

# On Designing Good LDPC Codes for Markov Channels

Andrew W. Eckford, *Member, IEEE*, Frank R. Kschischang, *Fellow, IEEE*, and Subbarayan Pasupathy, *Fellow, IEEE*

**Abstract**—This paper presents a reduced-complexity approximate density evolution (DE) scheme for low-density parity-check (LDPC) codes in channels with memory in the form of a hidden Markov chain. This approximation is used to design degree sequences representing some of the best known LDPC code ensembles for the Gilbert–Elliott channel, and example optimizations are also given for other Markov channels. The problem of approximating the channel estimation is addressed by obtaining a specially constructed message-passing schedule in which the channel messages all approach their stable densities. It is shown that this new schedule is much easier to approximate than the standard schedule, but has the same ultimate performance in the limits of long block length and many decoding iterations. This result is extended to show that all message-passing schedules that satisfy mild conditions will have the same threshold under density evolution.

**Index Terms**—Code optimization, density evolution, estimation-decoding, Gaussian assumption, irregular low-density parity-check (LDPC) codes, Markov channels, message-passing schedules.

## I. INTRODUCTION

LOW-density parity-check (LDPC) codes [1] are a class of linear codes with very sparse parity-check matrices. LDPC codes may be characterized by a *degree sequence*, which expresses the probability of finding a given number of ones in either the rows or the columns of the parity-check matrix. An LDPC code is *regular* if every row and every column contains a fixed number of ones, or *irregular* if the number of ones varies. It is known that the bit error performance of irregular LDPC codes is a function of the degree sequence, which can therefore be optimized to design good LDPC code ensembles [2].

Informally, the irregular LDPC code design problem can be posed as follows:

**Given** a particular communication channel

**Maximize** code rate

**Subject to**  $P_{\text{err}} < \epsilon$ ,

where  $P_{\text{err}}$  is the probability of symbol error, and  $\epsilon$  is the maximum acceptable probability of error. For a given degree se-

quence, the symbol-wise error performance of an LDPC code in the limit of long block length may be obtained analytically through *density evolution* (DE) [3], [4], which can be used to verify the probability of error criterion. However, exact DE can be inefficient for use in an optimization algorithm, even when optimized for speed of calculation [5]. Much recent effort has focused on fast approximations to DE which are more appropriate for degree sequence design, which includes the extrinsic information transfer (EXIT) approximation [6], and the semi-Gaussian approximation [7], the last of which has the advantage of allowing degree sequence optimization with linear programming.

The use of LDPC codes in channels with memory is attracting interest in the literature. In previous work [8], the use of regular LDPC codes in the Gilbert–Elliott channel was analyzed using DE, under an *estimation-decoding* strategy, in which intermediate results from the iterative decoding algorithm are used to estimate the channel state, and intermediate channel estimates are used to improve the decoding procedure, which improves both processes. These results both confirmed that the performance of LDPC codes is excellent in the Gilbert–Elliott channel, and showed that large gains can be achieved through the use of estimation-decoding. However, they also indicated that the performance of regular codes falls well short of the Shannon limit, motivating the search for good degree sequences in these channels. Furthermore, the Gilbert–Elliott channel is a simple example of a *Markov channel*, in which the channel noise is dependent on a hidden Markov chain, and which can be used to represent wireless fading channels. A design procedure, related to the one in this paper, was presented for multiple-input multiple-output fading channels in [9], which assessed the effect of demodulation on the design procedure, but which did not address channel estimation, since the receiver was assumed to know the channel.

The problem of finding very good LDPC degree sequences for estimation-decoding in Markov channels is an interesting open problem of great practical importance. To address the irregular LDPC design problem in Markov channels, we seek simple design tools which approximate DE. However, the task is complicated in that we must not only approximate DE for the LDPC code, but also obtain a related approximation for the channel estimation technique, so that the code can be optimized with respect to the entire estimation-decoding algorithm. In this paper, we will show that this complications can be addressed by the semi-Gaussian approximation in [7], as well as some novel message-passing schedules and theoretical results. Our emphasis in this paper is on designing *degree sequences*, which parameterize LDPC code ensembles, and then the particular code within this ensemble is selected at random, which is a good strategy when block length is asymptotically large (al-

Manuscript received December 9, 2003; revised August 16, 2006. The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Chicago, IL, June/July 2004.

A. W. Eckford was with The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto. He is now with the Department of Computer Science and Engineering, York University, Toronto, ON M3J 1P3, Canada (e-mail: aeckford@yorku.ca).

F. R. Kschischang and S. Pasupathy are with The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: frank@comm.utoronto.ca; pas@comm.utoronto.ca).

Communicated by K. Abdel-Ghaffar, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2006.887467

though proving that all codes have the same performance with  $\text{Pr} = 1$  requires a proof of a concentration theorem for Markov channels, which remains an open problem).

We will first use the Gilbert–Elliott channel as a motivating example, and subsequently generalize the framework developed for the Gilbert–Elliott channel to binary Markov channels with larger state alphabets. The technique works with general Markov channels, but we give binary Markov examples as a proof of concept, since they are the most challenging type of channel for our design method. The main contributions of this work include: an approximate DE algorithm for the Gilbert–Elliott channel based on the approximation in [7]; theoretical results showing the equivalence in ultimate performance between decoders with different message-passing schedules; and degree sequences based on the derived algorithms which represent the best known LDPC codes for the Gilbert–Elliott channel. The remainder of this paper is organized as follows. In Section II, we introduce the Markov channel model, and discuss joint estimation-decoding in this channel. In Section III, we introduce a particular message-passing schedule for Markov–LDPC estimation-decoding for which DE is straightforward to approximate, and show that its ultimate performance (in the limit of large numbers of iterations and long block length) is the same as the standard schedule. In Section IV, we state the approximate DE algorithm for the estimation-decoding scheme. Results for the Gilbert–Elliott channel are presented in Section V, and a generalized approximate DE algorithm for Markov channels is presented in Section VI, along with some results.

## II. SYSTEM MODELS

### A. Gilbert–Elliott and Markov Channels

A Markov channel is any channel with a vector of inputs  $\mathbf{X} \in \{0, 1\}^n$ , outputs  $\mathbf{Y} \in \mathcal{Y}^n$ , and hidden channel states  $\mathbf{S} \in \mathcal{S}^n$ , so that

- $\mathcal{S}$  is a discrete, finite index set, i.e.,  $\mathcal{S} = \{1, 2, \dots, |\mathcal{S}|\}$ ;
- $\mathbf{S}$  forms a regular Markov chain operating in steady state, independent of  $\mathbf{X}$ ; and
- the channel is conditionally memoryless given  $\mathbf{S}$ , i.e.,

$$f_{\mathbf{Y}|\mathbf{S}, \mathbf{X}}(\mathbf{y} | \mathbf{s}, \mathbf{x}) = \prod_{t=1}^n f_{Y_t|S_t, X_t}(y_t | s_t, x_t). \quad (1)$$

Because  $\mathbf{S}$  is assumed independent of  $\mathbf{X}$ , we do not consider partial response channels in this paper.

Equation (1) suggests that each state corresponds to a particular channel behavior. Let  $f_{Y|X}(y | x; \eta)$  represent a family of channel input–output probability density functions (pdfs), indexed by the parameter  $\eta$ . To each channel state  $i \in \mathcal{S}$ , we assign a channel parameter  $\eta_i$ , such that

$$f_{Y_t|S_t, X_t}(y_t | i, x_t) = f_{Y_t|X_t}(y_t | x_t; \eta_i)$$

that is, given the  $i$ th state at time  $t$ , the channel behavior is selected from the given family with parameter  $\eta_i$ . Thus, for the set of states  $\mathcal{S} = \{1, 2, \dots, |\mathcal{S}|\}$ , there is a corresponding vector  $\mathbf{n} = [\eta_1, \eta_2, \dots, \eta_{|\mathcal{S}|}]$  of channel parameters.

The Markov chain itself is parameterized by the state transition probability matrix  $\mathbf{P}$ , which is an  $|\mathcal{S}| \times |\mathcal{S}|$  matrix where

$$P_{ij} := p_{S_{t+1}|S_t}(i | j).$$

Given a particular family, the matrix  $\mathbf{P}$  and the vector  $\mathbf{n}$  parameterize any Markov channel; we write the parameters of any particular channel  $\mathbf{c}$  as

$$\mathbf{c} = (\mathbf{P}, \mathbf{n}).$$

For ease of representation, we extend the hidden state vector  $\mathbf{S}$  by one element, so  $\mathbf{S} \in \mathcal{S}^{n+1}$ , which changes nothing in the system. The input–output pdf, including the hidden  $\mathbf{S}$ , is written

$$f_{\mathbf{Y}, \mathbf{S}|\mathbf{X}}(\mathbf{y}, \mathbf{s} | \mathbf{x}) = p(s_1) \prod_{t=1}^n f_{Y_t|S_t, X_t}(y_t | s_t, x_t) p_{S_{t+1}|S_t}(s_{t+1} | s_t). \quad (2)$$

Note that each factor in (2) can be expressed in terms of the parameters  $\mathbf{c} = (\mathbf{P}, \mathbf{n})$ .

An important special case is the *Gilbert–Elliott* (GE) channel, which has its own notation. The GE channel is a binary Markov channel with two states, labeled  $\{B, G\}$ . Each state corresponds to a binary-symmetric channel (BSC), so  $\mathbf{n} = [\eta_B, \eta_G]$ , and by convention,  $\eta_B > \eta_G$ . In other words, state  $B$  corresponds to a “bad” BSC, while state  $G$  corresponds to a “good” BSC. Meanwhile, the matrix  $\mathbf{P}$  has elements

$$\mathbf{P} = \begin{bmatrix} 1-g & g \\ b & 1-b \end{bmatrix}$$

where  $g := p_{S_{t+1}|S_t}(G|B)$  and  $b := p_{S_{t+1}|S_t}(B|G)$ . Thus, if  $\mathbf{c}$  is a GE channel, it can be expressed as

$$\mathbf{c} = \left( \begin{bmatrix} 1-g & g \\ b & 1-b \end{bmatrix}, [\eta_B, \eta_G] \right)$$

with four parameters. Throughout this paper, we will use the shorthand notation  $\mathbf{c} = (b, g, \eta_B, \eta_G)$  to denote the fact that  $\mathbf{c}$  is a GE channel with the given four parameters.

### B. LDPC Estimation Decoding

For iteratively decoded codes such as LDPC codes, the channel can be estimated as the code is decoded, and intermediate decoding results can assist the estimation task, and *vice versa*. This method is called *estimation-decoding*, and is straightforward when factor graphs are used to express both the channel memory and the LDPC code structure.

Let  $\mathbf{H}$  represent the  $m \times n$  parity-check matrix of a given LDPC code. A given binary sequence  $\mathbf{x}$  is a valid codeword if and only if  $\mathbf{x}\mathbf{H}^T = \mathbf{0}$  (in mod-2 arithmetic), where  $\mathbf{0}$  is an all-zero vector with  $m$  elements. Thus, writing  $\mathbf{H}$  in terms of its rows as

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_m \end{bmatrix}$$

it is clear that  $\mathbf{x}$  is a codeword if and only if

$$\mathbf{x}\mathbf{h}_i^T = 0 \quad (3)$$

is true for all  $i = 1, 2, \dots, m$ . The rows  $\mathbf{h}_i$  are called *parity checks*, and each parity check is *satisfied* if (3) is true.

To express the LDPC code on a factor graph, we turn to the *characteristic function* of the code,  $h(\mathbf{x})$ , which is defined as

$$h(\mathbf{x}) := \begin{cases} 1, & \mathbf{x}\mathbf{H}^T = \mathbf{0} \\ 0, & \text{otherwise.} \end{cases}$$

Suppose that exactly  $d_i$  entries of the  $i$ th parity check,  $\mathbf{h}_i$ , are equal to 1 (where the rest are equal to 0), and let  $[\phi_i(1), \phi_i(2), \dots, \phi_i(d_i)]$  represent the indices for which the  $\mathbf{h}_i$  is equal to 1. Furthermore, let  $h_j(x_{\phi_i(1)}, x_{\phi_i(2)}, \dots, x_{\phi_i(d_i)})$  be the characteristic function for the parity check  $\mathbf{h}_i$ , where

$$h_i(x_{\phi_i(1)}, x_{\phi_i(2)}, \dots, x_{\phi_i(d_i)}) := \begin{cases} 1, & \mathbf{x}\mathbf{h}_i^T = 0 \\ 0, & \text{otherwise.} \end{cases}$$

Then it is easy to see that the characteristic function of the code is given by

$$h(\mathbf{x}) = \prod_{i=1}^m h_i(x_{\phi_i(1)}, x_{\phi_i(2)}, \dots, x_{\phi_i(d_i)}).$$

If each codeword is equally likely to be transmitted, then the probability of transmitting a particular vector  $\mathbf{x}$  is given by

$$\begin{aligned} p_X(\mathbf{x}) &= \frac{1}{c} h(\mathbf{x}) \\ &= \frac{1}{c} \prod_{i=1}^m h_i(x_{\phi_i(1)}, x_{\phi_i(2)}, \dots, x_{\phi_i(d_i)}) \end{aligned} \quad (4)$$

where  $c$  represents the number of valid codewords in the code.

Given (4), a factor graph can be drawn to represent an LDPC code, with variables  $x_1, x_2, \dots, x_n$  and factors  $h_1, h_2, \dots, h_m$ , with edges assigned randomly (according to an ensemble of possible edge connections described in [4]). For an edge selected uniformly at random from all possible edges, the degrees of the nodes incident to the edge are random variables, and the probability distributions of these degrees for check and variable nodes are called *degree sequences*. An LDPC code ensemble is characterized by the degree sequences  $(\lambda, \rho)$ , where  $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_{v_{\max}}]$ , in which  $\lambda_i$  is the probability that a randomly selected edge is incident to a variable node with degree  $i$ , and where  $v_{\max}$  is the maximum variable degree. Similarly,  $\rho = [\rho_1, \rho_2, \dots, \rho_{c_{\max}}]$ , in which  $\rho_j$  is the probability that a randomly selected edge is incident to a parity-check node with degree  $j$ , and where  $c_{\max}$  is the maximum check degree. If  $\rho_{c_{\max}} = \lambda_{v_{\max}} = 1$ , the code is said to be *regular*, otherwise, it is said to be *irregular*.

Assuming that the rows of the parity-check matrix are linearly independent, the rate of an LDPC code with a given degree sequence is given by [3]

$$R = 1 - \frac{\sum_{k=1}^{c_{\max}} \rho_k/k}{\sum_{k=1}^{v_{\max}} \lambda_k/k}. \quad (5)$$

In the Introduction, we stated that we would optimize (i.e., maximize) the rate of LDPC codes, subject to the constraint of successful decoding. However, from (5), the rate is a function of the degree sequence, and furthermore, code performance is also

affected by the degree sequence. Thus, in order to maximize the code rate, we must find an optimal degree sequence.

Combining the channel probability mass function (PMF) (2) with the factorization for  $h(\mathbf{x})$ , we obtain a PMF of the form

$$\begin{aligned} f_{\mathbf{Y}, \mathbf{S}, \mathbf{X}}(\mathbf{y}, \mathbf{s}, \mathbf{x}) &= p_{S_1}(s_1) \prod_{t=1}^n f_{Y_t|S_t, X_t}(y_t | s_t, x_t) p_{S_{t+1}|S_t}(s_{t+1} | s_t) \\ &\cdot \prod_{j=1}^m h_j(x_{\phi_j(1)}, x_{\phi_j(2)}, \dots, x_{\phi_j(d_j)}) \end{aligned} \quad (6)$$

which can be expressed on a factor graph as an LDPC subgraph connected to a Markov subgraph, as suggested in [10, Ch. 7] and [11]. Let  $\mathcal{L}$  represent this factor graph. An estimation-decoding algorithm defined over  $\mathcal{L}$  based on the Sum-Product Algorithm (SPA) [12] is discussed in detail in [8], and we assume that this algorithm is used for estimation-decoding. A section of  $\mathcal{L}$  is given in Fig. 1.

### C. Message-Passing Schedules and Local Neighborhoods

Estimation-decoding in the GE channel requires not only the SPA calculations, but also a message-passing schedule, which defines the order in which calculations occur. One of the main contributions of this paper is to use an unusual message-passing schedule as a component of a design tool for LDPC codes used in estimation-decoding.

As a starting point, a commonly used schedule is as follows [8].

*Standard Message-Passing Schedule:* Divide the set of nodes in the factor graph into the set of factor nodes (i.e., parity checks and channel factors), and the set of variable nodes (i.e., symbol variables and channel states). A full iteration proceeds in two steps: first perform the SPA at each factor node, and pass the appropriate messages to each attached variable node. Then perform the SPA at each variable node, and pass the appropriate messages to each attached factor node.

This can be thought of as a “flooding” schedule, which gives equal time to all nodes and variables. We will use the performance of estimation-decoding under the standard schedule as a reference point for comparison to other schedules.

Messages calculated and passed in the current iteration are functions of messages calculated and passed between different nodes in previous iterations. To analyze the messages passed under estimation-decoding, such as with density evolution, we form the subgraph of the overall factor graph containing all nodes and edges that participate in the calculation of a particular message along edge  $e$  at iteration number  $\ell$ . This subgraph is called the *local neighborhood* of the edge  $e$ , and written  $\mathcal{N}_e^{(\ell)}$  [4]. Some important properties of the local neighborhood are given as follows.

- *Cycle-Free.*  $\mathcal{N}_e^{(\ell)}$  is cycle-free with  $\Pr \rightarrow 1$  as the block length  $n \rightarrow \infty$ . This statement was originally proven for Markov-LDPC factor graphs in [8, Theorem 1].
- *Random.* Given a degree sequence, we choose a code randomly from the respective ensemble of possible codes. Thus, the degrees of the nodes surrounding an edge  $e$ , and

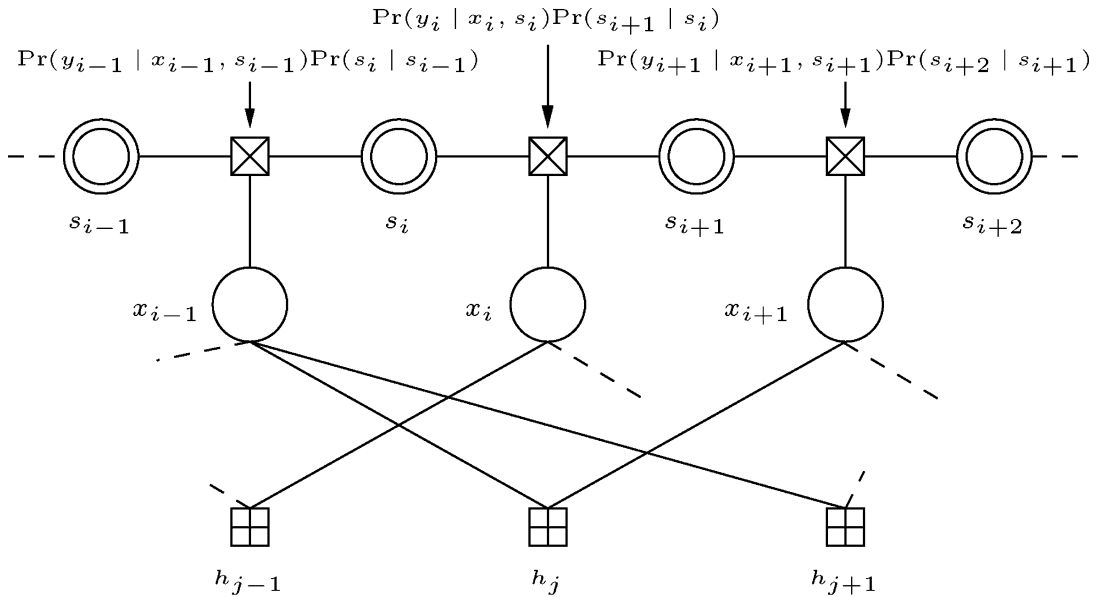


Fig. 1. A section of the Markov-LDPC factor graph.

the structure of the local neighborhood, are random variables. (This is not true of regular LDPC codes, since the degrees of the nodes are all the same, meaning that the structure of the local neighborhood is deterministic.)

Different message-passing schedules include different nodes and edges in the calculation, and hence give rise to different local neighborhoods. For example, the operations that make up a complete iteration in the standard schedule can be divided into two parts:

- the *LDPC Subiteration*, which represents message-passing operations at the parity-check nodes and at the symbol variable nodes; and
- the *Markov Subiteration*, which represents message-passing operations at the channel factor nodes and at the channel state variable nodes.

Using this distinction of the subiterations, the standard message-passing schedule has one LDPC subiteration for every Markov subiteration. We can also consider the following message-passing schedule.

*Message Passing Schedule A:* In one complete iteration, one LDPC subiteration is performed at the same time as  $N$  Markov subiterations, where  $N > 1$ .

This schedule is similar to the schedule from [13], where one operation of estimating intersymbol interference (ISI) was followed by  $K > 1$  message-passing operations in the LDPC subgraph, although Schedule A is the reverse of this schedule, with many Markov subiterations followed by one LDPC subiteration.

The local neighborhood is a function of both the iteration number and the message-passing schedule. Density evolution can be used to analyze the messages passed using any message-passing schedule in which the cycle-free property is satisfied, which is true for Schedule A (as long as  $N$  is finite). Furthermore, under the cycle-free assumption, density evolution can be used to analyze messages passed under any iterative method of constructing local neighborhoods, whether they could arise from a “real” message-passing schedule or not. We will use this property, along with local neighborhoods similar

to those from Schedule A, to develop a design tool for LDPC codes under estimation-decoding.

### III. FACTOR GRAPHS AND APPROXIMATION

In this section, we discuss the difficulties encountered in approximating DE for LDPC estimation-decoding in the GE channel. To address these difficulties, we propose an unusual way of iteratively assembling local neighborhoods, similar to Schedule A, which is useful in creating a design tool for LDPC codes under estimation-decoding. The usefulness of this method is then demonstrated by our main theoretical results: under mild conditions, all message-passing schedules, and iterative local neighborhood constructions, have the same threshold under density evolution. Following from the result that the message-passing schedule does not matter, degree sequences evaluated using the design tool will have the same threshold as under virtually any other message-passing schedule.

#### A. Shortcomings of Existing Design Tools

Existing design tools for LDPC codes fall into one of two categories.

- Expedited methods for performing exact density evolution (such as fast density evolution [5]), along with a nonconvex optimization technique.
- Approximating density evolution by tracking the evolution of a single parameter of the message pdfs, with the problem set up to allow convex optimization [7].

In either case, it is difficult to use these techniques with estimation-decoding in Markov channels, due to the channel information message  $T$ , which is a random variable, which carries the information available about a symbol arising from the channel observation and state estimation. This message is affected by the iterative estimation which occurs in an estimation-decoding algorithm, which is difficult to model. For the first category, to calculate the channel information density  $f_T(t)$  (i.e., the pdf of the channel information message), density evolution in Markov channels is either time consuming (for two-state Markov channels, such as the GE channel) [8], or computationally infeasible

(for higher order state spaces, since multivariate densities would have to be calculated). For the second category, this method works well because LDPC decoding messages are well approximated by the Gaussian density (with variance exactly twice the mean), so tracking a single parameter of the message pdf, and assuming Gaussianity, gives good results. However, there is no single-parameter family of pdfs that can represent the message pdfs of all possible Markov channels.

Our approach largely uses the single-parameter method (but as we point out in Section V-C, it is also usable with expedited forms of density evolution). To represent the parameter that approximates the extrinsic information after  $\ell$  iterations, we will use the notation  $\nu^{(\ell)}$ . Ideally, we would like a method to relate  $\nu^{(\ell)}$  to the pdfs of the channel information messages (i.e., channel information densities). That is, for each value of  $\nu$  used to approximate the extrinsic information, there should be a particular channel information density corresponding to that value. Thus,  $\nu^{(\ell)}$  would parameterize not only the extrinsic information density, but also an implicit family of channel information densities, represented by  $f_T(t)$ . In other words,  $f_T(t)$  would be dependent on exactly one extrinsic density  $\nu^{(\ell)}$ . This would allow precalculation of all possible  $f_T(t)$  for various predetermined values of  $\nu^{(\ell)}$ .

To accomplish this, consider the use of Schedule A, where  $N$  is very large. While the schedule is performing  $N$  Markov subiterations, the extrinsic information densities remain the same, since nothing is changing in the LDPC subgraph. We have observed that whenever the same extrinsic message density is applied over a large number of iterations, a stable channel message density will result, regardless of the initial channel message density. (A partial justification of this statement is presented in Appendix A.) Using this observation, suppose that the extrinsic information densities passed to the channel are replaced by their single-parameter approximations. These are parameterized by  $\nu^{(\ell)}$  at the  $i$ th iteration, where the prime superscript is used to distinguish “pure” extrinsic information that contains no channel information, such as would be passed to a channel factor node. For each value of  $\nu^{(\ell)}$ , a stable channel density can be obtained by applying a large number of operations through the Markov subgraph, using the approximate extrinsic density corresponding to that value of  $\nu^{(\ell)}$ . Thus, under Schedule A, the approximate channel density used to calculate  $\nu^{(\ell)}$  may be expressed as  $f_T(t) = \phi(\nu^{(\ell-1)})$ , that is, as a function of only one extrinsic density. (We will use  $\phi(\cdot)$  to represent the input–output characteristic of the parameter  $\nu$ .)

A small difficulty occurs since  $\nu^{(\ell)}$  and  $\nu^{\prime(\ell)}$  are calculated in different ways, as shown in Fig. 2. Since they are both obtained from extrinsic information at the  $(\ell - 1)$ th iteration, from the figure we see that  $\nu^{(\ell)}$  and  $\nu^{\prime(\ell)}$  are both functions of  $\nu^{(\ell-1)}$ . Now suppose we are interested in calculating  $\nu^{(\ell)}$ . From Fig. 2, we require the extrinsic information message to the parity checks, parameterized by  $\nu^{(\ell-1)}$ , and the extrinsic information message to the channel factors, parameterized by  $\nu^{\prime(\ell-1)}$ . Furthermore, since  $\nu^{\prime(\ell-1)}$  is in turn a function of  $\nu^{(i-2)}$ ,  $\nu^{(\ell)}$  may be calculated as a function of  $\nu^{(\ell-1)}$  and  $\nu^{(i-2)}$ .

We can do some further work to eliminate the dependence of  $\nu^{\prime(\ell)}$  on  $\nu^{(i-2)}$ , making it only a function of  $\nu^{(\ell-1)}$ , which is far more desirable since it makes the approximation “memoryless”

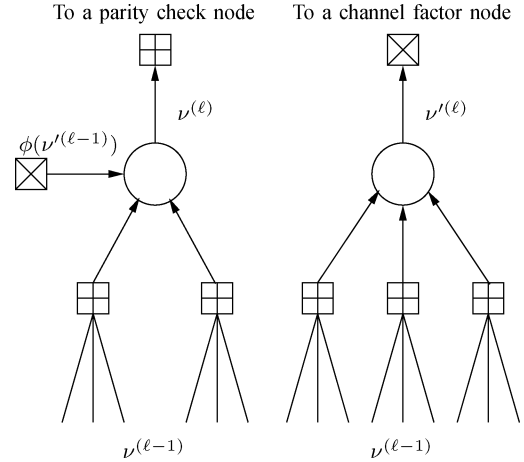


Fig. 2. Different extrinsic message calculations for the message to a parity check, and for a message to a channel factor. The input to both blocks is  $\nu^{(\ell-1)}$ , representing the extrinsic density at iteration  $i - 1$ . At the  $i$ th iteration,  $\nu^{(\ell)}$  represents the message passed to the parity check, while  $\nu^{\prime(\ell)}$  represents the message passed to the channel factor.

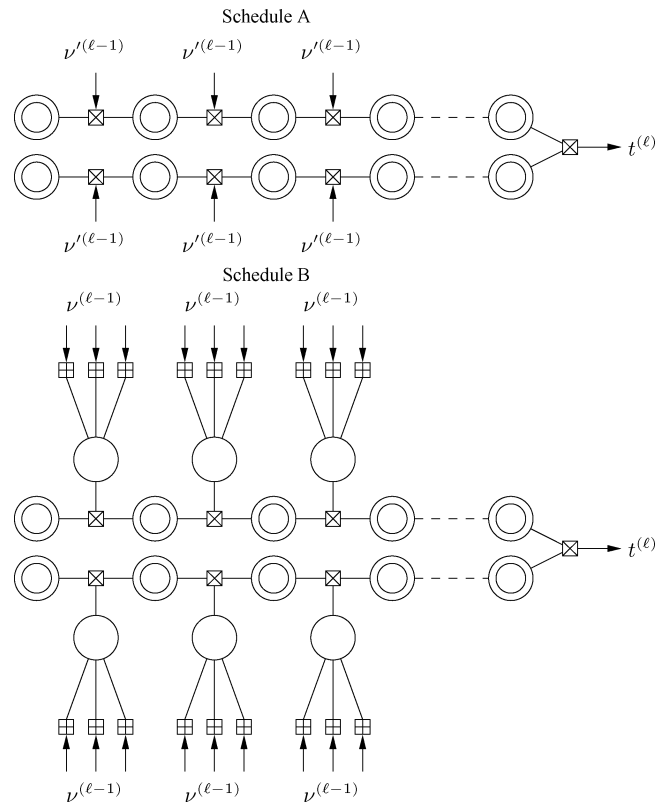


Fig. 3. Figure depicting the difference in calculating the channel message  $t^{(\ell)}$  between Schedule A (top) and Schedule B (bottom). In Schedule B, an additional LDPC subiteration before each channel factor node means that the messages can be functions of  $\nu^{(\ell-1)}$  rather than  $\nu^{\prime(\ell-1)}$ .

(i.e., independent of what happened at any iteration previous to  $i - 1$ ). To accomplish this, we only need to make an adjustment to Schedule A, to bring  $\nu^{(\ell)}$  into sync with  $\nu^{(\ell)}$ .

**Message-Passing Schedule B:** This schedule is largely the same as Schedule A, except that messages passed from the LDPC subgraph to the channel factor node are subjected to an additional iteration through the LDPC subgraph, as in Fig. 3.

This message-passing schedule can only be defined in terms of the structure of the local neighborhood, rather than in terms

of a “real” message-passing schedule that could be implemented in a system. Nonetheless, we get what we wanted: from the perspective of the approximation, we calculate  $\nu^{(\ell)}$  as a function of  $\nu^{(\ell-1)}$ ; we then obtain the stable channel message density using  $\nu^{(\ell)}$ ; and we finish the calculation of  $\nu^{(\ell)}$  as a function of  $\nu^{(\ell-1)}$  and this channel density. The only change in Fig. 2 is that the channel message is now  $\phi(\nu^{(\ell)})$  rather than  $\phi(\nu^{(\ell-1)})$ , since the calculations leading to  $\nu^{(\ell)}$  are split up and reordered to allow  $\nu^{(\ell)}$  to “catch up” with  $\nu^{(\ell)}$ .

### B. Schedules and Density Evolution

In this subsection, we show that the standard schedule has the same ultimate performance as both Schedules A and B. In other words, we can design our code under the assumption that Schedule B was used for message-passing, and expect similar performance under the standard message-passing schedule, as well as any other message-passing schedule that satisfies some mild conditions.

Using an irregular code, the local neighborhood of an edge  $e$  is a random variable, so there exists an *ensemble* of possible local neighborhoods. Using the standard message-passing schedule, let

$$\mathbb{N}^{(\ell)} = \left\{ (\mathcal{N}_e^{(\ell)}, \Pr(\mathcal{N}_e^{(\ell)})) \right\}$$

represent this ensemble, which contains all possible local neighborhoods under  $\ell$  iterations of the standard schedule, along with their probabilities of occurrence. The probability  $\Pr(\mathcal{N}_e^{(\ell)})$  is dependent on the degree sequence  $(\lambda, \rho)$ , since the degree sequence determined the occurrence probabilities of nodes of each type and degree. We assume the members of  $\mathbb{N}^{(\ell)}$  are cycle-free, since cycle-free local neighborhoods occur with  $\Pr \rightarrow 1$ . Let  $\mathbb{A}^{(\ell)}$  and  $\mathbb{B}^{(\ell)}$  represent the equivalent ensembles under  $\ell$  iterations of message-passing Schedules A and B, respectively. Also let  $P_{\text{err}}(\mathbb{N}^{(\ell)})$  represent the average probability of error for messages arising from the ensemble  $\mathbb{N}^{(\ell)}$ , and similarly for the other defined ensembles. Then the result may be stated formally as follows:

*Theorem 1:*

$$\lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathbb{A}^{(\ell)}) = \lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathbb{B}^{(\ell)}) = \lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathbb{N}^{(\ell)}).$$

In other words, whether the ensemble is constructed with message-passing Schedule A, message-passing Schedule B, or the standard schedule, the ultimate probability of error is the same, be it zero or some other value. The proof of the theorem is given in Appendix B, where we also give a corollary showing that the same result applies to any message-passing schedule that satisfies some mild conditions. Intuitively, the theorem holds because the local neighborhoods generated by  $\ell$  iterations of Schedules A and B both “fit inside” the local neighborhood  $\mathcal{N}_e^{(\ell^+)}$  for some  $\ell^+ \geq \ell$ , and “contain” the local neighborhood  $\mathcal{N}_e^{(\ell^-)}$  for some  $\ell^- \leq \ell$ . The probability of error for Schedules A and B is sandwiched between the probability of error for  $\mathcal{N}_e^{(\ell^+)}$  and  $\mathcal{N}_e^{(\ell^-)}$ , so if  $\ell^- \rightarrow \infty$  as  $\ell \rightarrow \infty$ , the schedules all share the same limit.

## IV. APPROXIMATE DE ALGORITHMS

### A. Overview and Definitions

In this section, we propose a design algorithm by adapting the semi-Gaussian approximation [7] to the requirements of Schedule B. We begin by reviewing the semi-Gaussian approximation and its useful variants, then explain how to calculate the stable message densities to account for channel estimation, and finally, bring the two together to find approximate density evolution algorithms for estimation-decoding.

We give several procedures in this section, which are related as follows.

- **Procedures A, B1, and B2** reiterate elements of the semi-Gaussian approximation from [7], and are given here for completeness.
  - **Procedure A** reiterates the semi-Gaussian approximation for memoryless channels, based on [7].
  - **Procedures B1 and B2** give a variant of the semi-Gaussian approximation, which separates the pure extrinsic information (from the code) and the channel information (from the channel observation). We divide it into two procedures, as we will reuse Procedure B1 (which deals with pure extrinsic information, and is independent of the channel).
- **Procedures C, D, and E** form the core of our method for dealing with channel estimation in a design tool, and adapting the semi-Gaussian approximation to the case of estimation-decoding.
  - **Procedure C** explains how to calculate a set of channel message densities for the GE channel using Schedule B, in a manner that is compatible with the semi-Gaussian approximation. (Later, we replace this with Procedure F, which generalizes Procedure C to arbitrary Markov channels.)
  - **Procedure D** adapts Procedure A to the case of estimation-decoding, and is dependent on Procedure C to precalculate a set of channel message densities.
  - **Procedure E** adapts Procedure B2 to the case of estimation-decoding, again relying on Procedure C to precalculate the channel message densities. It is used along with an unmodified Procedure B1.

The purpose of the procedures is to perform all the calculations necessary to set up the design problem. We show that, following these calculations, the design problem can be solved using linear programming.

We now give, or restate, useful notation and definitions which will be useful in this section. Our message notation partially follows the notation from [14]. The channel messages are given by  $t$ , and their densities are given by  $f_T(t)$ . Extrinsic messages are designated  $q$ , and their densities are given by  $f_Q(q)$ . The scalar  $\nu$  (approximately) parameterizes the messages, so when invoking the approximation, we will write  $f_T(t; \nu)$ , or  $f_Q(q; \nu)$ , to make the dependence explicit. For the value of  $\nu$  at iteration  $\ell$ , we will write  $\nu^{(\ell)}$ , to be consistent with the notation for local neighborhoods and ensembles. Thus, the functions  $\phi(\cdot)$ , which track the evolution of  $\nu$ , are defined iteratively, as  $\nu^{(\ell+1)} = \phi(\nu^{(\ell)})$  or  $\nu^{(\ell+1)} = \psi(\nu^{(\ell)})$  (where the latter notation indicates

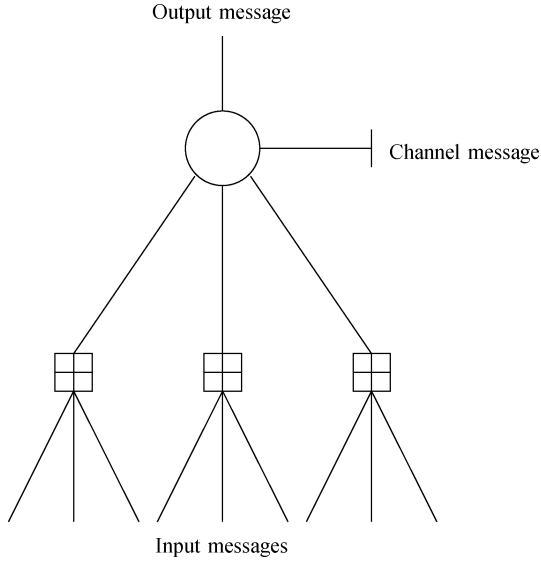


Fig. 4. An example factor graph segment over which the SGA is calculated, similar to a figure from [7].

that the input–output relation has been modified to account for estimation–decoding).

### B. The Semi-Gaussian Approximation

The semi-Gaussian approximation (SGA) has two features which distinguish it from the Gaussian approximation (GA) [15]. Recall that all approximations use a single parameter  $\nu$  to characterize extrinsic information densities. First, under the GA,  $\nu$  is the mean of the extrinsic message density; while under the SGA,  $\nu$  is the probability that the Gaussian density is less than zero (i.e., under the all-one codeword assumption, the probability of error in the message). Second, and more importantly, while the GA assumes that the densities at the output of both symbol variable nodes and parity-check nodes are both Gaussian, the SGA makes this assumption only about densities at the output of the symbol variable nodes, where the Gaussian assumption is generally more accurate, which makes it preferable for handling channels with non-Gaussian channel messages. The SGA thus approximates the densities only at the inputs and outputs of blocks such as in Fig. 4.

To implement the SGA for a particular (potentially non-Gaussian) channel and a particular degree sequence  $(\lambda, \rho)$ , we must find the intermediate input–output characteristic function given that the check degree sequence is  $\rho$  and the variable degree is  $k$ :

$$\nu^{(\ell+1,k)} = \phi_k(\nu^{(\ell)}; \rho). \quad (7)$$

Thus,  $\nu^{(\ell+1,k)}$  is the conditional probability of error in the message, given  $\rho$  and  $k$ . Using these functions, the overall input–output characteristic function for a variable degree sequence  $\lambda$ , designated  $\phi(\nu^{(\ell)}; \lambda, \rho)$ , may be calculated. Marginalizing over the variable degree, we can calculate the

value of  $\nu^{(\ell+1)}$ , which is the average probability of error in the message, as follows:

$$\begin{aligned} \nu^{(\ell+1)} &= \phi(\nu^{(\ell)}; \lambda, \rho) \\ &= \sum_{k=1}^{v_{\max}} \Pr(\text{err} \mid \text{degree } k; \nu^{(\ell)}, \rho) \Pr(\text{degree } k) \\ &= \sum_{k=1}^{v_{\max}} \phi_k(\nu^{(\ell)}; \rho) \lambda_k \end{aligned} \quad (8)$$

where “err” refers to the event that the message passed over an edge is in error, and “degree  $k$ ” refers to the event that the edge over which the message is passed is incident to a node of degree  $k$ . Thus,  $\phi(\nu^{(\ell)}; \lambda, \rho)$  is a linear combination of the individual input–output characteristic functions  $\phi_k(\nu^{(\ell)}; \rho)$ .

The following procedure is used to calculate the functions  $\phi_k(\nu^{(\ell)}; \rho)$ .

#### Procedure A: Fixed-Channel SGA:

- 1) (Initialization 1) Let  $f_T(t)$  represent the channel message density, and let  $p_0 := \int_{-\infty}^0 f_T(t) dt$  be the channel probability of error. Let  $\nu = \{\nu_1, \nu_2, \dots, \nu_{n_q}\}$  be a set of  $n_q$  quantization points for the interval  $[0, p_0]$ . Let  $k = 2$ .
- 2) (Initialization 2) For each  $\nu_i \in \nu$ , let  $m_i := (2 \operatorname{erfc}^{-1}(2\nu_i))^2$ , and let

$$f_Q(q; \nu_i) := \frac{1}{\sqrt{4\pi m_i}} \exp\left(-\frac{(q - m_i)^2}{4m_i}\right) \quad (9)$$

be the family of Gaussian extrinsic information densities  $f_Q(q; \nu_i)$  parameterized by  $\nu_i$ .

- 3) For each  $f_Q(q; \nu_i)$ , calculate exact density evolution over the factor graph in Fig. 4, using  $f_T(t)$  as the channel density,  $\rho$  as the check degree sequence, and  $k$  as the variable node degree. (A detailed explanation may be found in [3], [4].) Let  $\hat{f}_Q(q; \nu_i)$  be the resulting density.
- 4) For each  $\nu_i \in \nu$ , calculate the intermediate function

$$\phi_k(\nu_i; \rho) := \int_{q=-\infty}^0 \hat{f}_Q(q; \nu_i) dq \quad (10)$$

and interpolate between the points in  $\nu$  to find the value of the function at other points.

- 5) Let  $k := k + 1$ . If  $k > v_{\max}$ , stop; else, go to Step 3.

There is also a faster (yet slightly less accurate) algorithm from [7], using the principle of *intrinsic–extrinsic separation* (IES). Essentially, this technique approximates the output extrinsic message as Gaussian prior to the inclusion of the channel density  $f_T(t)$ , as in Fig. 5. Thus, the input–output characteristic function may be precalculated without knowledge of the channel density. To describe this procedure, we describe Procedure B1, in which the pure extrinsic information is calculated, and Procedure B2, in which the characteristic functions in (7) are calculated (separating these procedures will be useful in the next two sections):

**Procedure B1: Pure Extrinsic Information:** Calculate Procedure A, using  $f_T(t) := \delta(t)$ , i.e., a delta function at zero, in place of the true channel message density. Use  $\hat{\phi}_k(\nu_i; \rho)$  for  $\nu_i \in \nu$  to represent the output intermediate functions.

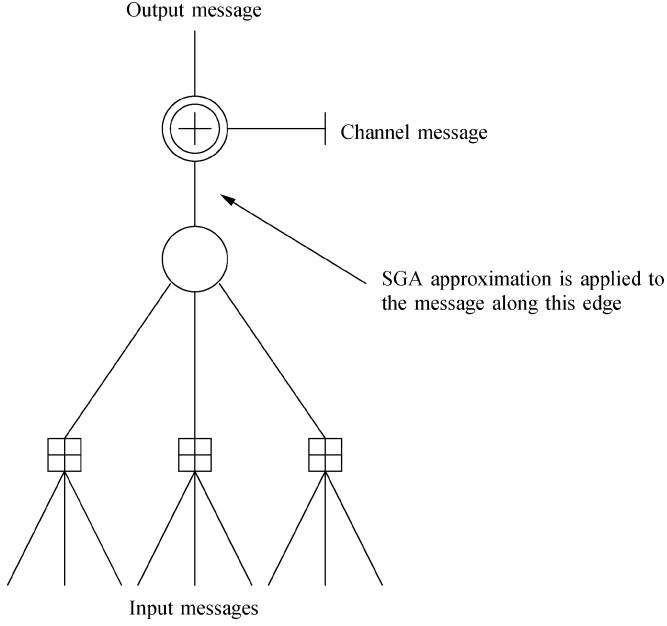


Fig. 5. Intrinsic–extrinsic separation—the SGA approximation is applied to the message at the output of the variable node, prior to the addition of the channel message.

**Procedure B2: Fixed Channel IES:** First, run Procedure B1. For each  $\nu_i \in \mathcal{V}$ , calculate  $m_i := (2 \operatorname{erfc}^{-1}(2\hat{\phi}_k(\nu_i; \rho)))^2$ , and obtain a Gaussian density  $\hat{f}_Q(q; \hat{\phi}_k(\nu_i; \rho))$  as in (9). Finally, for each  $k \in \{1, 2, \dots, v_{\max}\}$ , calculate

$$\phi_k(\nu_i; \rho) := \int_{\tau=-\infty}^0 \hat{f}_Q(\tau; \hat{\phi}_k(\nu_i; \rho)) \star f_T(\tau) d\tau$$

where the operator  $\star$  represents convolution over the dummy variable  $\tau$ .

In the next two subsections, we will use both the full SGA and the SGA using separation to design approximate DE algorithms.

### C. Full SGA

Here we implement an algorithm for approximating DE over Schedule B, using the SGA. Recall from Section II-C that  $N$  is the number of instances of channel estimation per iteration. As we argued in that section, if  $N$  is chosen to be sufficiently large, we can replace the channel densities  $f_T(t)$  with their stable counterparts, which are calculated on the assumption that the true extrinsic message density is replaced by one of the functions  $f_Q(q; \nu_i)$ , indexed by  $\nu_i$ .

The procedure for finding these stable densities is as follows.

#### Procedure C: Stable Densities:

- 1) Let  $\mathcal{V}' = \{\nu'_1, \nu'_2, \dots, \nu'_{n_q}\}$  be a quantization of the interval  $[0, 0.5]$ , so that  $\mathcal{V}'$  contains the parameters which approximate the extrinsic information to the channel.
- 2) For each  $\nu'_i \in \mathcal{V}'$ , let  $m_i := (2 \operatorname{erfc}^{-1}(2\nu'_i))^2$ , and calculate  $f_Q(q; \nu'_i)$  as in (9).
- 3) For each  $f_Q(q; \nu'_i)$ , obtain a channel density  $f_T(t; \nu'_i)$  by performing  $N \gg 1$  iterations of channel estimation DE

through the GE factor graph, using  $f_Q(q; \nu'_i)$  as the extrinsic message density. This is done in the same manner as in the DE algorithm described in [8].

An appropriate value for  $N$  can be found in an *ad hoc* manner, by examining the change in the channel message after each iteration of channel estimation DE in Step 3, and choosing  $N$  where the change becomes small. Intuitively, the value of  $N$  is the horizon on the number of symbols that participate in estimation for any given channel state, so this value will depend on the length of the channel memory (the longer the memory, the larger the required value of  $N$ ).

Now we concern ourselves with the intermediate functions. For each channel function  $f_T(t; \nu'_i)$  calculated using Procedure C, we need an input–output characteristic function dependent on this channel, which is obtained using Procedure A. To emphasize the dependence of this input–output characteristic on  $\nu'_i$ , and to distinguish it from the memoryless characteristic functions, we write this intermediate function as

$$\psi_k(\nu^{(\ell)}; \rho, \nu'_i).$$

Furthermore, as we argued in Section III, the input parameter  $\nu^{(\ell)}$  determines which of the  $\psi_k(\nu^{(\ell)}; \rho, \nu'_i)$  to use, so the functional relationship between  $\nu^{(\ell)}$  and  $\nu'_i$  must be found. From Schedule B, the extra iteration through the LDPC subgraph which produces  $\nu'_i$  represents pure extrinsic information, which is also used in the first step of the IES procedure. Thus, the relationship between  $\nu^{(\ell)}$  and  $\nu'_i$  can be calculated from intermediate functions using Procedure B1, represented by  $\hat{\phi}_k(\nu^{(\ell)}; \rho)$ .

These intermediate functions are calculated using the following procedure.

#### Procedure D: SGA With Channel Estimation:

- 1) (Initialization step) Define  $p_0$  and  $\mathcal{V}$  as before, and use  $f_Q(q; \nu_i)$  as in (9). Obtain a full variable degree sequence  $\lambda$  and a check degree sequence  $\rho$ . Let  $k = 1$ .
- 2) (Precalculation) Run Procedure C to obtain  $f_T(t; \nu'_i)$  for all  $\nu'_i \in \mathcal{V}'$ .
- 3) For each channel density  $f_T(t; \nu'_i)$ , and each variable degree  $j$ ,  $2 \leq j \leq v_{\max}$ , calculate each intermediate function  $\psi_j(\nu^{(\ell)}; \rho, \nu'_i)$ , obtained by calculating  $\phi_j(\nu^{(\ell)}; \rho)$  using Procedure A, with  $f_T(t; \nu'_i)$  as the channel density.
- 4) Obtain  $\hat{\phi}_k(\nu_i; \rho)$  for all  $\nu_i \in \mathcal{V}$ , and for all  $k \in \{2, 3, \dots, v_{\max}\}$  using Procedure B1.

Note that Step 2, marked “Precalculation,” only needs to be calculated once for each channel. These results may be stored and reused if a different check degree sequence is to be tested.

We now discuss how to obtain the overall input–output relationship from the intermediate functions, given some variable degree sequence  $\lambda$ . The extrinsic message to the channel is calculated similarly to the messages that carry information from symbol variable nodes to parity-check nodes. In density evolution, these variable-to-check message densities are averaged over  $\lambda$ , which contains the probabilities that a randomly selected edge is connected to a variable node of each degree. However, the extrinsic message to the channel is only passed along the edge connecting that symbol variable to its channel factor node, and there is exactly one of these edges per symbol variable node.

Thus, this message is averaged over  $\hat{\lambda}$ , which contains the probabilities that a randomly selected *node* has each degree, and has elements given by

$$\hat{\lambda}_k = \frac{\lambda_k/k}{\sum_{i=1}^{v_{\max}} \lambda_i/i}. \quad (11)$$

Furthermore, since the message is passed along the edge from the channel, all of the messages from the LDPC code are used, not all-but-one. We then calculate the input–output characteristic function for the extrinsic information destined for the channel, which is given similarly to (8) by

$$\hat{\phi}(\nu_i; \lambda, \rho) := \sum_{k=1}^{v_{\max}} \hat{\lambda}_k \phi_{k+1}(\nu_i; \rho) \quad (12)$$

for each  $\nu_i \in \mathbf{v}$ . A mapping  $h : \mathbf{v} \rightarrow \mathbf{v}'$  is then obtained by choosing  $h(\nu_i)$  as the element of  $\mathbf{v}'$  that is closest to  $\hat{\phi}(\nu_i; \lambda, \rho)$ . The overall input–output characteristic function is then calculated by

$$\psi(\nu_i; \lambda, \rho) := \sum_{k=1}^{v_{\max}} \lambda_k \psi_k(\nu_i; \rho, h(\nu_i)).$$

Note that both (12) and the above equation involve only linear combinations of the intermediate functions calculated using Procedure D.

#### D. SGA With IES

We can take advantage of IES to dramatically reduce the complexity of the full SGA algorithm. In Procedure D, much of the complexity stems from the requirement to calculate an input–output characteristic for each variable degree and each channel density function, in Step 3. Since IES uses an intermediate Gaussian approximation after the pure extrinsic information is calculated, but before the channel message is added, the approximation of the extrinsic information density is separated from the channel density.

Procedures B1 and B2 dealt with IES when the channel density is fixed. Because extrinsic information is unaffected by a changing channel density, Procedure B1 can be retained, while Procedure B2 is replaced by the following:

#### Procedure E: IES With Channel Estimation:

- 1) (Precalculation) As in Step 2 of Procedure D, run Procedure C to obtain  $f_T(t; \nu'_i)$  for all  $\nu'_i \in \mathbf{v}'$ .
- 2) (Precalculation) For each pair  $(\nu'(1), \nu'(2)) \in \mathbf{v}' \times \mathbf{v}'$ , calculate  $m := (2 \operatorname{erfc}^{-1}(2\nu'(1)))^2$ , obtain a Gaussian density  $\hat{f}_Q(q; \nu'(1))$ , and calculate

$$\begin{aligned} & \psi'(\nu_i; \nu'(1), \nu'(2)) \\ & := \int_{\tau=-\infty}^0 \hat{f}_Q(\tau; \nu'(1)) \star f_T(\tau; \nu'(2)) d\tau \end{aligned}$$

where the operator  $\star$  represents convolution over  $\tau$ .

- 3) As in Step 4 of Procedure D, run Procedure B1 to calculate  $\hat{\phi}_k(\nu_i; \rho)$  for all  $\nu_i \in \mathbf{v}$ .

Again, the step labeled “Precalculation” may be calculated in advance for all degree sequences.

Now, given a variable degree sequence  $\lambda$ , we calculate the function  $\psi(\nu_i; \lambda, \rho)$  by assembling the intermediate functions as follows. As in Procedure D, the function  $\hat{\phi}(\nu_i; \lambda, \rho)$  from (12)

is used to calculate the extrinsic messages to the channel, and the resulting function  $h : \mathbf{v} \rightarrow \mathbf{v}'$  in the same manner as before. However, we must also obtain the input–output relation for the pure extrinsic information, obtained prior to adding the channel message. We designate this quantity as  $\check{\phi}(\nu_i; \lambda, \rho)$ , and from the description of Procedure B1, this is given by

$$\check{\phi}(\nu_i; \lambda, \rho) := \sum_{k=1}^{v_{\max}} \lambda_k \hat{\phi}_k(\nu_i; \rho) \quad (13)$$

for each  $\nu_i \in \mathbf{v}$ . Let  $\tilde{h} : \mathbf{v} \rightarrow \mathbf{v}'$  represent the function that maps  $\check{\phi}(\nu_i; \lambda, \rho)$  to the closest value in  $\mathbf{v}'$ , similarly to  $h$ . Then the overall input–output relation is given by

$$\psi(\nu_i; \lambda, \rho) = \sum_{k=1}^{v_{\max}} \lambda_k \psi'(\nu_i; \tilde{h}(\nu_i), h(\nu_i)).$$

#### E. Degree Sequence Optimization

Under the SGA and all EXIT-like approximations, convergence occurs if and only if the decoding “tunnel” is open everywhere—that is, the transfer functions never intersect. In our case, this corresponds to the condition that  $\psi(\nu; \lambda, \rho) < \nu$  for all  $\nu$ . Our goal in optimizing these degree sequences is to obtain the degree sequence which maximizes rate, which still achieves  $P_{\text{err}} \rightarrow \epsilon$ . However, since the parity-check degree sequence is always fixed, from (5), maximizing  $R$  corresponds to maximizing  $\sum_{k=1}^{v_{\max}} \lambda_k/k$ . Thus, the optimization problem we gave in the introduction may be revised as follows:

**Given** a Markov channel with parameters  $\mathbf{c} = (\mathbf{P}, \mathbf{n})$  and a check degree sequence  $\rho$

**Maximize**  $\sum_{k=1}^{v_{\max}} \lambda_k/k$

**Subject to**  $\lambda_k > 0 \forall k$ ;  $\sum_{k=1}^{v_{\max}} \lambda_k = 1$ ; and  $\psi(\nu; \lambda, \rho) < \nu \forall \nu$ .

Since  $\psi(\nu; \lambda, \rho)$  is formed by a linear combination of functions, and since the objective function  $\sum_{k=1}^{v_{\max}} \lambda_k/k$  is linear, it appears that linear programming can be used to solve this optimization problem, as suggested in [7]. However, there is a small difficulty that must be resolved before linear programming can be directly applied. The extrinsic message to the channel is calculated by (12) and (13) in Procedures D and E, respectively, and the resultant mapping to a stable density  $f_T(t)$  is nonlinear. Thus, we may break the optimization into an iterative scheme: choose some arbitrary variable degree sequence  $\lambda^*$ , and use  $\lambda^*$  to calculate (12) or (13) for Procedure D and E, respectively. Then, keeping  $\lambda^*$  fixed, the above is then a linear optimization problem, and can be solved by linear programming. Finally, let  $\lambda^*$  equal the result of the linear programming procedure, and continue until the result is the same as  $\lambda^*$ .

## V. RESULTS FOR THE GE CHANNEL

### A. Illustration of Method

We first give some results that illustrate the operation of our algorithms. Consider a GE channel with parameters

$$(b, g, \eta_B, \eta_G) = (0.01, 0.01, 0.15, 0.01).$$

For this GE channel, we precalculate the channel information message densities  $f_T(t; \nu'_i)$  using Procedure C. Some examples of such densities for various values of  $\nu_i$  are given in Fig. 6,

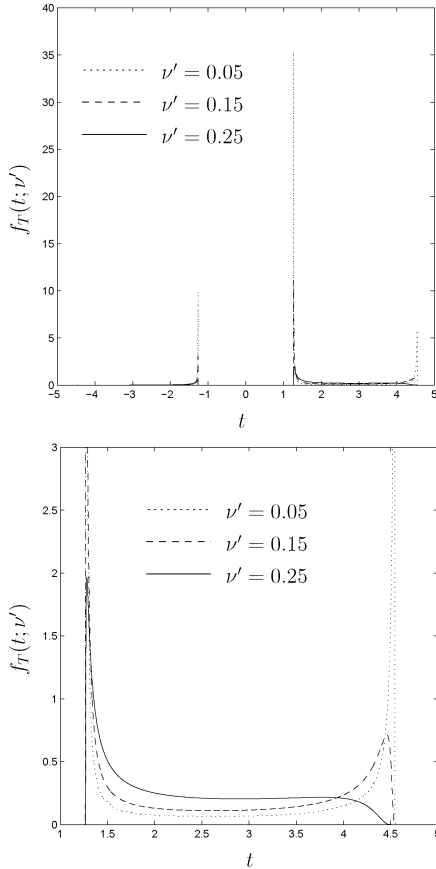


Fig. 6. An illustration of some stable channel message densities for various values of  $\nu'$ . The graph on the bottom is a close-up view of the positive values of the function on the top.

where  $N = 40$ . Note that with decreasing  $\nu'_i$  (corresponding to better extrinsic information), the “horns” of  $f_T(t; \nu'_i)$  become sharper. This phenomenon implies that, as expected, the channel estimates are of higher quality; if the channel states were perfectly known, these horns would become delta functions.

Our results that follow illustrate the effect of channel estimation on SGA analysis. In Procedure D, we obtained intermediate functions  $\psi_k(\nu; \rho, \nu')$  using the densities obtained in Procedure C. Some of these intermediate functions are given in Fig. 7 for the same GE channel as in Fig. 6, and using a check degree sequence with  $\rho_7 = 1$  as the only nonzero entry. For each variable degree  $k$ , we observe that decreasing  $\nu'$  (corresponding to better extrinsic information) causes the intermediate function to decrease everywhere. This has the effect of opening up the decoding “tunnel,” which makes it more likely for the decoding procedure to be successful.

## B. Code Results

Since we are using approximate DE, there is no guarantee of successful decoding in a physical system. In general, we have found that the design tool is often (but not always) overly “optimistic,” claiming that codes will decode successfully in channels where more accurate analysis (such as with exact DE) shows that they will not. This presents a problem, in that the code is designed with a particular channel  $c$  in mind (called the

TABLE I  
REGULAR DEGREE SEQUENCES, FOR COMPARISON

GE parameters ( $b, g, \eta_B, \eta_G$ )	Degree values	$R/C$
(0.01, 0.01, 0.5, 0.038)	$\lambda_3 = 1; \rho_4 = 1$	0.587
(0.01, 0.01, 0.197, 0.01)	$\lambda_3 = 1; \rho_6 = 1$	0.879

TABLE II  
DEGREE SEQUENCES OBTAINED USING PROCEDURE D FOR A CHANNEL WITH PARAMETERS ( $b, g, \eta_B, \eta_G$ ) = (0.01, 0.01, 0.22, 0.01), TESTED WITH DE

Degree values	Rate $R$	$R/C$
$\lambda_2 = 0.25, \lambda_3 = 0.3829, \lambda_7 = 0.1256,$ $\lambda_8 = 0.2415; \rho_6 = 0.13, \rho_7 = 0.87$	0.515	0.945
$\lambda_2 = 0.2, \lambda_3 = 0.4821, \lambda_9 = 0.3179;$ $\rho_6 = 0.04, \rho_7 = 0.96$	0.514	0.944
$\lambda_2 = 0.2, \lambda_3 = 0.282, \lambda_8 = 0.1948,$ $\lambda_9 = 0.1167, \lambda_{17} = 0.0253, \lambda_{18} = 0.1812;$ $\rho_8 = 0.4, \rho_9 = 0.6$	0.520	0.954
$\lambda_2 = 0.15, \lambda_3 = 0.3795, \lambda_{11} = 0.109,$ $\lambda_{12} = 0.2845, \lambda_{19} = 0.077;$ $\rho_8 = 0.55, \rho_9 = 0.45$	0.504	0.925

design threshold), but at the end of the procedure, the code might not decode successfully in  $c$ .

One straightforward method for improving a code is to decrease its average check degree. For example, a code with check degree 7, i.e.,  $\rho_7 = 1$ , can be improved by changing the check degree sequence to  $(\rho_6, \rho_7) = (0.1, 0.9)$ , though this is at the expense of code rate. In our results, we run the design tool with regular check degrees, which is widely believed to be an appropriate setting for capacity-achieving LDPC codes. Thus, there exists some  $c_{\max}$  where  $\rho_{c_{\max}} = 1$ . For each degree sequence obtained by the design tool, we check the sequence against the design threshold with exact DE, which was described for the GE channel in [8]. If decoding is not successful, we find the smallest  $\delta$  so that decoding is successful at the design threshold with  $(\rho_{c_{\max}-1}, \rho_{c_{\max}}) = (\delta, 1 - \delta)$ , and report that result.

It is obviously interesting to determine how closely the degree sequences obtained using Procedures D and E approach the Shannon limit. (We have generally set  $N = 40$ , as that value seems to be sufficient for good performance.) We use the ratio of code rate to channel capacity,  $R/C$ , as a metric in determining the goodness of the obtained degree sequences. In Tables II–VII, we give some degree sequences designed for various GE channels, and the value of  $R/C$  for this channel. For comparison, in Table I we give these values for the (3, 6)-regular and (3, 4)-regular LDPC codes. Three interesting conclusions can be drawn from these results:

- Comparing Table II to Table IV, which apply the same channel to Procedures D and E, respectively, the accuracy of Procedures D and E seems to be nearly the same. Thus, Procedure E seems to be preferable, due to its lower complexity.
- From Tables III–VI, Procedure E seems to be most accurate at rates greater than  $1/2$ , and the accuracy decays rapidly for low rates. However, degree sequences designed using the procedure have the largest values of  $R/C$  for rates close to  $1/2$ .
- From all tables, both Procedures D and E seem to be most accurate, and have highest  $R/C$ , when the  $v_{\max}$  and  $c_{\max}$  are relatively small. This is a concern, since channel capacity can only be attained as  $c_{\max}, v_{\max} \rightarrow \infty$ .

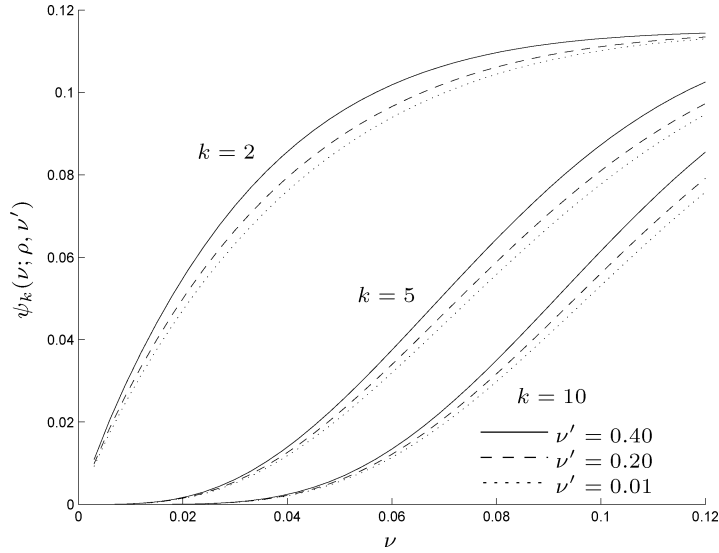


Fig. 7. An illustration of the effect of channel estimation on SGA approximation, with GE parameters  $b = g = \eta_G = 0.01$ , and  $\eta_B = 0.22$ ; and with  $\rho_7 = 1$ .

TABLE III

DEGREE SEQUENCES OBTAINED USING PROCEDURE E FOR A CHANNEL WITH PARAMETERS  $(b, g, \eta_B, \eta_G) = (0.01, 0.01, 0.15, 0.01)$ , TESTED WITH DE

Degree values	Rate $R$	$R/C$
$\lambda_2 = 0.26, \lambda_3 = 0.3984, \lambda_4 = 0.2245,$ $\lambda_5 = 0.1171; \rho_6 = 0.08, \rho_7 = 0.92$	0.577	0.919
$\lambda_2 = 0.25, \lambda_3 = 0.3683, \lambda_6 = 0.3718,$ $\lambda_7 = 0.0099; \rho_7 = 0.15, \rho_8 = 0.85$	0.590	0.939

TABLE IV

DEGREE SEQUENCES OBTAINED USING PROCEDURE E FOR A CHANNEL WITH PARAMETERS  $(b, g, \eta_B, \eta_G) = (0.01, 0.01, 0.22, 0.01)$ , TESTED WITH DE

Degree values	Rate $R$	$R/C$
$\lambda_2 = 0.25, \lambda_3 = 0.3858, \lambda_8 = 0.0784,$ $\lambda_9 = 0.2858; \rho_7 = 1$	0.516	0.948
$\lambda_2 = 0.2145, \lambda_3 = 0.338, \lambda_9 = 0.1302,$ $\lambda_{10} = 0.1934, \lambda_{12} = 0.1239;$ $\rho_7 = 0.45, \rho_8 = 0.55$	0.496	0.911
$\lambda_2 = 0.1877, \lambda_3 = 0.2996, \lambda_{10} = 0.3333,$ $\lambda_{16} = 0.0615, \lambda_{18} = 0.118;$ $\rho_8 = 0.48, \rho_9 = 0.52$	0.504	0.925
$\lambda_2 = 0.1668, \lambda_3 = 0.263, \lambda_9 = 0.2455,$ $\lambda_{10} = 0.0001, \lambda_{14} = 0.1444, \lambda_{30} = 0.1114,$ $\lambda_{33} = 0.0688; \rho_9 = 0.76, \rho_{10} = 0.24$	0.494	0.908

TABLE V

DEGREE SEQUENCE OBTAINED USING PROCEDURE E FOR A CHANNEL WITH PARAMETERS  $(b, g, \eta_B, \eta_G) = (0.01, 0.01, 0.30, 0.01)$ , TESTED WITH DE

Degree values	Rate $R$	$R/C$
$\lambda_2 = 0.2513, \lambda_3 = 0.2777, \lambda_6 = 0.0399,$ $\lambda_{10} = 0.2418, \lambda_{17} = 0.1893;$ $\rho_6 = 0.13, \rho_7 = 0.87$	0.439	0.920

These results confirm that the degree sequences rarely achieve  $P_{\text{err}} \rightarrow 0$  at the design threshold. We have found that a constraint on the value of  $\lambda_2$  can somewhat improve the accuracy of the approximation, at the expense of a small amount of rate, which can be observed in the second entry of Table II.

The best degree sequences in the literature are usually given for the memoryless Gaussian channel, and have thresholds extremely close to the Shannon limit. However, to obtain a fair comparison with other available degree sequences, we should

TABLE VI

DEGREE SEQUENCE OBTAINED USING PROCEDURE E FOR A CHANNEL WITH PARAMETERS  $(b, g, \eta_B, \eta_G) = (0.01, 0.01, 0.40, 0.01)$ , TESTED WITH DE

Degree values	Rate $R$	$R/C$
$\lambda_2 = 0.2323, \lambda_3 = 0.2537, \lambda_{10} = 0.1695,$ $\lambda_{12} = 0.0935, \lambda_{13} = 0.0099, \lambda_{41} = 0.0077,$ $\lambda_{46} = 0.2334; \rho_6 = 0.35, \rho_7 = 0.65$	0.347	0.815

TABLE VII

DEGREE SEQUENCE OBTAINED USING PROCEDURE E FOR A CHANNEL WITH PARAMETERS  $(b, g, \eta_B, \eta_G) = (0.0044, 0.0156, 0.50, 0.0192)$  (FROM [16]), TESTED WITH DE

Degree values	Rate $R$	$R/C$
$\lambda_2 = 0.25, \lambda_3 = 0.6209, \lambda_6 = 0.1291;$ $\rho_7 = 1$	0.5959	0.938

compare our results against degree sequences optimized for similar channels. Very little work exists in this area; however, the BSC has an impulsive channel density function, similarly to the GE channel density function. In [3], Richardson and Urbanke presented a rate-1/2 degree sequence for the BSC, directly optimized using DE, with  $R/C = 0.976$ . The best of our degree sequences designed using Procedures D and E typically have  $R/C$  values of around 0.95, implying that these are indeed good degree sequences, although there remains room for improvement. Nonetheless, these codes are the best known LDPC codes for the GE channel.

Furthermore, we remark that the values of  $R/C$  can be improved by searching for channels with lower capacity in which the codes nonetheless decode. For example, we have found using exact DE that the code in Table VII also succeeds in a channel with parameters

$$(b, g, \eta_B, \eta_G) = (0.0044, 0.0156, 0.5, 0.021)$$

for which  $R/C = 0.95$ . Similar channel searches lead to slightly improved  $R/C$  values for most of our codes. However, there is no systematic way of finding such channels, so this method is of limited use to a system designer.

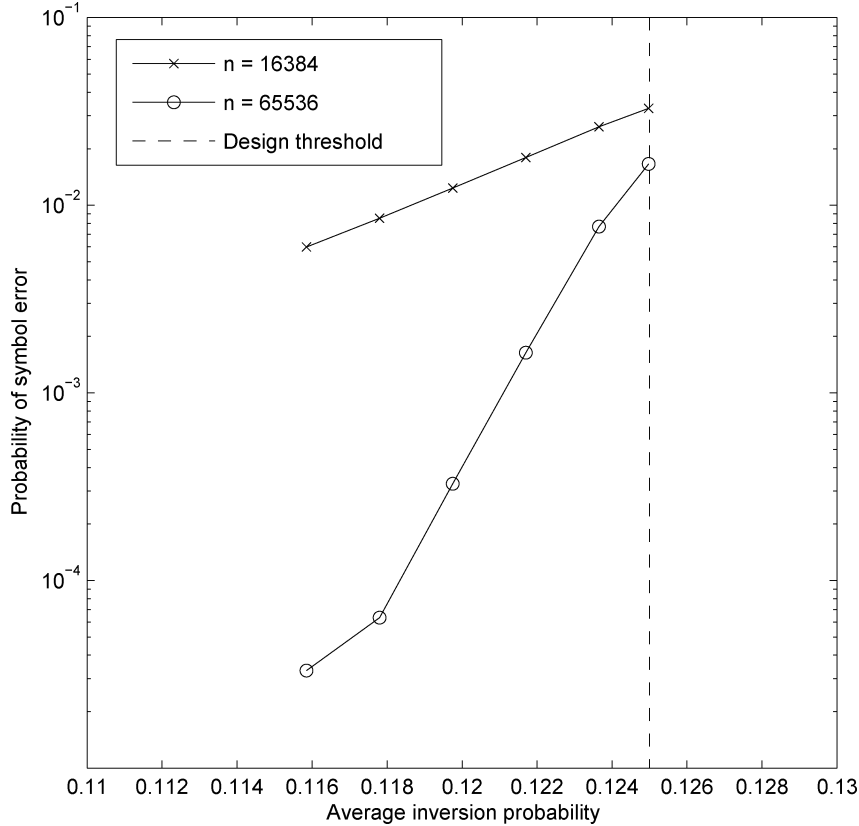


Fig. 8. Experimental results for the GE channel, using the code in Table VII. To produce the curve,  $\eta_G$  was varied to produce the given average inversion probability.

### C. Discussion

In this section, we consider some points raised by our design tool. First, in Fig. 8, we give experimental results for the code designed in Table VII, at code lengths  $n = 16384$  and  $n = 65536$ . Although the results roughly confirm the location of the threshold, we see that the memory inherent in the GE channel causes a significant gap to the threshold, even for these relatively long block lengths. This phenomenon was also observed for regular LDPC codes in [8], although the gap shrinks with increasing code length. Furthermore, the shape of Fig. 8 indicates an error floor. We have noticed that error floors can occur in these channels when  $\lambda_2$  is too large, which was also noticed in [3]. It is possible to modify the constraints in our design problem to restrict the maximum value of  $\lambda_2$ . However, we leave a criterion on the maximum value of  $\lambda_2$  to future work.

Second, the densities calculated by Procedure C can be used in concert with other methods of channel design, particularly with expedited forms of density evolution, such as fast density evolution [5]. The main stumbling block for density evolution analysis of estimation-decoding is the complexity of the messages in the Markov subgraph, which grows exponentially with the number of states. However, we can use the precalculated densities from Procedure C to approximately calculate the evolution of the channel message density. After each subiteration through the LDPC subgraph, an extrinsic message density

$f_Q(q)$  is obtained using an expedited form of density evolution. From  $f_Q(q)$ , the appropriate value of  $\nu$  can be calculated, and that value used to find a precalculated channel density  $f_T(t; \nu)$ . This new density is then fed into the expedited density evolution for the LDPC subiteration, and so on. The resulting (approximate) density evolution algorithm can be combined with any appropriate nonconvex optimization technique.

Finally, results have been presented for good turbo codes in the GE channel [16]. Due to a difference in approach, a direct comparison between these two codes is difficult. The method in [16] was to obtain a turbo code of a particular rate and then find, by simulation, the worst channel (according to some criterion) in which that code could be decoded successfully. In our case, we start with a channel, and find the highest rate code which successfully decodes in that channel in the limit of long block length. Thus, starting from the channel given in [16], which reported successful decoding with a turbo code at rate  $R = 1/2$ , we find a code that decodes successfully with rate  $R = 0.5959$ . When this code is shortened to the same block length as the simulated turbo code, we have observed by simulation that its performance is worse. However, its rate is considerably higher, so whether this code is “better” or “worse” depends on the designer’s criteria and perspective. To compare turbo and LDPC codes, at the same rate and in the same channel, would require optimization of both codes (e.g., the turbo code might need to be punctured), so we leave such a head-to-head comparison to future work.

## VI. DEGREE SEQUENCES FOR HIGHER ORDER MARKOV CHANNELS

System designers are most interested in capacity-approaching codes that work in a wide variety of channels. The design tool we have obtained in this paper was initially derived for the GE channel. As we shall show in this section, that design tool is tremendously versatile, and can be applied to virtually any Markov channel.

In our design tool, the current parameter  $\nu$  representing the extrinsic information was used to select a channel density function from a family of such functions, precalculated and corresponding to a quantized set of values for  $\nu$ . The family of functions was calculated by running exact DE through the channel factor graph, where the density of the extrinsic information message was a density corresponding to  $\nu$ . To adapt this method to a more complicated channel model, we need only find a way to obtain a similar family of channel message densities for an arbitrary channel.

Unfortunately, exact calculation of DE over the Markov subgraph is difficult when the number of states is greater than two. This is because the messages passed between the channel state nodes are probabilities over the channel states, so if there are  $|\mathcal{S}|$  states, the message passed between channel states is a vector with  $|\mathcal{S}|-1$  entries. The pdf of a vector is a multivariate function, whose evolution is difficult to track, even numerically. However, we are only interested in the channel message density, which is a univariate function regardless of the number of states or the nature of the channel. Furthermore, since the design tool implements approximate DE, the requirement to obtain an exactly accurate channel density function is somewhat relaxed. In particular, we can avoid the use of exact DE and instead use Monte Carlo methods to obtain a family of approximate channel densities. Such an approach has previously been used to evaluate the performance of turbo codes [17], and to implement DE in ISI channels [18].

Consider the following Monte Carlo procedure for obtaining a channel message density for some Markov channel  $c$ , intended to replace Procedure C.

### Procedure F: Monte Carlo Stable Densities:

- 1) Let  $\boldsymbol{\nu}' = \{\nu'_1, \nu'_2, \dots, \nu'_{n_q}\}$  be a quantization of the interval  $[0, 0.5]$ , so that  $\boldsymbol{\nu}'$  contains the parameters which approximate the extrinsic information to the channel.
- 2) For each  $\nu'_i \in \boldsymbol{\nu}'$ , let  $m_i := (2 \operatorname{erfc}^{-1}(2\nu'_i))^2$ , and calculate  $f_Q(q; \nu'_i)$  as in (9).
- 3) For each  $f_Q(q; \nu'_i)$ , generate a channel message as follows. Randomly generate a string of  $2N+1$  channel observations according to the channel model  $c$ , and a string of  $2N$  independent extrinsic messages according to the extrinsic density  $f_Q(q; \nu'_i)$ . Use the SPA to calculate the channel message corresponding to the  $(N+1)$ th channel observation.
- 4) Use  $M$  iterations of Step 3 and obtain a histogram of the resulting channel messages. This histogram, when normalized, is the Monte Carlo channel density  $f_T(t; \nu'_i)$ .

The value of  $M$  should be set very large, and we have found that setting it to  $10^5$ , or more, ensures that  $f_T(t; \nu'_i)$  is a reasonably accurate approximation of the true density.

To put our proposed method from the previous section into action, we choose a complicated channel for which no method exists to explicitly calculate DE: a binary Markov channel with three states. This channel is used as a motivating example; in principle, any Markov channel could be used in its place. The channel we tested has transition probability matrix

$$\mathbf{P} = \begin{bmatrix} 0.99 & 0.005 & 0.005 \\ 0.005 & 0.99 & 0.005 \\ 0.02 & 0.02 & 0.96 \end{bmatrix} \quad (14)$$

and inversion probability vector

$$\boldsymbol{n} = [0.01, 0.11, 0.5].$$

An optimized degree sequence for this channel was obtained using Procedure E. We obtained a degree sequence given by

$$\lambda_2 = 0.245, \lambda_3 = 0.4585, \lambda_6 = 0.1183, \lambda_8 = 0.1782; \\ \rho_7 = 1$$

which has a rate of  $R = 0.550$ . Experimental results illustrating this code's performance are given in Fig. 9, using a sum-product decoding algorithm. We observe that the actual performance of the code agrees reasonably closely with the designed channel. This result is highly encouraging to future work in designing codes for Markov channels.

## VII. CONCLUSION

In this paper, we have presented theoretical results concerning message-passing schedules, and have shown that a carefully chosen message-passing schedule can lead to an easier approximation for channel estimation under LDPC estimation-decoding. Using this result, we have proposed heuristics which allow irregular LDPC degree sequences to be optimized for Markov channels. These heuristics have given the best available codes for the GE channel, and shows promise in more general Markov channels.

The results presented in this paper are merely a few examples of the possible channels in which optimized LDPC codes may be obtained. For instance, it is easy to use this work to design codes for more complicated Markov channels, including Gaussian channels with memory. Nonetheless, the best of our codes achieved roughly 95% of the capacity of the corresponding GE channel. Although this is quite good, it falls short of the tiny gaps to capacity observed in memoryless channels. Future work should focus on closing the gap to capacity further.

## APPENDIX A

### STABLE DENSITIES FOR CHANNEL MESSAGES

In this appendix, we partially justify our approximation by showing that there exists a stable (or invariant) distribution for the channel messages in binary Markov channels. We do not address the more difficult question of convergence to this invariant distribution, although in applying the design techniques proposed in this paper, we have empirically observed this convergence in a wide range of scenarios.

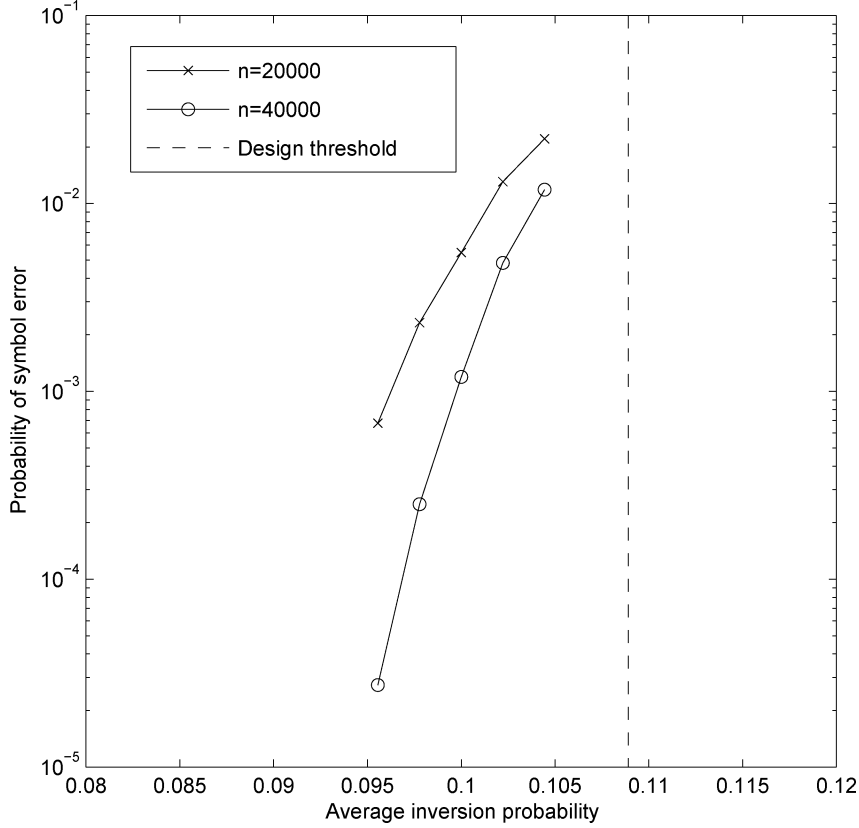


Fig. 9. Experimental results for the 3-state binary Markov channel.  $P_{\text{err}}$  is plotted with respect to  $\bar{\eta}$ , the average inversion probability. Results are given for channels with  $\mathbf{P}$  as in (14) and  $\mathbf{n} = [0.01, \eta_0, 0.5]$ , where  $\eta_0$  is varied to obtain the given average inversion probability.

We first show that the sequence of forward and backward messages form a Markov chain under two conditions: first, that the independence assumption holds, and second, that the extrinsic message density is always the same. The forward message  $\alpha_i$  forms a Markov chain if

$$f_{A_i|A_{i-1}, A_{i-2}, \dots}(\alpha_i | \alpha_{i-1}, \alpha_{i-2}, \dots) = f_{A_i|A_{i-1}}(\alpha_i | \alpha_{i-1}).$$

Under the SPA,  $\alpha_i$  is a function of the incoming forward message  $\alpha_{i-1}$ , the extrinsic message  $q_i$ , and the channel observation  $y_i$ . Thus

$$\begin{aligned} f_{A_i, Q_i, Y_i | A_{i-1}, A_{i-2}, \dots}(\alpha_i, q_i, y_i | \alpha_{i-1}, \alpha_{i-2}, \dots) &= \\ f_{A_i | Q_i, Y_i, A_{i-1}}(\alpha_i | q_i, y_i, \alpha_{i-1}) & \\ \cdot f_{Q_i}(q_i) f_{Y_i | A_{i-1}, A_{i-2}, \dots}(y_i | \alpha_{i-1}, \alpha_{i-2}, \dots) & \quad (15) \end{aligned}$$

since  $q_i$  is independent of the forward messages. Furthermore, if  $s_i$  is the channel state corresponding to the observation  $y_i$ , we can write

$$\begin{aligned} f_{Y_i | A_{i-1}, A_{i-2}, \dots}(y_i | \alpha_{i-1}, \alpha_{i-2}, \dots) &= \\ \sum_{s_i \in \mathcal{S}} f_{Y_i | S_i}(y_i | s_i) f_{S_i | A_{i-1}, A_{i-2}, \dots}(s_i | \alpha_{i-1}, \alpha_{i-2}, \dots) & \\ = \sum_{s_i \in \mathcal{S}} \sum_{s_{i-1} \in \mathcal{S}} f_{Y_i | S_i}(y_i | s_i) p_{S_i | S_{i-1}}(s_i | s_{i-1}) & \\ \cdot p_{S_{i-1} | A_{i-1}}(s_{i-1} | \alpha_{i-1}) & \\ = f_{Y_i | A_{i-1}}(y_i | \alpha_{i-1}) & \end{aligned}$$

where the second line follows from the fact that  $\alpha_{i-1}$  contains summarized probabilities of  $s_{i-1}$ , which are calculated using the SPA. Thus, from (15)

$$\begin{aligned} f_{A_i | A_{i-1}, A_{i-2}, \dots}(\alpha_i | \alpha_{i-1}, \alpha_{i-2}, \dots) &= \\ \sum_{y_i} \int_{q_i} f_{A_i | Q_i, Y_i, A_{i-1}}(\alpha_i | q_i, y_i, \alpha_{i-1}) f_{Q_i}(q_i) & \\ \cdot f_{Y_i | A_{i-1}}(y_i | \alpha_{i-1}) dq_i & \\ = f_{A_i | A_{i-1}}(\alpha_i | \alpha_{i-1}) & \end{aligned}$$

which is invariant for all  $i$  so long as all  $q_i$  have the same density. A similar argument can be made to show that the backward message is also a Markov chain.

Under some mild conditions, it is a well-known result that the state probabilities of a Markov chain approach a stable distribution if the state space  $\mathcal{S}$  is finite and countable. However, if  $\mathcal{S}$  is uncountable, the existence of a stable distribution over the state space is much more difficult to prove. In this appendix, we adapt results from [19], showing first that the forward message Markov chain satisfies the weak Feller property, which states that all bounded, continuous functions on the state space are mapped into bounded, continuous functions by the probability kernel [19, pp. 134-137], as follows.

*Lemma 1:* The sequence  $[\alpha_1, \alpha_2, \dots]$  forms a Markov chain which satisfies the weak Feller property and the Positivity/Regularity Criterion.

*Proof:* We prove the statement for the GE channel; the lemma can be easily generalized to any binary Markov channel. Let  $\mathcal{S}_{y_i}(\alpha_i, q_i)$  represent the SPA calculation of  $\alpha_{i+1}$  given  $y_i$ ,

$\alpha_i$ , and  $q_i$ . For fixed  $q_i$  and  $y_i$ , it is easy to show that this function is continuous. To show the weak Feller property, for some  $\alpha_i$  we can represent the mapping of some bounded, continuous function  $h(\alpha)$  under the probability kernel

$$\begin{aligned}
Ph(\alpha_{i+1}) &= E[h(S_{y_i}(\alpha_i, q_i))] \\
&= \int_{q_i} h(S_{y_i=0}(\alpha_i, q_i)) f_{Y_i|A_i}(0 | \alpha_i) f_{Q_i}(q_i) dq_i \\
&\quad + \int_{q_i} h(S_{y_i=1}(\alpha_i, q_i)) f_{Y_i|A_i}(1 | \alpha_i) f_{Q_i}(q_i) dq_i \\
&= (\alpha_i(1 - \eta_G) + (1 - \alpha_i)(1 - \eta_B)) \\
&\quad \cdot \int_{q_i} h(S_{y_i=0}(\alpha_i, q_i)) f_{Q_i}(q_i) dq_i \\
&\quad + (\alpha_i\eta_G + (1 - \alpha_i)\eta_B) \\
&\quad \cdot \int_{q_i} h(S_{y_i=1}(\alpha_i, q_i)) f_{Q_i}(q_i) dq_i. \tag{16}
\end{aligned}$$

Following the proof of [19, Proposition 6.1.2], each integral in the last line of (16) results in a continuous function of  $\alpha_i$ , and multiplication by linear functions of  $\alpha_i$  does not affect continuousness. Thus,  $Ph(\alpha_{i+1})$  is continuous, which is sufficient to show that the chain satisfies the weak Feller property.

To show the Positivity/Regularity Condition, we trivially select  $\mathcal{C} := \mathcal{S}$ , i.e., the entire state space, and select  $V(\alpha) = \alpha$ . Since  $\mathcal{S} = [0, 1]$ ,  $V(\alpha) \leq 1$  always. Thus, the Positivity/Regularity Condition is satisfied.  $\square$

Thus, we have the following result.

*Theorem 2:* There exists an invariant probability measure for the Markov chain sequence  $[\alpha_1, \alpha_2, \dots]$ .

*Proof:* The theorem follows directly from Lemma 1 and [19, Theorem 12.3.4].  $\square$

Again, a very similar argument can be made for the backward message. Thus, there exist invariant pdfs for both the forward and backward messages.

Finally, we note that the channel message is calculated directly from the forward and backward messages. Thus, it is obvious that the channel message has a stable density if the forward and backward messages have stable densities.

## APPENDIX B PROOF OF THEOREM 1

We will prove this result for the case of regular LDPC codes, and then generalize the proof to the case of irregular LDPC codes. In a regular code, there is only one possible cycle-free member of  $\mathbb{N}^{(\ell)}$ , since the degrees of all the variables and nodes are deterministic. Furthermore, for regular codes, it is known from [8, Theorem 3] that

$$P_{\text{err}}(\mathbb{N}^{(\ell)}) \leq P_{\text{err}}(\mathbb{N}^{(\ell-1)}) \tag{17}$$

which implies that  $\lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathbb{N}^{(\ell)})$  exists (since  $P_{\text{err}}(\mathbb{N}^{(\ell)})$  is bounded below by zero). The observation (17) is easy to generalize to irregular codes.

We will now give a brief result, required in the proof of the theorem, which relates local neighborhoods that are subgraphs

of one another. Let the factor graph  $\mathcal{U}_e$  be a local neighborhood of the edge  $e$ . Another local neighborhood  $\mathcal{V}_e$  is a *subgraph* of  $\mathcal{U}_e$ , written  $\mathcal{V}_e \subseteq \mathcal{U}_e$  if

- $\mathcal{V}_e$  contains some or all of the nodes and edges from  $\mathcal{U}_e$ , and no nodes or edges that are not present in  $\mathcal{U}_e$ ;
- any variable node in  $\mathcal{V}_e$  has the same degree as the corresponding variable in  $\mathcal{U}_e$ ; and
- for any factor node in  $\mathcal{V}_e$  with smaller degree (i.e., connected to fewer variables) than the respective factor node in  $\mathcal{U}_e$ , the factor contained in the node in  $\mathcal{V}_e$  is the same as the one in  $\mathcal{U}_e$ , marginalized over the missing variables with respect to their prior distributions.

The first condition in the definition is true of the subgraph of any graph. The second and third conditions are necessary in the case of the factor graph, and describes the properties of the pdf represented by the factor graph  $\mathcal{V}_e$  with respect to the pdf represented by  $\mathcal{U}_e$ .

Letting  $P_{\text{err}}(\mathcal{V}_e)$  represent the probability of error of the message passed along edge  $e$ , when this message arises from a local neighborhood  $\mathcal{V}_e$ , the definition of a subgraph leads to the following lemma.

*Lemma 2:* If  $\mathcal{V}_e \subseteq \mathcal{U}_e$ , then  $P_{\text{err}}(\mathcal{V}_e) \geq P_{\text{err}}(\mathcal{U}_e)$ .

*Proof:* Marginalizing over missing variables with respect to a prior distribution, as in the third property of the subgraph, is equivalent to replacing all channel observations with erasures and executing the SPA, because the SPA performs marginalization, and without observations, the densities of the variables are their prior densities, by definition. Since  $\mathcal{V}_e$  and  $\mathcal{U}_e$  are cycle-free, they are optimal detectors for the variable attached to edge  $e$ , and by the Side Information Lemma [8], discarding observations in an optimal detector cannot result in a lower probability of error. Thus, since  $\mathcal{V}_e$  is equivalent to  $\mathcal{U}_e$  with some observations discarded,  $P_{\text{err}}(\mathcal{V}_e) \geq P_{\text{err}}(\mathcal{U}_e)$ .  $\square$

Because it is formed by a flooding-type schedule, the local neighborhood  $\mathcal{N}_e^{(\ell)}$  contains all possible sequences of edges through the estimation-decoding factor graph of length  $2\ell$  starting at the factor node attached to  $e$  (such that  $e$  is not the first edge traversed, by the definition of a local neighborhood). Thus, for any properly initialized message-passing schedule, its local neighborhoods are subgraphs of  $\mathcal{N}_e^{(\ell^*)}$  for some  $\ell^*$ . Furthermore,  $\mathcal{N}_e^{(0)}$  is trivially a subgraph of all possible local neighborhoods with any schedule. With this in mind, for all  $\ell \in \{0, 1, 2, \dots\}$ , let  $n_\ell^+$  represent the largest integer such that  $\mathcal{A}_e^{(\ell)} \subseteq \mathcal{N}_e^{(n_\ell^+)}$  and  $\mathcal{B}_e^{(\ell)} \subseteq \mathcal{N}_e^{(n_\ell^+)}$ , and let  $n_\ell^-$  represent the smallest integer such that  $\mathcal{N}_e^{(n_\ell^-)} \subseteq \mathcal{A}_e^{(\ell)}$  and  $\mathcal{N}_e^{(n_\ell^-)} \subseteq \mathcal{B}_e^{(\ell)}$ . Then we have the following.

*Lemma 3:*  $n_\ell^+ \geq n_\ell^-$  for all  $\ell$ , and  $\lim_{\ell \rightarrow \infty} n_\ell^- = \infty$ .

*Proof:* The first statement is obvious. To prove the second statement, at least one subiteration takes place in each subgraph for every iteration, so the shortest sequence of edges in any local neighborhood is  $2\ell$ . Thus, all possible sequences of edges of length  $2\ell$  are in the local neighborhood, and the local neighborhood is a subgraph of  $\mathcal{N}_e^{(\ell)}$ , so  $n_\ell^- \geq \ell$ , which obviously approaches infinity as  $\ell \rightarrow \infty$ .  $\square$

Now we can prove Theorem 1 for regular LDPC codes.

*Proof:* In the case of a regular LDPC code, only one member each of the ensembles  $\mathbb{N}^{(\ell)}$ ,  $\mathbb{A}^{(\ell)}$ , and  $\mathbb{B}^{(\ell)}$  has nonzero probability: respectively, these are the local neighborhoods  $\mathcal{N}_e^{(\ell)}$ ,  $\mathcal{A}_e^{(\ell)}$ , and  $\mathcal{B}_e^{(\ell)}$ . For Schedule B, Lemma 2, and the definitions of  $n_\ell^+$  and  $n_\ell^-$ , we have that

$$P_{\text{err}}(\mathcal{N}_e^{(n_\ell^+)}) \leq P_{\text{err}}(\mathcal{B}_e^{(\ell)}) \leq P_{\text{err}}(\mathcal{N}_e^{(n_\ell^-)}). \quad (18)$$

From Lemma 3, both  $n_\ell^+$  and  $n_\ell^-$  approach  $\infty$  as  $\ell \rightarrow \infty$ . Thus

$$\begin{aligned} \lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathcal{N}_e^{(n_\ell^+)}) &= \lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathcal{N}_e^{(\ell)}) \\ &\leq \lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathcal{B}_e^{(\ell)}) \\ &\leq \lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathcal{N}_e^{(n_\ell^-)}) \\ &= \lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathcal{N}_e^{(\ell)}) \end{aligned}$$

so we end up with the sandwiching

$$\lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathcal{N}_e^{(\ell)}) \leq \lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathcal{B}_e^{(\ell)}) \leq \lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathcal{N}_e^{(\ell)}).$$

The derivation is exactly the same for Schedule A, and the theorem follows.  $\square$

To prove Theorem 1 for irregular codes, we need to modify Lemma 2 so as to allow the same sandwiching in (18) when more than one local neighborhood in the ensemble has nonzero probability. To do so, we define a *subensemble* as follows. Let  $\mathbb{U}$  and  $\mathbb{V}$  be ensembles of local neighborhoods. Also, define the conditional probability  $\Pr(\mathcal{U}_e | \mathcal{V}_e)$  to be the probability that  $\mathcal{U}_e$  is the local neighborhood surrounding  $e$ , given that  $\mathcal{V}_e$  surrounds  $e$  (for instance, if  $\mathcal{U}_e$  is a subgraph of  $\mathcal{V}_e$ , then  $\Pr(\mathcal{U}_e | \mathcal{V}_e) = 1$ ). Then  $\mathbb{V}$  is a subensemble of  $\mathbb{U}$ , written  $\mathbb{V} \sqsubseteq \mathbb{U}$ , if the following condition holds:

- For any  $\mathcal{U}_e \in \mathbb{U}$  and any  $\mathcal{V}_e \in \mathbb{V}$ ,  $\Pr(\mathcal{U}_e | \mathcal{V}_e) > 0$  only if  $\mathcal{V}_e$  is a subgraph of  $\mathcal{U}_e$ .

In other words, if we know that the local neighborhood around  $e$  is  $\mathcal{V}_e$ , then  $\mathcal{U}_e$  is a valid local neighborhood for  $e$  only if it contains  $\mathcal{V}_e$  as a subgraph. Furthermore

$$\Pr(\mathcal{U}_e, \mathcal{V}_e) = \Pr(\mathcal{U}_e | \mathcal{V}_e) \Pr(\mathcal{V}_e) \quad (19)$$

so it is also true that  $\Pr(\mathcal{U}_e, \mathcal{V}_e) > 0$  only if  $\mathcal{V}_e$  is a subgraph of  $\mathcal{U}_e$ . Thus, by marginalization,  $\Pr(\mathcal{U}_e) > 0$  can occur only if  $\mathcal{U}_e$  has at least one subgraph in  $\mathbb{V}$ , and  $\Pr(\mathcal{V}_e) > 0$  can occur only if  $\mathcal{V}_e$  is a subgraph of at least one member of  $\mathbb{U}$ . In other words, if  $\mathbb{V} \sqsubseteq \mathbb{U}$ , then  $\mathbb{V}$  contains those factor graphs which are the ‘‘building blocks’’ of  $\mathbb{U}$  as subgraphs.

The modified lemma is given as follows.

*Lemma 4:* If  $\mathbb{V} \sqsubseteq \mathbb{U}$ , then  $P_{\text{err}}(\mathbb{V}) \geq P_{\text{err}}(\mathbb{U})$ .

*Proof:* We can write the probability of error as

$$\begin{aligned} P_{\text{err}}(\mathbb{U}) &= \sum_{\mathcal{U}_e \in \mathbb{U}} P_{\text{err}}(\mathcal{U}_e) \Pr(\mathcal{U}_e) \\ &= \sum_{(\mathcal{V}_e, \mathcal{U}_e)} \Pr(\mathcal{V}_e, \mathcal{U}_e) P_{\text{err}}(\mathcal{U}_e). \end{aligned} \quad (20)$$

Let  $\mathcal{R}$  represent the set of pairs  $(\mathcal{U}_e, \mathcal{V}_e)$  so that  $\mathcal{V}_e$  is a subgraph of  $\mathcal{U}_e$ . Then the sum in (20) may be broken down as follows:

$$\begin{aligned} P_{\text{err}}(\mathbb{U}) &= \sum_{(\mathcal{U}_e, \mathcal{V}_e) \in \mathcal{R}} \Pr(\mathcal{V}_e, \mathcal{U}_e) P_{\text{err}}(\mathcal{U}_e) \\ &\quad + \sum_{(\mathcal{U}_e, \mathcal{V}_e) \notin \mathcal{R}} \Pr(\mathcal{V}_e, \mathcal{U}_e) P_{\text{err}}(\mathcal{U}_e) \\ &= \sum_{(\mathcal{U}_e, \mathcal{V}_e) \in \mathcal{R}} \Pr(\mathcal{V}_e, \mathcal{U}_e) P_{\text{err}}(\mathcal{U}_e) \end{aligned} \quad (21)$$

where the second line follows from the definition of  $\sqsubseteq$  and (19). By a similar argument, we may write

$$P_{\text{err}}(\mathbb{V}) = \sum_{(\mathcal{U}_e, \mathcal{V}_e) \in \mathcal{R}} \Pr(\mathcal{V}_e, \mathcal{U}_e) P_{\text{err}}(\mathcal{V}_e). \quad (22)$$

However, for each pair  $(\mathcal{U}_e, \mathcal{V}_e) \in \mathcal{R}$ ,  $P_{\text{err}}(\mathcal{U}_e) \leq P_{\text{err}}(\mathcal{V}_e)$  by Lemma 2, and thus each term in the sum from (21) is less than or equal to each term in the sum from (22). Thus,  $P_{\text{err}}(\mathbb{U}) \leq P_{\text{err}}(\mathbb{V})$ , which proves the lemma.  $\square$

For all  $\ell \in \{0, 1, 2, \dots\}$ , let  $k_\ell^+$  represent the largest integer such that  $\mathbb{A}^{(\ell)} \sqsubseteq \mathbb{N}^{(k_\ell^+)}$  and  $\mathbb{B}^{(\ell)} \sqsubseteq \mathcal{N}^{(k_\ell^+)}$ , and let  $k_\ell^-$  represent the smallest integer such that  $\mathbb{N}^{(k_\ell^-)} \sqsubseteq \mathbb{A}^{(\ell)}$  and  $\mathbb{N}^{(k_\ell^-)} \sqsubseteq \mathbb{B}^{(\ell)}$ . Lemma 3 is easily generalized to this case, and we give it without proof.

*Lemma 5:*  $k_\ell^+ \geq k_\ell^-$  for all  $\ell$ , and  $\lim_{\ell \rightarrow \infty} k_\ell^- = \infty$ .

Finally, we can write the proof of Theorem 1.

*Proof:* For Schedule B, from Lemma 4, we replace the sandwiching in (18) with

$$P_{\text{err}}(\mathbb{N}^{(k_\ell^+)}) \leq P_{\text{err}}(\mathbb{B}^{(\ell)}) \leq P_{\text{err}}(\mathbb{N}^{(k_\ell^-)}).$$

The proof then follows exactly the proof in the regular case, using Lemma 5 in place of Lemma 3.  $\square$

Finally, we note that the only schedule-specific properties we used were that Schedules A and B produce local neighborhoods that are cycle-free with  $\Pr \rightarrow 1$  as the block length goes to infinity, and that Lemma 5 holds. Thus, we can give the following corollary to Theorem 1:

*Corollary 1:* Let  $\mathbb{W}^{(\ell)}$  be an ensemble of local neighborhoods for an arbitrary message-passing schedule that satisfies both Lemma 5 and the cycle-free property. Then  $\lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathbb{W}^{(\ell)}) = \lim_{\ell \rightarrow \infty} P_{\text{err}}(\mathbb{N}^{(\ell)})$ .

In other words, the result in the theorem applies to a very wide class of message-passing schedules.

#### ACKNOWLEDGMENT

The authors wish to thank Prof. Masoud Ardakani of the University of Alberta for a number of helpful discussions.

#### REFERENCES

- [1] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: MIT Press, 1963.
- [2] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, ‘‘Improved low-density parity-check codes using irregular graphs,’’ *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.

- [3] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.
- [4] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.
- [5] H. Jin and T. J. Richardson, "A new fast density evolution," in *Proc. IEEE Information Theory Workshop*, Punta del Este, Uruguay, Mar. 2006, pp. 183–187.
- [6] S. ten Brink, "Convergence of iterative decoding," *Electron. Lett.*, vol. 35, pp. 806–808, May 1999.
- [7] M. Ardakani and F. R. Kschischang, "A more accurate one-dimensional analysis and design of LDPC codes," *IEEE Trans. Commun.*, vol. 52, no. 12, pp. 2106–2114, Dec. 2004.
- [8] A. W. Eckford, F. R. Kschischang, and S. Pasupathy, "Analysis of low-density parity-check codes for the Gilbert–Elliott channel," *IEEE Trans. Inf. Theory*, vol. 51, no. 11, pp. 3872–3889, Nov. 2005.
- [9] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 670–678, Apr. 2004.
- [10] N. Wiberg, "Codes and Decoding on General Graphs," Ph.D. dissertation, Linköping Univ., Linköping, Sweden, 1996.
- [11] A. P. Worthen and W. E. Stark, "Unified design of iterative receivers using factor graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 843–849, Feb. 2001.
- [12] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
- [13] N. Varnica and A. Kavčić, "Optimized low-density parity-check codes for partial response channels," *IEEE Commun. Lett.*, vol. 7, no. 4, pp. 168–170, Apr. 2003.
- [14] J. Garcia-Frias, "Decoding of low-density parity-check codes over finite-state binary Markov channels," *IEEE Trans. Commun.*, vol. 52, no. 11, pp. 1840–1843, Nov. 2004.
- [15] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.
- [16] J. Garcia-Frias and J. D. Villasenor, "Turbo decoding of Gilbert–Elliott channels," *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 357–363, Mar. 2002.
- [17] T. J. Richardson and R. L. Urbanke, "Thresholds for turbo codes," in *Proc. IEEE Int. Symp. Information Theory*, Sorrento, Italy, Jun./Jul. 2000, p. 317.
- [18] A. Kavčić, X. Ma, and M. Mitzenmacher, "Binary intersymbol interference channels: Gallager codes, density evolution, and code performance bounds," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1636–1652, Jul. 2003.
- [19] S. P. Meyn, *Markov Chains and Stochastic Stability*. London, U.K.: Springer-Verlag, 1993.