The Capacity of Low-Density Parity Check Codes under Message-Passing Decoding

Article *in* IEEE Transactions on Information Theory · March 2001

DOI: 10.1109/18.910577 · Source: IEEE Xplore

2 authors:

Tom Richardson
Qualcomm
119 PUBLICATIONS **17,752** CITATIONS

SEE PROFILE

Rüdiger L. Urbanke
École Polytechnique Fédérale de Lausanne
223 PUBLICATIONS **19,926** CITATIONS

SEE PROFILE

The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding

Thomas J. Richardson and Rüdiger L. Urbanke

***Abstract*—In this paper, we present a general method for determining the *capacity* of low-density parity-check (LDPC) codes under message-passing decoding when used over any binary-input memoryless channel with discrete or continuous output alphabets. Transmitting at rates below this capacity, a randomly chosen element of the given ensemble will achieve an arbitrarily small target probability of error with a probability that approaches one exponentially fast in the length of the code. (By concatenating with an appropriate outer code one can achieve a probability of error that approaches zero exponentially fast in the length of the code with arbitrarily small loss in rate.) Conversely, transmitting at rates above this capacity the probability of error is bounded away from zero by a strictly positive constant which is independent of the length of the code and of the number of iterations performed. Our results are based on the observation that the concentration of the performance of the decoder around its average performance, as observed by Luby *et al.* [1] in the case of a binary-symmetric channel and a binary message-passing algorithm, is a general phenomenon. For the particularly important case of belief-propagation decoders, we provide an effective algorithm to determine the corresponding capacity to any desired degree of accuracy. The ideas presented in this paper are broadly applicable and extensions of the general method to low-density parity-check codes over larger alphabets, turbo codes, and other concatenated coding schemes are outlined.**

***Index Terms*—Belief propagation, iterative decoding, low-density parity-check (LDPC) codes, message-passing decoders, turbo codes, turbo decoding.**

I. Introduction

In the wake of the phenomenal success of turbo codes [2], another class of codes exhibiting similar characteristics and performance was rediscovered [3], [4]. This class of codes, called *low-density parity-check* (LDPC) codes, was first introduced by Gallager in his thesis in 1961 [5]. In the period between Gallager's thesis and the invention of turbo codes, LDPC codes and their variants were largely neglected. Notable exceptions are the work of Zyablov and Pinsker [6], Tanner [7], and Margulis [8].

In their original (regular) incarnation, the performance of LDPC codes over the binary-input additive white Gaussian noise (BIAWGN) channel is only slightly inferior to that of parallel or serially concatenated convolutional codes (turbo codes). For example, a rate one-half LDPC code of block length 10 000 requires an E_b/N_0 of roughly 1.4 dB to achieve a bit-error probability of 10^{-5}, whereas an equivalent turbo code with comparable complexity achieves the same performance at, roughly, $E_b/N_0 = 0.8$ dB. Shannon capacity dictates that, in order to achieve reliable transmission at a rate of one-half bit per channel use over the continuous-input AWGN channel, an E_b/N_0 of at least 0 dB is required, and this increases to 0.187 dB if we restrict the input to be binary.

It is well known, that any linear code can be expressed as the set of solutions $\boldsymbol{x}$ of a parity-check equation $H\boldsymbol{x}^T = 0^T$. Furthermore, if the code is binary then H takes elements in $\mathrm{GF}(2)$ and the arithmetic is also over this field. A $(d_\mathrm{v}, d_\mathrm{c})$-regular LDPC code, as originally defined by Gallager, is a binary linear code determined by the condition that every codeword bit participates in exactly d_v parity-check equations and that every such check equation involves exactly d_c codeword bits, where d_v and d_c are parameters that can be chosen freely.[1] In other words, the corresponding parity-check matrix H has d_v ones in each column and d_c ones in each row. The modifier "low-density" conveys the fact that the fraction of nonzero entries in H is small, in particular it is linear in the block length n, as compared to "random" linear codes for which the expected fraction of ones grows like n^2. Following the lead of Luby *et al.* [1] we do not focus on particular LDPC codes in this paper, but rather analyze the performance of *ensembles* of codes. One way of constructing an ensemble of LDPC codes would be to consider the set of all parity-check matrices of length n which fulfill the above row and column sum constraints for some fixed parameters $(d_\mathrm{v}, d_\mathrm{c})$, and to equip this set with a uniform probability distribution. It is more convenient, however, to proceed as suggested in [1] and to define the ensemble of LDPC codes via *bipartite graphs*, see Section II-A, since the resulting ensemble is easier to analyze.[2] Many variations and extensions of Gallager's original definition are possible and these extensions are important in order to construct LDPC codes that approach capacity. To mention only the two most important ones: a) constructing irregular as opposed to regular codes [1], [9], [10], i.e., variables may participate in different numbers of checks, and check equations may involve different numbers of variables and b) allowing nodes to represent groups of bits rather than single bits [11] (see also [5]).

In his thesis, Gallager discussed several decoding algorithms that work directly on the nodes and edges of the bipartite graph (see Section II) representing the code. Here, one set of nodes,

Manuscript received November 16, 1999; revised August 18, 2000.

T. J. Richardson was with Bell Labs, Lucent Technologies, Murray Hill, NJ 07974 USA. He is now with Flarion Technologies, Bedminster, NJ 07921 USA (e-mail: richardson@flarion.com).

R. L. Urbanke was with Bell Labs, Lucent Technologies, Murray Hill, NJ 07974 USA. He is now with the EPFL, LTHC-DSC, CH-1015 Lausanne, Switzerland (e-mail: rudiger.urbanke@epfl.ch).

Communicated by F. R. Kschischang, Associate Editor for Coding Theory.

Publisher Item Identifier S 0018-9448(01)00737-4.

[1] As we will discuss in more detail in Section II-A, the *design rate* of such a code is equal to $1 - \frac{d_\mathrm{v}}{d_\mathrm{c}}$.

[2] In any case, these two ensembles are essentially identical if we consider large block lengths.

the "variable" nodes, correspond to the variables, i.e., the codeword bits $\boldsymbol{x}_i$ or, equivalently, the columns of H, the other set, the "check" nodes, correspond to the constraints or, equivalently, the rows of H. A variable node is connected to a constraint node in the graph if and only if the corresponding variable participates in the corresponding constraint. The decoding algorithms of interest work *iteratively.* Information is exchanged between neighboring nodes in the graph by passing messages along the edges. Each message can be associated to the codeword bit corresponding to the variable node incident to the edge carrying the message. Invariably, the messages can be interpreted as conveying an estimate of that bit's value along with some reliability information for that estimate. Associated with such a message is a hard decision: one can consider the bit's most likely value implied by the message. We will say that a message is "correct/incorrect" if its associated hard decision is correct/incorrect, i.e., does/does not agree with the true value of the codeword bit. In the sequel, the precise meaning will be clear from context.

Gallager [5] proposed the following step-by-step program to determine the *worst binary-symmetric channel* (BSC), i.e., the BSC with the largest crossover probability, over which an appropriately constructed $(d_{\mathsf{v}}, d_{\mathsf{c}})$-regular LDPC code in conjunction with a given iterative decoding algorithm can be used to transmit information reliably.

1) [Code Construction] For increasing length n, construct a sequence of $(d_{\mathsf{v}}, d_{\mathsf{c}})$-regular LDPC codes that do not contain cycles of length less or equal to $2\ell(n)$ where

$$\ell(n) := \frac{\ln n - \ln \frac{d_{\mathsf{v}}d_{\mathsf{c}} - d_{\mathsf{v}} - d_{\mathsf{c}}}{2d_{\mathsf{c}}}}{\ln[(d_{\mathsf{c}}-1)(d_{\mathsf{v}}-1)]}.$$

2) [Density Evolution and Threshold Determination] Determine the average fraction of incorrect messages passed at the ℓth iteration assuming that the graph does not contain cycles of length 2ℓ or less. For an iterative decoder with a finite message alphabet this fraction can be expressed by means of a system of coupled recursive functions which depend on the ensemble parameters $(d_{\mathsf{v}}, d_{\mathsf{c}})$, and the channel parameter. Determine the maximum channel parameter, called the *threshold*, with the property that for all parameters strictly less than this threshold the expected fraction of incorrect messages approaches zero as the number of iterations increases.

3) Conclude that if one applies the chosen decoding algorithm to this sequence of codes with $\ell(n)$ decoding rounds while operating over a BSC with parameter less than the threshold found in Step 2), then the bit-error probability decreases as $e^{-\alpha n^{\beta}}$, where α and β are positive constants which depend on $(d_{\mathsf{v}}, d_{\mathsf{c}})$ (one actually requires $d_{\mathsf{v}} > 2$). (Since the block-error probability is at most n times the bit-error probability, it follows that the block-error probability can be upper-bounded in the same way.)

Although the analysis presented in this paper contains many of the same elements as the program suggested by Gallager there is one crucial difference. As remarked earlier, we do not focus on any particular (sequence of) codes but rather on (sequences of) ensembles as suggested in [1]. The main advantage gained by this approach is the following: even in the regular case it is not an easy task to construct codes which contain only large cycles and this task becomes almost hopeless once we allow irregular graphs. Sampling an element of an (irregular) ensemble, on the other hand, is almost trivial. This approach was applied with great success in [1] to the analysis (and design) of LDPC ensembles used over the binary erasure channel (BEC) as well as the BSC when employed in conjunction with a one-bit message-passing decoder.

The contribution of the present paper is to extend the method of analysis in [1] to a very broad class of channels and decoding algorithms and to introduce some new tools for the analysis of these decoding systems. To clarify further the nature of the results we briefly describe an example and formulate some claims that represent what we consider to be the apex of the theory. Let us consider the ensemble of $(3, 6)$-regular LDPC codes (see Section II for its definition) of length n for use over a BIAWGNC, i.e., we transmit a codeword consisting of n bits $\boldsymbol{x}_i \in \{\pm 1\}$ and receive n values $y_i = \boldsymbol{x}_i + z_i$ where the z_i are independent and identically distributed (i.i.d.) zero-mean Gaussian random variables with variance σ^2. Choose a code at random from this ensemble (see Section II), choose a message with uniform probability from the set of messages, and transmit the corresponding codeword. Decode the received word using the belief-propagation algorithm. The following statements are consequences of the theory:

> [Concentration] Let $P_e^n(\ell)$ be the expected fraction of incorrect messages which are passed in the ℓth iteration, where the expectation is over all instances of the code, the choice of the message, and the realization of the noise. For any $\delta > 0$, the probability that the actual fraction of incorrect messages which are passed in the ℓth iteration for any particular such instance lies outside the range $(P_e^n(\ell) - \delta, P_e^n(\ell) + \delta)$ converges to zero exponentially fast in n.
>
> [Convergence to Cycle-Free Case] $P_e^n(\ell)$ converges to $P_e^{\infty}(\ell)$ as n tends to infinity, where $P_e^{\infty}(\ell)$ is the expected fraction of incorrect messages passed in the ℓth decoding round assuming that the graph does not contain cycles of length 2ℓ or less.
>
> [Density Evolution and Threshold Determination] $P_e^{\infty}(\ell)$ is computable by a deterministic algorithm. Furthermore, there exists a channel parameter σ^* (in this case $\sigma^* \simeq 0.88$),[3] the *threshold* with the following property: if $\sigma < \sigma^*$ then $\lim_{\ell\to\infty} P_e^{\infty}(\ell) = 0$; if, on the other hand, $\sigma > \sigma^*$ then there exists a constant $\gamma(\sigma) > 0$ such that $P_e^{\infty}(\ell) > \gamma(\sigma)$ for all $\ell \geq 1$. For the current example, using the methods outlined in Section III-B to efficiently implement density evolution, we get $\gamma(\sigma) \geq 0.068$ for $\sigma > \sigma^*$.

The first statement asserts that (almost) all codes behave alike and so the determination of the average behavior of the ensemble suffices to characterize the individual behavior of (almost)

[3]Gallager attempted to compute the threshold for the BSC by a combinatorial approach. The complexity of his approach precluded considering more than a few iterations though. For the threshold of the $(3, 6)$-regular LDPC ensemble he obtained the approximate lower bound of 0.07 whereas density evolution gives 0.084.

all codes. The second statement then claims that for long codes this average behavior is equal to the behavior which one can observe on cycle-free graphs and that for the cycle-free case this average behavior is computable by a deterministic algorithm. Finally, from the last statement we conclude that long codes will exhibit a threshold phenomenon, clearly separating the region where reliable transmission is possible from that where it is not. As a consequence, if we want to transmit over the BIAWGNC using codes from the $(3, 6)$-regular LDPC ensemble and a belief-propagation decoder we are confronted with the following dichotomy. If $\sigma < 0.88$ and if we are given any target bit-error probability, call it ϵ, then we can choose an appropriate number of decoding iterations $\ell(\epsilon, \sigma)$ and an appropriate block length $n(\epsilon, \sigma)$. We can then be assured that all but at most an exponentially (in n) small subset from the $(3, 6)$-regular LDPC ensemble of length $n(\epsilon, \sigma)$ if decoded for $\ell(\epsilon, \sigma)$ rounds will exhibit a bit-error probability of at most ϵ.[4] On the other hand, if $\sigma > 0.88$ and if we are willing to perform at most ℓ decoding rounds, where ℓ is a fixed natural number, then all but at most an exponentially (in n) small subset of codes of the $(3, 6)$-regular LDPC ensemble will exhibit a bit error probability of at least 0.05[5] if decoded by means of at most ℓ rounds of a belief-propagation decoder. With regards to this converse, we conjecture that actually the following much stronger statement is true—namely, that *all* codes in the $(3, 6)$-regular LDPC ensemble have bit-error probability of at least 0.05 *regardless of their length* and *regardless of how many iterations* are performed. This will follow if one can prove that cycles in the graph (and the resulting dependence) can only degrade the performance of the iterative decoder on *average*, [6] a statement which seems intuitive but has so far eluded proof.

Each of the above statements generalizes to some extent to a wide variety of codes, channels, and decoders. The concentration result holds in essentially all cases and depends mainly on the fact that decoding is "local." Similarly, the convergence of $P_e^n(\ell)$ to $P_e^\infty(\ell)$ is very general, holding for all cases of interest. It is a consequence of the fact that typically the decoding neighborhoods become "tree-like" if we fix the number of iterations and let the block length tend to infinity. Concerning the *density evolution*, for decoders with a message alphabet of size q the quantity $P_e^\infty(\ell)$ can in general be expressed by means of $(q-1)$ coupled recursive functions (see Section III-A). For message-passing algorithms with *infinite* message alphabets the situation is quite more involved. Nevertheless, for the important case of the sum-product or belief-propagation decoder we will present in Section III-B an efficient algorithm to calculate $P_e^\infty(\ell)$. Finally, the existence of a threshold as asserted in the last statement requires more assumptions. It depends on *both* the decoding algorithm used and the class of channels considered. As we will see, such a threshold always exists if we are dealing with a family of channels which can be ordered by *physical degradation* (see Section III-B1) and if the decoding algorithm respects this ordering. In some other instances, like in the case of decoders with finite message alphabets, the existence of a threshold can often be shown by analyzing the corresponding recursions directly.

Roughly speaking, the general statement is then of the following kind. We are given a particular ensemble of codes, a family of channels, and a particular decoding algorithm. From these quantities we will be able to calculate the critical channel parameter which is called the threshold. Then almost any long enough code can be used to provide sufficiently reliable transmission of information if decoded by the given decoder for a sufficiently large number of iterations provided that the actual channel parameter is below this threshold. Conversely, reliable transmission over channels with parameter above this threshold is not possible with most codes chosen at random from "long" ensembles. Therefore, one should think of the threshold as the equivalent of a "random capacity" for a given ensemble of codes and a particular decoder, except that in the case of random capacity the channel is usually fixed and one is interested in the largest rate whereas for the threshold we fix the rate and ask for the "worst" channel.

The outline of the paper is as follows. Section II introduces the class of codes, channels, and decoding algorithms considered in this paper. We show that under suitable symmetry conditions the error probability is independent of the transmitted codeword. Section III then focuses on the determination of $P_e^\infty(\ell)$ and the determination of the threshold value. We first investigate decoders with finite message alphabets. In this case, as pointed out above, $P_e^\infty(\ell)$ can be expressed by means of multiple coupled recursive functions, and the existence of a threshold can be shown by a closer investigation of these recursions. We will see that by a judicious choice of the messages the resulting thresholds can be made surprisingly close to the ultimate limit as given by the (Shannon) capacity bound. We next investigate the important case of belief-propagation decoders. Despite the fact that the message alphabet is infinite, in this case we describe an efficient algorithm that can be used to calculate $P_e^\infty(\ell)$ for a belief-propagation decoder for any binary-input memoryless output-symmetric channel to any desired accuracy. We also show that in this case the existence of a threshold is guaranteed if the channel family can be ordered by physical degradation, as is the case for many important families of channels, including the BEC, BSC, BIAWGNC, and the binary-input Laplace (BIL) channel. In Section IV, we will show that, as the length of the code increases, the behavior of individual instances concentrates around the expected behavior and that this expected behavior converges to the behavior for the cycle-free case.

The ideas presented in this paper are broadly applicable, and extensions of the general method to irregular LDPC codes, LDPC codes over larger alphabets, turbo codes, and other concatenated coding schemes are outlined in Section V.

[4]Although this is not the main focus of this paper, one can actually achieve an *exponetially decreasing probability of error.* To achieve this, simply use an outer code which is capable of recovering from a small linear fraction of errors. If properly designed, the LDPC code will decrease the bit probability of error below this linear fraction with exponential probability within a fixed number of decoding rounds. The outer code then removes all remaining errors. Clearly, the incurred rate loss can be made as small as desired.

[5]Note that γ, the probability of sending an *erroneous messages*, is at least 0.068. To this corresponds a *bit-error probability* which is at least 0.05.

[6]By constructing simple examples, it is not very hard to show that for *some specific* bits the probability of error can actually be decreased by the presence of cycles.

II. Basic Notation and Assumptions

In this section we will introduce and discuss some of the basic notation and assumptions that we will use throughout this paper. We start by giving a precise definition of the *ensemble* of $(d_{\mathsf{v}}, d_{\mathsf{c}})$-regular LDPC codes that we will consider. We then discuss the notion of *decoding neighborhoods.* This notion will play an important role when giving a proof of the concentration theorem. After a brief look at the class of *binary-input memoryless channels* we will introduce the class of *message-passing* decoders, which are iterative decoders working on the graph and which obey the *extrinsic information principle* well known from turbo decoding. Finally, we will see that under a suitable set of *symmetry assumptions* the conditional decoding probability will become independent of the transmitted codeword, allowing us in the sequel to assume that the *all-one* codeword was transmitted.

A. Ensembles

Let n be the length of the binary code given as the set of solutions $\boldsymbol{x}$ to the parity-check equation $H\boldsymbol{x}^T = 0^T$. We construct a bipartite graph with n *variable* nodes and $m := \frac{nd_{\mathsf{v}}}{d_{\mathsf{c}}}$ *check* nodes. Each variable node corresponds to one bit of the codeword, i.e., to one column of H, and each check node corresponds to one parity-check equation, i.e., to one row of H. Edges in the graph connect variable nodes to check nodes and are in one-to-one correspondence with the nonzero entries of H.

Since each check equation typically decreases the number of degrees of freedom by one, it follows that the *design* rate of the code is

$$r := \frac{n-m}{n} = 1 - \frac{d_{\mathsf{v}}}{d_{\mathsf{c}}}.$$

The actual rate of a given code may be higher since these check equations might not all be independent, but we shall generally ignore this possibility. There are $nd_{\mathsf{v}} = md_{\mathsf{c}}$ edges in the bipartite graph, d_{v} edges incident to each variable node on the left and d_{c} edges incident to each check node on the right. Fig. 1 gives an example of a $(3, 6)$-regular code of length 10.

The ensemble $\mathcal{C}^n(d_{\mathsf{v}}, d_{\mathsf{c}})$ of $(d_{\mathsf{v}}, d_{\mathsf{c}})$-regular LDPC codes of length n which we consider in this paper is defined as follows. Assign to each node d_{v} or d_{c} "sockets" according to whether it is a variable node or a check node, respectively. Label the variable and check sockets separately with the set $[nd_{\mathsf{v}}] := \{1, \ldots, nd_{\mathsf{v}}\}$ in some arbitrary fashion. Pick a permutation π on nd_{v} letters at random with uniform probability from the set of all $(nd_{\mathsf{v}})!$ such permutations. The corresponding (labeled) bipartite graph is then defined by identifying edges with pairs of sockets and letting the set of such pairs be $\{(i, \pi(i)), i = 1, \ldots, nd_{\mathsf{v}}\}$. This induces a uniform distribution on the set $\mathcal{C}^n(d_{\mathsf{v}}, d_{\mathsf{c}})$ (the set of *labeled bipartite graphs*). In practice, one usually modifies the permutation in an attempt to obtain the best possible performance for the given length. In particular, one avoids double edges between the nodes and excessive overlap of the neighbor sets of the nodes.

Strictly speaking, edges are unordered pairs $\{i, \pi(i)\}$, where i and $\pi(i)$ denote the corresponding variable node socket and check node socket, respectively. It is often more convenient to think of an edge as a pair of nodes $\{\mathsf{v}, \mathsf{c}\}$, where v and c are the variable and check nodes incident to the given edge. If the graph does not contain parallel edges then both descriptions are equivalent and we can freely switch between the two representations. Although this constitutes a moderate abuse of notation, we will maintain the notation $\mathsf{e} = \{\mathsf{v}, \mathsf{c}\}$ even if parallel edges are not excluded. The reader is then advised to think of e as *some* edge which connects the variable node v to the check node c. This simplifies the notation significantly and should not cause any confusion. By a *directed* edge $\vec{\mathsf{e}}$ we mean an ordered pair (v, c) or (c, v) corresponding to the edge $\mathsf{e} = \{\mathsf{v}, \mathsf{c}\}$. When we say that a message traverses a directed edge $\vec{\mathsf{e}}$ we mean that it traverses it in the indicated direction. When we say that a message traverses an undirected edge e we mean that it traverses it in some direction.

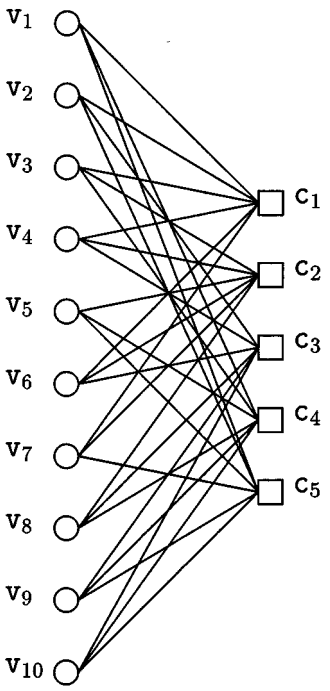

Fig. 1. A $(3, 6)$-regular code of length 10. There are 10 variable nodes and five check nodes. The $nd_{\mathsf{v}} = 30 = md_{\mathsf{c}}$ "sockets" of the variable and check nodes are labeled (not shown).

B. Decoding Neighborhoods

A *path* in the graph is a directed sequence of directed edges $\vec{\mathsf{e}}_1, \ldots, \vec{\mathsf{e}}_k$ such that, if $\vec{\mathsf{e}}_i = (\mathsf{u}_i, \mathsf{u}_i')$, then the $\mathsf{u}_i' = \mathsf{u}_{i+1}$ for $i = 1, \ldots, k-1$. The length of the path is the number of directed edges in it and we say that the path starts from u_1, ends at u_k', and connects u_1 to u_k'. Given two nodes in the graph, we say that they have *distance* $d \leq \infty$ if they are connected by a path of length d but not by a path of length less than d. (If the nodes are not connected then we say $d = \infty$.) For a given node u, we define its *neighborhood of depth* d, denoted by $\mathcal{N}_{\mathsf{u}}^d$, as the induced subgraph consisting of all nodes reached and edges traversed by paths of length at most d starting from u (including u). Note that for any two nodes u_1 and u_2 we have, by symmetry of the distance function

$$\mathsf{u}_1 \in \mathcal{N}_{\mathsf{u}_2}^d \Longleftrightarrow \mathsf{u}_2 \in \mathcal{N}_{\mathsf{u}_1}^d. \tag{1}$$

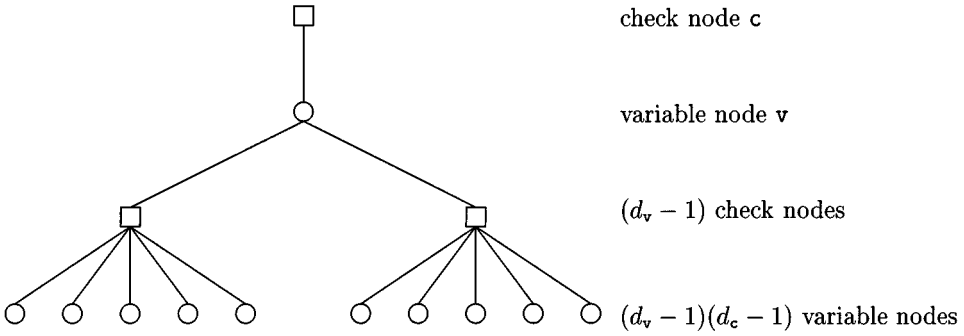

Fig. 2. The directed neighborhood of depth 2 of the directed edge $\vec{e} = (v, c)$.

Let e be a particular edge between variable node v and check node c, i.e., $e = \{v, c\}$ according to our convention. The *undirected neighborhood of depth* d of e, denoted by $\mathcal{N}_e^d$, is defined as $\mathcal{N}_e^d := \mathcal{N}_v^d \cup \mathcal{N}_c^d$. We claim that for any pair of edges $e = \{v, c\}$ and $e' = \{v', c'\}$

$$e \in \mathcal{N}_{e'}^d \Longleftrightarrow e' \in \mathcal{N}_e^d. \tag{2}$$

By symmetry, it is clearly enough to prove the implication in one direction. Hence, assume that $e \in \mathcal{N}_{e'}^d$. A directed version of e, denoted $\vec{e}$, lies on a path of length d starting from either v' or c'. It follows that there exists a path starting from either v' or c' and ending at either v or c of length at most $d-1$. Reversing the orientation of the directed edges in the path we obtain a path of length at most $d-1$ from either v or c to either v' or c'. Thus, there exists a path of length at most d starting from either v or c containing a directed version of the edge $\{v', c'\}$. We conclude that $e' \in \mathcal{N}_e^d$.

The *directed neighborhood of depth* d of $\vec{e} = (v, c)$, denoted by $\mathcal{N}_{\vec{e}}^d$, is defined as the induced subgraph containing all edges and nodes on paths $\vec{e}_1, \ldots, \vec{e}_d$ starting from v such that $\vec{e}_1 \neq \vec{e}$. Fig. 2 gives an example of such a directed neighborhood of depth 2. A directed neighborhood of a directed edge $\vec{e} = (c, v)$ is defined analogously. If the induced subgraph (corresponding to a directed or undirected neighborhood) is a tree then we say that the neighborhood is *tree-like*, otherwise, we say that it is not tree-like. Note that the neighborhood is tree-like if and only if all involved nodes are distinct. For $2 \leq d_v < d_c$, one can prove that the number of (distinct) nodes as well as the number of (distinct) edges in a neighborhood of a node or of an edge (undirected or directed) of depth 2ℓ is upper-bounded by $2(d_v d_c)^\ell$ (see [5, p. 92]).

C. Binary-Input Memoryless Channels

The class of channels we consider in this paper is the class of memoryless channels with binary input alphabet $\mathcal{I} := \{\pm 1\}$ and discrete or continuous output alphabet $\mathcal{O}$. We will, however, indicate some extensions in Section V.

Example 1 [BSC]: Let $\boldsymbol{x}_t$, $\boldsymbol{x}_t \in \mathcal{I} := \{\pm 1\}$, be the channel input at time t, $t \in \mathbb{Z}$. Let y_t, $y_t \in \mathcal{O} := \{\pm 1\}$, be the output at time t. The BSC with parameter ϵ is characterized by the relation $y_t = (-1)^{w_t} \boldsymbol{x}_t$, where w_t is a sequence of i.i.d. Bernoulli random variables with $\Pr\{w_t = 0\} = 1 - \epsilon$ and $\Pr\{w_t = 1\} = \epsilon$.

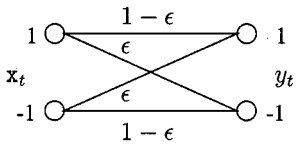

Fig. 3. The BSC with parameter ϵ.

The channel is depicted in Fig. 3. It is well known (see [12, p. 186]), that the (Shannon) capacity of this channel is

$$C_{\mathrm{BSC}}(\epsilon) := 1 - h(\epsilon) \tag{3}$$

where $h(x) := -x \log_2 x - (1-x) \log_2 (1-x)$ is the binary entropy function [12, p. 13].

Example 2 [Continuous Additive Channels: Gaussian and Laplace]: Consider a memoryless binary-input channel with continuous output alphabet and additive noise. More precisely, let the channel be modeled by $y_t := \boldsymbol{x}_t + z_t$, where $\boldsymbol{x}_t \in \{\pm 1\}$ and where z_t is a sequence of i.i.d. random variables with probability density $p(z)$. The best known example is the BIAWGN channel for which

$$p_{\mathrm{BIAWGN}}(z) := \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z^2}{2\sigma^2}}.$$

The capacity of the BIAWGN channel is given by

$$C_{\mathrm{BIAWGN}}(\sigma) := -\int \phi_\sigma(x) \log_2 \phi_\sigma(x)\, dx - \frac{1}{2} \log_2 (2\pi e \sigma^2) \tag{4}$$

where

$$\phi_\sigma(x) = \frac{1}{\sqrt{8\pi\sigma^2}} \left(e^{-\frac{(x+1)^2}{2\sigma^2}} + e^{-\frac{(x-1)^2}{2\sigma^2}} \right).$$

As a second example, we consider the BIL channel for which

$$p_{\mathrm{BIL}}(z) := \frac{1}{2\lambda} e^{-\frac{|z|}{\lambda}}.$$

The capacity of the BIL channel is given by

$$C_{\mathrm{BIL}}(\lambda) := \frac{-\pi + 4 \arctan(e^{-1/\lambda})}{2e^{1/\lambda} \log_e 2} - \log_2 \left(\frac{1 + e^{-2/\lambda}}{2} \right). \tag{5}$$

□

D. Message-Passing Decoders

Without loss of generality, we can assume that the channel output alphabet $\mathcal{O}$ is equal to the decoder input alphabet. Given a code and a channel model there are, in general, many reasonable decoding algorithms based on *message passing*—the class of algorithms we consider in this paper. These algorithms behave as follows. At time zero, every variable node v_i, $i \in [n]$, has an associated received message r_i, a random variable taking values in $\mathcal{O}$. Messages are exchanged between nodes in the graph along the edges in the graph in discrete time steps. First, each variable node v_i sends back to each neighboring check node c_j a *message* taking values in some *message alphabet* $\mathcal{M}$. Typically, at time zero, a variable node i sends r_i as its first message (this requires $\mathcal{O} \subset \mathcal{M}$). Each check node c_j processes the messages it receives and sends back to each neighboring variable node v_i a message taking values in $\mathcal{M}$. Each variable node v_i now processes the messages it receives together with its associated received value r_i to produce new messages which it then sends to its neighboring check nodes. For every time ℓ, $\ell \in \mathbb{N}$, a cycle or iteration of message passing proceeds with check nodes processing and transmitting messages followed by the variable nodes processing and transmitting messages.

An important condition on the processing is that a message sent from a node u along an adjacent edge e may not depend on the message previously received along edge e. There is a good reason for excluding the incoming message along edge e in determining the outgoing message along edge e. In turbo coding terminology, this guarantees that only *extrinsic* information is passed along. This is known to be an important property of good message-passing decoders. Even more importantly, it is exactly this restriction that makes it possible to analyze the behavior of the decoder.

Let $\Psi_{\mathsf{v}}^{(\ell)}: \mathcal{O} \times \mathcal{M}^{d_{\mathsf{v}}-1} \to \mathcal{M}$, $\ell \geq 1$, denote the variable node message map and let $\Psi_{\mathsf{c}}^{(\ell)}: \mathcal{M}^{d_{\mathsf{c}}-1} \to \mathcal{M}$, $\ell \geq 0$, denote the check node message map as a function of $\ell \in \mathbb{N}$. These functions represent the processing performed at the variable nodes and constraint nodes, respectively.

Note that, because of the imposed restriction on the dependence of messages, the outgoing message only depends on $(d_{\mathsf{v}} - 1)$ incoming messages at a variable node and $(d_{\mathsf{c}} - 1)$ incoming messages at a check node. Note also that we allow these maps to depend on the iteration number. We assume that each node of the same degree invokes the same message map for each edge and that all edges connected to such a node are treated equally. For completeness, let $\Psi_{\mathsf{v}}^{(0)}: \mathcal{O} \to \mathcal{M}$ denote the initial message map, i.e., node v_i initially transmits the message $\Psi_{\mathsf{v}}^{(0)}(r_i)$ to all of its neighbors.

For any given set A, let Π_A denote the space of probability distribution defined over A. In our analysis the messages will be random variables, and we are interested in tracking the evolution of their distributions during the execution of the algorithm. To this end, we define the map

$$^{\star}\Psi_{\mathsf{v}}^{(\ell)}: \Pi_{\mathcal{O}} \times \Pi_{\mathcal{M}}^{d_{\mathsf{v}}-1} \to \Pi_{\mathcal{M}}$$

as follows. If m_0 is a random variable distributed according to $P_0 \in \Pi_{\mathcal{O}}$, and m_i, $i = 1, \ldots, d_{\mathsf{v}} - 1$, are random variables distributed according to $P \in \Pi_{\mathcal{M}}$, and all indicated random variables are independent, then the distribution of $\Psi_{\mathsf{v}}^{(\ell)}(\mathsf{m}_0, \mathsf{m}_1, \ldots, \mathsf{m}_{d_{\mathsf{v}}-1})$ is, by definition, $^{\star}\Psi_{\mathsf{v}}^{(\ell)}(P_0, P, \ldots, P)$.

We define the map

$$^{\star}\Psi_{\mathsf{c}}^{(\ell)}: \Pi_{\mathcal{M}}^{d_{\mathsf{c}}-1} \to \Pi_{\mathcal{M}}$$

in an analogous way.

E. Symmetry Assumptions: Restriction to All-One Codeword

It is helpful to think of the messages (and the received values) in the following way. Each message represents an estimate of a particular codeword bit. More precisely, it contains an estimate of its *sign* and, possibly, some estimate of its *reliability*. To be concrete, consider a discrete case and assume that the output alphabet is

$$\mathcal{O} := \{-q_o, -(q_o - 1), \ldots, -1, 0, 1, \ldots, (q_o - 1), q_o\}$$

and that the message alphabet is

$$\mathcal{M} := \{-q, -(q-1), \ldots, -1, 0, 1, \ldots, (q-1), q\}.$$

The sign of the message indicates whether the transmitted bit is estimated to be -1 or $+1$, and the absolute value of the message is a measure of the reliability of this estimate. The sign of the particular value 0, which represents an *erasure*, is equally likely to be a $+1$ or a -1. In the continuous case we may assume that $\mathcal{O} = \mathcal{M} = \mathbb{R}$, or even $\mathcal{O} = \mathcal{M} = [-\infty, +\infty]$. Again, the sign of the message indicates whether the transmitted bit is estimated to be -1 or $+1$, and the absolute value of the message is a measure of the reliability of this estimate.

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

Definition 1 [Symmetry Conditions]:

- **Channel symmetry:** The channel is *output-symmetric*, i.e.,
$$p(y_t = q|\boldsymbol{x}_t = 1) = p(y_t = -q|\boldsymbol{x}_t = -1).$$
- **Check node symmetry:** Signs factor out of check node message maps
$$\Psi_{\mathsf{c}}^{(\ell)}(b_1\mathsf{m}_1, \ldots, b_{d_{\mathsf{c}}-1}\mathsf{m}_{d_{\mathsf{c}}-1}) = \Psi_{\mathsf{c}}^{(\ell)}(\mathsf{m}_1, \ldots, \mathsf{m}_{d_{\mathsf{c}}-1}) \left(\prod_{i=1}^{d_{\mathsf{c}}-1} b_i\right)$$
for any ± 1 sequence $(b_1, \ldots, b_{d_{\mathsf{c}}-1})$.
- **Variable node symmetry:** Sign inversion invariance of variable node message maps holds
$$\Psi_{\mathsf{v}}^{(\ell)}(-\mathsf{m}_0, -\mathsf{m}_1, \ldots, -\mathsf{m}_{d_{\mathsf{v}}-1}) = -\Psi_{\mathsf{v}}^{(\ell)}(\mathsf{m}_0, \mathsf{m}_1, \ldots, \mathsf{m}_{d_{\mathsf{v}}-1}), \qquad \ell \geq 1$$
and $\Psi_{\mathsf{v}}^{(0)}(-\mathsf{m}_0) = -\Psi_{\mathsf{v}}^{(0)}(\mathsf{m}_0)$.

Lemma 1 [Conditional Independence of Error Probability Under Symmetry]: Let $\mathcal{G}$ be the bipartite graph representing a given binary linear code (not necessarily an LDPC code) and for a given message-passing algorithm let $P_e^{(\ell)}(\boldsymbol{x})$ denote the conditional (bit or block) probability of error after the ℓth decoding iteration, assuming that codeword $\boldsymbol{x}$ was sent. If the channel and

the decoder fulfill the symmetry conditions stated in Definition 1 then $P_e^{(\ell)}(\boldsymbol{x})$ is independent of $\boldsymbol{x}$.

Proof: To prove the claim, let $p(q)$ denote the channel transition probability $p(y = q|\boldsymbol{x} = 1)$. Now note that any binary-input memoryless output-symmetric channel can be modeled multiplicatively as

$$y_t = \boldsymbol{x}_t z_t$$

where $\boldsymbol{x}_t$ is the input bit, y_t is the channel output, and z_t are i.i.d. random variables with distribution defined by $\Pr\{z_t = q\} = p(q)$. Let $\boldsymbol{x}$ denote an arbitrary codeword and let $y = \boldsymbol{x}z$ be an observation from the channel after transmitting $\boldsymbol{x}$, where z denotes the channel realization (multiplication is componentwise and all three quantities are vectors of length n).

Let v_i denote an arbitrary variable node and let c_j denote one of its neighboring check nodes. For any received word w, let $\mathsf{m}_{ij}^{(\ell)}(w)$ denote the message sent from v_i to c_j in iteration ℓ assuming w was received and let $\mathsf{m}_{ji}^{(\ell)}(w)$ denote the message sent from c_j to v_i in iteration ℓ assuming w was received. From the variable node symmetry at $\ell = 0$ we have $\mathsf{m}_{ij}^{(0)}(y) = \boldsymbol{x}_i \mathsf{m}_{ij}^{(0)}(z)$. Assume now that in iteration ℓ we have $\mathsf{m}_{ij}^{(\ell)}(y) = \boldsymbol{x}_i \mathsf{m}_{ij}^{(\ell)}(z)$. Since $\boldsymbol{x}$ is a codeword, we have $\prod_{k:\, \exists \mathsf{e} = (\mathsf{v}_k, \mathsf{c}_j)} \boldsymbol{x}_k = 1$.[7] From the check node symmetry condition we conclude that

$$\mathsf{m}_{ji}^{(\ell+1)}(y) = \boldsymbol{x}_i \mathsf{m}_{ji}^{(\ell+1)}(z).$$

Furthermore, from the variable node symmetry condition it follows that in iteration $\ell + 1$ the message sent from variable node v_i to check node c_j is

$$\mathsf{m}_{ij}^{(\ell+1)}(y) = \boldsymbol{x}_i \mathsf{m}_{ij}^{(\ell+1)}(z).$$

Thus, by induction, all messages to and from variable node v_i when y is received are equal to the product of $\boldsymbol{x}_i$ and the corresponding message when z is received. Hence, both decoders commit exactly the same number of errors (if any), which proves the claim. Moreover, we observe that the entire behavior of the decoder can be predicted from its behavior assuming transmission of the all-one codeword. □

In the sequel we will assume that the symmetry conditions are fulfilled and that the all-one codeword was transmitted.

III. Density Evolution and Threshold Determination

Recall from the introduction that there are three components to our proposed analysis of message-passing decoders: 1) *concentration*: show that almost all instances behave nearly the same; 2) *convergence to cycle-free case*: show that the average behavior converges (in the block length n) to that of the cycle-free case; 3) *density evolution and threshold determination*: analyze the cycle-free case via what we termed *density evolution* as well as determine the *threshold* for the cycle-free case.

In this section we will concentrate on the third step, i.e., we will assume that the decoding neighborhood of depth 2ℓ is tree-like and we will analyze the evolution of the densities as a function of the iteration number.

[7]In the case of parallel edges, $\boldsymbol{x}_k$ has to be counted according to the multiplicity of the edge.

We say that a message passed along an edge is *correct* if the sign of the message agrees with the transmitted bit, and we say that a message is *incorrect* otherwise. Recall from Section II-E that we may assume that the all-one codeword was transmitted, and, hence, the number of incorrect messages is equal to the number of messages with nonpositive sign. (By convention, we will count an erasure as being incorrect with probability $1/2$.)

We start in Section III-A by focusing on the case of message-passing decoders with *discrete* message alphabets. As remarked already, in this case the expected fraction of erroneous messages passed in the ℓth iteration can be expressed through a system of linked recursive functions which depend on $(d_\mathsf{v}, d_\mathsf{c})$ and the channel parameter. By investigating this system of recursive functions one can usually prove the existence of a threshold and this threshold can then be determined numerically to any desired degree of accuracy.[8] Furthermore, we show that by a judicious choice of the message maps one can construct iterative coding systems with a threshold which is surprisingly close to the ultimate (Shannon) limit at very low complexities. For example, in many cases, decoder E (see Example 5), which uses a message alphabet of size three, performs surprisingly close to belief propagation, which uses an infinite message alphabet, and significantly better than Gallager's decoder B (see Example 4), which uses a message alphabet of size two.

In Section III-B, we investigate the most important case of a message-passing decoder with an *infinite* message alphabet, namely, the sum-product or belief-propagation decoder. In this case, one has to keep track of the evolution of message *densities.* We will present an efficient algorithm to accomplish this task. We will also show that, in the case of the belief-propagation decoders, thresholds always exist if we consider a family of channels which can be ordered by *physical degradation.*

A. Discrete Alphabets

We start with message-passing algorithms that use discrete message alphabets. Let $p_k^{(0)}$, $k \in \{-q, \ldots, -1, 0, 1, \ldots, q\}$, be the probability that the message sent at time zero is equal to k. Let $p_k^{(\ell)}$ denote the corresponding probabilities at iteration ℓ (assuming that edge $\vec{\mathsf{e}}$ has a tree-like directed neighborhood up to at least depth 2ℓ). Similarly, let $q_k^{(\ell)}$ denote the probability of messages sent from check nodes to variable nodes in the ℓth iteration. Assume we are given a particular message-passing algorithm, i.e., particular message maps $\Psi_\mathsf{v}^{(\ell)}$ and $\Psi_\mathsf{c}^{(\ell)}$. As was shown by Gallager, it is then, in principle, possible to write down a recursion, expressing $p_k^{(\ell)}$ as a function of $p_k^{(0)}, p_k^{(\ell-1)}$ the code parameters $(d_\mathsf{v}, d_\mathsf{c})$, and the channel parameter.

Example 3 [Gallager's Decoding Algorithm A]: Consider the ensembles of $(d_\mathsf{v}, d_\mathsf{c})$-regular graphs. Let the channel be the BSC of Example 1 with input and output alphabet $\{-1, 1\}$ and crossover probability ϵ. Then, clearly, $p_1^{(0)} = 1 - \epsilon$ and $p_{-1}^{(0)} = \epsilon$. The message alphabet is given by $\mathcal{M} = \{-1, 1\}$. The message maps are time-invariant, i.e., they do not depend on the iteration number, and are given by $\Psi_\mathsf{v}(\mathsf{m}_0) = \mathsf{m}_0$, $\Psi_\mathsf{v}(\mathsf{m}_0, \mathsf{m}_1, \ldots, \mathsf{m}_{d_\mathsf{v}-1}) = -\mathsf{m}_0$ if $\mathsf{m}_1 = \mathsf{m}_2 = \cdots =$

[8]In some instance, it is even possible to determine the threshold analytically (see , e.g., [13]).

$m_{d_v-1} = -m_0$ and $\Psi_v(m_0, m_1, \ldots, m_{d_v-1}) = m_0$ otherwise, and

$$\Psi_c(m_1, \ldots, m_{d_c-1}) = \prod_{i=1}^{d_c-1} m_i.$$

In words, check nodes send a message indicating the modulo two sum of the *other* neighboring variables. The variable nodes send their received value unless the incoming messages are unanimous, in which case the sign indicated by these messages is sent.

It follows that

$$\begin{aligned}\left(q_{-1}^{(\ell)}, q_1^{(\ell)}\right) &= {}^{\star}\Psi_c\left(\left(p_{-1}^{(\ell-1)}, p_1^{(\ell-1)}\right), \ldots, \left(p_{-1}^{(\ell-1)}, p_1^{(\ell-1)}\right)\right) \\ &= \frac{1}{2}\left(1 - \left(1-2p_{-1}^{(\ell-1)}\right)^{d_c-1}, \right. \\ &\qquad \left. 1 + \left(1-2p_{-1}^{(\ell-1)}\right)^{d_c-1}\right)\end{aligned}$$

and

$$\begin{aligned}&\left(p_{-1}^{(\ell)}, p_1^{(\ell)}\right) \\ &\quad = {}^{\star}\Psi_v\left(\left(q_{-1}^{(\ell)}, q_1^{(\ell)}\right), \ldots, \left(q_{-1}^{(\ell)}, q_1^{(\ell)}\right)\right) \\ &\quad = \left(p_1^{(0)}\left(q_{-1}^{(\ell)}\right)^{d_v-1} + p_{-1}^{(0)}\left(1-(q_1^{(\ell)})^{d_v-1}\right), \right. \\ &\qquad \left. p_{-1}^{(0)}\left(q_1^{(\ell)}\right)^{d_v-1} + p_1^{(0)}\left(1-\left(q_{-1}^{(\ell)}\right)^{d_v-1}\right)\right).\end{aligned}$$

Eliminating q we obtain, as in Gallager [5, p. 48]

$$\begin{aligned}p_{-1}^{(\ell)} = p_{-1}^{(0)} - p_{-1}^{(0)}&\left[\frac{1+\left(1-2p_{-1}^{(\ell)-1}\right)^{d_c-1}}{2}\right]^{d_v-1} \\ &+ \left(1-p_{-1}^{(0)}\right)\left[\frac{1-\left(1-2p_{-1}^{(\ell-1)}\right)^{d_c-1}}{2}\right]^{d_v-1}. \quad (6)\end{aligned}$$

From (6) it is easy to see that, for a fixed value of $p_{-1}^{(\ell-1)}$, $p_{-1}^{(\ell)}$ is an increasing function of $p_{-1}^{(0)}$. Likewise, for a fixed value of $p_{-1}^{(0)}$, $p_{-1}^{(\ell)}$ is an increasing function of $p_{-1}^{(\ell-1)}$. Hence, by induction we conclude that $p_{-1}^{(\ell)}$ is an increasing function of $p_{-1}^{(0)}$. Let ϵ^* denote the supremum of all values of $p_{-1}^{(0)} \in [0, 1]$ such that $\lim_{\ell\to\infty} p_{-1}^{(\ell)} = 0$. By the above argument it follows that, for all values $p_{-1}^{(0)} < \epsilon^*$, $\lim_{\ell\to\infty} p_{-1}^{(\ell)} = 0$. These *threshold values* ϵ^* are listed for various pairs of (d_v, d_c) in Table I, where most of the entries were already given by Gallager.[9] We note that this decoding algorithm is *universal* in that it does not require knowledge of the channel parameter. □

Example 4 [Gallager's Decoding Algorithm B]: For $d_v > 3$ the following algorithm, also due to Gallager [5, p. 50], is more efficient. The message along edge $\vec{e} = (v_i, c_j)$ equals the received value r_i unless at least b incoming check messages (excluding the message along edge $\vec{e}$) disagree with the received value, in which case the opposite of r_i is sent. The value of $b = b^{(\ell)}$ is in general a function of d_v, d_c, and ℓ. Formally, we have the following: the message maps are the same as in the previous example, except that $\Psi_v^{(\ell)}(m_0, m_1, \ldots, m_{d_v-1}) = -m_0$ if $|\{i: m_i = -m_0\}| \geq b_\ell$ and $\Psi_v^{(\ell)}(m_0, m_1, \ldots, m_{d_v-1}) = m_0$ otherwise.

[9] *Exact* threshold values for this case can be found in [13].

TABLE I
MAXIMAL PARAMETER ϵ^* FOR THE BSC AND GALLAGER'S DECODING ALGORITHMS A AND B, ALGORITHM E (ERASURES IN THE DECODER), AND BELIEF PROPAGATION (BP). NOTE THAT ALGORITHMS A AND B ONLY DIFFER FOR $d_v > 3$. ALSO, LISTED IS THE MAXIMUM ALLOWED VALUE ϵ_{opt} ACCORDING TO (3)

d_v	d_c	Rate	ϵ^* (A)	ϵ^* (B)	ϵ^* (E)	ϵ^* (BP)	ϵ_{opt}
3	6	0.5	0.04	0.04	0.07	0.084	0.11
4	8	0.5	0.047	0.051	0.059	0.076	0.11
5	10	0.5	0.027	0.041	0.055	0.068	0.11
3	5	0.4	0.061	0.061	0.096	0.113	0.146
4	6	0.333	0.066	0.074	0.09	0.116	0.174
3	4	0.25	0.106	0.106	0.143	0.167	0.215

The evolution of $p_{-1}^{(\ell)}$ is now given by (see [5, p. 50])

$$\begin{aligned}p_{-1}^{(\ell)} = p_{-1}^{(0)} &- p_{-1}^{(0)} \sum_{k=b_l}^{d_v-1} \binom{d_v-1}{k} \\ &\cdot \left[\frac{1+\left(1-2p_{-1}^{(\ell-1)}\right)^{d_c-1}}{2}\right]^{k} \\ &\cdot \left[\frac{1-\left(1-2p_{-1}^{(\ell-1)}\right)^{d_c-1}}{2}\right]^{d_v-1-k} \\ &+ \left(1-p_{-1}^{(0)}\right) \sum_{k=b_l}^{d_v-1} \binom{d_v-1}{k} \\ &\cdot \left[\frac{1-\left(1-2p_{-1}^{(\ell-1)}\right)^{d_c-1}}{2}\right]^{k} \\ &\cdot \left[\frac{1+\left(1-2p_{-1}^{(\ell-1)}\right)^{d_c-1}}{2}\right]^{d_v-1-k}.\end{aligned}$$

The optimal choice of b_ℓ was also determined in [5] and is given by the smallest integer b for which

$$\frac{1-p_{-1}^{(0)}}{p_{-1}^{(0)}} \leq \left[\frac{1+\left(1-2p_{-1}^{(\ell-1)}\right)^{d_c-1}}{1-\left(1-2p_{-1}^{(\ell-1)}\right)^{d_c-1}}\right]^{2b-d_v+1}.$$

Again, let ϵ^* denote the supremum of all values of $p_{-1}^{(0)}$ such that $\lim_{\ell\to\infty} p_{-1}^{(\ell)} = 0$. By an argument similar to that in the previous example, it follows that, for all values $p_{-1}^{(0)} < \epsilon^*$, $\lim_{\ell\to\infty} p_{-1}^{(\ell)} = 0$. The threshold values ϵ^* are listed for various combinations of (d_v, d_c) in Table I.

Example 5 [BSC with Erasures in the Decoder:[10] Algorithm E]: Assume now that we extend the previous example by

[10] This decoder was independently conceived and analyzed by M. Mitzenmacher [14].

allowing erasures in the decoder, i.e., the alphabet is now $\mathcal{M} := \{-1, 0, 1\}$. The decoding rule is the following. At step ℓ, the message node v_i takes the real sum of all incoming messages, except the message along edge $\vec{e}=(\mathsf{c}_j, \mathsf{v}_i)$, plus $w^{(\ell)}$ times the received message r_i, where $w^{(\ell)}$ is an appropriately chosen weight. The message along edge $\vec{e} = (\mathsf{v}_i, \mathsf{c}_j)$ is now chosen as the sign of this sum. At check node c_j, the message along edge $\vec{e}=(\mathsf{c}_j, \mathsf{v}_i)$ is equal to the product of all incoming messages except the incoming message along edge $\vec{e}=(\mathsf{v}_i, \mathsf{c}_j)$. Thus, in this example, the message maps are specified by $\Psi_{\mathsf{v}}^{(0)}(\mathsf{m}_0) = \mathsf{m}_0$

$$\Psi_{\mathsf{v}}^{(l)}(\mathsf{m}_0, \mathsf{m}_1, \ldots, \mathsf{m}_{d_{\mathsf{v}}-1}) = \mathsf{sgn}(w^{(\ell)}\mathsf{m}_0 + \sum_{i=1}^{d_{\mathsf{v}}-1} \mathsf{m}_i)$$

and

$$\Psi_{\mathsf{c}}(\mathsf{m}_1, \ldots, \mathsf{m}_{d_{\mathsf{v}}-1}) = \prod_{i=1}^{d_{\mathsf{c}}-1} \mathsf{m}_i.$$

We easily obtain

$$q_1^{(\ell)} = \frac{1}{2}\left[\left(p_1^{(\ell-1)}+p_{-1}^{(\ell-1)}\right)^{d_{\mathsf{c}}-1} + \left(p_1^{(\ell-1)} - p_{-1}^{(\ell-1)}\right)^{d_{\mathsf{c}}-1}\right]$$
$$q_{-1}^{(\ell)} = \frac{1}{2}\left[\left(p_1^{(\ell-1)}+p_{-1}^{(\ell-1)}\right)^{d_{\mathsf{c}}-1} - \left(p_1^{(\ell-1)}-p_{-1}^{(\ell-1)}\right)^{d_{\mathsf{c}}-1}\right]$$
$$q_0^{(\ell)} = 1-\left(1-p_0^{(\ell-1)}\right)^{d_{\mathsf{c}}-1}.$$

These equations explicitly represent the equation

$$Q^{(\ell)} = {}^{\star}\Psi_{\mathsf{c}}^{(\ell)}(P^{(\ell-1)}, \ldots, P^{(\ell-1)})$$

where

$$P_\ell := (p_1^{(\ell)}, p_{-1}^{(\ell)}, p_0^{(\ell)})$$

and

$$P^{(\ell)} = {}^{\star}\Psi_{\mathsf{c}}^{(\ell)}(Q^{(\ell)}, \ldots, Q^{(\ell)}).$$

The quantities $p_0^{(\ell)}$, $p_1^{(\ell)}$, and $p_{-1}^{(\ell)}$ can now be expressed as

$$\begin{aligned} p_0^{(\ell)} &= p_0^{(0)} \sum_{(i,j):\, i-j=0} \binom{d_{\mathsf{v}}-1}{i,\, i,\, d_{\mathsf{v}}-1-2i} \\ &\quad \cdot \left(q_1^{(\ell-1)}\right)^i \left(q_{-1}^{(\ell-1)}\right)^i \left(q_0^{(\ell-1)}\right)^{d_{\mathsf{v}}-1-2i} \\ &\quad + p_1^{(0)} \sum_{(i,j):\, i-j=-w_\ell} \binom{d_{\mathsf{v}}-1}{i,\, j,\, d_{\mathsf{v}}-1-i-j} \\ &\quad \cdot \left(q_1^{(\ell-1)}\right)^i \left(q_{-1}^{(\ell-1)}\right)^j \left(q_0^{(\ell-1)}\right)^{d_{\mathsf{v}}-1-i-j} \\ &\quad + p_{-1}^{(0)} \sum_{(i,j):\, i-j=w_\ell} \binom{d_{\mathsf{v}}-1}{i,\, j,\, d_{\mathsf{v}}-1-i-j} \\ &\quad \cdot \left(q_1^{(\ell-1)}\right)^i \left(q_{-1}^{(\ell-1)}\right)^j \left(q_0^{(\ell-1)}\right)^{d_{\mathsf{v}}-1-i-j} \\ p_1^{(\ell)} &= p_0^{(0)} \sum_{(i,j):\, i-j>0} \binom{d_{\mathsf{v}}-1}{i,\, j,\, d_{\mathsf{v}}-1-i-j} \\ &\quad \cdot \left(q_1^{(\ell-1)}\right)^i \left(q_{-1}^{(\ell-1)}\right)^j \left(q_0^{(\ell-1)}\right)^{d_{\mathsf{v}}-1-i-j} \\ &\quad + p_1^{(0)} \sum_{(i,j):\, i-j>-w_\ell} \binom{d_{\mathsf{v}}-1}{i,\, j,\, d_{\mathsf{v}}-1-i-j} \\ &\quad \cdot \left(q_1^{(\ell-1)}\right)^i \left(q_{-1}^{(\ell-1)}\right)^j \left(q_0^{(\ell-1)}\right)^{d_{\mathsf{v}}-1-i-j} \\ &\quad + p_{-1}^{(0)} \sum_{(i,j):\, i-j>w_\ell} \binom{d_{\mathsf{v}}-1}{i,\, j,\, d_{\mathsf{v}}-1-i-j} \\ &\quad \cdot \left(q_1^{(\ell-1)}\right)^i \left(q_{-1}^{(\ell-1)}\right)^j \left(q_0^{(\ell-1)}\right)^{d_{\mathsf{v}}-1-i-j} \\ p_{-1}^{(\ell)} &= 1 - p_1^{(\ell)} - p_0^{(\ell)}. \end{aligned}$$

It remains to specify the sequence of weights $w^{(\ell)}$, $\ell \geq 1$ (at $\ell = 0$ there are no incoming messages at the variable nodes and, hence, no weight is used). Note that, at any given step ℓ, different choices of the weight $w^{(\ell)}$ will result in different densities $(p_{-1}^{(\ell)}, p_0^{(\ell)}, p_1^{(\ell)})$ and that there is no clear (linear) ordering among those alternatives. To find a good sequence of weights we could proceed in the following way. Assume that we would like to minimize $p_{-1}^{(\ell)}+\alpha p_0^{(\ell)}$ at the ℓth decoding iteration, where α is some positive number, e.g., $\alpha = 1/2$. We can then find the optimum weights $w^{(1)}, \ldots, w^{(\ell)}$ by means of dynamic programming. For the $(3, 6)$-regular code, the optimum weight sequence found by dynamic programming is $w^{(1)} = 2$, and $w^{(\ell)} = 1$, $\ell \geq 2$. The advantage of such an approach is that it is widely applicable regardless of how many alternative maps there are at any step and regardless of how many levels of quantization we have. The major drawback is that this scheme is computationally intensive and that it quickly becomes infeasible if the size of the message alphabet becomes large. It is often almost equally effective, and much less labor-intensive, to find sensible heuristics. In the sequel, we will assume that the weight $w^{(\ell)}$, $w^{(\ell)} \in \mathbb{N}$, maximizes

$$1 - p_0^{(\ell)} + h\left(p_0^{(\ell)}\right) - h\left(p_0^{(\ell)}, p_{-1}^{(\ell)}\right) \tag{7}$$

which is the capacity of a memoryless symmetric channel with binary input and ternary output with a crossover probability of $p_{-1}^{(\ell)}$ and an erasure probability of $p_0^{(\ell)}$. For the $(3, 6)$-regular code this leads to the same sequence of weights, but no claim is made regarding the optimality of this decision scheme in general. Table I summarizes the largest achievable parameters ϵ^* for the various code parameters. We see that all entries are significantly larger than their corresponding entries for Gallager A and B. Particularly impressive is the performance for the $(3, 6)$-regular code. We will later see that with belief propagation, the code has a threshold for the BSC of roughly 8.4%. Hence, the simple decoder with erasures performs almost as well as a belief-propagation decoder.

Example 6 [Quantized Continuous Channels with Erasures in the Decoder]: Encouraged by the large increase in the threshold for the BSC when allowing erasures in the decoder, we apply the same kind of decoder to continuous output channels.

Assume first that we pick a symmetric threshold τ around zero, and quantize the continuous output of the channel into negative values, erasures, and positive values, depending on whether the continuous output r fulfills $r \leq -\tau$, $-\tau < r < \tau$, or $r \geq \tau$, respectively.

We can now apply exactly the same algorithm as in the previous example. Clearly, the resulting threshold σ^* will depend on the choice of the threshold value τ, and we can try to find the optimum such threshold value. We can make even better use

of the received values if we adapt the quantization at every step of the decoding algorithm. More precisely, rather than picking the threshold value τ once and for all, we can pick a suitable threshold value τ_ℓ at every step of the decoding algorithm. At every step ℓ we pick that threshold τ_ℓ and that weight $w^{(\ell)}$ which maximize (7). For the BIAWGN channel and the $(3, 6)$-regular code, the threshold is $\sigma^* = 0.743$, which corresponds to a raw bit-error probability of roughly 8.9%. Hence, by providing soft information at the input to the decoder, even when the messages are only ternary, the threshold raw bit-error probability can be raised by almost 2% compared to a decoder which is fed hard decisions. □

Example 7 [Quantized Gaussian Channel, 3-Bit Messages, $(3, 6)$-Regular Codes]: From the previous example it is evident that a very coarse quantization of the input and a very small message alphabet can already result in very good performance. We now present a very simple time-invariant decoder whose performance on the $(3, 6)$-regular code over the Gaussian channel is close to the performance of belief propagation.

Let the message alphabet and the received alphabet be equal to

$$\mathcal{O} = \mathcal{M} = \{-4, -3, -2, -1, 1, 2, 3, 4\}.$$

In this example, we look at practical situation: a quantized version of the Gaussian channel. If the output of the channel at time t is $\boldsymbol{x}_t + z_t$ then the corresponding log-likelihood ratio $\log \frac{p_1}{p_{-1}}$ is given by $\frac{2}{\sigma^2}(\boldsymbol{x}_t + z_t)$. Let the threshold set τ be given by

$$\begin{aligned}\tau &= \{\tau_{-\infty}, \tau_{-3}, \tau_{-2}, \tau_{-1}, \tau_0, \tau_1, \tau_2, \tau_3, \tau_\infty\} \\ &= \{-\infty, -4.2, -2.8, -1.4, 0.0, 1.4, 2.8, 4.2, \infty\}\end{aligned}$$

and quantize the log-likelihood value according to these thresholds, e.g., if $1.4 \leq \frac{2}{\sigma^2}(X_t + Z_t) < 2.8$, then the associated received value is 2.

The check message map $\Psi_{\mathsf{c}}(\mathsf{m}_1, \ldots, \mathsf{m}_{d_{\mathsf{c}}-1})$ is specified as

$$\Psi_{\mathsf{c}}(\mathsf{m}_1, \ldots, \mathsf{m}_{d_{\mathsf{c}}-1}) = sM$$

where the *sign* s is given by

$$s := \prod_{i=1}^{d_{\mathsf{c}}-1} \operatorname{sgn} \mathsf{m}_i$$

and where M is the *reliability* specified by the following decision tree. Let n_q be the number of received messages out of all $(d_{\mathsf{c}} - 1)$ messages under consideration with reliability (absolute value) q, $q \in \{1, 2, 3, 4\}$. For simplicity, we represent the decision as a sequence of tests. Thus, in the list below, we perform the tests in order, stopping as soon as M is determined.

1) If $n_1 > 0$ or $n_2 > 3$ then $M = 1$.
2) If $n_2 > 1$ then $M = 2$.
3) If $n_2 = 1$ and $n_3 > 1$ then $M = 2$.
4) If $n_2 = 1$ or $n_3 > 2$ then $M = 3$.
5) $M = 4$.

We define the variable message map by first mapping the received messages and the incoming messages into suitable integers so that the decision rule can be based on the sum of all these values. Let

$$\phi_{\mathcal{O}}\colon \mathcal{O} \to \{-21, -15, -9, -3, 3, 9, 15, 21\}$$

be defined by

$$\phi_{\mathcal{O}}(m) = 6m - 3 \operatorname{sgn} m$$

and let

$$\phi_{\mathcal{M}}\colon \mathcal{M} \to \{-12, -8, -6, -2, 2, 6, 8, 12\}$$

be defined by

$$\phi_{\mathcal{M}}(m) = 2 \operatorname{sgn} m \left(|m| + \left\lfloor \frac{|m|}{2} \right\rfloor \right).$$

The variable message map $\Psi_{\mathsf{v}}(\mathsf{m}_0, \mathsf{m}_1, \mathsf{m}_2)$ is determined by $\phi_{\mathcal{O}}(\mathsf{m}_0) + \phi_{\mathcal{M}}(\mathsf{m}_1) + \phi_{\mathcal{M}}(\mathsf{m}_2)$ and by the threshold set

$$\{-\infty, -18, -12, -6, 0, 6, 12, 18, \infty\}.$$

For example, if $\mathsf{m}_0 = 1$, $\mathsf{m}_1 = -4$, and $\mathsf{m}_2 = 2$ then

$$\phi_{\mathcal{O}}(1) + \phi_{\mathcal{M}}(-4) + \phi_{\mathcal{M}}(2) = 3 - 12 + 6 = -3$$

so that the outgoing message is -1 (since $-6 < -3 < 0$).

It is evident that the computational requirements to implement this decoder are very small. This decoder on the $(3, 6)$-regular code has a threshold[11] value of $\sigma^* = 0.847$, corresponding to a raw error rate of about 11.9%. Later we will see that with belief propagation the code has a threshold of roughly $\sigma^* = 0.88$, equivalent to a raw error rate of about 13.0%. Hence, the performance of this simple decoder is very close to that of belief propagation.

Actually, as specified, this decoder has an error floor of about 0.06%. The error floor is easily removed by replacing

$$\phi_{\mathcal{O}}\colon \mathcal{O} \to \{-21, -15, -9, -3, 3, 9, 15, 21\}$$

with the corresponding

$$\phi_{\mathcal{O}}\colon \mathcal{O} \to \{-9, -9, -9, -3, 3, 9, 9, 9\}$$

after sufficiently many iterations. □

From the previous examples the general method should now be clear. For a given discrete memoryless channel and a decoder with discrete message alphabet it is always possible, although cumbersome, to write down the recursion describing the evolution of the fraction of incorrect messages which are passed. If the channel is parameterized by a single real value, then we can find the supremum of all such values for which the fraction of incorrect messages passed at the ℓth decoding iteration converges to zero as ℓ tends to infinity. This threshold value is, of course, a function of the code parameters. We will see in Section IV that this quantity, which we derived under the assumption that the graph did not contain small cycles describes the average behavior over all inputs. Quite surprisingly, it also describes the behavior for almost all randomly chosen graphs and almost all inputs up to an arbitrarily small deviation, once the code length is sufficiently large.

It should also be clear from the previous examples that there is an enormously rich family of message-passing algorithms, whose surface we have only scratched, and that by choosing different elements of this family we can trade off complexity with performance. From a practical point of view, it is interesting to note that very simple decoders (like the ones used in Examples 5 and 6) can already yield extremely good performance.

[11]We have not demonstrated monotonicity of this decoder, but monotonicity is apparent.

B. Continuous Message Alphabets: Belief Propagation

Assume now that the message alphabet is continuous and that the channel output is discrete or continuous. From our previous experience with binary or ternary message-passing algorithms, it seems a formidable task to determine the threshold in the continuous case. But for the most important case, namely, the case of a belief-propagation decoder, there is an efficient way of describing and evaluating density evolution, which we will now present.

To begin, we will assume that our alphabets are the reals and that probability density functions of interest are absolutely continuous with respect to Lebesgue measure. This assumption is not required, and we will consider discrete (input) probability densities later, but it simplifies the mathematical statements. Throughout, we shall refer to probability density functions simply as densities.

Roughly speaking, in belief propagation the messages sent on an edge e represent posterior densities on the bit associated with the incident variable node. A (conditional) probability density on a bit is a pair of nonnegative reals p_{-1}, p_1 satisfying $p_{-1} + p_1 = 1$. Such a pair can be represented by the corresponding log-likelihood ratio $\log \frac{p_1}{p_{-1}}$ and, to be specific, we shall assume that the messages use this representation. Each node acts under the assumption that each density communicated to it in a given round is a conditional distribution on the bit, and that each message is conditionally independent of all others, i.e., the random variables on which the different messages are based are independent. After receiving its messages, a node transmits to each neighboring node the conditional distribution of the bit conditioned on *all* information not coming from that particular neighboring node. More precisely, consider the message emitted by a variable node v_i along the edge $\vec{\mathsf{e}} = (\mathsf{v}_i, \mathsf{c}_j)$ in the ℓth decoding round. Assuming that there are no cycles of length 2ℓ or less, this message is $\log \frac{p_1}{p_{-1}}$, where p_{-1} is the *a posteriori* probability that the value of the associated variable node is -1 given the observed values of all nodes in the directed neighborhood $\mathcal{N}_{\vec{\mathsf{e}}}^{2\ell}$. An equivalent statement is true for the messages sent out from check nodes.

If $l_1, l_2, \ldots, l_k$ are likelihood ratios of conditional distributions of a given bit value conditioned on independent random variables, then the likelihood ratio of the bit value conditioned on all of the random variables is $\prod_{i=1}^{k} l_i$. Therefore, since we are using log likelihoods for messages, we have

$$\Psi_{\mathsf{v}}(\mathsf{m}_0, \mathsf{m}_1, \ldots, \mathsf{m}_{d_{\mathsf{v}}-1}) := \sum_{i=0}^{d_{\mathsf{v}}-1} \mathsf{m}_i.$$

Given densities on the real quantities $\mathsf{m}_0, \mathsf{m}_1, \ldots, \mathsf{m}_{d_{\mathsf{v}}-1}$, it follows that the density of $\Psi_{\mathsf{v}}(\mathsf{m}_0, \ldots, \mathsf{m}_{d_{\mathsf{v}}-1})$ is simply the convolution (over the reals) of those densities. In other words, we have

$${}^{\star}\Psi_{\mathsf{v}}(P_0, P_1, \ldots, P_{d_{\mathsf{v}}-1}) = P_0 \otimes P_1 \otimes \cdots \otimes P_{d_{\mathsf{v}}-1}$$

where $\otimes$ denotes convolution. Letting $\mathcal{F}$ denote the Fourier transform (over the reals in this case) we have

$${}^{\star}\Psi_{\mathsf{v}}(P_0, P_1, \ldots, P_{d_{\mathsf{v}}-1}) = \mathcal{F}^{-1}(\mathcal{F}(P_0)\mathcal{F}(P_1)\ldots\mathcal{F}(P_{d_{\mathsf{v}}-1})).$$

In our computations, we shall have $P_1 = P_2 = \cdots = P_{d_{\mathsf{v}}-1}$ and all densities will be quantized; we can therefore efficiently obtain ${}^{\star}\Psi_{\mathsf{v}}(P_0, P_1, \ldots, P_{d_{\mathsf{v}}-1})$ using the fast Fourier transform (FFT).

Let us now consider a check node c_j. We are interested in the outgoing message along edge $\vec{\mathsf{e}} = (\mathsf{c}_j, \mathsf{v}_i)$, where v_i is some variable node incident to c_j. Label all other edges (excluding e) incident to c_j with the set $[d_{\mathsf{c}} - 1]$, and let the incoming message along edge $k \in [d_{\mathsf{c}} - 1]$ be given by $\log \frac{p_1^k}{p_{-1}^k}$. We can think of each such message as representing a random variable on $\{\pm 1\}$ that takes on the value ± 1 with probability $p_{\pm 1}^k$. Furthermore, all of these $(d_{\mathsf{c}} - 1)$ random variables are assumed independent. The outgoing message along edge e is then $\log \frac{\tilde{p}_1}{\tilde{p}_{-1}}$, where $\tilde{p}_{\pm 1}$ is the probability that the product of these random variables has value ± 1.

To calculate $\tilde{p}_{\pm 1}$ it is more convenient to switch to the following alternative representation: assume that the random variables take values in $\{0, 1\}$, instead of $\{\pm 1\}$, with the correspondence $0 \leftrightarrow 1$, $1 \leftrightarrow -1$, $p_0 \leftrightarrow p_1$, and $p_1 \leftrightarrow p_{-1}$. The value $\tilde{p}_0$ is now the probability that the $\bmod 2$ sum of the $(d_{\mathsf{c}} - 1)$ independent $\{0, 1\}$-valued random variables is equal to zero.

Note that the probability vector $(\tilde{p}_0, \tilde{p}_1)$ is given by the cyclic convolution of the $(d_{\mathsf{c}} - 1)$ probability vectors (p_0^k, p_1^k). An efficient way of performing these convolutions is by means of a Fourier transform. In general, if f represents a function over $\mathrm{GF}(2) = \mathbb{Z}/2\mathbb{Z}$, then its Fourier transform is defined over $\{0, 1\}$ and is given by

$$\mathcal{F}(f)(0) = f(0) + f(1)$$

and

$$\mathcal{F}(f)(1) = f(0) - f(1).$$

When f is a probability mass function, we have, of course, $\mathcal{F}(f)(0) = 1$. Thus, we have

$$\tilde{p}_0 - \tilde{p}_1 = \prod_{k=1}^{d_{\mathsf{c}}-1} (p_0^k - p_1^k)$$

(and also $\tilde{p}_0 + \tilde{p}_1 = \prod_{k=1}^{d_{\mathsf{c}}-1}(p_0^k + p_1^k) = 1$).

If m is a log-likelihood $\log \frac{p_0}{p_1}$, then it follows that

$$p_0 - p_1 = \frac{e^{\mathsf{m}} - 1}{e^{\mathsf{m}} + 1} = \tanh\left(\frac{\mathsf{m}}{2}\right).$$

Conversely, if $q = p_0 - p_1$, then $\log \frac{p_0}{p_1} = \log \frac{1+q}{1-q}$. Thus, the appropriate definition for the check message map is

$$\Psi_{\mathsf{c}}(\mathsf{m}_1, \ldots, \mathsf{m}_{d_{\mathsf{c}}-1}) := \log\left(\frac{1 + \prod_{i=1}^{d_{\mathsf{c}}-1} \tanh \frac{1}{2}\mathsf{m}_i}{1 - \prod_{i=1}^{d_{\mathsf{c}}-1} \tanh \frac{1}{2}\mathsf{m}_i}\right).$$

Let us consider the evolution of the densities of the messages at the check nodes according to the above scheme. It turns out that, for this purpose, the check node update is more conveniently expressed using yet another representation of the messages. A probability density (p_0, p_1) can be represented as a log-likelihood $\log \frac{p_0}{p_1}$, as used above, and it can also be represented using the following ordered pair:

$$(\mathsf{lg\,sgn}(p_0 - p_1), -\log|p_0 - p_1|) \in \mathrm{GF}(2) \times [0, \infty)$$

where we define

$$\mathsf{lg\,sgn}\, q := \begin{cases} 1, & q < 0 \\ 0, & q > 0. \end{cases}$$

(Ideally, $\mathsf{lg\,sgn}0$ is defined probabilistically, taking either value 0 or 1 with equal probability.) Note that the above essentially decomposes $p_0 - p_1$ into its sign and its (log) magnitude. The advantage of this representation arises from the fact that, under this representation, the check node message map in the given space $\mathrm{GF}(2) \times [0, \infty)$ is simply addition.

Given a density P of log-likelihoods we can find the equivalent density $\tilde{P}$ over $\mathrm{GF}(2) \times [0, \infty)$ by making the appropriate change of measure. Let m denote a log-likelihood with density P, then we wish to derive the density of $(\mathsf{lg\,sgn}\mathsf{m}, -\log|\tanh(\frac{\mathsf{m}}{2})|))$, where we have used the fact that

$$\mathsf{lg\,sgn}\,\tanh\left(\frac{\mathsf{m}}{2}\right) = \mathsf{lg\,sgn}\,\mathsf{m}.$$

If $\mathsf{m} > 0$, then, letting $y := -\log\tanh(\frac{\mathsf{m}}{2})$, we find the surprising symmetrical relation $\mathsf{m} = -\log\tanh(\frac{y}{2})$. Similarly, if $\mathsf{m} < 0$ then letting

$$y := -\log\left(-\tanh\left(\frac{\mathsf{m}}{2}\right)\right) = -\log\left(\tanh\left(\frac{-\mathsf{m}}{2}\right)\right)$$

we have $-\mathsf{m} = -\log\tanh(\frac{y}{2})$. Let $\tilde{P}^0(y)$, $y \in [0, \infty)$, be defined by $\tilde{P}^0(y) = \tilde{P}(y, 0)$ and let $\tilde{P}^1(y)$, $y \in [0, \infty)$, be defined by $\tilde{P}^1(y) = \tilde{P}(y, 1)$. By differentiating we obtain (for $y > 0$)

$$\tilde{P}^0(y) = \frac{1}{\sinh(y)}\, P\left(-\log\tanh\frac{y}{2}\right)$$

and

$$\tilde{P}^1(y) = \frac{1}{\sinh(y)}\, P\left(\log\tanh\frac{y}{2}\right).$$

Similarly, given $\tilde{P}$ we have

$$P(\mathsf{m}) = \frac{1}{\sinh(\mathsf{m})}\, \tilde{P}^0\left(-\log\tanh\frac{\mathsf{m}}{2}\right)$$

for $\mathsf{m} > 0$ and, for $\mathsf{m} < 0$

$$P(\mathsf{m}) = \frac{1}{\sinh(-\mathsf{m})}\, \tilde{P}^1\left(-\log\tanh\frac{-\mathsf{m}}{2}\right).$$

Note that the space $\mathrm{GF}(2) \times [0, \infty)$ is the direct product of two fields and that we are interested in computing the densities of the sum of putatively independent random variables over this space given their individual densities. It follows, and this is the point of the representation, that the density of this sum is the convolution of the densities of the summands. This convolution can be efficiently computed using (generalized) Fourier transforms. Here, the Fourier transform is the Laplace transform in the second coordinate (in practice, we use the Fourier transform over the reals) and the discrete Fourier transform over $\mathrm{GF}(2)$ in the first coordinate.

More specifically, let $\hat{\tilde{P}}^0$, $\hat{\tilde{P}}^1$ denote the Laplace transforms of $\tilde{P}^0$, $\tilde{P}^1$, respectively. The Fourier transform of the density $\tilde{P}$ is given by

$$\mathcal{F}(\tilde{P})(s, 0) = \hat{\tilde{P}}^0(s) + \hat{\tilde{P}}^1(s)$$

and

$$\mathcal{F}(\tilde{P})(s, 1) = \hat{\tilde{P}}^0(s) - \hat{\tilde{P}}^1(s).$$

Let $\tilde{Q}$ denote the density of $\sum_{i=1}^{k} \tilde{\mathsf{m}}_i$ where $\tilde{\mathsf{m}}_i \in \mathrm{GF}(2) \times [0, \infty)$ corresponds to a log-likelihood message m_i distributed according to P_i. We then have

$$\mathcal{F}(\tilde{Q})(s, 0) = \prod_{i=1}^{k} (\hat{\tilde{P}}_i^0(s) + \hat{\tilde{P}}_i^1(s))$$

and

$$\mathcal{F}(\tilde{Q})(s, 1) = \prod_{i=1}^{k} (\hat{\tilde{P}}_i^0(s) - \hat{\tilde{P}}_i^1(s)).$$

Thus, $\tilde{Q}$ may be obtained by performing the inverse Fourier transform. The density $Q = {}^{\star}\Psi_{\mathsf{c}}(P_1, \ldots, P_{d_{\mathsf{c}}-1})$ can now be obtained from $\tilde{Q}$ by a change of variables.

Let us summarize how we perform density evolution for belief propagation for $(d_{\mathsf{v}}, d_{\mathsf{c}})$-regular graphs. Let $P^{(\ell)}$ denote the common density associated with the messages from variable nodes to check nodes in the lth round, and let P_0 denote the density of the received values. First we find the density $\tilde{P}^{(\ell)}$ corresponding to $P^{(\ell)}$. This is done using the change of measure described above. We then determine the density $\tilde{Q}^{(\ell)}$ according to

$$\begin{aligned} \hat{\tilde{Q}}^{(\ell),0} - \hat{\tilde{Q}}^{(\ell),1} &= \left(\hat{\tilde{P}}^{(\ell-1),0} - \hat{\tilde{P}}^{(\ell-1),1}\right)^{d_{\mathsf{c}}-1} \\ \hat{\tilde{Q}}^{(\ell),0} + \hat{\tilde{Q}}^{(\ell),1} &= \left(\hat{\tilde{P}}^{(\ell-1),0} + \hat{\tilde{P}}^{(\ell-1),1}\right)^{d_{\mathsf{c}}-1} \end{aligned} \tag{8}$$

where $\hat{\tilde{Q}}^{(\ell),s}$ denotes the Laplace transform of $\tilde{Q}^{(\ell),s}$. Finally, we obtain $Q^{(\ell)}$ by performing the appropriate change of measure.

Thus, completing the iteration, we have

$$\mathcal{F}\left(P^{(\ell+1)}\right) = \mathcal{F}\left(P^{(0)}\right)\left(\mathcal{F}\left(Q^{(\ell)}\right)\right)^{d_{\mathsf{v}}-1}. \tag{9}$$

The above described algorithm enables one to effectively compute $P^{(\ell+1)}$ from $P^{(\ell)}$ via judicious use of the FFT.

Although we have assumed that the density $P^{(0)}$ is absolutely continuous with respect to Lebesgue measure, the analysis above easily extends to more general densities. In particular, we may consider discrete densities. In some cases, we may have $\int_{0-}^{0+} P^{(\ell)}(x)\,dx > 0$ for some ℓ, in which case special consideration must be taken for this probability mass. This can be done in a straightforward manner, as we indicated earlier, and will not be described further.

The above algorithm can also be used to study quantization effects. If the messages are quantized versions of the messages that would be obtained using belief propagation, then the performance can be evaluated as above by simply, for given messages, computing the density of return messages assuming belief propagation, and then quantizing the densities in accordance with the quantization of the messages.

Essentially, all our finite-alphabet examples can be interpreted as quantized versions of belief propagation. The eight-level message passing algorithm of Example 7, in particular, was designed by hand to approximate belief propagation near the threshold value for that decoder. The quantization levels were chosen visually from plots of the message densities arising from belief propagation and partially quantized belief propagation (we introduce quantization in stages). The rules given to describe the algorithm are consistent with the effect of the quantization but are written in a form which emphasizes their simplicity. We have not attempted to properly optimize the quantization levels, and we restricted ourselves to time-invariant quantization. Clearly, the algorithm is not the optimal eight-level algorithm but it is surely close and its

construction indicates general principles that can guide the design of low-complexity decoders.

1) Monotonicity—Threshold Effects: Assume we are given a class of channels fulfilling the required symmetry condition and that this class is parameterized by α. This parameter may be real-valued, as is the case for all our examples (the crossover probability ϵ for the BSC, the standard deviation σ for the BIAWGNC, the parameter λ for the BILC), or may take values in a larger domain. For a fixed parameter α we can use the above algorithm to determine if, for a given $(d_{\mathsf{v}}, d_{\mathsf{c}})$, the expected fraction of incorrect messages tends to zero with an increasing number of iterations under density evolution.

For all our examples, the parameter α reflects a natural ordering of the channels—the capacity decreases with increasing parameter α. It is, therefore, natural to ask whether in such cases the convergence for a parameter α automatically implies the convergence for every parameter α' such that $\alpha' \leq \alpha$. More generally, we might want to define a partial ordering of channels with respect to a given code and a belief-propagation decoder.

Let a channel W be represented by its transition probability $p_W(y|\boldsymbol{x})$. We say that a channel W' is *physically degraded* with respect to W if $p_{W'}(y'|\boldsymbol{x}) = p_Q(y'|y)p_W(y|\boldsymbol{x})$ for some auxiliary channel Q, see [12].

Theorem 1 [Monotonicity for Physically Degraded Channels]: Let W and W' be two given memoryless 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 ℓth decoding iteration assuming tree-like neighborhoods and transmission over channel W, and let p' denote the equivalent quantity for transmission over channel W'. Then $p \leq p'$.

Proof: Let $\vec{e} = (\mathsf{v}_i, \mathsf{c}_j)$ be a given edge and assume that $\mathcal{N}_{\vec{e}}^{2\ell}$ is tree-like. Let R be the result of passing a randomly chosen codeword through channel W, and let R' be the result of passing R through an additional auxiliary channel Q which is chosen so that the concatenation of W and Q results in the channel W'. Consider the following three maximum-likelihood (ML) estimators of the bit value associated to v_i given all observations in $\mathcal{N}_{\vec{e}}^{2\ell}$. The first estimator has input R, the second one has input (R, R'), and the third one has input R'. Since the transmitted bit associated with variable node v_i has uniform *a priori* probability, an ML estimator is equal to a maximum *a posteriori* estimator, which is known to yield the minimum probability of error of all estimators based on the same observation. The first two of the above three estimators clearly yield the same probability of error since the additional input R' is conditionally independent of the transmitted bit given R. Furthermore, we can think of the third estimator as an estimator which has input (R, R') but ignores R in his decision. Clearly, such an estimator cannot have smaller probability of error than the optimal (ML) estimator given input (R, R').

The claim now follows by observing that, for a belief-propagation decoder, the sign of the message sent along edge $\vec{e}$ in the ℓth decoding iteration is equal to the estimate of an ML estimator based on the observations in $\mathcal{N}_{\vec{e}}^{2\ell}$ [15]. Our above statements then translate into the fact that a belief-propagation decoder operating on the output of the channel W has a smaller probability of error than a belief-propagation decoder operating on the output of the channel W', i.e., $p \leq p'$. □

If a class of channels is parameterized by a real-valued parameter α, we will say that the class of channels is *monotone with respect to a particular decoder* if the convergence for a parameter α implies the converge for every parameter α' such that $\alpha' < \alpha$.

Example 8 [BSC: Monotonicity by Self-Concatenation]: Assume we are given two BSCs with parameters ϵ' and ϵ, respectively. It is then easy to see that their concatenation (in any order) is again a BSC and that the parameter of the concatenated channel is

$$\epsilon'' = (1-\epsilon')\epsilon + (1-\epsilon)\epsilon'. \tag{10}$$

Moreover, for any $\epsilon'' \geq 0$ and any $\epsilon' \leq \epsilon''$ there exists a positive ϵ such that (10) is fulfilled. By Theorem 1, this proves the monotonicity of the class of BSCs with respect to a belief-propagation decoder.

Equivalent statements are true for the class of BECs, the class of BIAWGNCs channels, and the class of Cauchy channels with

$$p_{\text{Cauchy}}(z) := \frac{\lambda}{\pi(\lambda^2 + z^2)}.$$ □

Example 9 [Monotonicity of the BILC]: As discussed in Section II, for the class of BILCs we have

$$p_{\text{BIL}}(z) := \frac{1}{2\lambda}\, e^{-\frac{|z|}{\lambda}}.$$

Although the concatenation of a BILC with a Laplace channel (LC) is not a BILC, one can check that the concatenation of a BILC with parameter λ' with an additive memoryless channel with

$$p(z) := \left(\frac{\lambda}{\lambda'}\right)^2 \delta(z) + \left(1 - \left(\frac{\lambda}{\lambda'}\right)^2\right) \frac{1}{2\lambda}\, e^{-\frac{|z|}{\lambda}}$$

where δ denotes the Dirac delta function, results in a BILC with parameter λ. Hence, by Theorem 1, the class of BILC is monotone with respect to a belief-propagation decoder. □

2) Thresholds for Belief Propagation—Examples: For each of the examples below we computed the corresponding threshold value. Note that, in order to carry out the computations, the involved quantities have to be discretized. In order to avoid some of the resulting numerical precision problems, we made sure that the discretization was symmetric with respect to the input parity. As a consequence, the computations correspond to actual implementable message-passing algorithms (quantized, possibly probabilistic, approximations of belief propagation). Since the values reported are those for which the probability of error converged to zero (within machine precision,) it follows that the reported thresholds are guaranteed to be lower bounds. In all cases, the values presented are believed (after extensive numerical checking) to be accurate (rounded down) up to the number of digits given.

TABLE II
THRESHOLD VALUE σ^* FOR THE BIAWGNC UNDER BELIEF PROPAGATION FOR VARIOUS CODE PARAMETERS. ALSO LISTED IS THE MAXIMUM ALLOWED VALUE σ_{opt} ACCORDING TO (4)

d_{v}	d_{c}	Rate	σ^*	σ_{opt}
3	6	0.5	0.88	0.979
4	8	0.5	0.83	0.979
5	10	0.5	0.79	0.979
3	5	0.4	1.0	1.148
4	6	0.333	1.01	1.295
3	4	0.25	1.26	1.549

TABLE III
THRESHOLD VALUE λ^* FOR THE BILC UNDER BELIEF PROPAGATION FOR VARIOUS CODE PARAMETERS. ALSO LISTED IS THE MAXIMUM ALLOWED VALUE λ_{opt} ACCORDING TO EQUATION (5)

d_{v}	d_{c}	Rate	λ^*	λ_{opt}
3	6	0.5	0.65	0.752
4	8	0.5	0.62	0.752
5	10	0.5	0.58	0.752
3	5	0.4	0.77	0.914
4	6	0.333	0.78	1.055
3	4	0.25	1.02	1.298

Example 10 [BSC Under Belief Propagation]: We start by applying the above algorithm to the BSC with crossover probability ϵ. The initial density P_0 is given by

$$P_0(x)=\epsilon\delta\left(x+\log\left(\frac{1-\epsilon}{\epsilon}\right)\right)+(1-\epsilon)\delta\left(x-\log\left(\frac{1-\epsilon}{\epsilon}\right)\right).$$

The resulting threshold values are given in Table I.

Example 11 [BIAWGN Channel Under Belief Propagation]: We next apply the above algorithm to the BIAWGN channel with standard deviation σ. In this case we have

$$P_0\left(\frac{2}{\sigma^2}x\right)=\frac{\sigma}{2\sqrt{2\pi}}\exp\frac{-(x-1)^2}{2\sigma^2}.$$

The resulting threshold values are given in Table II. □

Example 12 [BILC Under Belief Propagation]: As a final example, we apply the above algorithm to the BILC with parameter λ. In this case we have

$$P_0(x)=\frac{1}{2}\,\delta\left(x+\frac{2}{\lambda}\right)+\frac{1}{2}\,e^{-\frac{2}{\lambda}}\delta\left(x-\frac{2}{\lambda}\right)+\frac{1}{4}\,e^{-\frac{x+\frac{2}{\lambda}}{2}}\chi_{|x|\le\frac{2}{\lambda}}$$

where

$$\chi_{|x|\le\frac{2}{\lambda}}:=\begin{cases}1, & |x|\le\frac{2}{\lambda}\\ 0, & |x|>\frac{2}{\lambda}.\end{cases}$$

The resulting threshold values are given in Table III. □

As we have seen in the previous example, the monotonicity theorem is a powerful tool. Nevertheless, in some situations it would be desirable to have alternative sufficient conditions for convergence of the density evolution according to a belief-propagation decoder. These conditions should depend only on the shape of the input density, regardless of whether this density corresponds to the output of an actual channel or not. One situation where such a criterion would be of great value is the situation of a channel mismatch. Assume, for example, that it is known that the channel belongs to a certain class parameterized by α but the exact value of α is unknown. Assume further that the class is monotone and that the threshold is α^*. In such a situation, it is natural to inquire if a belief-propagation decoder is "universal" in the sense that it may assume a parameter α^* and that it will decode correctly with high probability as long as the actual parameter α is strictly less than α^*. Unfortunately, this question cannot be answered by means of the monotonicity theorem since the input to the decoder will, in general, not correspond to the output of an actual channel.

IV. CONCENTRATION AND CONVERGENCE TO THE CYCLE-FREE CASE

In this section, we will show that the average behavior of individual instances (of the code and the noise) concentrates around its expected behavior when the length of the code grows and we will show that this average behavior converges to the behavior of the cycle-free case.

The concentration result which we derive applies regardless of whether the average fraction of incorrect messages passed at the ℓth iteration converges to zero as $\ell\to\infty$ or not. Hence, assuming that the codeword length n is large enough, for almost all codes in the ensemble $\mathcal{C}^n(d_{\text{v}},d_{\text{c}})$ transmission will be reliable if and only if the parameter of the channel is below the calculated threshold value. More precisely, if the parameter of the channel is below the calculated threshold value, then, given any arbitrarily small target error probability, it can be achieved using ℓ iterations of the decoder (for some fixed ℓ depending on the target error probability) with probability approaching one exponentially in n by choosing a code at random from $\mathcal{C}^n(d_{\text{v}},d_{\text{c}})$. If the parameter of the channel is above the calculated threshold value, then there is an $\epsilon>0$ such that, for any fixed number of iterations ℓ, the decoder will yield an error rate larger than ϵ with probability approaching one exponentially in n when using a code chosen at random from $\mathcal{C}^n(d_{\text{v}},d_{\text{c}})$. (Note that if we could prove that loops in the graph degrade average performance, and we conjecture that this is actually the case, then we could allow the number of iterations to be arbitrary.)

In order to simplify some of the subsequent notation, we will assume that the number of iterations that the decoder performs is fixed and we will denote this number by ℓ. All subsequent quantities are then the quantities at iteration ℓ and, hence, we often omit the index ℓ.

Assume that we are in the ℓth iteration and recall that the message passed from variable node v to check node c is a function of the chosen graph and the input to the decoder. Let Z be the number of incorrect messages among all $d_{\text{v}}n$ variable-to-check messages sent out in the ℓth iteration and let $\mathbb{E}[Z]$ be the expected value of Z where the expectation is over all graphs and all decoder inputs. For a given edge $\vec{\mathsf{e}}$ whose directed neighborhood of depth 2ℓ is tree-like, let p be the expected number of incorrect messages (including half the number of erasures) passed along this edge at the ℓth iteration, averaged over all inputs. We have seen in the previous section how p can be determined in

most cases of interest. Note that in the case of continuous message alphabets, p is given as

$$p := \int_{-\infty}^{0^-} P(x)\,dx + \tfrac{1}{2}\int_{0^-}^{0^+} P(x)\,dx$$

where $P(x)$ describes the density of the messages at the ℓth iteration. Although we will not make use of this fact in the sequel, we note that the same concentration result applies to the more general case of a random variable Z_x that corresponds to messages in the range of $(-\infty, x]$ for any $x \in \mathbb{R}$. The previous case is then just the special case $Z = Z_0$. More importantly, since Z_x and $\mathbb{E}[Z_x]$ are increasing functions with range $[0, 1]$, uniform convergence to within ϵ follows from convergence at finitely many points determined by $\mathbb{E}[Z_x]$. For example, if $\mathbb{E}[Z_x]$ is continuous in x, then we may take the points where $\mathbb{E}[Z_x] = \epsilon/2 + m\epsilon$ for $m = 0, 1, \ldots, \lceil \frac{1}{\epsilon} \rceil - 1$. If Z_x is within $\epsilon/2$ of $\mathbb{E}[Z_x]$ at these points, then Z_x is within ϵ of $\mathbb{E}[Z_x]$ for all x. If $\mathbb{E}[Z_x]$ is discontinuous, then, in general, we must consider left and right limits at each x and we obtain uniform convergence of these limits. These uniform convergence results show that the cumulative message distributions computed for tree-like neighborhoods uniformly approximates the actual message distributions to within ϵ with a probability that decays exponentially in n.

The equivalent of the following main theorem for the binary erasure channel and binary messages was proved in [1].

Theorem 2: Over the probability space of all graphs $\mathcal{C}^n(d_\mathsf{v}, d_\mathsf{c})$ and channel realizations let Z be the number of incorrect messages among all nd_v variable-to-check node messages passed at iteration ℓ. Let p be the expected number of incorrect messages passed along an edge with a tree-like directed neighborhood of depth at least 2ℓ at the ℓth iteration. Then, there exist positive constants $\beta = \beta(d_\mathsf{v}, d_\mathsf{c}, \ell)$ and $\gamma = \gamma(d_\mathsf{v}, d_\mathsf{c}, \ell)$ such that

[Concentration Around Expected Value] For any $\epsilon > 0$ we have

$$\Pr\{|Z - \mathbb{E}[Z]| > nd_\mathsf{v}\epsilon/2\} \le 2e^{-\beta\epsilon^2 n}. \tag{11}$$

[Convergence to Cycle-Free Case] For any $\epsilon > 0$ and $n > \frac{2\gamma}{\epsilon}$ we have

$$|\mathbb{E}[Z] - nd_\mathsf{v}p| < nd_\mathsf{v}\epsilon/2. \tag{12}$$

[Concentration Around Cycle-Free Case] For any $\epsilon > 0$ and $n > \frac{2\gamma}{\epsilon}$ we have

$$\Pr\{|Z - nd_\mathsf{v}p| > nd_\mathsf{v}\epsilon\} \le 2e^{-\beta\epsilon^2 n}. \tag{13}$$

Proof: As noted before, Theorem 2 is an extension of [1, Theorem 1] to nonbinary channel output alphabets and nonbinary messages. As we will see, we can use essentially the same arguments as were given in [1]. Since the proof in [1] is rather brief, we give an explicit account of the necessary steps.

First note that (13) follows immediately from (11) and (12).

We start by proving (12). Let $\mathbb{E}[Z_i]$, $i \in [nd_\mathsf{v}]$, be the expected number of incorrect messages passed along edge $\vec{e}_i$, where the average is over all graphs and all decoder inputs. Then, by linearity of expectation and by symmetry

$$\mathbb{E}[Z] = \sum_{i \in [nd_\mathsf{v}]} \mathbb{E}[Z_i] = nd_\mathsf{v}\mathbb{E}[Z_1].$$

Furthermore

$$\mathbb{E}[Z_1] = \mathbb{E}\left[Z_1 | \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is tree-like}\right] \Pr\left\{\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is tree-like}\right\} + \mathbb{E}\left[Z_1 | \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}\right] \Pr\left\{\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}\right\}$$

where $\mathcal{N}_{\vec{e}_1}^{2\ell}$ denotes the directed neighborhood of depth 2ℓ of edge $\vec{e}_1$. It is shown in Appendix A that

$$\Pr\{\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like }\} \le \frac{\gamma}{n}$$

for some positive constant γ. Furthermore, we have

$$\mathbb{E}[Z_1 | \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is tree-like}] = p$$

by definition and

$$|\mathbb{E}[Z_1 | \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}]| \le 1$$

trivially, hence

$$nd_\mathsf{v}p\left(1 - \frac{\gamma}{n}\right) \le \mathbb{E}[Z] \le nd_\mathsf{v}\left(p + \frac{\gamma}{n}\right)$$

and $|\mathbb{E}[Z] - nd_\mathsf{v}p| \le d_\mathsf{v}\gamma$. It follows that if $n > \frac{2\gamma}{\epsilon}$ then

$$|\mathbb{E}[Z] - nd_\mathsf{v}p| < nd_\mathsf{v}\epsilon/2.$$

It remains to prove (11). Recall that Z denotes the number of incorrect variable-to-check node messages among all nd_v variable-to-check node messages passed in the ℓth iteration for a particular $(G, R) \in \Omega$, where G is a graph in the ensemble $\mathcal{C}^n(d_\mathsf{v}, d_\mathsf{c})$, R is a particular input to the decoder, and Ω is the probability space. Let $=_i$, $0 \le i \le (d_\mathsf{v} + 1)n =: m$, be a sequence of equivalence relations on Ω ordered by refinement, i.e., $(G', R') =_i (G'', R'')$ implies $(G', R') =_{i-1} (G'', R'')$. These equivalence classes are defined by partial equalities. In particular, suppose we expose the $d_\mathsf{v}n$ edges of the graph one at a time, i.e., at step $i \le d_\mathsf{v}n$ we expose the particular check node socket $\pi(i)$ which is connected to the ith variable node socket, and, similarly, in the following n steps we expose the n received values r_i one at a time. Then we have $(G', R') =_i (G'', R'')$ if and only if the information revealed in the first i steps for both pairs is the same.

Now, define $Z_0, Z_1, \ldots, Z_m$ by

$$Z_i(G, R) := \mathbb{E}[Z(G', R') | (G', R') =_i (G, R)].$$

By construction $Z_0, Z_1, \ldots, Z_m$ is a *Doob's Martingale Process* [16, p. 90], where $Z_0 = \mathbb{E}[Z]$ and $Z_m = Z$. As noted in Appendix B, one can give bounds on

$$\Pr\{|Z - \mathbb{E}[Z]| > nd_\mathsf{v}\epsilon/2\} = \Pr\{|Z_m - Z_0| > nd_\mathsf{v}\epsilon/2\}$$

if we can prove that

$$|Z_{i+1}(G, R) - Z_i(G, R)| \le \alpha_i, \qquad i = 0, \ldots, h - 1 \tag{14}$$

for some suitable constants α_i which may depend on d_v, d_c, and ℓ, but preferably not on n.

We first prove (14) for $i \in [nd_\mathsf{v}]$, i.e., for the steps where we expose the edges. Recall that $\pi(i) = j$ means that the ith variable node socket is connected to the jth check node socket. Let $\mathcal{G}(G, i)$ be the set of graphs in the ensemble $\mathcal{C}^n(d_\mathsf{v}, d_\mathsf{c})$ such that the first i edges are equal to the edges in G, i.e.,

$$\mathcal{G}(G, i) = \{G' \colon (G', R) =_i (G, R)\}.$$

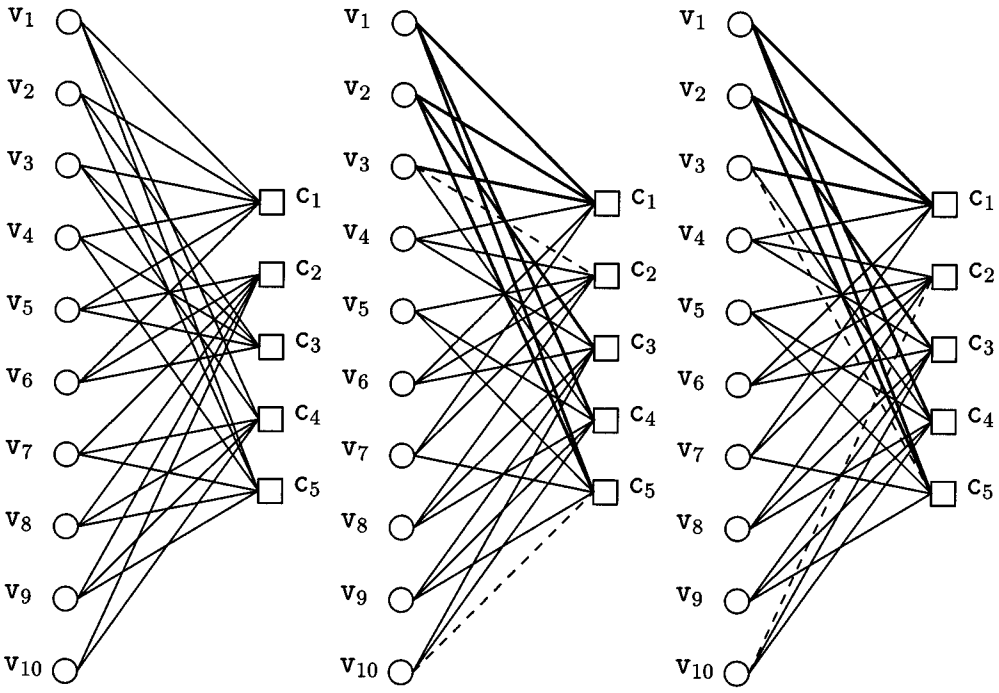

Fig. 4. Left: a graph G in the ensemble $\mathcal{C}^{10}(3, 6)$. Middle: a graph H from the ensemble $\mathcal{G}_2(G, 7)$ (note that the labels of the sockets are not shown—these labels should be inferred from the order of the connections in the middle figure); the first seven edges that H has in common with G are drawn in bold. Right: the associated graph $\phi_{2,5}(H)$. The two dashed lines correspond to the two edges whose end points are switched.

Let $\mathcal{G}_j(G, i)$ be the subset of $\mathcal{G}(G, i)$ consisting of those graphs for which $\pi(i+1) = j$. Thus, $\mathcal{G}(G, i) = \cup_j \mathcal{G}_j(G, i)$.

We have

$$\begin{aligned} Z_i(G, R) &= \mathbb{E}[Z(G', R')|G' \in \mathcal{G}(G, i)] \\ &= \sum_{j \in [md_c]} \mathbb{E}[Z(G', R')|G' \in \mathcal{G}_j(G, i)] \\ &\quad \cdot \Pr\{G' \in \mathcal{G}_j(G, i)|G' \in \mathcal{G}(G, i))\}. \quad (15) \end{aligned}$$

We claim that if j and k are such that

$$\Pr\{G' \in \mathcal{G}_j(G, i)|G' \in \mathcal{G}(G, i)\} \neq 0$$

and

$$\Pr\{G' \in \mathcal{G}_k(G, i)|G' \in \mathcal{G}(G, i)\} \neq 0$$

then

$$|\mathbb{E}[Z(G', R')|G' \in \mathcal{G}_j(G, i)] - \mathbb{E}[Z(G', R')|G' \in \mathcal{G}_k(G, i)]| < 8(d_v d_c)^\ell. \quad (16)$$

To prove this claim define a map $\phi_{j,k}$: $\mathcal{G}_j(G, i) \to \mathcal{G}_k(G, i)$ as follows. Let π be the permutation defining the edge assignment for a given graph $H \in \mathcal{G}_j(G, i)$ and let $i' = \pi^{-1}(k)$. Define a permutation π' by $\pi' = \pi$ except that $\pi'(i+1) = k$ and $\pi'(i') = j$. Let H' denote the resulting graph, then $H' \in \mathcal{G}_k(G, i)$. By definition, $H' = \phi_{j,k}(H)$. The construction is shown in Fig. 4. Clearly, $\phi_{j,k}$ is a bijection and, since every (edge-labeled) graph in the ensemble has uniform probability, such a bijection preserves probabilities. We claim that

$$|Z(H, R) - Z(\phi_{j,k}(H), R)| \leq 8(d_v d_c)^\ell \quad (17)$$

for all decoder inputs R. To see this, note that for any edge $\vec{e}$ the message along this edge is only a function of the directed neighborhood $\mathcal{N}_{\vec{e}}^{2l}$. Therefore, a message is only affected by an exchange of the endpoints of two edges if one (or both) of the two edges is (are) in the directed neighborhood $\mathcal{N}_{\vec{e}}^{2l}$. As stated in the beginning of this section, a neighborhood contains at most $2(d_v d_c)^\ell$ distinct edges and, by the symmetry property expressed in (2), an edge can be in at most $2(d_v d_c)^\ell$ neighborhoods. It follows that at most $8(d_v d_c)^\ell$ neighborhoods can be affected by the exchange of the endpoints of two edges, which proves claim (17).

Since $\phi_{j,k}$ is a bijection and preserves probability, it follows that

$$\begin{aligned} &\mathbb{E}[Z(G', R')|G' \in \mathcal{G}_k(G, i)] \\ &\qquad = \mathbb{E}[Z(\phi_{j,k}(G'), R')|G' \in \mathcal{G}_j(G, i)]. \end{aligned}$$

By (17), any pair $Z(H, R)$ and $Z(\phi_{j,k}(H), R)$ has difference bounded by $8(d_v d_c)^\ell$ and, since for any random variable $|\mathbb{E}[W]| \leq \mathbb{E}[|W|]$, claim (16) follows. Finally, note that, by definition, $Z_{i+1}(G, R)$ is equal to $\mathbb{E}[Z(G', R')|G' \in \mathcal{G}_j(G, i)]$ for some $j \in \Psi_i$ where $\Psi_i \subset [md_c]$ denotes the set of unoccupied sockets on the check node side after revealing i edges of G, i.e., $\Psi_i = [md_c] \backslash \{\pi(l)\colon l \leq i\}$. Hence

$$\begin{aligned} &|Z_{i+1}(G, R) - Z_i(G, R)| \\ &\quad \leq \max_{j \in \Psi_i} |\mathbb{E}[Z(G', R')|G' \in \mathcal{G}_j(G, i)] - Z_i(G, R)| \\ &\quad \leq \max_{j,k \in \Psi_i} |\mathbb{E}[Z(G', R')|G' \in \mathcal{G}_j(G, i)] \\ &\qquad - \mathbb{E}[Z(G', R')|G' \in \mathcal{G}_k(G, i)]| \\ &\quad \leq 8(d_v d_c)^\ell \end{aligned}$$

where we have used the representation of $Z_i(G, R)$ given in (15). This proves (14) for $i \in [nd_v]$ with $\alpha_i = 8(d_v d_c)^\ell$.

It remains to show that the inequality is also fulfilled for the last n steps. The idea of the proof is very similar and we will be

brief. When we reveal a received value at a particular message node v then messages whose directed neighborhood include the node v can be affected. By the node symmetry property (1) this is equal to the number of nodes in the neighborhood $\mathcal{N}_{\mathsf{v}}^{2l}$ and by our above estimate this number is upper-bounded by $2(d_{\mathsf{v}}d_{\mathsf{c}})^{\ell}$. This proves (14) for $i \in nd_{\mathsf{v}}+1, \ldots, (n+1)d_{\mathsf{v}}$ with $\alpha_i = 2(d_{\mathsf{v}}d_{\mathsf{c}})^{\ell}$.

Theorem 2 now follows by applying Azuma's inequality to the given martingale. The parameter $1/\beta$ can be chosen as $544d_{\mathsf{v}}^{2\ell-1}d_{\mathsf{c}}^{2\ell}$. We remark that this is by no means the best possible constant. □

As discussed before, it follows from this concentration theorem that (for sufficiently large n) almost all codes can transmit reliably up to the threshold value, but that they have an error probability bounded away from zero above the threshold value. The required size of n, according to the proofs, could be absurdly large. The proofs, however, are very pessimistic. We assume, for example, that any loop effectively introduces an error into every message it affects. With similar pessimism we assume that replacing one received value with another independent one can produce an error in every message the replacement affects. In summary, the analysis rests on the idea that loops and local changes are perturbations that are irrelevant asymptotically. In practice, however, large loops may have only a small effect and local changes tend to disappear under correct decoding. The effects are therefore much milder than predicted and the n required to observe near threshold performance is significantly smaller than any bound that can be obtained from the results.

Fig. 5 depicts some simulation results for the BSC. Fig. 6 depicts corresponding simulation results for the BIAWGN channel. The curves indicate bit-error probabilities observed as a function of the channel parameter. Results are indicated for $n = 10^3$, 10^4, and 10^5 for several different decoding algorithms. Also shown is the predicted threshold value for each algorithm. We observe the convergence to a sharp threshold effect at the predicted value as n increases.

We remark that the bit-error probabilities are the result of running the decoder for 100 iterations. (For shorter lengths this is far more than necessary.) In some cases, in particular for Gallager's decoding algorithm A, it is possible to slightly lower the measured bit-error rate by exploiting the observation that when the decoding fails the number of errors is typically larger than the uncoded error rate (this is not true for belief propagation). Since decoding failure is virtually always detected, one can decode to the received values and reap a small gain in this case.

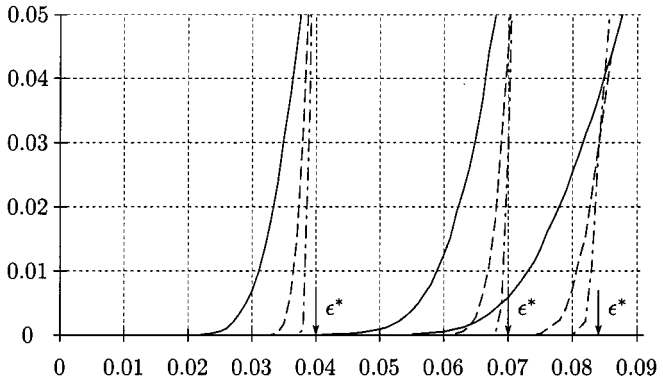

Fig. 5. Bit-error probability versus parameter ϵ for the $(3, 6)$-regular ensemble transmitting over the BSC channel for three decoding algorithms. The leftmost curves correspond to Gallager's decoding algorithm A, the middle curves correspond to decoding with erasures and the rightmost curves correspond to a belief-propagation decoder. The solid curves correspond to a codeword length of 1000, whereas the dashed and the dotted-dashed curves correspond to codeword lengths of 10 000 and 100 000, respectively. The arrows indicate the threshold values $\epsilon^* = 0.04$, 0.07, and 0.084, respectively. Observe how the lines move closer to these threshold values for increasing codeword lengths.

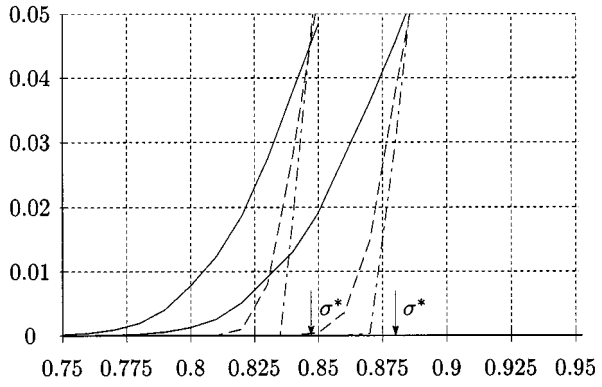

Fig. 6. Bit-error probability versus parameter σ for the $(3, 6)$-regular code used over the BIAWGN channel. The left curves correspond to the message passing algorithm with eight messages described in Example 7, whereas the right curves correspond to belief propagation. The solid curves correspond to a codeword length of 1000, whereas the dashed and the dotted–dashed curves correspond to codeword lengths of 10 000 and 100 000, respectively. The arrows indicate the corresponding threshold values $\sigma^* = 0.847$ and $\sigma^* = 0.88$. Observe how the lines move closer to this threshold value for increasing codeword lengths.

V. Extensions

In this section, we outline some extensions that are slated to be subjects of further work.[12]

First, consider irregular graphs. As proposed in [1], better LDPC codes can be constructed if we allow varying degrees, i.e., if we allow variable nodes to participate in different numbers of checks and check equations to involve different numbers of variables. One can check that, as long as the degrees in the graph are bounded, the concentration theorem applies essentially unchanged to such an ensemble of codes, requiring only adjustments of the constants. Let λ and ρ be polynomials representing the variable node and check node degree distributions, respectively. More precisely

$$\lambda(x) := \sum_{i\geq 2}^{d_{\mathsf{v}}^{\max}} \lambda_i x^{i-1}$$

and

$$\rho(x) := \sum_{i\geq 2}^{d_{\mathsf{c}}^{\max}} \rho_i x^{i-1}$$

where λ_i and ρ_i denote the fraction of edges incident to variable and check nodes with degree i, respectively. The calculation of

[12]Some of the proposed work has been carried out subsequent to the preparation of this paper, see, e.g., [10].

the message densities under the tree-like assumption requires only minor modifications to accommodate irregular degree sequences. In fact, the only changes required are the following: (8) becomes

$$\hat{\hat{Q}}^{(\ell),0} - \hat{\hat{Q}}^{(\ell),1} = \rho\left(\hat{\hat{P}}^{(\ell-1),0} - \hat{\hat{P}}^{(\ell-1),1}\right)$$
$$\hat{\hat{Q}}^{(\ell),0} + \hat{\hat{Q}}^{(\ell),1} = \rho\left(\hat{\hat{P}}^{(\ell-1),1} + \hat{\hat{P}}^{(\ell-1),1}\right)$$

and (9) becomes

$$\mathcal{F}\left(P^{(\ell)}\right) = \mathcal{F}\left(P^{(0)}\right)\lambda\left(\mathcal{F}\left(Q^{(\ell-1)}\right)\right).$$

Hence, for given degree sequences and a given memoryless channel, we can calculate the resulting threshold of a belief-propagation decoder in an efficient manner. In [1], the authors propose a particular degree sequence for a rate one-half code. The degree distributions are given by the following:[13]

$$\lambda(x) = \frac{1}{6}(x^2+x^4+x^8+x^{16}+x^{32}+x^{64})$$
$$\rho(x) = \rho_7 x^6+\rho_8 x^7+\rho_{19}x^{18}+\rho_{20}x^{19}+\rho_{84}x^{83}+\rho_{85}x^{84}$$

where

$$\rho_7 = 0.154078 \qquad \rho_8 = 0.147471 \qquad \rho_{19} = 0.121201$$
$$\rho_{20} = 0.228596 \qquad \rho_{84} = 0.218999 \qquad \rho_{85} = 0.129654.$$

Although this particular degree distribution was optimized for erasure channels, we can use our extended algorithm to calculate the resulting threshold for other channels. For example, we get $\epsilon^* = 0.094$ for the BSC and $\sigma^* = 0.92$ ($\sigma^* = 0.92$ corresponds to $E_b/N_0 = 0.72$) for the BIAWGNC. Some very good degree distributions for various channels and message-passing algorithms are described in [10].

Next, let us consider LDPC codes over larger fields, or rings, under belief-propagation decoding. Again it is easy to check that the concentration results apply to both regular and, with proper conditions as mentioned above, irregular graphs.

Another area into which the methods used in this paper extend is that of turbo codes. Although this is not immediately apparent, the notion of a neighborhood of a variable and the notion of a tree-like such neighborhood, which played important roles in proving concentration results for LDPC codes, can be suitably extended to the setting of turbo codes. This is the key observation in proving equivalent concentration results for turbo codes. The determination of the average performance (under tree-like assumptions) may not be directly computationally feasible, but simulations should nevertheless be sufficient to determine the densities of the messages passed between constituent decoders. This opens the possibility of calculating threshold values for turbo codes under turbo decoding. For a more detailed account of this extension, see [17] and [18]. As we saw in this paper, LDPC codes admit a large family of decoding algorithms, allowing a tradeoff between performance and complexity. To date there are only a few decoding algorithms known for turbo codes, such as the standard turbo decoding using the Bahl–Cocker–Jelinek–Raviv (BCJR) algorithm, or the simpler version employing the SOVA algorithm [19]. It is an interesting and important open question to find a larger class of decoding algorithms for turbo codes that allow one to trade off performance with complexity in a larger range.

We conclude this section by presenting our view of the overall structure of LDPC codes and belief-propagation decoders. Let us first consider the structure of the code and the meaning of the graph. In general,

- the variables v_i take values r_i in some ring and
- check nodes represent weighted sum constraints. More precisely, each check node c_j represents a constraint of the form

$$0 = \sum_{i:\, \exists e = (\mathsf{v}_i, \mathsf{c}_j)} w_i r_i$$

where the weights w_i are elements of the ring.

Under belief propagation we have the following general description of the meaning of the messages.

- A message is a (representation of a) conditional distribution of the variable associated with the variable node.
- All messages arriving at a node at a particular stage (including the received value in the case of a variable node) are assumed to be conditional distributions of the associated variable, each independent of the others, i.e., each conditioned on independent random variables.
- At the variable nodes the outgoing messages are pointwise products of the densities from incoming messages.
- The outgoing messages from the check nodes represent the distribution of the (additive group) inverse of some weighted sum of variables. The variables participating in the sum are assumed to have the distributions indicated by the associated incoming message, and each is assumed independent of the others. Thus, the outgoing message represents the convolution of the participating incoming messages (suitably adjusted for any multiplicative factor). Typically, the additive group possesses a Fourier transform so that efficient computation of the convolution can be done in the Fourier domain.

Thus, at the message nodes, the distributions should be interpreted in the Fourier domain, and then the outgoing messages are just pointwise products. For example, in the case of GF (2^n), the additive group operative at the check nodes is GF $(2)^n$. Thus, the appropriate Fourier transform is just the multidimensional version of the one used for binary parity-check codes. This observation leads to a new efficient implementation of the decoder for this case.

We now address the problem of *computing average asymptotic performance.* Here, we interpret the messages themselves as random variables for which we desire the density.

- In each case, i.e., at both sets of nodes, we move to a log domain so that pointwise products become pointwise sums.
- Once in the log domain, the density of an outgoing message is the convolution of the density of incoming messages. Typically, the log domain can be suitably repre-

[13]Note that the left degrees are of the form $2^n + 1$ (the exponents of the polynomials are one less than the corresponding degrees).

sented so that a Fourier transform exists over the space on which the densities are defined. The update of the density can then be computed efficiently in this Fourier domain. This applies both at the variable nodes and at the check nodes, but the spaces may be different in both cases.

In the case of GF(2^m), for example, the Fourier transform of the densities can be viewed as a real function over $F(2)^m$, taking values in the interval $[-1, 1]$. Thus, the appropriate log domain is a multidimensional version of the one appearing in the binary case. Unfortunately, the dimensionality of the space renders computation of the densities infeasible except for quite small m (e.g., $m = 2$).

Other examples of interest are codes over Z/q or codes over $Z^2/(q_1, q_2)$. The latter example introduces the possibility of designing LDPC codes directly for QAM-type signaling schemes. Again, it is the existence of the Fourier transform over these spaces that renders the necessary computation at the check nodes efficient.

Appendix A
Probability of Tree-Like Neighborhood

In this section we will give a short proof that

$$\Pr\{\mathcal{N}_{\vec{e}}^{2\ell^*} \text{ is not tree-like}\} \leq \frac{\gamma}{n}$$

for some constant γ, where $\vec{e} = (\mathsf{v}, \mathsf{c})$ is a given edge in a randomly chosen element of $\mathcal{C}^n(d_\mathsf{v} d_\mathsf{c})$ and $2\ell^*$ is a fixed depth.

Note that there are, in total,

$$M_{\ell^*} := \sum_{i=0}^{l} (d_\mathsf{v} - 1)^i (d_\mathsf{c} - 1)^i$$

variable nodes and

$$C_{\ell^*} := 1 + (d_\mathsf{v} - 1) \sum_{i=0}^{l-1} (d_\mathsf{v} - 1)^i (d_\mathsf{c} - 1)^i$$

check nodes in $\mathcal{N}_{\vec{e}}^{2\ell^*}$ assuming it is tree-like. Recall that m denotes the number of check nodes in the full graph. Fix ℓ^* and let $\ell < \ell^*$. Assuming that $\mathcal{N}_{\vec{e}}^{2\ell}$ is tree-like, we ask this: what is the probability that $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is tree-like? We obtain a bound by revealing the outgoing edges of the variable node leaves of the tree given by $\mathcal{N}_{\vec{e}}^{2\ell}$ one at a time and bounding the probability that this revelation creates a loop. Assume that k additional edges have been revealed at this stage without creating a loop; then the probability that the next revealed edge does not create a loop is $\frac{(m - C_\ell - k)d_\mathsf{c}}{md_\mathsf{c} - C_\ell - k}$. Assuming that n is sufficiently large we have

$$\begin{aligned} \frac{(m - C_\ell - k)d_\mathsf{c}}{md_\mathsf{c} - C_\ell - k} &= 1 - \frac{(C_\ell + k)(d_\mathsf{c} - 1)}{md_\mathsf{c} - C_\ell - k} \\ &\geq 1 - \frac{C_{\ell^*}}{m}. \end{aligned}$$

Thus, the probability that $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is tree-like, given $\mathcal{N}_{\vec{e}}^{2\ell}$ is tree-like, is lower-bounded by $(1 - \frac{C_{\ell^*}}{m})^{C_{\ell+1} - C_\ell}$.

Assume now that $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is tree-like. As above, we reveal the outgoing edges of the check node leaves of $\mathcal{N}_{\vec{e}}^{2\ell+1}$ one at a time. Assuming that k variable nodes have been revealed without creating a loop, then the probability that the next revealed edge does not create a loop is $\frac{(n - M_\ell - k)d_\mathsf{v}}{nd_\mathsf{v} - M_\ell - k}$. Assuming that n is sufficiently large, we now have

$$\begin{aligned} \frac{(n - M_\ell - k)d_\mathsf{v}}{nd_\mathsf{v} - M_\ell - k} &= 1 - \frac{(M_\ell + k)(d_\mathsf{v} - 1)}{nd_\mathsf{v} - M_\ell - k} \\ &\geq 1 - \frac{M_{\ell^*}}{n}. \end{aligned}$$

Thus, the probability that $\mathcal{N}_{\vec{e}}^{2\ell+2}$ is tree-like given $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is tree-like is lower-bounded by $(1 - \frac{M_{\ell^*}}{n})^{M_{\ell+1} - M_\ell}$.

It now follows that the probability that $\mathcal{N}_{\vec{e}}^{2\ell^*}$ is tree-like is lower-bounded by

$$\left(1 - \frac{M_{\ell^*}}{n}\right)^{M_{\ell^*}} \left(1 - \frac{C_{\ell^*}}{m}\right)^{C_{\ell^*}}.$$

Hence, for n sufficiently large

$$\Pr\{\mathcal{N}_{\vec{e}}^{2\ell^*} \text{ is not tree-like}\} \leq \frac{M_{\ell^*}^2 + \frac{d_\mathsf{c}}{d_\mathsf{v}} C_{\ell^*}^2}{n}. \qquad \square$$

Appendix B
Azuma's Inequality

Let $Z = \sum Z_i$ be a sum of independent random variables. We can then use the Chernoff bound to give an exponential bound on the probability that Z will deviate from its mean by more than a fraction ϵ.

Sometimes it is possible to give strong bounds even when the random variables are not independent. In one particularly useful case the random variables form a martingale.

Theorem 3 [Azuma's Inequality]: Let $Z_0, Z_1, \ldots$ be a martingale sequence such that for each $k \geq 1$

$$|Z_k - Z_{k-1}| \leq \alpha_k$$

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

$$\Pr\{|Z_\ell - Z_0| \geq \lambda\} \leq 2e^{-\frac{\lambda^2}{2\sum_{k=1}^{\ell} \alpha_k^2}}.$$

For proofs and applications of Azuma's inequality see, for example, [16], [20].

Acknowledgment

The authors would like to thank Richard Blahut, Amin Shokrollahi, Emre Telatar, and Igal Sason for their comments on an earlier draft of this paper. They would also like to thank the reviewers for their many helpful suggestions.

References

[1] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Analysis of low density codes and improved designs using irregular graphs. [Online]. Available: http://www.icsi.berkeley.edu/~luby/

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding," in *Proc. Int. Communications Cconf. (ICC'93)* Geneva, Switzerland, May 1993, pp. 1064–1070.

[3] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[4] M. Sipser and D. Spielman, "Expander codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1710–1722, Nov. 1996.

[5] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
[6] V. Zyablov and M. Pinsker, "Estimation of the error-correction complexity of Gallager low-density codes," *Probl. Pered. Inform.*, vol. 11, pp. 23–26, Jan. 1975.
[7] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.
[8] G. A. Margulis, "Explicit construction of graphs without short cycles and low density codes," *Combinatorica*, vol. 2, no. 1, pp. 71–78, 1982.
[9] D. MacKay, S. Wilson, and M. Davey, "Comparison of constructions of irregular Gallager codes," *IEEE Trans. Commun.*, vol. 47, pp. 1449–1454, Oct. 1999.
[10] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.
[11] M. C. Davey and D. J. C. MacKay, "Low density parity check codes over GF (q)," *IEEE Commun. Lett.*, vol. 2, pp. 165–167, June 1998.
[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
[13] L. Bazzi, T. Richardson, and R. Urbanke, "Exact thresholds for the binary symmetric channel and Gallager's decoding algorithm A," *IEEE Trans. Inform. Theory*, to be published.
[14] M. Mitzenmacher, "A note on low density parity check codes for erasures and errors,", SRC Technical Note 1998-017, December 1998.
[15] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann, 1988.
[16] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.
[17] E. Gelblum, R. Calderbank, and J. Boutros, "Understanding serially concatenated codes from a support tree approach," in *Proc. Int. Symp. Turbo Codes and Related Topics*, Brest, France, Sept. 1997, pp. 271–274.
[18] T. Richardson and R. Urbanke, "Thresholds of turbo codes," paper, to be published.
[19] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. GLOBECOM '95*, Dallas, TX, Nov. 1995, pp. 1680–1686.
[20] N. Alon, J. Spencer, and P. Erdös, *The Probabilistic Method*. New York: Wiley, 1992.