# Bilayer Low-Density Parity-Check Codes for Decode-and-Forward in Relay Channels

Peyman Razaghi, *Student Member, IEEE*, and Wei Yu, *Member, IEEE*

***Abstract*—This paper describes an efficient implementation of binning for decode-and-forward (DF) in relay channels using low-density parity-check (LDPC) codes. Bilayer LDPC codes are devised to approach the theoretically promised rate of the DF relaying strategy by incorporating relay-generated parity bits in specially designed bilayer graphical code structures. While conventional LDPC codes are sensitively tuned to operate efficiently at a certain channel parameter, the proposed bilayer LDPC codes are capable of working at two different channel parameters and two different rates: that at the relay and at the destination. To analyze the performance of bilayer LDPC codes, bilayer density evolution is devised as an extension of the standard density evolution algorithm. Based on bilayer density evolution, a design methodology is developed for the bilayer codes in which the degree distribution is iteratively improved using linear programming. Further, in order to approach to the theoretical DF rate for a wide range of channel parameters, this paper proposes two different forms of bilayer codes: the bilayer-expurgated and bilayer-lengthened codes. It is demonstrated that the rate of a properly designed bilayer LDPC code can closely approach the theoretical DF limit. Finally, it is shown that a generalized version of the proposed bilayer code construction is applicable to relay networks with multiple relays.**

***Index Terms*—Binning, decode-and-forward, density evolution, low-density parity-check (LDPC) codes, relay channel.**

## I. Introduction

LOW-density parity-check (LDPC) codes have proved to be powerful in approaching the capacity of single-user communication channels. The key idea of LDPC codes is to practically implement the random coding scheme of Shannon by enforcing a set of random parity-check constraints on information bits. While random coding is a fundamental element of single-user information theory, binning is of fundamental importance in multiuser scenarios. In this paper, we explore the use of LDPC codes to practically implement binning for the relay channel.

The classic work of Cover and El Gamal [1] describes two basic strategies for the relay channel: a decode-and-forward (DF) strategy in which the relay completely decodes the transmitted message and partially forwards the decoded message using a binning technique to allow the complete resolution of the message at the decoder, and a more complex compress-and-forward strategy in which the relay does not need to decode the source message. Cover and El Gamal proved that the DF strategy is capacity achieving for a class of degraded relay channels [1].

This paper focuses on practical implementation of the DF strategy for the relay channel. We restrict our attention to Gaussian relay channels at low signal-to-noise ratios (SNRs) for which binary linear codes are suitable. We show that, within a linear coding framework, the binning strategy in which a bin index of the codeword is transmitted by the relay to the destination can be interpreted as a *parity-forwarding* scheme. Further, the optimal code design for the DF strategy entails the design of an LDPC code working at two different channel SNRs: a higher SNR at the relay and a lower SNR at the destination. This represents a novel embedded LDPC code construction, named *bilayer LDPC codes* in this paper.

The main results of this paper are as follows. Two new ensembles of LDPC codes, bilayer-expurgated codes and bilayer-lengthened LDPC codes, are proposed to simultaneously approach the capacities of two Gaussian channels at two different SNRs. The performance analysis and the design methodologies for these new ensembles of bilayer LDPC codes are developed by generalizing density evolution [2] for standard LDPC codes to bilayer codes. A design technique based on linear programming is developed to optimize the variable degrees of the bilayer code. The two forms of bilayer code structure are devised to accommodate the optimization of check degrees. Together, it is shown that specially structured irregular bilayer LDPC code with carefully chosen variable and check degree sequences can asymptotically approach the theoretical DF rate (for binary inputs) of the relay channel to within a fraction of a decibel for a wide range of different channel conditions. Finally, the proposed code design is generalized for relay networks with multiple relays and it is shown that the proposed coding approach is applicable to more general networks.

### A. Related Work

Recent interests in wireless ad-hoc and sensor networks have fueled a new surge of research activities both on the theoretical aspects (e.g., [3]–[5]) and on practical implementation of cooperative communication schemes (e.g., [6]–[13]). Earlier work on the application of coding to the relay channel can be traced back to distributed coding diversity ideas [6], [14]–[16]. To provide coding diversity using a relay terminal, distributed turbo codes are employed in [6], [15], and [16], where relay-generated parity bits are used to enhance diversity.

Manuscript received September 1, 2006; revised May 24, 2007. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under a discovery Grant and under the Canada Research Chairs program. The material in this paper was presented in part at the IEEE International Conference on Communications (ICC), Istanbul, Turkey, June 2006.

The authors are with The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario ON M5S 3G4, Canada (e-mail: peyman@comm.utoronto.ca; weiyu@comm.utoronto.ca).

Communicated by G. Kramer, Guest Editor for the Special Issue on Relaying and Cooperation.

Digital Object Identifier 10.1109/TIT.2007.904983

While the papers mentioned above focus on enhancing cooperative diversity in a turbo code context, the present paper focuses on approaching the information-theoretical achievable DF rate using LDPC codes. LDPC codes are chosen in this paper because of their capacity-approaching performance on single-user channels and their flexibility in code construction. A comprehensive set of design techniques for the single-user channel is also available, allowing us to develop similar procedures for the relay channel.

In the context of LDPC coding, the code construction presented in this paper is related to several recent papers on coding for decode-and-forward for the half-duplex and full-duplex relay channels [17]–[20], [10]. In [17], an LDPC code structure similar to the bilayer expurgated LDPC codes proposed in this paper is devised for the relay channel, and a density evolution approach is used for code design. The code analysis and design methodology of [17] is based on density evolution for the single-user channel, where the performance of bilayer codes is approximated by that of a conventional LDPC code. The work of [18] considers the use of independent source and relay codebooks in a relay channel, which is optimal for relay channels where the relay-destination link is strong. In this case, successful decoding of the expurgated code at the destination is guaranteed, and the optimal code design simplifies to the design of a conventional LDPC code for the source-relay channel. It should be noted that although the use of independent codebooks is also optimal for certain fading channels in which coherent transmission is not possible [3, Th. 6], in general as will be discussed later, conventional LDPC codes are not adequate for DF in stringent channel scenarios, where the bandwidth and/or power available to the source-relay, relay-destination, and source-destination channels are the minimum required to achieve a certain DF rate.

In another series of work, [19] and [20] apply conventional LDPC codes optimized for single-user channels to the fading relay channel, and employ random puncturing to make an LDPC code work at two different rates. Using this approach, the authors report threshold gaps of 0.4–0.7 dB for several channel configurations with fixed relaying parameters. (For optimized relaying parameters, the threshold gap can be more than 1 dB [19, Fig. 6], [20, Fig. 8].) Finally, [10] uses a scheme based on the puncturing of conventional LDPC codes, where multiple-input–multiple-output coding schemes are adapted for DF.

The present work differs from [10], [17]–[19] in that instead of using random puncturing techniques to make a conventional LDPC code work at two different rate, we employ *structured* expurgating and lengthening techniques and explicitly design bilayer LDPC codes, which are capable of simultaneously approaching the capacities of the source-relay and source-destination links. Explicit design of LDPC codes is necessary for approaching the theoretically promised DF rate, because conventional LDPC codes are sensitively tuned to operate efficiently at a certain channel parameter; thus, a standard LDPC source code cannot be capacity approaching over both the source-relay channel and the source-destination channel due to the difference in the respective channel parameters. This is an important design issue if the channel from the relay to the destination is to be used efficiently for maximizing the overall DF rate. Finally, this paper shows that two different LDPC graph structures, namely bilayer expurgated and bilayer lengthened codes, are needed to approach the theoretical DF rate for the entire range of channel parameters.

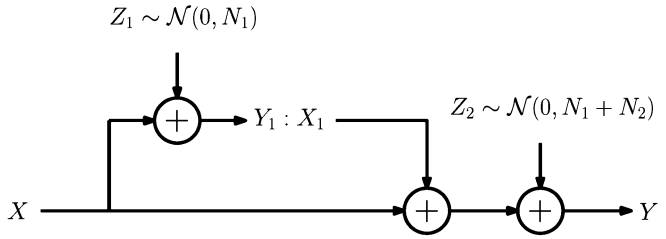


Fig. 1. The Gaussian relay channel: $X$ is power constrained to $P$; $X_1$ is power constrained to $P_1$.

The code design problem for the relay channel is also related to the general concept of *rateless* codes, popularized by the invention of fountain codes [21] and, more recently, raptor codes [22]. The bilayer LDPC codes proposed in this paper are similar to rateless codes in that both are capable of working at multiple rates. However, the code design requirement for the relay channel is also fundamentally different in the following aspect. In a relay channel, the extra information for the second code is transmitted via a separately coded channel from the relay to the destination. In contrast, the channel model for rateless codes typically assumes that additional bits are sent through the same channel (thus, are corrupted by the channel noise). In addition, fountain codes and raptor codes are designed specifically for the binary erasure channel (BEC); whether practical rateless codes exist for more general channel models is still an open research issue [23]. In this sense, the design methodology for fountain codes and raptor codes cannot be directly applied to the general relay channel, except in the erasure case.

Finally, the code construction proposed in this paper is also related to the use of punctured rate-compatible LDPC codes for incremental redundancy hybrid automatic repeat request (IR-HARQ) protocols for wireless transmission channels (e.g., [24]–[30]). In the HARQ setting, additional coded data bits are sent when the decoder fails to decode. Again, the additional bits can be potentially corrupted by the same channel, resulting in a different coding problem compared to the relay setting. Thus, the existing HARQ coding schemes are not directly applicable to DF.

## II. Coding for the Relay Channel

### A. DF Strategy

A Gaussian relay channel, as shown in Fig. 1, is defined by

$$Y_1 = X + Z_1 \tag{1}$$

$$Y = X + X_1 + Z_2 \tag{2}$$

where $Z_1 \sim \mathcal{N}(0, N_1)$ and $Z_2 \sim \mathcal{N}(0, N_1 + N_2)$ denote additive Gaussian noises at the relay and at the destination, respectively. The source has a power constraint $P$; the relay has a power constraint $P_1$.

We begin by briefly reviewing the DF strategy of [1, Section IV], and by identifying the structure of capacity-achieving codes in DF. In the DF strategy, the source selects a new message $w_i \in \{1, 2, \ldots, 2^{nR}\}$ in each block $i$. The set of source messages $\{1, 2, \ldots, 2^{nR}\}$ is randomly partitioned into $2^{nR_1}$ bins $(R_1 \leq R)$ of size $2^{n(R-R_1)}$. The relay message in block $i$ is represented by $s_i$ which is the bin index of $w_{i-1}$, the source message in block $i-1$. In block $i$, the relay transmits $\boldsymbol{X}_1(s_i)$, and the source transmits

$$\boldsymbol{X}(w_i \,|\, s_i) = \tilde{\boldsymbol{X}}(w_i) + \sqrt{\frac{(1-\alpha)P}{P_1}}\boldsymbol{X}_1(s_i) \tag{3}$$

where $\tilde{\boldsymbol{X}}(w_i)$ and $\boldsymbol{X}_1(s_i)$ are Gaussian random vectors of length $n$. $\tilde{\boldsymbol{X}}(w_i)$ and $\boldsymbol{X}_1(s_i)$ encode $w_i$ and $s_i$ via random codebooks of sizes $2^{nR}$ and $2^{nR_1}$, generated according to probability distributions $p_{\tilde{X}}(\tilde{x}) \sim \mathcal{N}(0, \alpha P)$ and $p_{X_1}(x_1) \sim \mathcal{N}(0, P_1)$, respectively. Note that (3) implies that the source optimally divides its total power budget $P$ into a fraction $\alpha P$ for transmitting new message $w_i$ and a fraction of $(1-\alpha)P$ for cooperatively transmitting the bin index $s_i$ of the previous source message $w_{i-1}$.

The decoding process in block $i$ goes as follows. The relay first decodes $w_i$ based on

$$\boldsymbol{Y}_1 = \boldsymbol{X} + \boldsymbol{Z}_1 = \tilde{\boldsymbol{X}}(w_i) + \sqrt{\frac{(1-\alpha)P}{P_1}}\boldsymbol{X}_1(s_i) + \boldsymbol{Z}_1. \tag{4}$$

It then computes $s_{i+1}$, the bin index of $w_i$, to be transmitted in the next block. Since $\boldsymbol{X}_1(s_i)$ is known at the relay, it can be subtracted. Therefore, successful decoding of $\tilde{\boldsymbol{X}}(w_i)$ is possible if

$$R \leq I(X; Y_1 \,|\, X_1) = \frac{1}{2}\log\left(1 + \frac{\alpha P}{N_1}\right). \tag{5}$$

The destination observes

$$\begin{aligned}\boldsymbol{Y} &= \boldsymbol{X} + \boldsymbol{X}_1 + \boldsymbol{Z}_2 \\ &= \tilde{\boldsymbol{X}}(w_i) + \left(1 + \sqrt{\frac{(1-\alpha)P}{P_1}}\right)\boldsymbol{X}_1(s_i) + \boldsymbol{Z}_2.\end{aligned} \tag{6}$$

The decoding of $w_i$ takes place in two stages.[1] First, the decoder decodes $s_i$, the bin index of $w_{i-1}$, while regarding $\tilde{\boldsymbol{X}}(w_i)$ as noise. The decoding is successful if

$$R_1 \leq \frac{1}{2}\log\left(1 + \frac{(\sqrt{P_1} + \sqrt{(1-\alpha)P})^2}{\alpha P + N_1 + N_2}\right). \tag{7}$$

With $s_i$ known, the destination now subtracts $\boldsymbol{X}_1(s_i)$ and proceeds with the decoding of $w_i$ in the second stage. This is performed in the next coding block, after the bin index $s_{i+1}$ is decoded. The bin index $s_{i+1}$ restricts the candidate $w_i$ into a set of size $2^{n(R-R_1)}$. Thus, decoding of $w_i$ (in block $i+1$) is successful if

$$R - R_1 \leq I(\tilde{X}; Y \,|\, X_1) = \frac{1}{2}\log\left(1 + \frac{\alpha P}{N_1 + N_2}\right). \tag{8}$$

Combining (5), (7), and (8), we see that the overall DF rate for the Gaussian relay channel is

$$R = \max_{\alpha} \min\left\{\frac{1}{2}\log\left(1 + \frac{\alpha P}{N_1}\right), \frac{1}{2}\log\left(1 + \frac{P + P_1 + 2\sqrt{(1-\alpha)PP_1}}{N_1 + N_2}\right)\right\}. \tag{9}$$

The optimal cooperation factor $\alpha$ in the above expressions is $\alpha = 1$ if $P/N_1 \geq P_1/N_2$, which is the case when the optimal strategy is *not* to allocate any portion of the transmitter's power to cooperate with the relay message [1]. Thus, no coherent transmission is needed between the relay and the transmitter [3, Remark 42].

When $P_1/N_2 < P/N_1$, coherent transmission of $s_i$ is beneficial. In this case, the optimal $\alpha$ is strictly between 0 and 1. Its numerical value can be found by equating the two terms within the minimization expression in (9); see [1].

### B. Coding for DF

Observe that the code design problem for optimal DF strategy involves the construction of two codes: $\mathcal{X}_1$ of rate $R_1$ and $\tilde{\mathcal{X}}$ of rate $R$. While the relay's codebook $\mathcal{X}_1$ can be constructed as a conventional error-correcting code that guarantees successful decoding at the destination, the source's codebook $\tilde{\mathcal{X}}$ must be constructed so that it can be decoded *both* at the relay and at the destination. The relay must be able to decode $\tilde{\mathcal{X}}$ at an SNR

$$\text{SNR}_+ = \frac{\alpha P}{N_1} \tag{10}$$

while the destination must be able to decode $\tilde{\mathcal{X}}$ under a different SNR

$$\text{SNR}_- = \frac{\alpha P}{N_1 + N_2} \tag{11}$$

but with the help of extra bin index information from the relay.

The code construction problem is abstracted in a schematic depicted in Fig. 2, where

$$R_+ = \frac{1}{2}\log(1 + \text{SNR}_+) \tag{12}$$

$$R_- = \frac{1}{2}\log(1 + \text{SNR}_-) \tag{13}$$

are used to denote the effective source-relay and the effective source-destination rates. The overall DF rate in terms of $R_+$ and $R_-$ now becomes

$$R = \min\{R_+, R_1 + R_-\}. \tag{14}$$

Observe that whenever $P_1/N_2 \leq P/N_1$, the optimal choice of the power allocation factor $\alpha$ in (9) always leads to $R_+ = R_1 + R_-$. This implies that the source's codebook $\tilde{\mathcal{X}}$ must be designed to *simultanously* approach the capacities of the effective source-relay and the effective source-destination channels at two different SNRs and at two different rates.

[1]Note that joint decoding of the source and relay messages is also possible. The use of joint decoding gives the same theoretical achievable rate, but may result in a lower error probability in practice. Although the proposed design methodology is based on successive decoding, the resulting codes can be used for joint decoding as well.

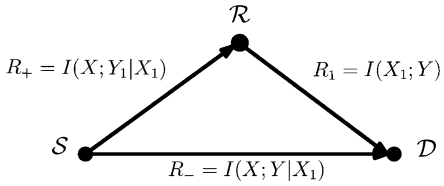


Fig. 2. The code construction problem for DF corresponds to two subproblems: constructing a source codebook to simultaneously approach rates $R_+$ and $R_-$, and constructing a conventional relay codebook to approach the rate $R_1 = R_+ - R_-$.

The condition $R_+ = R_1 + R_-$ does not necessarily hold, if the relay-destination link is very strong (i.e., $P_1/N_2 > P/N_1$), or if the power allocation factor $\alpha$ is not chosen optimally. (The latter case would lead to a strictly suboptimal overall DF rate.) In these cases, a conventional code designed for single-user channels can be sufficient [10], [18]–[20].

For example, when $R_+ \ll R_1 + R_-$, the overall DF rate is limited to $R_+$. The relay may transmit excess information about the source codeword to the destination at no cost to the overall DF rate. In this case, the source-destination code may operate at a rate well below $R_-$. Similarly, when $R_+ \gg R_1 + R_-$, the overall DF rate is limited to $R_1 + R_-$. The source-relay code may operate at a rate strictly below $R_+$. In both special cases, one of the two constraints in $\min\{R_+, R_1 + R_-\}$ is not tight. Consequently, a single-user conventional LDPC code designed for the tighter constraint suffices for achieving $\min\{R_+, R_1 + R_-\}$. In general, whenever $P_1/N_2 \leq P/N_1$, one would always optimize the relay operation in order to maximize the overall rate. This means $R_+ = R_1 + R_-$, which implies the need for a source code that is simultaneously capacity-achieving at two different rates and at two different SNRs. The rest of the paper treats this more stringent case.

### C. Binning Via Parity Generation

A main ingredient of the decode-and-forward strategy is binning. How can binning be implemented in practice? If we restrict our attention to Gaussian channels at low SNR (i.e., $R < 1$) for which binary signaling and linear codes are optimum, then binning may be implemented by generating extra parity bits on the codewords of $\tilde{\mathcal{X}}$. The generation of parity bits (or syndromes) is a natural way of partitioning a linear codebook into bins, with codewords in each bin satisfying a particular set of parity equations. The parity bits are exactly the bin indices. The idea of implementing structured binning via syndromes has been used in the past for Slepian–Wolf coding [31], [32] and for channel and source coding with side information [33].

To implement binning and block-Markov coding using this idea, the relay decodes the transmitted codeword $\tilde{\boldsymbol{X}}(w_i)$ in block $i$, generates extra parity bits for $\tilde{\boldsymbol{X}}(w_i)$, encodes them using an independent codebook $\mathcal{X}_1$, and sends the encoded bits to the destination in the next block. The destination decodes $\tilde{\boldsymbol{X}}(w_i)$ by utilizing the extra parity bits. Therefore, the DF strategy is a parity-forwarding strategy, and it gives rise to a bilayer code construction. The coding scheme based on this idea is described in Fig. 3.

Before considering the design of bilayer codes for approaching the Gaussian relay channel capacity, it is useful

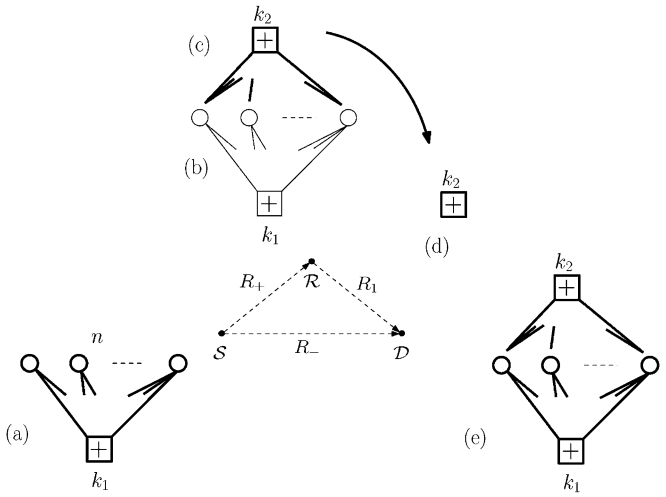


Fig. 3. Parity-forwarding implementation of DF using LDPC codes. (a) The source message is encoded using an $(n, n-k_1)$ LDPC code. (b) The relay decodes the source's codeword. (c) The relay then generates $k_2$ extra parity bits. (d) The $k_2$ parity bits are transmitted to the destination using a separate codebook. (e) The destination first decodes the extra $k_2$ parity bits, then decodes the source message over the bilayer code by searching for a codeword that satisfies $k_1$ zero parity bits and $k_2$ nonzero parity bits.

to ask whether such code exists in theory. It is well known that random linear codes are capacity-achieving for the binary symmetric channel (and for the Gaussian channel at a low SNR) under the maximum likelihood decoding. A subcode of a random code is also a random code. Therefore, under the maximum likelihood decoding, a bilayer code that is capacity-achieving at two different SNRs can be found.

The question becomes more interesting, if we consider practical iterative decoding methods. In this realm, theoretical results are available only for the BEC, for which capacity-achieving degree sequences for LDPC codes under iterative decoding methods have been identified [34], [35], [36]. We show in Appendix I that these codes can indeed be used to achieve the capacity of a particular class of erasure relay channels. The rest of the paper focuses on LDPC code design for the Gaussian case.

## III. Design of Bilayer Expurgated LDPC Codes

We now propose our first bilayer LDPC code construction for DF for the Gaussian relay channel. Let $\tilde{\mathcal{X}}$ be a linear $(n, n-k_1)$ LDPC code of rate $(n-k_1)/n$. The codebook $\tilde{\mathcal{X}}$ should be a capacity approaching code for the source-relay channel at $\text{SNR}_+$ with a rate $R_+$. Let $k_2$ be the number of extra random parity bits on the source codeword $\tilde{\boldsymbol{X}}(w_i)$, generated by the relay and provided to the destination. Then, a subcode of $\tilde{\boldsymbol{X}}(w_i)$ which satisfies two sets of parities: $k_1$ zero parities enforced by the source's codebook and $k_2$ extra presumably nonzero parity bits provided by the relay, should form an $(n, n-k_1-k_2)$ capacity-approaching code for decoding at the destination, i.e., at $\text{SNR}_-$ with a rate $R_-$. Note that the performance of a practical bilayer code is characterized by two gaps to the capacities at $\text{SNR}_+$ and at $\text{SNR}_-$.

The decoding of the subcode of $\tilde{\mathcal{X}}$ with the extra $k_2$ nonzero parity bits can be performed in the exact same way as the decoding of a conventional LDPC code. Since these subcodes are

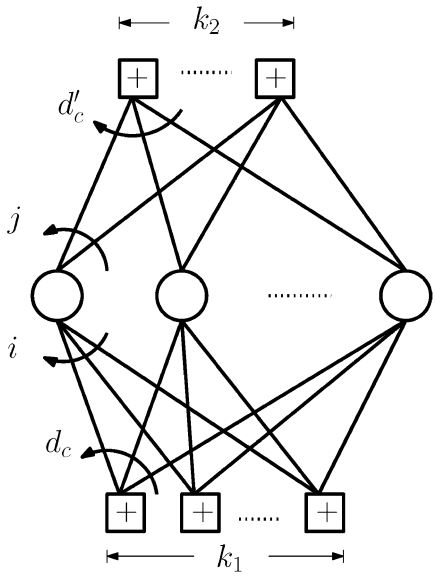


Fig. 4. The bilayer-expurgated code. The lower subgraph represents an LDPC code for source-relay channel. The overall graph represents an LDPC code for the destination.

cosets corresponding to different values of the $k_2$ extra parity checks, and are identical to each other geometrically (via a linear one-to-one mapping), we only need to ensure that the subcode represented by the zero-codeword coset leader is well designed.

The proposed LDPC code structure is shown in Fig. 4. We call the proposed code structure *bilayer-expurgated LDPC code*, as the overall graph represents an expurgated subcode of the lower-layer code. The first (lower) layer corresponds to an $(n, n-k_1)$ code and the second (upper) layer consists of the $k_2$ extra parity bits which modify the first layer in a way that the resulting $(n, n-k_1-k_2)$ subcode represented by the overall graph is suitable for the source-destination channel.

Although the code construction problem described in this section is derived for a full-duplex relay channel, the same code construction problem arises for relay channels with other multiple-access components, including the half-duplex relay channel and relay channels with a digital relay-destination link, as long as the relay-destination rate $R_1$ satisfies $R_1 \geq R_+ - R_-$.

This paper focuses on the practical design of $\tilde{\mathcal{X}}$, while ignoring multiple-access and interference subtraction issues at the destination. Practical implementations of superposition coding and interference subtraction have been well studied in the multiuser detection literature [37]. Our code construction also ignores error propagation, whereby an incorrect decoding of $\boldsymbol{X}_1(s_{i-1})$ negatively affects the decoding of $\tilde{\boldsymbol{X}}(w_i)$. In practice, the probability of error for the DF protocol is approximated by the sum of failure probabilities of the decoding of the source message at the relay, the decoding of relay message at the destination, and the decoding of the source message at the destination. While error propagation does not impact the design of capacity-achieving codes, it is practically important, especially in terms of outage probability in a wireless fading channel.

### A. Bilayer-Expurgated LDPC Code Ensemble

We now address the design of bilayer-expurgated LDPC codes. Our design methodology is based on a variation of the linear programming approach for LDPC rate optimization, first appeared in [38] and later modified in [39]. The exact optimization steps used in this paper are based on the approach of [40], which is also inspired by EXIT-chart based methods [41]–[44]. The complete design methodology as applied to conventional LDPC code design is presented in detail in Appendix II. The rest of this section deals with the design of bilayer codes.

An ensemble of bilayer-expurgated LDPC codes is defined as follows. The bilayer graph of the code, as shown in Fig. 4, consists of three sets of nodes and two sets of edges. The three sets of nodes correspond to one set of variable nodes, and two sets of check nodes: the lower check nodes corresponding to the check nodes in the lower subgraph of Fig. 4, and the upper check nodes corresponding to the check nodes in the upper subgraph in Fig. 4. Edges are grouped in two sets: those connecting the variable nodes to the lower check nodes, and those connecting the variable nodes to the upper check nodes. We call an edge a lower edge if it connects a variable node to a lower check node. Similarly, an upper edge denotes an edge belonging to the upper subgraph in Fig. 4.

The lower degree of a variable node is defined as the number of lower edges connected to it. Likewise, the upper degree of a variable node is the number of upper edges connected to it. The lower degree of an edge is defined as the lower degree of the variable node it is connected to, and similarly the upper degree of an edge is the upper degree of the variable node connected to that edge. The minimum lower variable degree is 2 as the lower subgraph should be a valid LDPC code for the source-relay channel. The minimum upper variable degree is 0, since some variable nodes may not participate in any of the $k_2$ extra parity checks generated by the relay. A variable node is said to have degree $(i,j)$ if it has a lower degree $i$ and an upper degree $j$. Similarly, an edge is of degree $(i,j)$ if it is connected to a degree $(i,j)$ variable node.

We assume regular check degrees for check nodes in the lower and upper graphs. The lower check degree of a bilayer graph $d_c$ denotes the number of edges connected to check nodes in the lower subgraph. Likewise, the upper check degree $d'_c$ equals to the number of edges connected to an upper check node. The ensemble of bilayer LDPC codes can be characterized by a variable degree distribution $\lambda_{i,j}, i \geq 2, j \geq 0$, which defines the percentage of edges with lower degree $i$ and upper degree $j$, and a parameter $\eta$ which defines the percentage of lower edges in the bilayer graph. Note that $\sum_{i\geq 2, j\geq 0} \lambda_{i,j} = 1$, and $0 < \eta < 1$. Note also that a bilayer LDPC code reduces to a conventional LDPC code if $\lambda_{i,j} = \lambda_i \lambda_j, i \geq 2, j \geq 0$ for some set of parameters $\lambda_i$ with $\lambda_0 = \lambda_1 = 0$.

### B. Bilayer Density Evolution

Because the ensemble of bilayer-expurgated LDPC codes is statistically different from a conventional LDPC code ensemble, conventional density evolution algorithm must be modified in order to accurately predict the performance of the bilayer code.

In the conventional density evolution analysis, the input message densities to all check nodes at each density evolution iteration are the same, since the probability that an edge, emanating from a check node, is connected to a degree $i$ variable node is equal to $\lambda_i$ for *all* check nodes. However, in a bilayer-expurgated code, there is a distinction between lower edges and upper edges. Therefore, evolution of two densities should be tracked: the lower density corresponding to the density of messages in the lower subgraph, and the upper density corresponding to the density of messages in the upper subgraph.

Let $p^l$ and $q^l$ denote the message probability density functions (pdf) at the input of the lower and upper check nodes in the lower and upper subgraphs, respectively, at the beginning of the $l$th decoding iteration. The message densities after a check update can be computed for $p^l$ and $q^l$ using the conventional density evolution check update as described in [2]. Let $p'^l$ and $q'^l$ denote the evolved versions of $p^l$ and $q^l$ after the check updates. For log-likelihood sum-product decoding with parallel updates, the density-evolution update at a degree $(i,j)$ variable node can be computed from $p'^l$ and $q'^l$ to obtain the message densities, $p_{i,j}^{l+1}$ and $q_{i,j}^{l+1}$, as follows:

$$p_{i,j}^{l+1} = (\otimes^{i-1} p'^l) \otimes (\otimes^{j} q'^l) \otimes p_c, \quad i \geq 2, j \geq 0 \quad (15)$$

$$q_{i,j}^{l+1} = (\otimes^{i} p'^l) \otimes (\otimes^{j-1} q'^l) \otimes p_c, \quad i \geq 2, j \geq 1 \quad (16)$$

where $p_c$ denotes the density of the log-likelihood ratio received over the channel, and $\otimes^i$ denotes convolution of order $i$. (By convention, for any density $f$, $\otimes^1 f = f$ and $\otimes^0 f = \delta$, where $\delta$ denotes the Dirac delta function.) The input message densities to the lower and upper check nodes, at the beginning of the $(l+1)$th iteration can be computed as follows:

$$p^{l+1} = \sum_{i\geq 2, j\geq 0} \frac{i}{i+j} \lambda_{i,j} p_{i,j}^{l+1} \quad (17)$$

$$q^{l+1} = \sum_{i\geq 2, j\geq 0} \frac{j}{i+j} \lambda_{i,j} q_{i,j}^{l+1}. \quad (18)$$

Note that the probability that a degree $(i,j)$ edge is a lower edge is given by $i/(i+j)$.

The lower-graph degree $(i,j)$ error profile function $e_{i,j}^1(p^l, q^l)$ is defined for a bilayer-expurgated LDPC code as the message error probability corresponding to the density $p_{i,j}^{l+1}$, after one density evolution iteration with input message densities $p^l$ and $q^l$. Similarly, $e_{i,j}^2(p^l, q^l)$ is defined as the message error probability corresponding to $q_{i,j}^{l+1}$ after one density evolution iteration for input message densities $p^l$ and $q^l$. Let $e(p^{l+1}, q^{l+1})$ denote the overall message error probability in the bilayer graph corresponding to the message densities $p^{l+1}$ and $q^{l+1}$. The overall message error probability at the beginning of the $(l+1)$th decoding iteration $e(p^{l+1}, q^{l+1})$ can be computed as a linear combination of $e_{i,j}^1(p^l, q^l)$ and $e_{i,j}^2(p^l, q^l)$ as follows:

$$e(p^{l+1}, q^{l+1}) = \sum_{i\geq 2, j\geq 0} \lambda_{i,j} \left( \frac{i}{i+j} e_{i,j}^1(p^l, q^l) + \frac{j}{i+j} e_{i,j}^2(p^l, q^l) \right). \quad (19)$$

The above formulation allows an iterative linear programming optimization of $\lambda_{i,j}$.

### C. Bilayer-Expurgated LDPC Code Optimization

The design of a bilayer-expurgated LDPC code involves finding a variable degree distribution $\lambda_{i,j}, i \geq 2, j \geq 0$, a parameter $\eta$, and a pair of check degrees $d_c$ and $d_c'$ such that the lower subgraph represents a capacity-approaching LDPC code over the source-relay channel at $\text{SNR}_+$, and the overall bilayer code is capacity approaching at $\text{SNR}_- < \text{SNR}_+$.

One way to formulate the design problem is to fix $d_c$, $d_c'$, and jointly optimize $\lambda_{i,j}$ and $\eta$. This approach is taken in our previous work [45]. It is equivalent to a joint optimization of both the lower subgraph and the overall graph to achieve the highest overall rate, but leads to high computational complexity. In this paper, we utilize a simpler approach of fixing the lower graph code to be an optimal capacity-approaching LDPC code at $\text{SNR}_+$ and searching for a variable degree distribution $\lambda_{i,j}$ that is consistent with the lower graph code and is capacity approaching at $\text{SNR}_-$.

Based on the iterative linear programming LDPC code optimization procedure described in Appendix II, a rate maximization problem is formulated for the bilayer code as follows. Fixing the check degrees $d_c$, $d_c'$, the rate of the bilayer graph is related to the parameter $\eta$, since $\eta$ depends on the number of check nodes in the graph via

$$\eta = \frac{d_c k_1}{d_c k_1 + d_c' k_2}. \quad (20)$$

By fixing the lower graph, i.e., fixing $n$, $k_1$ and the lower variable degree distribution $\lambda_i$, the rate of the bilayer code, defined by $1 - (k_1 + k_2)/n$, can be maximized by minimizing $k_2$ or equivalently maximizing $\eta$. The distribution $\lambda_i$ is related to $\lambda_{i,j}$ as follows:

$$\lambda_i = \frac{1}{\eta} \sum_{j\geq 0} \frac{i}{i+j} \lambda_{i,j}. \quad (21)$$

For a fixed $\lambda_i$, (21) can be rewritten in a linear format in terms of $\lambda_{i,j}$ and $\eta$

$$\sum_{j\geq 0} \frac{i}{i+j} \lambda_{i,j} - \eta \lambda_i = 0. \quad (22)$$

Fixing $d_c$, $d_c'$, and $\lambda_i$, an iterative linear programming update for $\lambda_{i,j}$ and $\eta$ can be formulated using (19) to iteratively maximize $\eta$ as follows:

$$\max_{\lambda_{i,j}, \eta} \quad \eta \quad (23a)$$

$$\text{s.t.} \quad \sum_{j\geq 0} \frac{i}{i+j} \lambda_{i,j} - \eta \lambda_i = 0 \quad i \geq 2 \quad (23b)$$

$$\sum_{i\geq 2, j\geq 0} \lambda_{i,j} \left( \frac{i}{i+j} e_{i,j}^1(p^l, q^l) + \frac{j}{i+j} e_{i,j}^2(p^l, q^l) \right) < \mu^h e(p^l, q^l), \qquad l = 1 \cdots L \quad (23c)$$

$$\sum_{i\geq 2, j\geq 0} \lambda_{i,j} = 1 \quad (23d)$$

where $h$ is the optimization iteration number, and $l$ is the decoding iteration number. The coefficient $0 < \mu^h < 1$ plays the

same role as the $\mu^h$ in (36) (see Appendix II) and is slightly increased at each optimization iteration, eventually approaching 1. The error profiles $e^1_{i,j}(p^l, q^l)$, $e^2_{i,j}(p^l, q^l)$, and $e(p^l, q^l)$ are recomputed at the end of each optimization iteration using bilayer density evolution, given the new $\lambda_{i,j}$ and $\eta$. We start with a large stepsize for increasing $\mu^h$, but use a backtracking algorithm so that if the resulted code fails to converge after an iteration step, the step size is reduced and the iteration is repeated. Typically, around 10 optimization iterations are sufficient to obtain a reasonably close-to-capacity rate.

To initialize the above iterative optimization, an initial degree distribution $\lambda_{i,j}$, which is consistent with the lower graph degree distribution in terms of (22) and guarantees a fast decoding convergence with a small $\mu^0 > 0$, should be found. Such a degree distribution can be found using a linear programming optimization that minimizes $\eta$, since minimizing $\eta$ or equivalently maximizing $k_2$ corresponds to adding as many extra parity bits as possible which ensures a fast decoding convergence. The initializing linear programming problem can be cast as follows:

$$\min_{\lambda_{i,j},\eta} \quad \eta \tag{24a}$$

$$\text{s.t.} \sum_{i\geq 2, j\geq 0} \lambda_{i,j} = 1 \tag{24b}$$

$$\sum_{j\geq 0} \frac{i}{i+j}\lambda_{i,j} - \eta\lambda_i = 0 \quad i \geq 2. \tag{24c}$$

To complete the design methodology of bilayer LDPC codes, we need to pick appropriate check degrees $d_c$ and $d'_c$. An appropriate check-degree pair $d_c$ and $d'_c$ can be found for a bilayer code by searching over a reasonable range of values for $d_c$ and $d'_c$. For each pair of $d_c$ and $d'_c$, the variable-degree optimization procedure needs to be repeated.

The main complexity of the overall optimization scheme is due to density evolution, which is used to update the coefficients in (23). We use discrete density evolution with a low resolution to find a proper pair of check degrees and also for the first few optimization iterations with small $\mu^h$. A high resolution discrete density evolution is then used to optimize the last few optimization iterations where $\mu^h$ is close to 1. The complexity of the overall optimization scheme is roughly equivalent to 10–15 rounds of performing density evolution for the final code.

The optimal check degree for a conventional LDPC code is often concentrated around a mean value (see Appendix II). Thus, when the gap between $\text{SNR}_+$ and $\text{SNR}_-$ is small, the difference between the optimal $d_c$ and $d'_c$ is likely to be small, and this scheme works well. However, if the gap between $\text{SNR}_+$ and $\text{SNR}_-$ is large, the optimal check degree $d'_c$ is often much smaller than $d_c$, resulting in a larger gap to capacity.

In the extreme case corresponding to $d'_c = 1$, the presented analysis is no longer valid, since the degree-one check nodes completely determine the values of variable nodes connected to them. Thus, a variable node connected to a degree-one check node can be removed, totally reshaping the structure of the graph. Hence, the effect of low-degree extra check nodes on the structure of the graph, in the extreme case, is to *reveal* the value of variable nodes connected to them. However, it would be more efficient for the relay to use a separate code to intelligently help the destination recover the value of a subset of variable nodes, rather than forwarding degree-one extra parity bits. In light of this intuition, in the next section, we consider another code ensemble that is suitable for decoding the source codeword at the destination while part of the codeword is revealed with the relay's help.

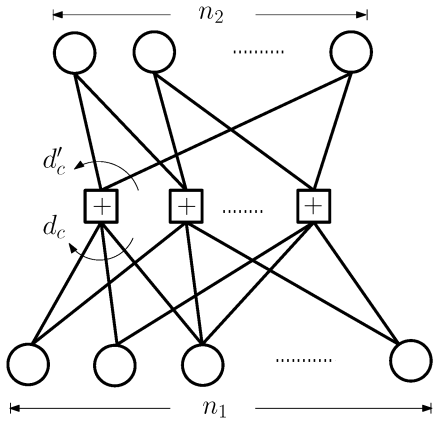


Fig. 5. The bilayer-lengthened LDPC code. The relay decodes the overall code and provides the value of upper variable nodes to the destination, using a separate codebook. The destination decodes the lower subgraph.

## IV. Design of Bilayer-Lengthened LDPC Codes

We now propose our second coding structure for DF based on code lengthening, which is designed to address the problem of the expurgated structure for large SNR differences, as explained above. Lengthening of a linear code refers to the process of increasing the codeword length while keeping the number of parity check equations fixed [46]. Fig. 5 depicts a bilayer-lengthened LDPC code in which the overall graph corresponds to a lengthened version of the lower code. Note that both the lower graph and the overall (bilayer) graph have the same set of parity-check nodes. Designing a bilayer-lengthened code corresponds to finding an overall graph so that the lower graph corresponds to a good LDPC code at rate $R_-$ optimized for $\text{SNR}_-$, while the overall bilayer graph (which can be constructed by adding extra variable nodes to the lower graph) represents a good LDPC code at rate $R_+$ optimized for $\text{SNR}_+$. Being capacity-approaching at two different rates is a core feature of this code.

The relaying scheme using the bilayer-lengthened structure is depicted in Fig. 6. The source encodes its data using the bilayer LDPC code corresponding to the overall graph as shown in Fig. 6(a). Thus, each codeword satisfies all parity-check nodes present in the bilayer graph (in contrast to the earlier bilayer code in which the source encodes its data over the lower subgraph).

The relay first decodes the source codeword over the bilayer graph. It then helps the destination by sending the values of upper $n_2$ variable nodes in the next block using the following scheme. The relay generates a set of $k_2 = (1 - R_-)n_2$ extra parity bits for upper variable nodes, using the parity check matrix of a separate conventional LDPC code $\mathcal{C}_2$ of rate $R_-$ optimized for $\text{SNR}_-$. The relay forwards these (presumably

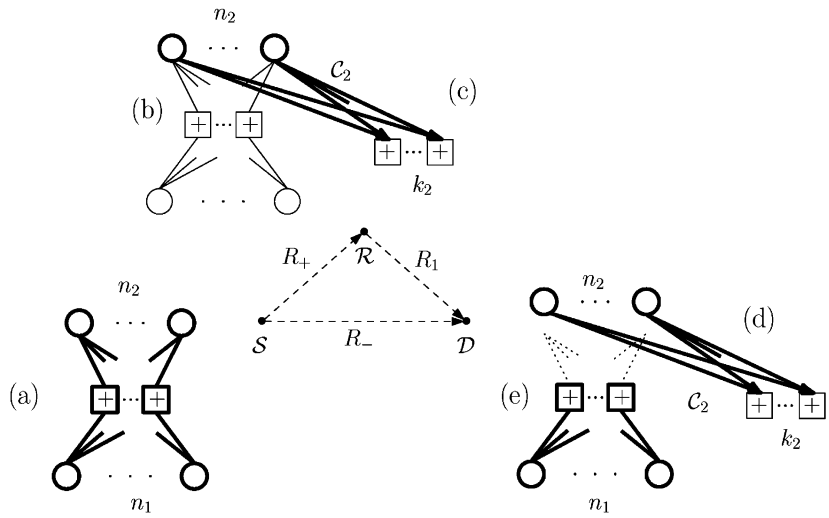


Fig. 6. Parity forwarding scheme using the bilayer-lengthened code structure. (a) The source encodes its data over the bilayer LDPC graph. (b) The relay decodes the entire source codeword. (c) The relay generates $k_2$ parity bits for the upper $n_2$ variable nodes using an LDPC code $\mathcal{C}_2$. The relay sends the set of $k_2$ parity bits to the destination using a separate codebook of rate $R_1$. (d) The destination first decodes the extra $k_2$ parity bits provided by the relay. Using the code $\mathcal{C}_2$, the destination decodes the upper $n_2$ variable nodes. (e) Upon decoding the upper variables, the destination removes the upper part of the graph and decodes the remaining variable nodes using the lower graph. (Note that steps (e) and (d) can be performed jointly, i.e., joint decoding over the entire graph is viable.)

nonzero) extra parity bits to the destination using another codebook of rate $R_1$. Note that it can be shown that the number of extra parity bits $k_2$ is exactly $nR_1 = n(R_+ - R_-)$, where $n$ is the block length of the overall code.[2]

In each block, the destination first decodes the set of $k_2$ extra parity bits for upper $n_2$ variable bits provided by the relay. These $k_2$ parity bits are used to decode the upper $n_2$ variable nodes of the source codeword in the previous block. Since $\mathcal{C}_2$ is a good LDPC code of rate $R_-$ optimized for SNR$_-$, the destination can recover the upper $n_2$ variable nodes with a low error probability at a large block length. Upon recovering the upper part of the bilayer graph, the destination removes the top $n_2$ variables, then updates the value of parity-check nodes in the graph. For example, the new value of a parity-check node corresponding to the constraint $v_1 + v_2 + v_3 + v_4 = 0$ after removing $v_1$ and $v_2$ would be $v_1 + v_2$ corresponding to the constraint $v_3 + v_4 = v_1 + v_2$. Note that for this procedure to work it is necessary that the code $\mathcal{C}_2$ is decoded correctly; thus, the decoding error probability of this procedure is bounded from below by the error probability for code $\mathcal{C}_2$. In practice, joint decoding by message passing over the overall graph [combined graphs of Fig. 6(e) and (d)] can be performed to reduce the error probability.

Finally, the destination decodes the remainder of the codeword over the lower subgraph which represents an optimized LDPC code of rate $R_-$ for SNR$_-$. (In contrast to the earlier bilayer code in which the destination decodes the source's codeword over the overall graph.) Note that in this coding scheme, the source codeword is split into two components such that both parts correspond to good LDPC codewords at SNR$_-$ [the lower subgraph and the code $\mathcal{C}_2$ in Fig. 6(d)]. This is in contrast to the previous bilayer-expurgated LDPC code in which the source LDPC code graph is modified in a way that the overall graph is suitable for SNR$_-$ [see Fig. 3(e)].

Many of features of the bilayer-lengthened LDPC code are the dual of the bilayer-expurgated code: the roles of variable nodes and check nodes are interchanged in the bilayer graph; the source encodes its data over the lower graph in one, and over the overall graph in the other. The bilayer-expurgated code performs well when the gap between SNR$_+$ and SNR$_-$ is small; the bilayer-lengthened code works well for larger gaps.

The advantage of this scheme is that the check degrees are reduced after the removal of the upper graph. Therefore, this code structure is suitable for a relay channel with a large gap between the SNR$_+$ and SNR$_-$. The bilayer-lengthened code structure considered in this paper is inspired by a code construction, called Matrioshka codes, introduced in [47] for the universal Slepian–Wolf source coding problem.

### A. Bilayer-Lengthened LDPC Code Ensemble

Similar to the bilayer-expurgated code, the bilayer-lengthened graph consists of three sets of nodes and two sets of edges (Fig. 5). The nodes are grouped into one set of check nodes (in contrast to the earlier bilayer graph in which there is one set of variable nodes), and two sets of variable nodes (in contrast to the earlier bilayer graph in which there are two sets of check nodes): the lower variable nodes corresponding to the variable nodes in the lower subgraph of Fig. 5, and the upper variable nodes corresponding to the variable nodes in the upper subgraph in Fig. 5. The edges are grouped in two sets: those connecting the check nodes to the lower variable nodes, and those connecting check nodes to the upper variable nodes. We call an edge a lower edge, if it connects a check node to a lower variable node. Similarly, an upper edge denotes an edge belonging to the upper subgraph in Fig. 5.

Assuming regular check degrees, each check node in the bilayer-lengthened graph has $d_c$ edges in the lower subgraph and

[2]This is because $k_2 = (1-R_-)n_2 = (1-R_-)(n-n_1) = (1-R_-)(n - n(1-R_+)/(1-R_-)) = n(R_+ - R_-) = nR_1$, where $n_1$ is the number of lower variable nodes. The third equality is based on the relation $n_1 = n\frac{1-R_+}{1-R_-}$, which follows the fact that the lower graph is of the rate $R_-$ and the overall graph is of rate $R_+$.

$d'_c$ edges in the upper subgraph. Similar to a conventional LDPC code, the degree of a variable node is defined as the number of edges connected to it. An edge is said to have a variable degree $i$ if it is connected to a variable node of degree $i$.

The ensemble of bilayer-lengthened LDPC codes is defined by the lower variable degree distribution, the upper variable degree distribution, and two regular check degrees $d_c$ and $d'_c$. The lower variable degree distribution $\lambda^1_i, i \geq 2$, defines the percentage of lower edges of various degrees in the lower subgraph, i.e., the probability that a lower edge is connected to a degree $i$ variable node is given by $\lambda^1_i$. Similarly, the upper variable-degree distribution $\lambda^2_i, i \geq 2$, gives the probability that an upper edge is of degree $i$. The lower and upper distributions $\lambda^1_i$ and $\lambda^2_i$ satisfy $\sum_{i\geq 2} \lambda^1_i = 1$, and $\sum_{i\geq 2} \lambda^2_i = 1$.

Note that the ensemble of bilayer-lengthened LDPC codes is not equivalent to either the conventional LDPC codes or the bilayer-expurgated LDPC codes discussed earlier, because in both of these earlier code ensembles, the variable degree distributions for all variable nodes are the same. Hence, density evolution tools for conventional LDPC codes and for bilayer-expurgated codes are not valid for the bilayer-lengthened LDPC code and should be modified.

### *B. Bilayer Density Evolution*

The densities of messages over lower and upper edges are in general not the same in a bilayer-lengthened LDPC code. This is because the lower and upper edges have different variable degree distributions. Thus, similar to the case of the bilayer-expurgated LDPC codes, to predict the performance of an infinite-length bilayer-lengthened LDPC code, we need to track the evolutions of two densities in the upper and lower subgraphs of the lengthened graph.

Let $p^l$ and $q^l$ denote the message densities in the lower and upper parts of the graph at the beginning of the $l$th decoding iteration. Let $p'^l$ and $q'^l$ denote the evolved versions of $p^l$ and $q^l$ after check updates. Let $\oplus$ denote the check density-update operation as described in [2], e.g., $f = f_1 \oplus f_2$ is the output message density after an update at a check node of degree 3. Then, the output message density at a check node of degree $d$ with input message density $f$ can be computed as $\oplus^{d-1} f = f \oplus f \oplus \cdots \oplus f$. Hence, $p'^l$ and $q'^l$ can be computed using the check density-update operation as follows:

$$p'^l = (\oplus^{d_c - 1} p^l) \oplus (\oplus^{d'_c} q^l), \quad d_c > 1 \tag{25a}$$

$$q'^l = (\oplus^{d_c} p^l) \oplus (\oplus^{d'_c - 1} q^l), \quad d'_c > 1 \tag{25b}$$

and for $d'_c = 1$

$$p'^l = (\oplus^{d_c - 1} p^l) \oplus (q^l), \quad d_c > 1 \tag{25c}$$

$$q'^l = (\oplus^{d_c} p^l) \tag{25d}$$

where $\oplus^1 f \triangleq f$ for any density $f$.

The computation of variable density updates is straightforward using the convolution operation. Let $p^{l+1}_i$ denote the output message density after a variable update at a variable node of degree $i$ in the lower subgraph, with an input message density $p'^l$. Let $q^{l+1}_i$ denote the output message density after a variable update at a variable node of degree $i$ in the upper subgraph, with an input message density $q'^l$. Using the convolution operation $\otimes$, we have

$$p^{l+1}_i = \otimes^{i-1} p'^l \otimes p_c, \quad i \geq 2 \tag{26}$$

$$q^{l+1}_i = \otimes^{i-1} q'^l \otimes p_c, \quad i \geq 2 \tag{27}$$

where $p_c$ is the channel message density.

The message densities in the lower and upper subgraphs after the variable update (i.e., at the beginning of $(l+1)$th decoding iteration), $p^{l+1}$ and $q^{l+1}$, can be computed from $p^{l+1}_i$ and $q^{l+1}_i$ as follows:

$$p^{l+1} = \sum_{i\geq 2} \lambda^1_i p^{l+1}_i \tag{28}$$

$$q^{l+1} = \sum_{i\geq 2} \lambda^2_i q^{l+1}_i. \tag{29}$$

Let $e(p^{l+1}, q^{l+1})$ denote the message error probability of the message densities $p^{l+1}$ and $q^{l+1}$ at the beginning of the $(l+1)$th decoding iteration. Let $e^1_i(p^l, q^l)$ denote the message error probability corresponding to $p^{l+1}_i$, which is the message density of degree-$i$ lower nodes after one density evolution iteration with input message densities $p^l$ and $q^l$. Similarly, let $e^2_i(p^l, q^l)$ denote the message error probability corresponding to $q^{l+1}_i$, which is the message density of degree-$i$ upper nodes after one density evolution iteration with input message densities $p^l$ and $q^l$. The overall message error probability at the beginning of the $(l+1)$th iteration, $e(p^{l+1}, q^{l+1})$, can be found as a linear combination of $e^1_i(p^l, q^l)$ and $e^2_i(p^l, q^l)$ functions as follows:

$$e(p^{l+1}, q^{l+1}) = \sum_{i\geq 2} \eta \lambda^1_i e^1_i(p^l, q^l) + (1-\eta)\lambda^2_i e^2_i(p^l, q^l). \tag{30}$$

where $\eta = d_c/(d_c + d'_c)$ denotes the percentage of lower edges in the bilayer-lengthened graph. The approximate linear structure of (30) is used to form an iterative linear programming procedure to update the variable-degree distributions $\lambda^1_i$ and $\lambda^2_i$ as discussed in the next subsection.

### *C. Bilayer-Lengthened LDPC Code Optimization*

The design of a bilayer-lengthened LDPC code involves finding a pair of variable degree distributions $\lambda^1_i$ and $\lambda^2_i$ $(i \geq 2)$ and a pair of check degrees $d_c$ and $d'_c$ for the lower and upper subgraphs in the bilayer structure of Fig. 5, such that the overall graph is a capacity-approaching LDPC code for a Gaussian channel at $\text{SNR}_+$, while the lower graph is a capacity-approaching LDPC code at $\text{SNR}_-$.

The design scheme is based on the optimization procedure for conventional LDPC codes described in Appendix II. Similar to the previous design, we fix the check degrees $d_c$ and $d'_c$. (Appropriate check degrees $d_c$ and $d'_c$ can be found by an exhaustive search over a reasonable range of values for $d_c$ and $d'_c$.) We also fix the lower variable-degree distribution $\lambda^1_i$ to be a capacity-approaching distribution for a conventional LDPC code optimized at $\text{SNR}_-$ (which is found independently). The design problem is now reduced to finding an upper variable-degree distribution $\lambda^2_i$ such that the overall lengthened graph represents a

TABLE I
BILAYER-EXPURGATED LDPC CODES

| Code | (A)* | | | | (B) | | (C)* | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\lambda_{i,j}$ | $\lambda_{i,0}$ | $\lambda_{i,1}$ | $\lambda_{i,3}$ | $\lambda_{i,4}$ | $\lambda_{i,0}$ | $\lambda_{i,1}$ | $\lambda_{i,0}$ | $\lambda_{i,1}$ | $\lambda_{i,3}$ | $\lambda_{i,4}$ | $\lambda_{i,6}$ |
| $i=2$ | 0.1398 | 0.0408 | | | 0.2417 | 0.0496 | 0.0998 | 0.0805 | | | |
| $i=3$ | 0.1323 | 0.0885 | | | 0.1702 | 0.0501 | 0.0827 | 0.1331 | | | |
| $i=6$ | 0.083 | | | | 0.1182 | | | 0.0086 | 0.0920 | | |
| $i=7$ | 0.0295 | 0.1332 | | | 0.0056 | 0.1348 | | | 0.1725 | | |
| $i=18$ | | | | | | 0.1573 | | | | | |
| $i=19$ | | | 0.0267 | 0.0458 | | | | | | | |
| $i=20$ | | | 0.2600 | 0.0928 | | | | | | 0.0845 | 0.2463 |
| $\eta,(d_c,d_c')$ | 0.8982,(15,8) | | | | 0.9435,(6,2) | | 0.8253,(15,4) | | | | |
| $R_-,R_+$ | 0.6363,0.7000 | | | | 0.2753,0.3856 | | 0.4618,0.7000 | | | | |
| $\text{Gap}_-$ | 0.1727 dB | | | | 0.4612 dB | | 0.5143 dB | | | | |
| $\text{Gap}_+$ | 0.08474 dB | | | | 0.2162 dB | | 0.08474d dB | | | | |

* The lower-graph of codes (A) and (C) is obtained from [48].

$\text{Gap}_-$ and $\text{Gap}_+$ denote the gaps between the convergence threshold and the theoretical limit with binary inputs for the lower-rate and higher-rate codes, respectively.

capacity-approaching code at $\text{SNR}_+$. (Note that in contrast to the design problem of a bilayer-expurgated code, the lower rate code is fixed here, and the higher rate code is optimized.)

The rate of the overall bilayer-lengthened code is $1-k/(n_1+n_2)$, where $k$ denotes the number of check nodes, $n_1$ is the number of lower variable nodes, and $n_2$ is the number of upper variable nodes. The number of upper variable nodes $n_2$ is given by $d_c'k\sum_{i\geq 2}\lambda_i^2/i$. Thus, fixing the lower graph code and $d_c'$, the rate of the overall graph can be maximized by maximizing $\sum_{i\geq 2}\lambda_i^2/i$. To ensure convergence of the overall code, we make use of the error profile function (30). More specifically, fixing $\eta$, $d_c$, and $d_c'$, the linear programming update for $\lambda_i^2$ can be formulated as follows:

$$\max_{\lambda_i^2} \quad \sum_{i\geq 2}\lambda_i^2/i \tag{31a}$$

$$\text{s.t.} \quad \sum_{i\geq 2}\eta\lambda_i^1 e_i^1(p^l,q^l)+(1-\eta)\lambda_i^2 e_i^2(p^l,q^l) < \mu^h e(p^l,q^l), \quad l=1\cdots L \tag{31b}$$

$$\sum_{i\geq 2}\lambda_i^2=1 \tag{31c}$$

where $h$ denotes the optimization iteration round, and $l$ is the decoding iteration number. The new upper variable-degree distribution is used to update the coefficients $e_i^1(p^l,q^l)$, $e_i^2(p^l,q^l)$, and $e(p^l,q^l)$ for the next optimization round through bilayer density evolution. The coefficient $\mu^h$ is slowly increased toward 1. This enforces an approximate local linearity condition with respect to $\lambda_i^2$ [in the same way as in (23c) and (35)]. As an initialization value for $\lambda^2$, we set $\lambda^2_{\max(d_v)}=1$.

The bilayer-lengthened LDPC code is a suitable code structure, if the gap between $\text{SNR}_+$ and $\text{SNR}_-$ is large. When the gap between $\text{SNR}_+$ and $\text{SNR}_-$ is small, the bilayer-expurgated LDPC code design of Section III have a good performance. In fact, the rate difference can be arbitrarily small for the bilayer-expurgated code. Thus, the bilayer-expurgated LDPC code and the bilayer-lengthened LDPC code are complementary structures that cover the entire range of rates and SNRs.

TABLE II
BILAYER-LENGTHENED LDPC CODES

| Code | (D) | | (E) | | (F) | |
|---|---|---|---|---|---|---|
| $\lambda_{i,j}$ | $\lambda_i^1$ | $\lambda_i^2$ | $\lambda_i^1$ | $\lambda_i^2$ | $\lambda_i^1$ | $\lambda_i^2$ |
| $i=2$ | 0.3227 | 0.1655 | 0.3227 | 0.1580 | 0.2421 | 0.1468 |
| $i=3$ | 0.2107 | 0.2617 | 0.2107 | 0.2045 | 0.2039 | 0.2331 |
| $i=5$ | | 0.1505 | | 0.0461 | | |
| $i=6$ | 0.1247 | | 0.1247 | 0.2171 | 0.1677 | |
| $i=7$ | 0.1194 | | 0.1194 | | 0.0829 | 0.3039 |
| $i=8$ | | | | | | 0.0298 |
| $i=10$ | | 0.2977 | | | | |
| $i=11$ | | 0.1246 | | | | |
| $i=12$ | | | | 0.0058 | | |
| $i=13$ | | | | 0.3685 | | |
| $i=19$ | | | | | | 0.2864 |
| $i=20$ | 0.2225 | | 0.2225 | | 0.3034 | |
| | $d_c=5$ | $d_c'=33$ | $d_c=5$ | $d_c'=1$ | $d_c=8$ | $d_c'=6$ |
| $R_-,R_+$ | 0.2871,0.8932 | | 0.2871,0.3843 | | 0.4877,0.6906 | |
| $\text{Gap}_-$ | 0.2369 dB | | 0.2369 dB | | 0.1641 dB | |
| $\text{Gap}_+$ | 0.1357 dB | | 0.2364 dB | | 0.1208 dB | |

$\text{Gap}_-$ and $\text{Gap}_+$ denote the gaps between the convergence threshold and the theoretical limit for the lower-rate and higher-rate codes, respectively.

## V. CODE CONSTRUCTION AND NUMERICAL RESULTS

Using the described schemes, three bilayer expurgated codes, listed in Table I, and three bilayer lengthened codes, listed in Table II, are designed for binary-input Gaussian channels with various channel parameters. We fix a target rate which corresponds to a target SNR. Then, the lower graph is optimized by using the iterative linear optimization process described in Appendix I to find the highest rate converging LDPC code for the target SNR. Fixing the lower layer, bilayer optimization is performed to find the highest rate overall code for the target SNR. Finally, the gap between the convergence threshold of the code and the theoretical limit corresponding to the optimal achieved rates is computed for the optimized codes. The maximum variable degree, $\max(d_v)$, for all cases, is chosen to be 20 (in each layer). Asymptotic infinite-length convergence thresholds of the codes are computed using the discretized density evolution approach of [39] with 13-bit quantization and a maximum

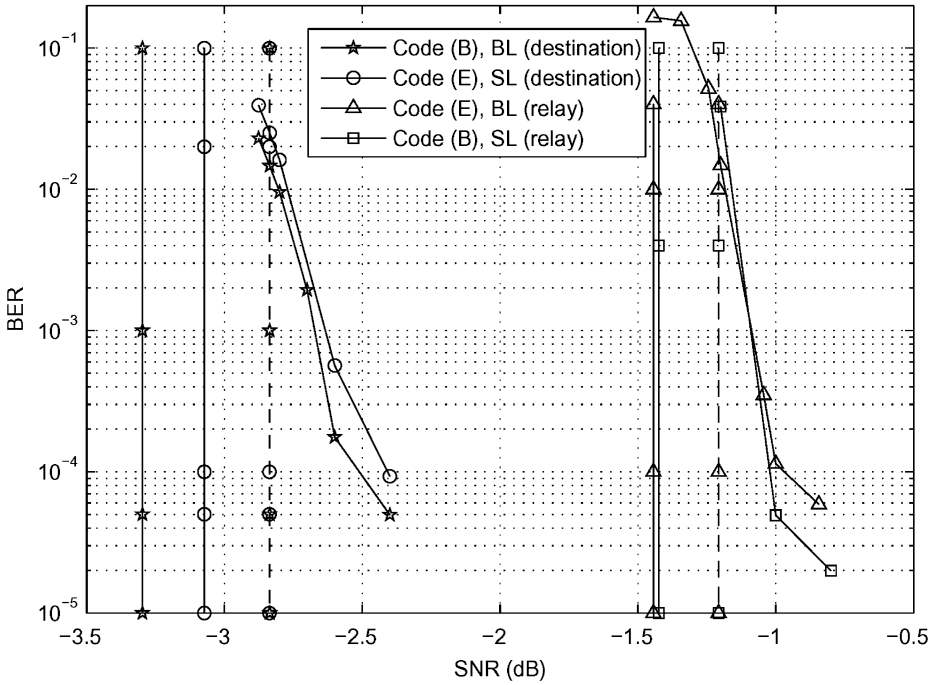


Fig. 7. Comparison of bilayer expurgated code (B) and bilayer lengthened code (E). Solid straight lines represent theoretical limits, and dashed lines represent the convergence threshold. BL stands for bilayer (the overall graph), and SL stands for single layer (the lower graph). Note that Code (E) has a smaller gap to the capacity. Block lengths are $100\,000 \pm 50$.

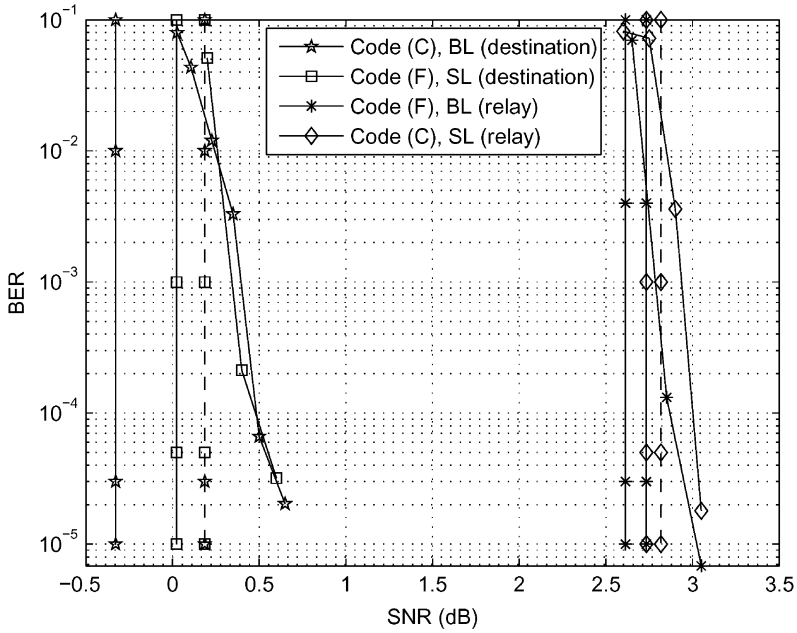


Fig. 8. Comparison of expurgated and lengthened codes (C) and (F). Solid straight lines represent theoretical limits; dashed lines represent the convergence threshold. Code (F) shows a smaller gap to the capacity for a block lengths of $100\,000 \pm 50$.

log-likelihood value 25. We use the sum-product decoding algorithm with parallel update scheduling. The maximum number of decoding iterations is set to 600. The empirical bit-error-rate (BER) performance curves for the source-relay and source-destination codes are shown in Figs. 7–10 for a block length of 100 000. (In these figures, the SNR axis corresponds to $\mathrm{SNR}_+$ for the source to the relay code, and $\mathrm{SNR}_-$ for the source to the destination code.)

Figs. 7 and 8 compare the BER performance of pairs of bilayer expurgated and bilayer lengthened LDPC codes (B) and (E), and (C) and (F), designed for the target rates $(R_-, R_+) = (0.3, 0.4)$ and $(R_-, R_+) = (0.5, 0.7)$, respectively. The convergence thresholds of the lengthened codes (E) and (F) are found to be within 0.24 dB to the theoretical limit. For the expurgated codes (B) and (C), the gap to the theoretical limit is below 0.52 dB. The BER curves confirm the asymptotic convergence

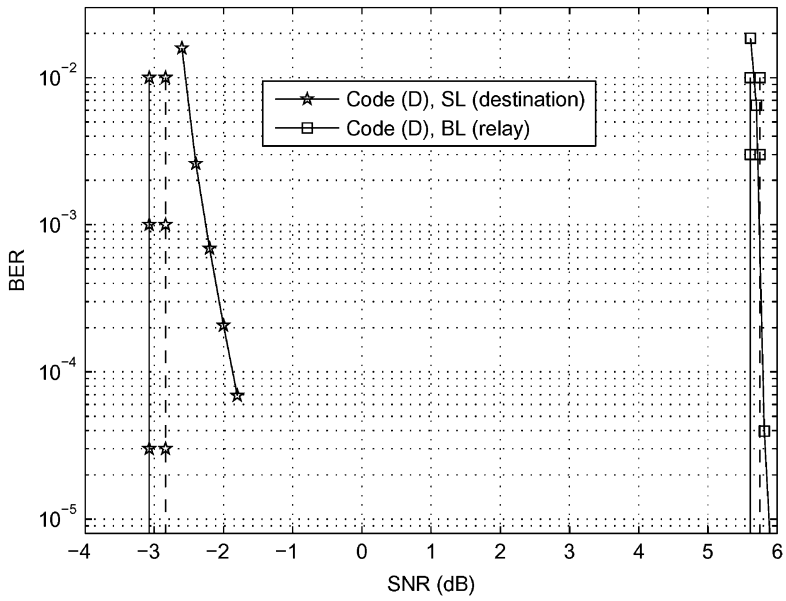


Fig. 9. Bilayer lengthened code for large SNR difference between the relay and the destination. Solid straight lines represent theoretical limits; dashed lines represent the convergence threshold. Block lengths are $100000 \pm 50$.

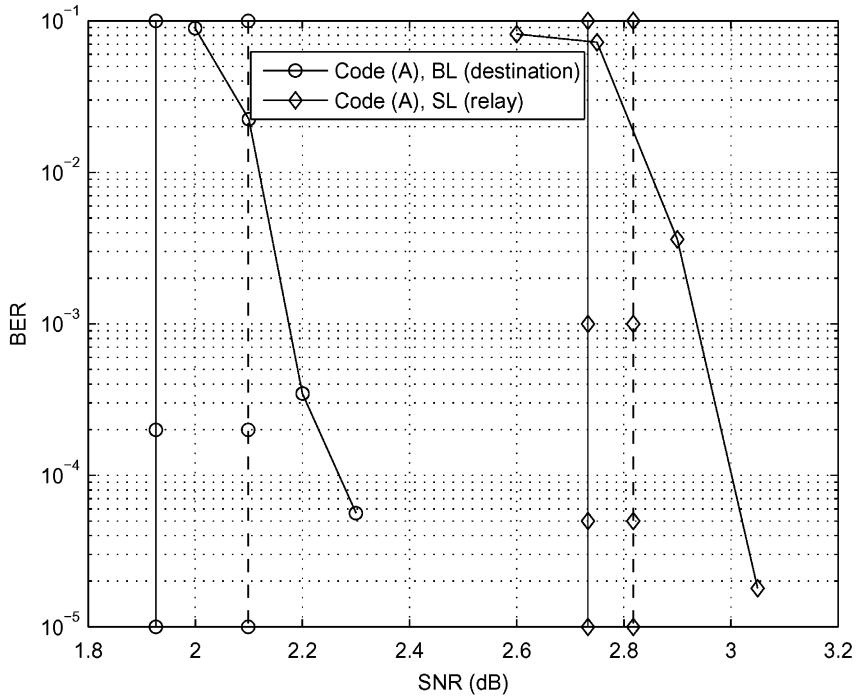


Fig. 10. Bilayer expurgated code for small SNR difference between the relay and the destination. Solid straight lines represent theoretical limits; dashed lines represent the convergence threshold. Block lengths are $100\ 000 \pm 50$.

threshold analysis, indicating a gap of less than 0.5 dB to the convergence threshold at a BER of $10^{-4}$. It is observed that at these rates, the performances of expurgated and lengthened codes are comparable with lengthened codes performing slightly better.

As mentioned earlier, the lengthened code structure is more suitable when the SNR difference at the relay and at the destination is large. Code (D) is designed for target rates $(R_-, R_+) = (0.3, 0.9)$ which corresponds to an SNR difference of close to 9 dB. Table II presents the parameters of this code. Note that the difference between the lower-graph check degree and the upper-graph check degree is quite large (5 versus 33). The gap of the convergence threshold of the bilayer code (overall graph) to the theoretical limit is as small as 0.14 dB. The gap of the single-layer graph (designed as a conventional LDPC code, using the described scheme in Appendix II) to the theoretical limit is less than 0.24 dB. The BER performance of this code is shown in Fig. 9. The SNR gap to the convergence threshold at a BER of $10^{-4}$ is negligible for the bilayer higher rate code. The single-layer code corresponding to the lower graph shows a larger gap (about 1 dB) to the analytical convergence threshold. This can be explained by the fact that the asymptotic analytical evaluation of the convergence threshold is valid for large block lengths; whereas, the block length of the lower graph of Code (D) is

14 000, which is quite small compared to the overall bilayer block length of 100 000.

On the other hand, the expurgated code has a good performance if the SNR gap between the relay and the destination is small. Code (A) is designed for a pair of close target rates $R_- = 0.65$ and $R_+ = 0.7$. The higher rate code corresponding to the lower graph is obtained from [48]. The designed lower rate code achieves close to 98% of the promised theoretical rate corresponding to a SNR gap of less than 0.18 dB to the theoretical limit. Fig. 10 shows the BER curves for the lower rate and higher rate components of this code. At $\mathrm{BER} = 10^{-4}$, the SNR gap to the convergence threshold for the bilayer component (lower-rate code) is less than 0.2 dB, which confirms the asymptotic convergence threshold analysis.

The BER curves indicate the presence of an error floor. In general, purely randomly constructed LDPC codes suffer from a high level of error floor (especially in codes with rates very close to the capacity, due to the large percentage of degree-two variable nodes). One approach to reduce the error floor is using a systematic encoder to avoid associating information bits to the degree-two variable nodes. A common and more effective approach is to remove cycles of length 4, and possibly 6 or more. Another approach is to enhance the performance of the finite-length code by using more sophisticated graph construction algorithms such as the progressive edge growth (PEG) scheme [49], or multi-edge graph construction [50]. The graph constructions in our simulations are purely random[3] with the exception of removing cycles of length two, i.e., parallel edges.

Note that the exhibited error floor is not necessarily a result of the bilayer configuration. This can be observed by comparing the level of error floor in the single-layer codes (SL curves) and bilayer codes (BL curves) in Figs. 7–10. For example, in some cases, the error floor of the bilayer component is slightly lower than that for the (standard LDPC) single-layer codes (e.g., Fig. 8), and in some other cases, the bilayer error floor is higher (e.g., Fig. 7).

## VI. Multilayer LDPC Codes for Relay Networks

Thus far, we have focused on the single-relay channel and shown that bilayer LDPC codes can be designed to approach the best DF rate in this classical setting. In a more general setting, bilayer codes (or multilayer codes) can also be adopted for multiple-relay networks.

Multiple-relay networks can have many different topologies. One way to generalize the DF rate to multiple-relay networks is to impose a linear ordering on the intermediate relays, and let each relay completely decode the source message with the help of relays prior to itself, then participate in transmission of the source message to subsequent relays and to the destination. The capacity of this DF strategy has been studied in [5] and [3]. In [51], the authors cast the multiple-relay network within a parity-forwarding framework, and have been able to enlarge the DF rate of [5] and [3]. This section focuses on two-relay networks and illustrate two fundamental ways that multiple relays can help each other and help the ultimate decoding of information at the destination. The main purpose of this section is to show that practical bilayer codes can be readily applied in these cases.

[3]This is because removing cycles (without significantly affecting the degree distribution) in a bilayer structure could be computationally costly for a large block length. Nevertheless, the raw performance of the codes are presented; further improvement is viable by using error floor reduction techniques.

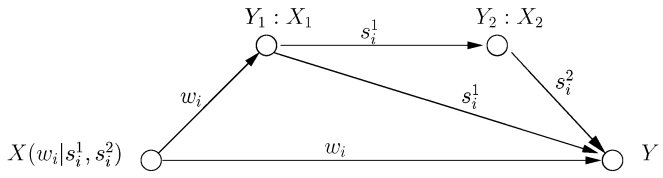


Fig. 11. A two-relay network in which the second relay facilitates the transmission of parity bits from the first relay to the destination.

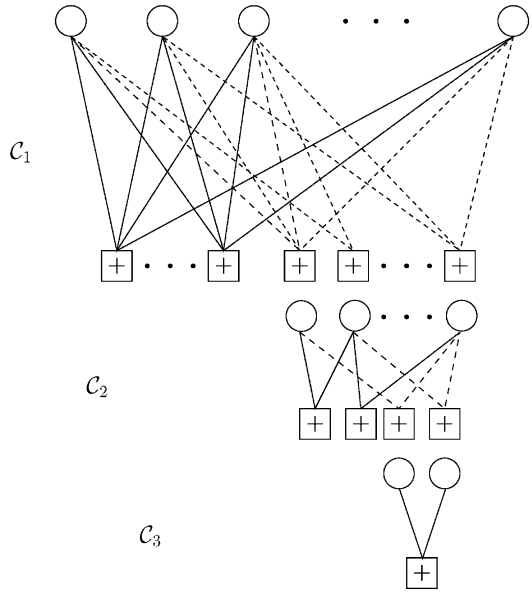


Fig. 12. Cascaded bilayer codes for the two-relay network in Fig. 11.

### A. Cascade Bilayer Codes for Two-Relay Networks

Consider a two-relay network depicted in Fig. 11. In this case, the first relay decodes the message from the source $w_i$, then sends out a parity $s_i^1$, just as in the single-relay case. However, suppose that the channel from the source to the second relay is weak. So, the second relay is not able to decode the source message (even with the help of $s_i^1$), although it is able to decode $s_i^1$ itself. However, for this channel, the second relay may still help the ultimate decoding at the destination by sending out parities of parities, denoted here as $s_i^2$, to help the destination decode $s_i^1$. This "helping-the-helper" strategy can be shown to be capacity-achieving for a doubly degraded network [51], and it enlarges the achievable rates in [5] and [3].

The code construction for this relay network is shown in Fig. 12. It consists of a cascade of two bilayer codes. The source message is coded by a bilayer code $\mathcal{C}_1$. Upon decoding $\mathcal{C}_1$, the first relay computes additional parities for $\mathcal{C}_1$ and re-encodes them using $\mathcal{C}_2$, which is another bilayer code. The second relay decodes $\mathcal{C}_2$, then computes extra parities for $\mathcal{C}_2$ and re-encodes them using $\mathcal{C}_3$. Finally, the destination first decodes $\mathcal{C}_3$ to recover the extra parities needed to decode $\mathcal{C}_2$. Then, it decodes $\mathcal{C}_2$ to recover the parities of $\mathcal{C}_1$. Finally, the destination decodes $\mathcal{C}_1$.

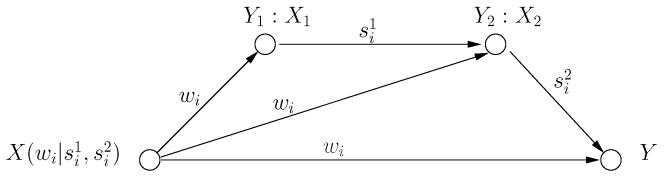


Fig. 13. A two-relay network in which the first relay helps the second relay to decode the source message.

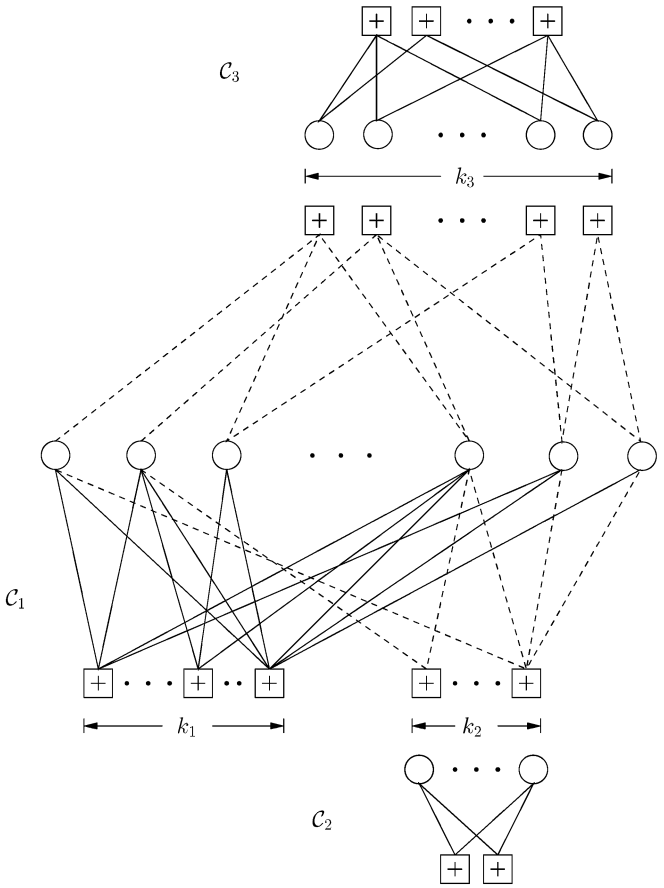


Fig. 14. Doubly bilayer codes for the two-relay network in Fig. 13.

Clearly, the bilayer codes that have been devised for single-relay channels can be directly cascaded to design coding systems capable of approaching the best achievable rate in this network.

### *B. Doubly Bilayer Codes for Two-Relay Networks*

Consider a different two-layer network depicted in Fig. 13 in which the channel between the first relay and the destination is weak. In this case, the optimal strategy is for the first relay to help the second relay, so that the second relay can later help the destination.

The code construction for this relay network is shown in Fig. 14. It is a doubly bilayer code in the following sense. The source encodes its message using $\mathcal{C}_1$. The first relay decodes $w_i$, computes $k_2$ parities bits, and re-encodes parities using $\mathcal{C}_2$ for the second relay. The second relay decodes $w_i$ with the help of $k_2$ parities. Then, it computes separate $k_3$ parity bits to be re-encoded by $\mathcal{C}_3$. The destination decodes $\mathcal{C}_3$, then $\mathcal{C}_1$, the source message. The achievable rate using the above strategy is a special case of the achievable rate in [5].

For this relay network, $\mathcal{C}_2$ and $\mathcal{C}_3$ are conventional LDPC codes. However, $\mathcal{C}_1$ must be specially designed as two bilayer codes extended from the same base code. The code design methodology described in the previous section can again be used for this network. For example, Code (A) and Code (C) can be utilized to construct codebooks for implementing this protocol with a source rate of $R = 0.7$, since the lower-layer component of these two codes are the same (i.e., two separate second layers can be added to the common lower-layer code of rate 0.7). The first relay decodes the source codeword at $R = 0.7$; the second relay, with the help of $k_2$ parity bits from the first relay at a rate 0.7–0.6363, can use Code (A) to decode the source codeword. The second relay then sends out $k_3$ parity bits to the destination at a rate 0.7–0.4618, which enables the destination to decode the source codeword using Code (C).

## VII. Concluding Remarks

Binning is of fundamental importance in multiuser information theory. This paper provides a practical implementation of the binning strategy for the relay channel from a linear coding perspective, in which extra parity bits are generated at the relay to facilitate the overall communication between the source and the destination. A key feature of the code design is the construction of a bilayer LDPC code that is capable of approaching the Gaussian channel capacity at two different SNRs and at two different rates. We show that conventional code design techniques must be significantly modified for the design of these multirate codes in order to achieve capacity-approaching performances.

The code construction in this paper shows that the binning operation for the relay channel is fundamentally easier to implement in practice than the binning techniques for source and channel coding with side information. The former is an error-correcting problem; the latter essentially a quantization problem for which efficient coding methods are not yet known.

The concept of bilayer codes can be extended to relay networks in which cascades of bilayer codes and multilayer LDPC codes may be needed. While in principle these codes can be designed and optimized for a given network topology, as the network size grows, the encoding and decoding protocols become increasingly complex, and the tuning of the code parameters increasingly involved. The code structure illustrated in this paper suggests that practical protocols for the relay network should involve universal and rateless codes. The bilayer code design methodology described in this paper is a first step toward this goal.

## Appendix I
## Coding for an Erasure Relay Channel

Consider a binary erasure relay channel as shown in Fig. 15, where the source-relay channel is a BEC with erasure probability $\epsilon_1$, the source-destination channel is an independent BEC with erasure probability $\epsilon_2 \geq \epsilon_1$, and the relay-destination channel is a digital link with capacity $R_1$. When $R_1 \leq (1-\epsilon_1) - (1-\epsilon_2)$, the capacity of this channel is known

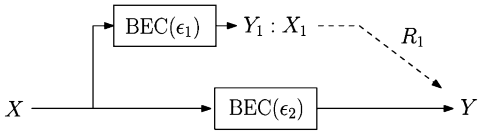


Fig. 15. The binary-erasure relay channel with a digital link from the relay to destination.

to be $C = R_1 + (1 - \epsilon_2)$ [52]; decode-and-forward strategy achieves the capacity on this channel.

Can practical codes achieve the capacity for binary erasure relay channel? In [21], Luby showed that for the binary erasure channel, instead of using conventional LDPC codes, where codewords satisfying a set of parity constraints are transmitted through the channel, it is possible to devise *universal* low-density generator-matrix (LDGM) codes, termed LT code, to achieve the BEC capacity for any arbitrary erasure probability. In the LT code construction, random parities generated from a carefully chosen degree distribution are transmitted through the BEC. Luby proved that using a parity generation function of average degree $O(\ln(m))$, one only needs $m + o(m)$ parities to decode the transmitted bit sequence with high probability. Thus, as $m \to \infty$, one can approach the BEC capacity regardless of the erasure probability.

LT codes can be easily adapted to create capacity-achieving codes for the erasure relay channel. Instead of using LT codes as an online code, consider a block code with $m$ source bits and $n + o(n)$ encoded parity bits sent by $X$, with rate $\frac{m}{n} = R_1 + (1-\epsilon_2)$. Since $R_1 \leq (1-\epsilon_1)-(1-\epsilon_2)$, the relay would receive a sufficient number of parities to decode the source bits with a high probability. The relay then independently re-encodes the source bits using the same degree distribution, and sends the additional parities to the destination via the digital link at rate $R_1$. The total number of independent parities at the destination is then $(1 - \epsilon_2)n$ bits from the source plus $R_1 n$ bits from the relay. Thus, decoding would be successful for arbitrary rates below $C = R_1 + (1 - \epsilon_2)$.

The above argument shows that practical capacity-achieving codes exists for the erasure relay channel. In fact, the above scheme can be further improved in practice by using Raptor codes [22] instead of LT codes to achieve linear-time encoding and decoding performance. However, as mentioned earlier, neither Raptor codes nor LT codes can be used to achieve the Gaussian relay channel capacity, as capacity-achieving rateless codes for general binary symmetric channels have not been found [23].

## Appendix II
## Iterative Linear Programming LDPC Code Design

In this Appendix, we review a linear-programming-based LDPC code design method, which forms the basis of our bilayer code optimization. An LDPC code can be described by a bipartite graph consisting of two sets of nodes: variable nodes and check nodes. An ensemble of irregular LDPC codes is described by two sets of parameters: the variable degree distribution $\lambda_i$, $i \geq 2$, and the check degree distribution $\rho_j$, $j \geq 2$. The variable and check degree distributions define the percentage of edges in the graph that are connected to various variable and check degrees.

The LDPC code design problem is to find a pair of variable and check degree distributions that maximize the rate of the code, while ensuring successful decoding at a given SNR. The rate of the code is determined by the variable and check degree distributions as follows. Let $E$ be the total number of edges in the LDPC graph. Then, the total number of variable nodes is given by $E\sum_{i\geq 2}\lambda_i/i$; the total number of check nodes in the graph is given by $E\sum_{j\geq 2}\rho_j/j$. Thus, the rate of the code is given by

$$R = 1 - \frac{\sum_{j\geq 2}\rho_j/j}{\sum_{i\geq 2}\lambda_i/i}. \tag{32}$$

The code design problem is equivalent to maximizing (32), while ensuring successful decoding at a given SNR.

In LDPC code design, it is common to fix a regular check degree and optimize the code over the variable degree distribution only (see [38, Sec. 3.3] for a justification).[4] The optimum check degree is often found by trying different values. Note that the optimal check degree for codes at different rates can be very different. This fact leads us to two different structures for the design of bilayer codes, which have to operate at two different rates.

With a fixed check degree $d_c$, the rate maximization problem (32) is equivalent to the maximization of $\sum_{i\geq 2}\lambda_i/i$. Assuming a fixed check-degree distribution, the basic idea is to start with some variable-degree distribution $\lambda_i$, then iteratively improve the overall rate using linear programming, while ensuring convergence by identifying a better $\lambda_i$.

For a fixed $\lambda_i$, the performance of an infinite-length LDPC code under the sum-product decoding algorithm can be computed via density evolution [2] (or discretized density evolution [39]). Let us define a decoding iteration to be a set of check updates followed by a set of variable updates for all messages (assuming parallel message passing). Let $p^l$ denote the message pdf at beginning the $l$th decoding iteration, $l = 1 \cdots L$, where $L$ is the maximum number of iterations. Let $q^l$ denote the message pdf after check updates in the $l$th iteration. The pdf $q^l$ is given by $q^l = \oplus^{d_c-1}p^l$, where $\oplus$ denotes the (discrete) message density check-update operation. Let $p_i^l$ denote the message density at the output of a variable node of degree $i$. Using $q^l$, the pdf $p_i^l$ can be computed as $p_i^l = \otimes^{i-1}q^l$, where $\otimes$ denotes convolution. The message pdf at the beginning the $(l+1)$th decoding iteration $p^{l+1}$ is computed as $p^{l+1} = \sum_{i\geq 2}\lambda_i p_i^l$.

Let $e(\cdot)$ be the function returning the message error probability corresponding to a given message density, i.e., $e(f(x))$ returns the message error probability corresponding to its input message pdf $f(x)$.[5] The *error profile function* corresponding to degree $i$ variable nodes is defined by $e_i(p^l) = e(p_i^{l+1})$, i.e., $e_i(p^l)$ equals to the error probability of messages passing along edges of degree $i$ at beginning the $(l+1)$th iteration. Using $e_i(p^l)$'s, $e(p^{l+1})$, the message error probability at beginning the

[4] See also examples of optimized degree distributions available at [48].

[5] Assuming that the all-ones sequence (corresponding to the all-zero codeword) is transmitted and log-likelihood messages are used in the message passing algorithm, $e(f(x))$ equals to $\int_{-\infty}^{0} f(x)dx$. See [2] for more details for the definition of message error probability.

$(l+1)$th iteration, can be computed as a function of $p^l$ as follows:

$$e(p^{l+1}) = \sum_{i\geq 2} \lambda_i e_i(p^l).$$

The message-passing decoding algorithm converges if the message error probability of the code decreases with each decoding iteration. This can be formulated by a set of convergence inequalities as follows:

$$e(p^{l+1}) = \sum_{i\geq 2} \lambda_i e_i(p^l) < e(p^l), \quad l = 1 \cdots L. \tag{33}$$

Using the above convergence criterion a (nonlinear) optimization can be formulated to find the optimal set of $\lambda_i$'s as follows:

$$\max_{\lambda_i, i\geq 2} \quad \sum_{i\geq 2} \lambda_i / i \tag{34a}$$

$$\text{s.t.} \quad \sum_{i\geq 2} \lambda_i e_i(p^l) < e(p^l), \quad l = 1 \cdots L \tag{34b}$$

$$\sum_{i\geq 2} \lambda_i = 1. \tag{34c}$$

The above optimization problem is nonlinear in $\lambda_i, i \geq 2$, since $e_i(p^l)$'s depend nonlinearly on $\lambda_i$. Nevertheless, (33) can still be used to formulate an iterative linear programming to update $\lambda_i$. The idea is to fix $e_i(p^l)$'s and update $\lambda_i$ slowly by enforcing a more stringent convergence condition

$$\sum_{i\geq 2} \lambda_i e_i(p^l) < \mu e(p^l), \quad l = 1 \cdots L \tag{35}$$

where $\mu$ is a *convergence factor* that increases slowly from 0 to 1 in the iterative design process. The updated degree distribution is expected to be converging (i.e., the corresponding message passing algorithm converges) because a small change in $\mu$ corresponds to only a small change in the convergence behavior of the code, and thus a small change in error profile $e_i(p^l)$.

Using (35) and (34), an iterative optimization scheme for updating the variable degree distribution can be formulated as follows. A sequence of linear programming problems

$$\max_{\lambda_i, i\geq 2} \quad \sum_{i\geq 2} \lambda_i / i \tag{36a}$$

$$\text{s.t.} \quad \sum_{i\geq 2} \lambda_i e_i(p^l) < \mu^h e(p^l), \quad l = 1 \cdots L \tag{36b}$$

$$\sum_{i\geq 2} \lambda_i = 1 \tag{36c}$$

are solved successively, where $h$ denotes the optimization iteration number, $l$ is the decoding iteration number, and $\mu^h$ is the convergence factor which is increased in each optimization iteration towards 1. The optimization procedure begins with all variable degrees set to $d_v^{\max}$ where $d_v^{\max}$ is the maximum allowed variable degree, i.e., $\lambda_{\max(d_v)} = 1$. This initial $\lambda_i$ is used to compute initial $e_i(p^l)$ coefficients in (36b). This ensures a small initial $\mu^0$ (as long as an appropriate check degree is selected). The resulting linear programming program is then solved to obtain an updated $\lambda_i$. For this $\lambda_i$, $e_i(p^l)$ is recomputed, and the linear programming problem is solved again with a slightly increased $\mu^h$. The slight increase in $\mu^h$ ensures that the change in $e_i(p^l)$ is small as compared to the previous iteration. The new variable degree distribution $\lambda_i$ obtained from (36) is then used to update $e_i(p^l)$ coefficients. The optimization is repeated with $\mu^{h+1}$, until $\mu^h$ eventually reaches 1.

This procedure is reminiscent of the EXIT-chart approach, because the value of $\mu$ defines the shape of the convergence behavior. The difference is that this scheme is entirely based on density evolution and no approximation of message densities is used. To speed up the $\mu$-update process, we also use a backtracking algorithm: at the end of the $h$th iteration, a greedy increase in $\mu^h$ is performed. If the resulting degree distribution does not correspond to a converging LDPC code (i.e., (36b) cannot be satisfied with $\mu^{h+1}$), $\mu^{h+1}$ is reduced and the optimization is repeated.

## ACKNOWLEDGMENT

The authors would like to thank Dr. M. Ardakani for insightful discussions on LDPC code optimization and for providing his softwares.

## REFERENCES

[1] T. M. Cover and A. A. E. Gamal, "Capacity theorems for the relay channel," *IEEE Trans. Inf. Theory*, vol. 25, no. 5, pp. 572–584, Sep. 1979.

[2] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 2, pp. 599–618, Feb. 2001.

[3] G. Kramer, M. Gastpar, and P. Gupta, "Cooperative strategies and capacity theorems for relay networks," *IEEE Trans. Inf. Theory*, vol. 51, no. 9, pp. 3037–3063, Sep. 2005.

[4] A. Høst-Madsen and J. Zhang, "Capacity bounds and power allocation for wireless relay channel," *IEEE Trans. Inf. Theory*, vol. 51, pp. 2020–2040, Jun. 2005.

[5] L.-L. Xie and P. R. Kumar, "An achievable rate for the multiple level relay channel," *IEEE Trans. Inf. Theory*, vol. 51, no. 4, pp. 1348–1358, Apr. 2005.

[6] M. Janani, A. Hedayat, T. Hunter, and A. Nosratinia, "Coded cooperation in wireless communications: Space-time transmission and iterative decoding," *IEEE Trans. Signal Process.*, vol. 52, no. 2, pp. 362–371, Feb. 2004.

[7] R. U. Nabar, H. Bölcskei, and F. W. Kneübuhler, "Fading relay channels: Performance limits and space-time signal design," *IEEE J. Sel. Areas Commun.*, vol. 22, pp. 1099–1109, Aug. 2004.

[8] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity—Part I: System description," *IEEE J. Sel. Areas Commun.*, vol. 51, pp. 1927–1938, Nov. 2003.

[9] J. N. Laneman and G. W. Wornell, "Distributed space-time-coded protocols for exploiting cooperative diversity in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 49, pp. 2415–2425, Oct. 2003.

[10] G. Kramer, "Distributed and layered codes for relaying," in *Conf. Rec. 39th Asilomar Conf. Signals, Syst., Comp.*, Oct. 2005, pp. 1752–1756.

[11] G. Kramer, "Communication strategies and coding for relaying," in *Wireless Communications: IMA Volumes in Mathematics and its Applications*, P. Agrawal, D. M. Andrews, P. J. Fleming, G. Yin, and L. Zhang, Eds. New York: Springer-Verlag, 2007, vol. 143, pp. 163–175.

[12] M. A. Khojastepour, N. Ahmed, and B. Aazhang, "Code design for the relay channel and factor graph decoding," in *Conf. Rec. 38th Asilomar Conf. Signals, Syst., Comp.*, 2004, vol. 2, pp. 2000–2004.

[13] C. Li, M. A. Khojastepour, G. Yue, X. Wang, and M. Madihian, "Performance analysis and code design for cooperative relay channels," in *Proc. 40th Ann. Conf. Inf. Sci., Syst. (CISS)*, 2006.

[14] T. E. Hunter and A. Nosratinia, "Cooperative diversity through coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2002, p. 220.

[15] B. Zhao and M. C. Valenti, "Distributed turbo coded diversity for relay channel," *Electronics Letters*, vol. 39, pp. 786–787, May 2003.

[16] Z. Zheng and T. M. Duman, "Capacity-approaching turbo coding and iterative decoding for relay channels," *IEEE Trans. Commun.*, vol. 53, no. 11, pp. 1895–1905, Nov. 2005.

[17] A. Chakrabarti, A. de Baynast, A. Sabharwal, and B. Aazhang, "Low density parity check codes for the relay channel," *IEEE J. Select. Areas Commun.*, vol. 25, pp. 280–291, Feb. 2007.
[18] J. Ezri and M. Gastpar, "On the performance of independently designed LDPC codes for the relay channel," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2006, pp. 977–981.
[19] J. Hu and T. M. Duman, "Low density parity check codes over half-duplex relay channels," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2006, pp. 972–976.
[20] J. Hu and T. M. Duman, "Low density parity check codes over wireless relay channels," *IEEE Trans. Wireless Commun.*, 2007, to appear.
[21] M. G. Luby, "LT codes," in *Proc. 43rd Ann. IEEE Symp. Found. Comp. Sci. (FOCS)*, 2002, pp. 271–282.
[22] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 5, pp. 2551–2567, Jun. 2006.
[23] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2033–2051, Jun. 2006.
[24] M. R. Yazdani and A. H. Banihashemi, "On construction of rate-compatible low-density parity-check codes," *IEEE Commun. Lett.*, vol. 8, no. 3, pp. 159–161, Mar. 2004.
[25] J. Kim, W. Hur, A. Ramamoorthy, and S. Mclaughlin, "Design of rate-compatible irregular LDPC codes for incremental redundancy hybrid ARQ systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2006, pp. 1139–1143.
[26] J. Kim, A. Ramamoorthy, and S. Mclaughlin, "Design of efficiently-encodable rate-compatible irregular LDPC codes," in *Proc. IEEE Int. Commun. Conf. (ICC)*, Jun. 2006, vol. 3, pp. 1131–1136.
[27] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2824–2836, Nov. 2004.
[28] E. Soljanin, N. Varnica, and P. Whiting, "Incremental redundancy hybrid ARQ with LDPC and raptor codes," in *Proc. IEEE Int. Symp. Information Theory*, Adelaide, Australia, Sep. 2005, pp. 995–999.
[29] B. Zhao and M. C. Valenti, "Practical relay networks: A generalization of hybrid-ARQ," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 7–18, Jan. 2005.
[30] S. Sesia, G. Caire, and G. Vivier, "Incremental redundancy hybrid ARQ schemes based on low-density parity-check codes," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1311–1321, Aug. 2004.
[31] A. Wyner, "Recent results in the Shannon theory," *IEEE Trans. Inf. Theory*, vol. 20, no. 1, pp. 2–10, Jan. 1974.
[32] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): Design and construction," *IEEE Trans. Inf. Theory*, vol. 49, no. 3, pp. 626–643, Mar. 2003.
[33] R. Zamir, S. Shamai, and U. Erez, "Nested linear/lattice codes for structured multiterminal binning," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1250–1276, Jun. 2002.
[34] A. Shokrollahi, "Capacity-achieving sequences," in *Codes, Systems, and Graphical Models: IMA Volumes in Mathematics and its Applications*, B. Marcus and J. Rosenthal, Eds. New York: Springer-Verlag, 2000, vol. 123, pp. 153–166.
[35] A. Shokrollahi, "New sequences of linear time erasure codes approaching the channel capacity," in *Proc. 13th Int. Symp. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, 1999, pp. 65–76.
[36] P. Oswald and A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 2, pp. 2017–3028, Dec. 2002.
[37] S. Verdú, *Multiuser Detection Cambridge*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[38] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Analysis of low density codes and improved designs using irregular graphs," in *Proc. 30th Ann. ACM Symp. Theory of Computing*, 1998, pp. 249–258.
[39] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.
[40] B. Smith, M. Ardakani, W. Yu, and F. Kschischang, "Design of low-density parity-check codes with optimized complexity-rate tradeoff," *IEEE Trans. Inf. Theory*, submitted for publication.
[41] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.
[42] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: Model and erasure channel properties," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2657–2673, Nov. 2004.
[43] S. ten Brink and G. Kramer, "Design of repeat-accumulate codes for iterative detection and decoding," *IEEE Trans. Signal Process.*, vol. 51, no. 11, pp. 2764–2772, Nov. 2003.
[44] E. Gamal and A. Hammons, "Analysing the turbo decoder using the Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, pp. 671–686, Feb. 2001.
[45] P. Razaghi and W. Yu, "Bilayer LDPC codes for the relay channel," in *Proc. IEEE Int. Commun. Conf. (ICC)*, Jun. 2006, pp. 1574–1579.
[46] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[47] A. Eckford and W. Yu, "Rateless Slepian-Wolf codes," in *Proc. Asilomar Conf. Signals, Syst. Comp.*, Oct. 2005, pp. 1757–1761.
[48] *A Fast And Accurate Degree Distribution Optimizer For LDPC Code Ensembles*, [Online]. Available: http://lthcwww.epfl.ch/research/ldpcopt/
[49] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, 2005.
[50] T. Richardson and R. Urbanke, Multi-Edge Type LDPC Codes [Online]. Available: http://lthcwww.epfl.ch/papers/multiedge.ps
[51] P. Razaghi and W. Yu, "Parity-forwarding for multiple-relay networks," in *Proc. IEEE Int. Symp. Inf. Theory*, Seattle, WA, Jul. 2006, pp. 1678–1682.
[52] Z. Zhang, "Partial converse for a relay channel," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 1106–1110, Sep. 1988.