with respect to the channel with parameter  $\alpha$ . Examples of families of channels ordered by physical degradation in this way include the BSC, the AWGNC, the Laplace channel, the erasure channel, and many others.

Our most general result on the question of monotonicity of performance is the following.

**THEOREM 7.1 (Monotonicity for Physically Degraded Channels).** *Let  $W$  and  $W'$  be two given memoryless binary input channels that fulfill the required channel symmetry conditions. Assume that  $W'$  is physically degraded with respect to  $W$ . For a given code and a belief propagation decoder, let  $p$  be the expected fraction of incorrect messages passed at the  $\ell$ -th decoding step assuming tree-like neighborhoods and transmission over channel  $W$ , and let  $p'$  denote the equivalent quantity for transmission over channel  $W'$ . Then  $p \leq p'$ .*

For a proof see [30]. The main idea of the proof is that belief propagation is optimal, i.e., maximum likelihood, in graphs that are trees. As an alternative one could, in principle, degrade the observations before performing maximum likelihood decoding. This can clearly not be any better than performing maximum likelihood decoding on the original data and it is equivalent to decoding the output from the weaker channel.

The arguments used to prove monotonicity of performance under belief propagation do not carry over to other – non maximum-likelihood – message-passing decoders. Nevertheless, for many natural decoders of interest monotonicity can be proved directly. Virtually all message-passing decoders of interest can be interpreted as approximations to belief propagation and it is not surprising that many of them inherit the monotonicity property. We do not know of any decoder of interest for which monotonicity on any family of physically degraded channel has been disproved. There are several examples of decoders of interest, however, for which such monotonicity has not been proved. For various examples we refer the reader to

[30].

Given an ordered family of channels, an ensemble of codes, and a decoder for which monotonicity holds one can then define a *threshold* as the maximum channel parameter  $\sigma^*$  such that the probability of error under density evolution converges to zero if  $\sigma < \sigma^*$ . In some degenerate cases this threshold can actually be zero. For example, for an LDPC code ensemble with a non-zero proportion of degree one variable nodes the threshold is zero since in this case these degree one variables prevent the error probability from reaching zero (see Section 8.3). In most cases of interest, however, a non-trivial threshold will exist demarking the noise limit beyond which the iterative decoding system can not be expected to perform well regardless of the length of the code and below which iterative decoding is expected to work well. In practice one observes that codes approach their asymptotic performance from below. Finite size effects only degrade performance. Also, the orderings induced by the asymptotic performance tend to hold over a wide range of lengths.

**8. Analysis of Density Evolution for Belief Propagation.** In the preceding section we stated a general result on the monotonicity of the asymptotic performance of the belief propagation decoder. There are many other interesting properties associated to density evolution of belief propagation. In this section we present the most important results in this direction.

**8.1. Consistency .** In the following, we call a distribution  $f$  on  $\bar{\mathbb{R}}$  *consistent* if it satisfies  $f(x) = f(-x)e^x$  for all  $x \in \bar{\mathbb{R}}$ . Let us first consider a few examples of received distributions. Note that in each case  $P_0$  is consistent.

EXAMPLE 8 (BEC). *For the BEC the initial message distribution is  $\delta\Delta_0 + (1 - \delta)\Delta_\infty$ , where  $\Delta_x$  denotes a delta function at position  $x$ .*  $\square$

EXAMPLE 9 (BSC). *For the BSC the initial message distribution is  $P_0(y) := \epsilon\Delta_{-\log \frac{1-\epsilon}{\epsilon}} + (1 - \epsilon)\Delta_{\log \frac{1-\epsilon}{\epsilon}}$ .*  $\square$

EXAMPLE 10 (AWGNC). *Here, the initial message distribution is  $P_0(y) := \sqrt{\frac{\sigma^2}{8\pi}}e^{-\frac{(y - \frac{\sigma^2}{8})^2 \sigma^2}{\sigma^2}}$ . The consistency condition is then easily verified:*

$$P_0(y) = \sqrt{\frac{\sigma^2}{8\pi}}e^{-\frac{(y - \frac{\sigma^2}{8})^2 \sigma^2}{\sigma^2}} = \sqrt{\frac{\sigma^2}{8\pi}}e^{-\frac{(-y - \frac{\sigma^2}{8})^2 \sigma^2}{\sigma^2}}e^y = P_0(-y)e^y.$$

$\square$

In [29] the following proposition was stated and proved.

PROPOSITION 8.1. *Suppose we are given a binary-input, output-symmetric memoryless channel. Then  $P_0$ , the initial message distribution in log-likelihood ratio form under the all-one word assumption, is consistent.*

It was next shown in [29] that the set of consistent distributions is invariant under LDPC density evolution, i.e., each distribution  $R_\ell$  and  $P_\ell$

which arise from density evolution is consistent. It turns out that the consistency condition is much more general than this.

PROPOSITION 8.2. *Suppose we are given a binary-input, output-symmetric memoryless channel and let  $x$  be a bit from a binary linear code. Then  $P$ , the density of the conditional log-likelihood of  $x$  assuming the all-one codeword was transmitted and the entire codeword observed, is consistent.*

For a proof of this proposition we refer the reader to [31]. This result implies Proposition 8.1 and it also implies that the distributions of extrinsic information which arise in asymptotic turbo decoding are consistent. Consistency is an important property both for theoretical and for practical reasons. As explained in more detail below, the consistency condition can be used to improve the accuracy of some numerical algorithms and it is also helpful in giving a compact description of the *stability condition*, see Section 8.4.

Let  $f(x)$  be a consistent distribution. We define the error probability operator as<sup>11</sup>

$$\Pr_{\text{err}}(f) := \int_{-\infty}^0 f(x) \, dx .$$

An easy but helpful consequence of the consistency condition is noted in the following

COROLLARY 8.1. *If  $f(x)$  is a consistent distribution then  $\Pr_{\text{err}}(f) = 0$  if and only if  $f = \Delta_{\infty}$ .*

Hence, requiring the probability of error to converge to zero is equivalent to requiring the message density to converge to a delta function at infinity.

**8.1.1. Application of Consistency to Density Evolution for Turbo Codes.** One important application of the consistency condition is to the numerically accurate estimation of the threshold for turbo codes. For turbo codes, the message distributions induce distributions on the state probability vectors (usually denoted  $\alpha$  and  $\beta$ ) and the outgoing messages (extrinsic information) are functionals of the state probability vectors. It is not known (and seems unlikely) whether the distribution of the extrinsic information can be computed without first computing the distribution of the state probability vectors. The state probability vectors are, in general, vectors in  $\mathbb{R}^{2^m - 1}$ . Thus, even for quite small  $m$ , e.g.,  $m = 3$ , representations of distributions of state probability vectors are prohibitively complex. Thus, direct computation of density evolution appears to be practically intractable except in the simplest cases.

Nevertheless, it is possible to estimate the distributions of density evolution via simulation of the decoding process. We simulate decoding on a

---

<sup>11</sup>If  $f$  has a point mass at 0 then exactly half of that mass is included in  $\Pr_{\text{err}}(f)$ .

very long trellis in which all variables are independently sampled from their known distributions, determining the outgoing extrinsic information for each variable. These outgoing messages thus provide an empirical sample of the true message distribution. Since the decoding process, in this sense, is ergodic, the empirical distribution converges weakly to the true message distribution. In actual computations the distributions are quantized and hence discrete, thus, with sufficiently long simulation, an arbitrarily accurate estimate of the message distribution can be obtained with arbitrarily high probability.

A substantial savings in simulation time can be had by exploiting the consistency of the true message distribution. In simulation we need only determine the distribution of the *magnitude* of the extrinsic log-likelihoods. For a given log-likelihood magnitude  $x$  then we have  $P(x) = \frac{1}{1+e^{-x}}(P(x) + P(-x))$  and  $P(-x) = \frac{1}{1+e^x}(P(x) + P(-x))$ . Since  $P(x) + P(-x)$  is exponentially larger than  $P(-x)$  this provides a very significant acceleration of the convergence of the empirical distribution to the true message distribution with regard to estimates of rare error events.

**8.2. Fixed Points.** Recall from Example 7 that for the BEC and a belief propagation decoder the state of the system at any iteration is described by the remaining fraction of erasures. Denote the remaining fraction of erasures by  $x$  and let  $h(x)$  be the remaining fraction of erasures after a further iteration. Then the threshold is given by the maximum fraction  $x_0$  such that

$$(8.1) \quad \forall 0 < x < x_0 : h(x) < x.$$

Recall further from Example 1 that a similar statement is true for the BSC under Gallager's decoding algorithm A. In this case  $x$  signifies the remaining number of errors.

In these cases, an alternative description of the threshold can be given in terms of fixed points. The threshold is given by the maximum number  $x_0$  such that the equation

$$(8.2) \quad h(x) = x$$

has no solutions for  $x \in (0, x_0)$ .

Assume now we are given a general discrete memoryless channel which fulfills the channel symmetry conditions and we employ a belief propagation decoder. The state of the system is now described by the message density. Let  $f$  denote such a density and let  $h(f)$  denote the corresponding density after a further iteration. Clearly, there is no characterization of the threshold which corresponds to (8.1) since there is, a priori, no given linear ordering of densities. The alternative formulation in (8.2) involving fixed points looks more promising. Unfortunately, the non-existence of fixed-points for iterated function systems in more than one dimension is in

general not enough to guarantee convergence. So it is quite surprising, as it turns out, that, for message distributions of belief propagation decoders, fixed points sufficiently characterize the convergence of the sequences.

Let  $P_\ell(x)$  denote the distribution of the messages at the  $\ell$ -th iteration assuming, as usual, that the all-one word was transmitted. Suppose that we were to decide on the transmitted bit according to the sign of the message. In this case the conditional error probability is equal to  $\Pr_{\text{err}}(P_\ell)$ . But, because of the symmetry conditions, this is equal to the error probability even without conditioning on the transmitted codeword. Since the sign of the message is equal to the MAP estimate, this error probability is clearly a non-increasing function of  $\ell$ .

This monotonicity property is the key to the fixed point theorem below. It is not hard to see that there is actually a whole family of such monotonicity conditions.

**THEOREM 8.1 (General Monotonicity Law).** Let  $P_\ell$  be the message distribution at the  $\ell$ -th decoding step and let  $g$  be a consistent distribution on  $\bar{\mathbb{R}}$ . Then

$$\Pr_{\text{err}}(P_\ell \otimes g)$$

is a non-increasing function of  $\ell$ .

**THEOREM 8.2 (Fixed Point Theorem).** Let  $f$  be a consistent distribution and assume that  $f$  has support over the entire real axis. Let  $P_{\ell_1}(x)$  and  $P_{\ell_2}(x)$  denote the message distributions at the  $\ell_1$ -th and  $\ell_2$ -th iteration respectively. If

$$(8.3) \quad \Pr_{\text{err}}(P_{\ell_1} \otimes f) = \Pr_{\text{err}}(P_{\ell_2} \otimes f)$$

then  $P_{\ell_1}(x) = P_{\ell_2}(x)$  for  $\ell \geq \min\{\ell_1, \ell_2\}$  i.e.,  $P_{\ell_1}(x) = P_{\ell_2}(x)$  is a fixed point of density evolution.

We note that the above theorem has some very strong implications. Although the message distributions for a belief propagation decoder live nominally on  $\mathbb{R}^{\mathbb{R}}$ , the above theorem implies that the possible message distributions can be ordered by one real parameter (the projection). Hence, roughly speaking, we are dealing with a one-dimensional manifold embedded in an infinite dimensional space. This is very much analogous to the simple one-dimensional case we encounter in the case of an erasure channel, or Gallager's decoding algorithm A.

**8.3. Stability.** Consider transmitting over the BEC with erasure probability  $\delta$  using LDPC codes. Recall from example 7 that the expected fraction of erasure messages at the  $\ell$ -th iteration is given by

$$(8.4) \quad x_\ell = \delta\lambda(1 - \rho(1 - x_{\ell-1})),$$

where  $x_0 = \delta$ . As pointed out earlier, the threshold  $\delta^*$  is the single most important parameter describing the performance of an iterative coding system. We recall that  $\delta^*$  is the supremum of all values of  $\delta$  such that  $x_\ell$  tends

to zero as  $\ell$  tends to infinity. In general, it is not an easy task to determine the threshold analytically but we can give an analytical upper bound on  $\delta^*$  by considering the behavior of (8.4) for very small values of  $x_\ell$ . Hence, let  $h(x) := \delta\lambda(1 - \rho(1 - x))$ . Then  $h(x) = \delta\lambda'(0)\rho'(1)x + O(x^2)$ . Therefore, to first order in  $x$ , the fraction of erasure messages will evolve from  $x$  to  $\delta\lambda'(0)\rho'(1)x$ . Clearly, if we want the fraction of erasure messages to tend to zero then we need  $\lambda'(0)\rho'(1) < 1/\delta$ . From this we can deduce the bound  $\delta^* < \frac{1}{\lambda'(0)\rho'(1)}$ . Vice versa, if  $\lambda'(0)\rho'(1) < 1/\delta$  then there exists an  $\delta > 0$  such that the values of the recursion tend to zero if the recursion is initialized with a value which does not exceed  $\delta$ . The condition  $\lambda'(0)\rho'(1) < 1/\delta$ , first discussed in [37], can be seen as a *stability condition* of the fixed point  $x = 0$ .

Such a stability condition can be given in a much more general setting and it plays an important role in the theory of iterative coding systems. As we have seen it leads to an upper bound on the threshold of an iterative coding system. In the case of, e.g., *cycle codes* and the BSC this upper bound is tight, i.e., the stability condition determines the threshold of cycle codes for the BSC exactly [8]. The stability condition can also be interpreted in an alternative way. Assume we are trying to construct degree sequence pairs which achieve capacity on a given channel. In this case the desired threshold is determined by the capacity formula. E.g., for the case of the BEC we have  $\delta^* = 1 - r$ , where  $r$  is the rate of the code. Now we can write the stability condition as  $\lambda'(0) \leq \frac{1}{(1-r)\rho'(1)}$ , i.e., the stability condition imposes an upper bound on the fraction of edges which connect to degree two nodes. It has been shown in the case of the BEC that capacity achieving sequences must fulfill the above inequality with equality and we conjecture that this is true in general. We will now describe some important instances of the stability condition.

**8.4. Stability Condition for LDPC Codes and Belief Propagation.** Consider now a transmission over a general binary-input, memoryless, output-symmetric channel and assume that we use a belief propagation decoder. Observe that if  $P_\ell = \Delta_\infty$  for some  $\ell \geq 0$  then  $P_{\ell+i} = \Delta_\infty$  for any  $i \geq 0$ , i.e.,  $\Delta_\infty$  is a fixed point of density evolution. Since we desire that the error probability associated with density evolution converge to zero, it is clear that the fixed point described above should, in some sense, be an attractor (recall Corollary 8.1). To analyze local convergence to this fixed point we shall consider a linearization of density evolution about the fixed point.

Consider a density  $\epsilon P + (1 - \epsilon)\Delta_\infty$  where  $P$  is any (consistent) density,  $P \neq \Delta_\infty$ . After a complete iteration of density evolution this density will evolve to

$$\lambda'(0)\rho'(1)\epsilon P \otimes P_0 + (1 - \lambda'(0)\rho'(1)\epsilon)\Delta_\infty + O(\epsilon^2),$$

where  $\lambda'(x)$  and  $\rho'(x)$  denote the derivatives of  $\lambda(x)$  and  $\rho(x)$ , respectively.

In other words, the linearization of density evolution at the fixed point  $\Delta_\infty$  is given by  $P \rightarrow \lambda'(0)\rho'(1)P \otimes P_0$ .

One would expect, therefore, that a necessary condition for convergence to zero of the probability of error be that the probability of error associated with

$$(\lambda'(0)\rho'(1))^n P \otimes P_0^{\otimes n}$$

be decaying to zero as  $n$  grows. Under very general conditions (see Lemma 8.1) the limit

$$(8.5) \quad r := - \lim_{n \rightarrow \infty} \frac{1}{n} \log \Pr_{\text{err}}(P_0^{\otimes n})$$

is well defined. In this case, since  $P \neq \Delta_\infty$ , the quantity

$$(\lambda'(0)\rho'(1))^n \Pr_{\text{err}}(P \otimes P_0^{\otimes n})$$

converges to zero if and only if  $(\lambda'(0)\rho'(1))^n \Pr_{\text{err}}(P_0^{\otimes n})$  converges to zero. Although the above argument is only heuristic, the assertion is correct. This is summarized in the following.

**THEOREM 8.3.** *If  $\lambda'(0)\rho'(1) > e^r$ , then the probability of error of density evolution is strictly bounded away from 0. Conversely, if  $\lambda'(0)\rho'(1) < e^r$ , then there exists  $\epsilon > 0$  such that if density evolution is initialized with a consistent message distribution  $P$  satisfying  $\Pr_{\text{err}}(P) < \epsilon$ , then the probability of error will converge to zero.*

We note that in all cases of interest the exponent  $r$  can be computed using moment generating functions as stated in the following.

**LEMMA 8.1.** *Let  $g(s)$  be the moment generating function corresponding to the distribution  $P_0(x)$ , i.e.,  $g(s) = E_{P_0}[e^{sX}]$ , and assume that  $g(s) < \infty$  for all  $s$  in some neighborhood of zero. Then  $r = -\log(\inf_{s < 0} g(s))$ . Further, if  $P_0$  is consistent then  $r = -\log(2 \int_0^\infty P_0^+(x) e^{-x/2} dx)$ .*

**EXAMPLE 11 (BEC).** *For the BEC (see Example 2) we have*

$$e^{-r} = 2 \int_0^\infty \left[ \frac{\delta}{2} \Delta_0 + (1 - \delta) \Delta_\infty \right] e^{-x/2} dx = \delta .$$

Therefore, the stability condition reads

$$\lambda'(0)\rho'(1) < \frac{1}{\delta},$$

as noted above. □

**EXAMPLE 12 (BSC).** *For the BSC (see Example 9) we have*

$$e^{-r} = 2 \int_0^\infty (1 - \epsilon) \Delta_{\log \frac{1-\epsilon}{\epsilon}} e^{-x/2} dx = 2\sqrt{\epsilon(1-\epsilon)} .$$

It follows that the stability condition for the BSC is given by

$$\lambda'(0)\rho'(1) < \frac{1}{2\sqrt{\epsilon(1-\epsilon)}}.$$

□

EXAMPLE 13 (AWGC). For the AWGNC (see Example 10) we have

$$e^{-r} = 2 \int_0^\infty \sqrt{\frac{\sigma^2}{8\pi}} e^{-\frac{(x-\frac{\sigma^2}{8})\sigma^2}{8}} e^{-x/2} dx = e^{-\frac{1}{2\sigma^2}}.$$

Thus, the stability condition reduces to

$$\lambda'(0)\rho'(1) < e^{\frac{1}{2\sigma^2}}.$$

□

### 8.5. Examples of Other Instances of the Stability Condition.

The stability condition is not limited to LDPC codes and belief propagation decoders. Consider, e.g., the case of LDPC codes decoded with Gallager's decoding algorithm A and transmission over a binary symmetric channel with cross-over probability  $\epsilon$ , as discussed in Example 1. In this case it was shown in [2] that the appropriate stability condition reads

$$\frac{1 - \lambda'(0)\rho'(1)}{\lambda'(1)\rho'(1) - \lambda'(0)\rho'(0)} > \epsilon.$$

We note that for some codes e.g., the (4, 8), (5, 10) and the (4, 6) regular codes, this condition is tight, i.e., it determines the threshold exactly.

Consider next the special case of a parallel concatenated turbo code with the simple component code  $[1, \frac{1}{1+D}]$ , no puncturing and transmission over the BEC with erasure probability  $\delta$ . In this case it was shown in [31] that the erasure probability  $x_\ell$  evolves as

$$x_\ell = \frac{x_{\ell-1}\delta^2(2 - 2\delta + x_{\ell-1}\delta)}{(1 - \delta(1 - x_{\ell-1}))^2}.$$

The stability condition is easily found to be  $\delta < \frac{1}{2}$ . Again, the stability is tight, i.e., the threshold  $\delta^*$  can be shown to be equal to  $\frac{1}{2}$ . The general form of the stability condition for turbo codes has not been explored in any depth so far and this promises to be a fruitful direction for future research.

**9. Acknowledgment.** We would like to thank David Proietti and the anonymous reviewer for pointing out to us some typos in a previous version of this paper.

## REFERENCES

- [1] N. ALON, J. SPENCER, AND P. ERDÖS, *The Probabilistic Method*, John Wiley & Sons, Inc., New York, 1992.
- [2] L. BAZZI, T. RICHARDSON, AND R. URBANKE, *Exact thresholds and optimal codes for the binary symmetric channel and Gallager's decoding algorithm A*. submitted IEEE IT.
- [3] S. BENEDETTO AND G. MONTORSI, *Unveiling turbo codes: Some results on parallel concatenated coding schemes*, IEEE Trans. Inform. Theory, 42 (1996), pp. 409–428.
- [4] C. BERROU, A. GLAVIEUX, AND P. THITIMAJSHIMA, *Near Shannon limit error-correcting coding and decoding*, in Proceedings of ICC'93, Geneva, Switzerland, May 1993, pp. 1064–1070.
- [5] S.-Y. CHUNG. personal communication.
- [6] T. M. COVER AND J. A. THOMAS, *Elements of Information Theory*, Wiley, New York, 1991.
- [7] M. C. DAVEY AND D. J. C. MACKAY, *Low density parity check codes over  $GF(q)$* , IEEE Communications Letters, 2 (1998).
- [8] L. DECREUSEFOND AND G. ZÉMOR, *On the error-correcting capabilities of cycle codes of graphs*, Combinatorics, Probability and Computing, (1997), pp. 27–38.
- [9] D. DIVSALAR, *A simple tight bound on error probability of block codes with application to turbo codes*. TMO Progress Report 42-139.
- [10] T. M. DUMAN AND M. SALEHI, *Performance bounds for turbo-coded modulation systems*, IEEE Transactions on Communications, 47 (1999), pp. 511–521.
- [11] D. FORNEY, *Codes on graphs: Generalized state realizations*. submitted to IEEE Trans. Inform. Theory.
- [12] B. FREY AND F. KSCHISCHANG, *Probability propagation and iterative decoding*, in Allerton Conf. on Communication, Control and Computing, 1996.
- [13] R. G. GALLAGER, *Low-Density Parity-Check Codes*, M.I.T. Press, Cambridge, Massachusetts, 1963.
- [14] J. GARCIA-FRIAS AND J. D. VILLASENOR, *Combining hidden Markov source models and parallel concatenated codes*, IEEE Communications Letters, 1 (1997), pp. 111–113.
- [15] J. GARCIA-FRIAS AND J. D. VILLASENOR, *Exploiting binary Markov channels with unknown parameters in Turbo decoding*, in Proc. Globecom'98, Sydney, Australia, Nov. 1998, pp. 3244–3249.
- [16] ———, *Turbo decoders for Markov channels*, IEEE Commun. Lett., 2 (1998), pp. 257–259.
- [17] E. A. GELBLUM, A. R. CALDERBANK, AND J. BOUTROS, *Understanding serially concatenated codes from a support tree approach*, in Proceedings of the International Symposium on Turbo Codes and Related Topics, Brest, France, Sept. 1997, pp. 271–274.
- [18] E. K. HALL AND S. G. WILSON, *Design and analysis of turbo codes on Rayleigh fading channels*, IEEE Journal of Selected Areas in Communications, 16 (1998), pp. 160–174.
- [19] P. HOEHER, J. LODGE, R. YOUNG AND J. HAGENAUER, *Separable map "filters" for the decoding of product and concatenated codes*, in Proceedings of ICC'93, Geneva, Switzerland, May 1993, pp. 1740–1745.
- [20] F. KSCHISCHANG AND B. FREY, *Iterative decoding of compound codes by probability propagation in graphical models*, IEEE Journal on Selected Areas in Communications, (1998), pp. 219–230.
- [21] F. KSCHISCHANG, B. FREY, AND H.-A. LOELIGER, *Factor graphs and the sum-product algorithm*. submitted to IEEE Trans. Inform. Theory.
- [22] M. LUBY, M. MITZENMACHER, A. SHOKROLLAHI, AND D. SPIELMAN, *Analysis of low density codes and improved designs using irregular graphs*, in Proceedings

- of the 30th Annual ACM Symposium on Theory of Computing, 1998, pp. 249–258.
- [23] M. LUBY, M. MITZENMACHER, A. SHOKROLLAHI, D. SPIELMAN, AND V. STEMANN, *Practical loss-resilient codes*, in Proceedings of the 29th annual ACM Symposium on Theory of Computing, 1997, pp. 150–159.
  - [24] D. J. C. MACKAY AND R. M. NEAL, *Good codes based on very sparse matrices*, in Cryptography and Coding. 5th IMA Conference, C. Boyd, ed., no. 1025 in Lecture Notes in Computer Science, Springer, Berlin, 1995, pp. 100–111.
  - [25] I. D. MARSLAND AND P. MATHIOPOULOS, *Multiple differential detection of parallel concatenated convolutional (turbo) codes in correlated fast rayleigh fading*, IEEE Journal of Selected Areas in Communications, 16 (1998), pp. 265–275.
  - [26] R. McELIECE, E. RODEMICH, AND J.-F. CHENG, *The turbo decision algorithm*, in Proceedings of the 33rd Allerton Conference on Communication, Control, and Computing, Monticello, IL, 1995.
  - [27] R. MOTWANI AND P. RAGHAVAN, *Randomized Algorithms*, Cambridge University Press, Cambridge, 1995.
  - [28] J. PEARL, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann Publishers, 1988.
  - [29] T. RICHARDSON, A. SHOKROLLAHI, AND R. URBANKE, *Design of provably good low-density parity check codes*. submitted IEEE IT.
  - [30] T. RICHARDSON AND R. URBANKE, *The capacity of low-density parity check codes under message-passing decoding*. submitted IEEE IT.
  - [31] ———, *The capacity of turbo codes and other concatenated codes under message-passing decoding*. in preparation.
  - [32] ———, *Concentrate!*, in Allerton Conf. on Communication, Control and Computing, 1999.
  - [33] P. ROBERTSON, *Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes*, in Proceedings of GLOBECOM'94, Nov. 1994, pp. 1298–1303.
  - [34] I. SASON AND S. SHAMAI (SHITZ), *Improved upper bounds on the decoding error probability of parallel and serial concatenated turbo codes via their ensemble distance spectrum*, in 1998 IEEE International Symposium on Information Theory, Boston, MA, Aug. 16–21 1998, p. 30.
  - [35] ———, *Bounds on the error probability of ML decoding for block and turbo-block codes*, Annales de Telecommunications, 54 (1999), pp. 61–78.
  - [36] A. SHAMIR AND J. SPENCER, *Sharp concentration of the chromatic number on random graphs  $G_{n,p}$* , Combinatorica, 7 (1987), pp. 121–129.
  - [37] A. SHOKROLLAHI, *New sequences of linear time erasure codes approaching the channel capacity*, in Proceedings of AAEECC-13, Lecture Notes in Computer Science 1719, 1999, pp. 65–76.
  - [38] M. SIPSER AND D. SPIELMAN, *Expander codes*, IEEE Trans. on Information Theory, 42 (1996).
  - [39] N. SOURLAS, *Spin-glass models as error-correcting codes*, Nature, 339 (1989), pp. 693–695.
  - [40] R. M. TANNER, *A recursive approach to low complexity codes*, IEEE Trans. Inform. Theory, 27 (1981), pp. 533–547.
  - [41] A. VITERBI AND J. OMURA, *Principles of Digital Communication and Coding*, McGraw-Hill, 1979.
  - [42] A. VITERBI, A. VITERBI, J. NICOLAS, AND N. SINDHUSHAYANA, *Perspective on interleaved concatenated codes with iterative soft-output decoding*, in Proceedings of the International Symposium on Turbo Codes and Related Topics, Brest, France, Sept. 1997, pp. 47–54.
  - [43] N. WIBERG, *Codes and Decoding on General Graphs*, PhD thesis, Linköping University, S-581 83, Linköping, Sweden, 1996.
  - [44] N. WIBERG, H.-A. LOELIGER, AND R. KÖTTER, *Codes and iterative decoding on general graphs*, European Transactions in Telecommunications, 6 (1995),

- pp. 513–526.
- [45] V. ZYABLOV AND M. PINSKER, *Estimation of the error-correction complexity of Gallager low-density codes*, Problemy Peredachi Informatsii, 11 (1975), pp. 23–26.