Spatially Coupled Sparse Codes on Graphs: Theory and Practice

Daniel J. Costello, Jr., Lara Dolecek, Thomas E. Fuja, Jörg Kliewer, David G. M. Mitchell, and Roxana Smarandache

Daniel J. Costello, Jr., Thomas E. Fuja, David G. M. Mitchell, and Roxana Smarandache are with the University of Notre Dame.

Lara Dolecek is with the University of California, Los Angeles.

Jörg Kliewer is with the New Jersey Institute of Technology.

This work was supported in part by the National Science Foundation under Grant Numbers CCF-1161754, CCF-1161774, CCF-1161798, and CCF-1252788.

ABSTRACT

Since the discovery of turbo codes 20 years ago and the subsequent rediscovery of low-density parity check codes a few years later, the field of channel coding has experienced a number of major advances. Until that time, code designers were usually happy with performance that came within a few decibels of the Shannon Limit, primarily due to implementation complexity constraints, whereas the new coding techniques now allow performance within a small fraction of a decibel of capacity with modest encoding and decoding complexity. Due to these significant improvements, coding standards in applications as varied as wireless mobile transmission, satellite TV, and deep space communication are being updated to incorporate the new techniques. In this article, we review a particularly exciting new class of low-density parity check codes called spatially coupled codes, which promise excellent performance over a broad range of channel conditions and decoded error rate requirements.

INTRODUCTION

Low-density parity check (*LDPC*) codes, combined with iterative *belief propagation* (BP) decoding, have emerged in recent years as the most promising method of achieving the goal set by Shannon [1] in his landmark 1948 paper: to communicate reliably over a noisy transmission channel at a rate approaching channel capacity. Indeed, many applications have recently adopted LDPC codes as industry standards, such as wireless LANs (IEEE 802.11n), WiMAX (IEEE 802.16e), digital video broadcasting (DVB-S2), 10GBase-T Ethernet (IEEE 802.3an), and the International Telecommunication Union — Telecommunication Standards Sector (ITU-T) standard for networking over power lines, phone lines, and coaxial cable (G.hn/G.9960). The key feature that sets LDPC codes apart from other capacity approaching codes is that with suboptimal iterative BP decoding, complexity grows only linearly with code block length, resulting in practically realizable decoder implementations for powerful (long block length) codes. (The decoding complexity of optimum *maximum likelihood* [ML] decoding, on the other hand, grows exponentially with block length, making it impractical for large block lengths.) *LDPC block code* (*LDPC-BC*) designs can be classified in two types: regular and irregular. *Regular* codes, as originally proposed by Gallager [2] in 1962, are *asymptotically good* in the sense that their *minimum distance* grows linearly with block length. This guarantees, with ML decoding, that the codes do not suffer from the *error floor* phenomenon, a flattening of the *bit error rate* (BER) curve that results in poor performance at high *signal-to-noise ratios* (SNRs), and similar behavior is observed with iterative BP decoding as well. However, the iterative decoding behavior of regular codes in the so-called *waterfall*, or moderate BER, region of the performance curve falls short of capacity, making them unsuitable for severely power-constrained applications, such as uplink cellular data transmission or digital satellite broadcasting systems, that must achieve the best possible performance at moderate BERs. On the other hand, *irregular* codes, pioneered by Luby *et al.* [3] in 2001, exhibit capacity approaching performance in the waterfall but are normally subject to a visible error floor, making them undesirable in applications, such as data storage and optical communication, that require very low decoded BERs. Typical performance characteristics of regular and irregular LDPC-BCs on a binary-input *additive white Gaussian noise channel* (AWGNC) are illustrated in Fig. 1, where the channel SNR is expressed in terms of E_b/N_0, the *information bit SNR*.

In this article, we highlight a particularly exciting new class of LDPC codes, called *spatially coupled LDPC* (SC-LDPC) codes, which promise robustly excellent performance over a broad range of channel conditions, including both the waterfall and error floor regions of the BER curve. We also show how SC-LDPC codes can be viewed as a type of *LDPC convolutional code* (LDPC-CC), since spatial coupling is equivalent to introducing memory into the encoding process. In channel coding parlance, the key feature of SC-LDPC codes that distinguishes them from standard LDPC codes is their ability to combine the best features of regular and irregular codes in a single design:

- Capacity approaching iterative decoding *thresholds*,[1] characteristic of optimized irregular codes, thus promising excellent performance in the waterfall

[1] Roughly speaking, the threshold associated with a particular code/decoder is the lowest SNR at which the decoder can operate reliably.

0163-6804/14/$25.00 © 2014 IEEE

- Linear growth of minimum distance with block length, characteristic of regular codes, thus promising the elimination of an error floor

As discussed in more detail in the following section, this is achieved by introducing a slight *structured irregularity* into the Tanner graph representation of a regular LDPC code. An added feature of the SC-LDPC code design is that the resulting graph retains the essential implementation advantages associated with the structure of regular codes compared to typical irregular designs. The research establishing the performance characteristics of SC-LDPC codes relies on ensemble average asymptotic methods, that is, the capacity approaching thresholds and asymptotically good minimum distance behavior are shown to hold for typical members of SC-LDPC code ensembles[2] as the block length tends to infinity. (Following the lead of Shannon, coding theorists often find it easier and more insightful to analyze the average asymptotic behavior of code ensembles than to determine the exact performance of specific codes.) These research results are summarized in the following section.

The section after that discusses issues related to realizing the exceptional promise of SC-LDPC codes with specific code and decoder designs suitable for low-complexity implementation at block lengths typically employed in practice:

- The use of high-throughput, parallel, pipeline decoding
- The use of sliding-window decoding strategies for reduced latency and computational complexity

We then present a short summary of several open research problems. The final section includes some concluding remarks along with a brief discussion of the promising use of the spatial coupling concept beyond the realm of channel coding.

SC-LDPC Codes: Basic Structure and Asymptotic Properties

A (J, K)-*regular* LDPC-BC of rate $R = k/n$ and block length n is defined as the null space of an $(n - k) \times n$ binary parity-check matrix $\mathbf{H}$, where each row of $\mathbf{H}$ contains exactly K ones, each column of $\mathbf{H}$ contains exactly J ones, and both J and K are small compared to the number of rows in $\mathbf{H}$. An LDPC code is called *irregular* if the row and column weights are not constant. It is often useful to represent the parity-check matrix $\mathbf{H}$ using a bipartite graph called the *Tanner graph*. In the Tanner graph representation, each column of $\mathbf{H}$ corresponds to a *code bit* or *variable node*, and each row corresponds to a *parity-check* or *constraint node*. If position (i, j) of $\mathbf{H}$ is equal to one, constraint node i is connected by an *edge* to variable node j in the Tanner graph; otherwise, there is no edge connecting these nodes. Figure 2 depicts the parity-check matrix and associated Tanner graph of a (3, 6)-regular LDPC-BC. In this example, we see that all variable nodes have degree 3, since they are connected to exactly 3 constraint nodes, and similarly all constraint nodes have degree 6. In the case of irregular codes, the notion of *degree distribution* is used to characterize the variations of constraint and variable node degrees [3].

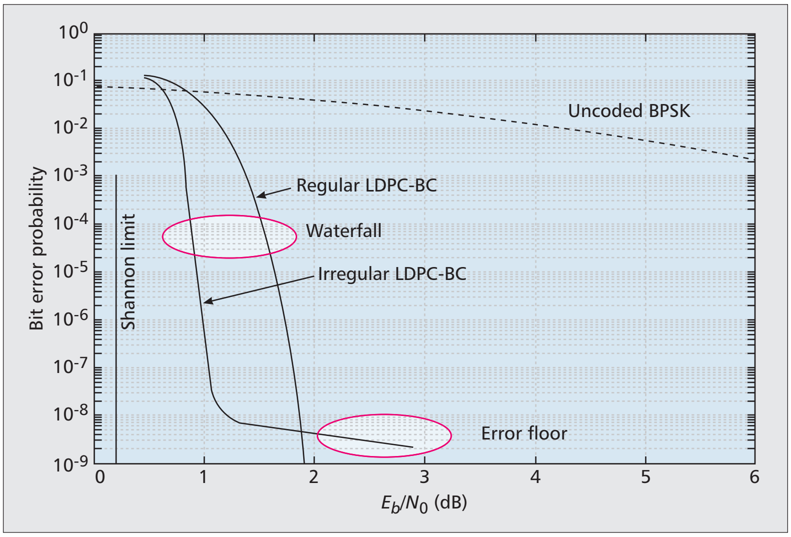

Figure 1. A sketch of typical LDPC-BC decoded BER performance over the AWGNC. Also shown for comparison are the channel capacity, or Shannon limit, and the performance of uncoded binary phase shift keying (BPSK) transmission.

Using the Tanner graph, iterative BP decoding can be viewed as passing messages back and forth between variable and constraint nodes [4]. On an AWGNC, for example, the messages are typically *log-likelihood ratios* (LLRs) associated with the (in general soft-valued) received symbols, which serve as indicators of the probability that a particular code bit is a 1 or a 0. These LLRs are then passed across the graph and adjusted iteratively to reflect the parity constraints until some stopping condition is satisfied, indicating that the received symbols can be decoded reliably.

Certain properties of the Tanner graph can serve as useful indicators of the performance characteristics of iterative decoding. In Fig. 2b, the darkened edges indicate a *cycle* of length 4. In general, codes with short cycles do not perform well in the waterfall due to the buildup of correlation in the iterative process. Hence, it is desirable to choose codes with large *girth*, the length of the shortest cycle, for good waterfall performance. In terms of the error floor performance, minimum distance is the best indicator for ML decoding, and asymptotically good codes are typically not subject to an error floor. For iterative decoding, however, certain substructures of the Tanner graph, such as trapping sets and absorbing sets (an important subclass of trapping sets[3]), can cause iterative decoders to fail apart from minimum distance considerations, resulting in the emergence of an error floor. Hence, it is desirable to select graphs without problematic trapping or absorbing sets for good error floor performance.

SC-LDPC codes can be viewed as a type of LDPC-CC, first introduced in the open literature by Jimenez-Felstrom and Zigangirov in 1999 [5]. A rate $R = b/c$ LDPC-CC can be rep-

[2] *A code ensemble is the collection of all codes sharing some common set of characteristics.*

[3] *An absorbing set is a subset of variable nodes and constraint nodes (a subgraph) that can cause an iterative decoder to get stuck (fail to decode) if the variable nodes in the subset receive unreliable LLRs from the channel. In other words, the decoder may be unable to correct errors associated with the variable nodes in an absorbing set.*

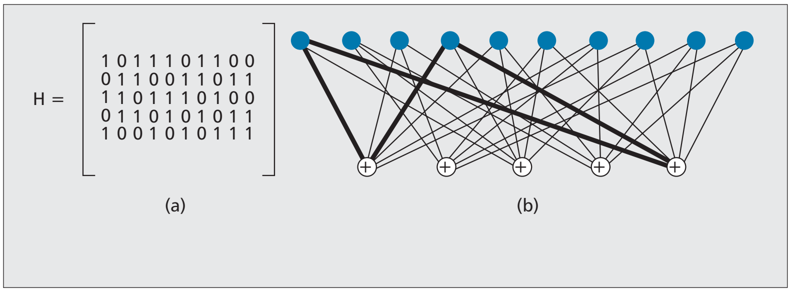

Figure 2. a) Parity-check matrix of a (3, 6)-regular LDPC-BC with block length $n = 10$; b) the associated (3, 6)-regular Tanner graph. The green circles represent code bits, or variable nodes; the open circles ⊕ represent parity checks, or constraint nodes, and the darkened edges represent a cycle of length 4.

resented by a bi-infinite parity check matrix, represented by Eq. 1, composed of a diagonal band of $(c - b) \times c$ submatrices $\mathbf{H}_i(t)$, $0 \le i \le m_s$, $t = 0, 1, 2, \ldots$, where the rows and columns of $\mathbf{H}_{cc}$ are sparse; in other words, they contain a small number of non-zero entries. If $\mathbf{H}_{cc}$ contains only zeros and ones, the code is binary; otherwise, it is non-binary. The *syndrome former memory* is denoted by m_s, where $m_s + 1$ is the width of each row in submatrices, and $\nu_s = (m_s + 1)c$, the width of each row in symbols, is called the *decoding constraint length*. If $\mathbf{H}_{cc}$ contains a fixed number J of ones in each column and a fixed number K of ones in each row, it represents a (J, K)-regular LDPC-CC; otherwise, the code is irregular. In general, $\mathbf{H}_{cc}$ describes a *time-varying* LDPC-CC, and if the rows of $\mathbf{H}_{cc}$ vary periodically, the code is *periodically time-varying*. If the rows of $\mathbf{H}_{cc}$ do not vary with time, the code is *time-invariant*.

Using a technique termed *unwrapping* in [5], it is possible to take any good LDPC-BC and *unwrap* it to form an LDPC-CC with improved BER performance. The unwrapping procedure applies cut-and-paste and diagonal matrix extension operations to the parity check matrix $\mathbf{H}$ of an LDPC-BC to produce a bi-infinite parity-check matrix $\mathbf{H}_{cc}$ of an LDPC-CC, as illustrated in Fig. 3a, where $\mathbf{H}$ represents a (3, 6)-regular block code with block length $n = 10$, and $\mathbf{H}_{cc}$ represents a (3, 6)-regular convolutional code with constraint length $\nu_s = 10$. The bi-infinite (convolutional) Tanner graph representation of $\mathbf{H}_{cc}$ is shown in Fig. 3b, and we see that the unwrapping procedure preserves the graph structure of the underlying LDPC-BC, that is, all node degrees remain the same, and the local connectivity of nodes is unchanged.

Extensive computer simulation results [6] have verified that for practical code lengths, LDPC-CCs obtained by unwrapping an LDPC-BC achieve a substantial convolutional gain compared to the underlying LDPC-BC, where both codes have the same computational complexity with iterative decoding, and the block length of the LDPC-BC equals the constraint length of the LDPC-CC. An example illustrating this convolutional gain is shown in Fig. 4.

Even though the Tanner graph representation of an LDPC-CC extends infinitely both forward and backward in time, in practice there is always some finite starting and ending time; that is, the Tanner graph is terminated at both the beginning and the end (Fig. 3c). A remarkable feature of this graph termination, first noted numerically in the paper by Lentmaier *et al.* [7] for both the *binary erasure channel* (BEC) and the AWGNC, and then shown analytically (for the BEC) by Kudekar *et al.,* [8], is the so-called *threshold saturation* effect. Consider for purposes of illustration the (3, 6)-regular LDPC-BC ensemble with AWGNC iterative BP decoding threshold $E_b/N_0 = 1.11$ dB, which is also the threshold of the associated (unterminated) LDPC-CC ensemble. As the graph termination length L becomes large, the threshold of the (terminated) LDPC-CC ensemble improves all the way to 0.46 dB, the threshold of the (3, 6)-regular LDPC-BC ensemble with ML decoding.[4] In other words, terminated LDPC-CCs with BP decoding are capable of achieving the same performance as comparable LDPC-BCs with (much more complex, and impractical) ML decoding! This “step-up” of the BP threshold to the ML threshold is referred to as threshold saturation. Note that, after termination, the LDPC-CC code ensemble can be viewed as an LDPC-BC ensemble with block length $n = (m_s + 1)cL = \nu_s L$. However, compared to typical LDPC-BC designs that have no restrictions on the location of the ones in the parity-check matrix and hence allow connections across the entire graph, the LDPC-CC code ensemble has a highly localized graph structure, since the non-zero portion of the parity-check matrix is restricted to a diagonal band of width ν_s. We will see later that this structure, in addition to yielding excellent iterative decoding thresholds, also gives rise to an efficient decoder implementation.

Threshold saturation is a result of the termination, which introduces a slight structured irregularity in the graph. Termination has the effect of introducing lower constraint node degrees (i.e., a structured irregularity) at each end of the graph (Fig. 3c). In the context of iterative BP decoding, the smaller degree constraint nodes pass more reliable messages to neighboring variable nodes, and this effect propagates

$$\mathbf{H}_{cc} = \begin{bmatrix} \ddots & \ddots & & & \ddots & & & \\ \mathbf{H}_{m_s}(t-1) & \mathbf{H}_{m_s-1}(t-1) & & \cdots & \mathbf{H}_0(t-1) & & & \\ & \mathbf{H}_{m_s}(t) & \mathbf{H}_{m_s-1}(t) & & \cdots & \mathbf{H}_0(t) & & \\ & & \mathbf{H}_{m_s}(t+1) & \mathbf{H}_{m_s-1}(t+1) & & \cdots & \mathbf{H}_0(t+1) & \\ & & \ddots & \ddots & & & & \end{bmatrix} \quad (1)$$

[4] *For small termination lengths* **L***, the terminated LDPC-CC suffers a rate loss compared to the underlying LDPC-BC, but this rate loss vanishes for large* **L***.*

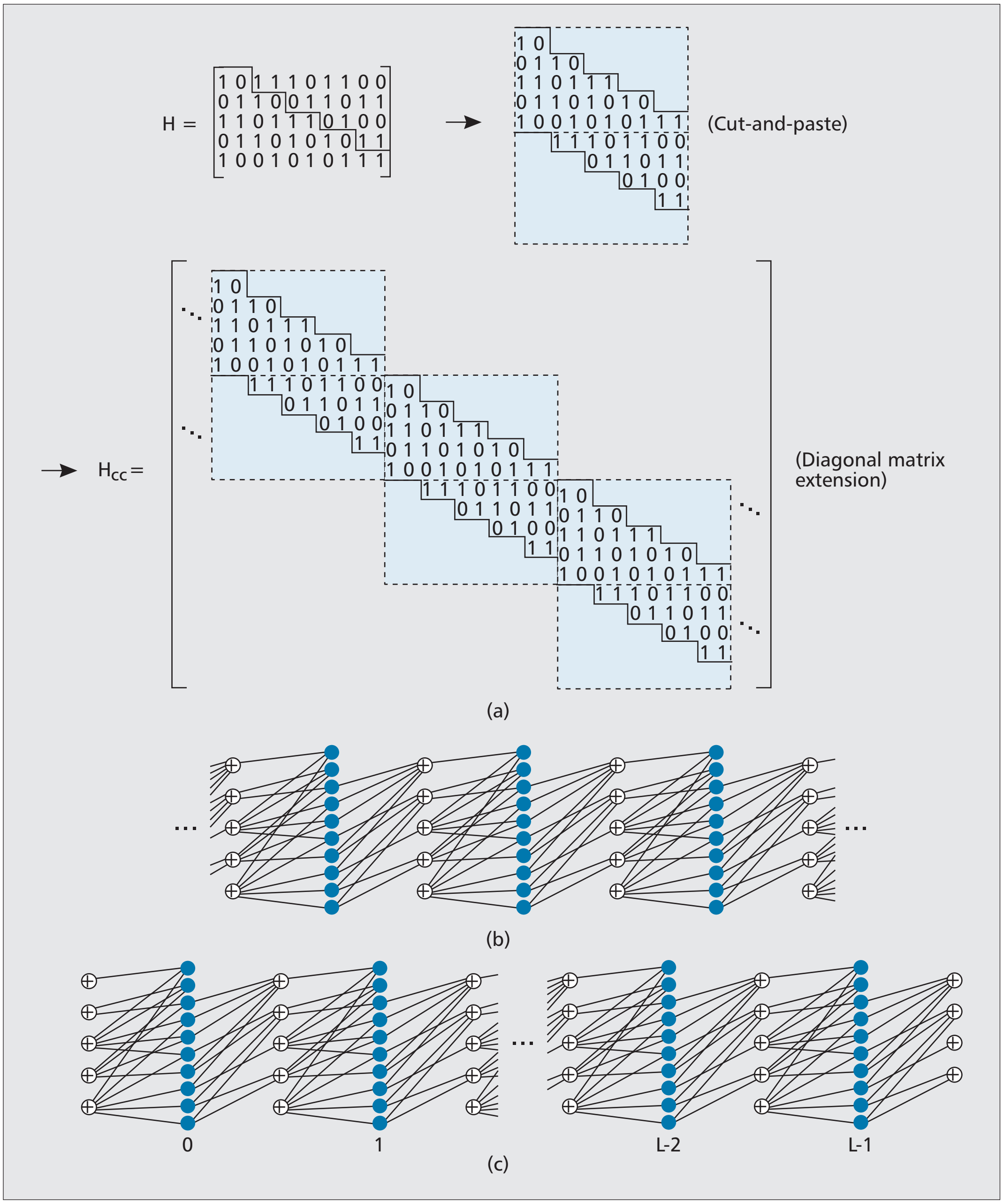

Figure 3. a) An illustration of the unwrapping procedure for a (3, 6)-regular LDPC-BC; b) the Tanner graph associated with the unwrapped (3, 6)-regular LDPC-CC; c) the terminated Tanner graph associated with the unwrapped (3, 6)-regular LDPC-CC.

throughout the graph as iterations increase. This results in BP thresholds for terminated LDPC-CC ensembles that, for large enough degree densities (J and K for regular codes), actually achieve capacity as the constraint length ν_s and termination length L go to infinity. In addition, for regular LDPC-CCs, the terminated (slightly irregular) ensembles are still asymptotically good in the sense that their minimum distance grows linearly with block length n.

The net result of these effects is captured in Fig. 5, which illustrates the trade-offs between the AWGNC BP decoding threshold (in E_b/N_0), the minimum distance growth rate ($d_{\min}/n$), and the code rate (R) for several (J, $2J$)-regular terminated LDPC-CC ensembles as a function of the termination length L. We observe that, in general, as the termination length L increases, the LDPC-CC rate approaches the rate of the underlying LDPC-BC and, in contrast to (J, $2J$)-regular LDPC-BC ensembles, the BP thresholds of the terminated LDPC-CC ensembles approach capacity as J increases.[5] Also, linear distance growth is maintained for any finite value of L. In addition to regular ensembles, Fig. 5 also includes terminated LDPC-CC ensembles based on the irregular ARJA codes designed by Divsalar et al. [9], an irregular LDPC-BC ensemble with linear distance growth and better thresholds than comparable regular ensembles. (Irregular LDPC-BC ensembles with optimized degree profiles already have thresholds close to capacity, and they do not possess linear distance growth, so little is to be gained by applying the

Threshold saturation is a result of the termination, which introduces a slight structured irregularity in the graph. Termination has the effect of introducing lower constraint node degrees, that is, a structured irregularity, at each end of the graph.

[5] *The BP thresholds of the terminated LDPC-CC ensembles are approaching the ML thresholds of the underlying LDPC-BC ensembles, which approach capacity as* J *increases.*

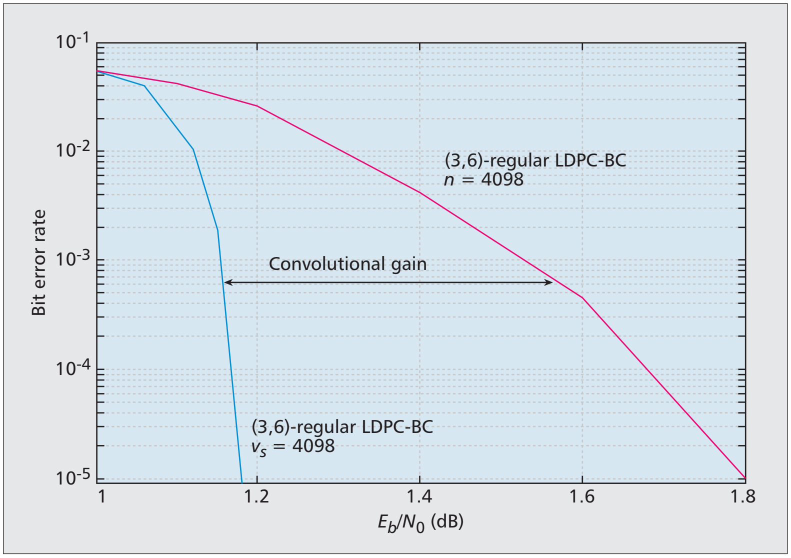

Figure 4. An illustration of convolutional gain.

terminated LDPC-CC construction in these cases.) The major advantage of the regular terminated LDPC-CC constructions highlighted above is that they can achieve the same thresholds as the optimized irregular designs *without sacrificing* linear distance growth, while maintaining the desirable structural features of regular codes [10].

An insightful way of viewing the design of terminated LDPC-CCs is to use a *protograph* representation of the code ensemble [11]. A block code protograph is a small bipartite graph, with c variable nodes and $c - b$ constraint nodes, that is used to represent the parity-check matrix of a rate $R = b/c$ block code with block length c, where c and b are typically small integers. An example of a block code protograph with $c = 2$ variable nodes of degree 3 and $c - b = 1$ constraint node of degree 6 is shown in Fig. 6a. The corresponding $(c - b) \times c = 1 \times 2$ parity-check matrix in this case is given by $\mathbf{B} = [3, 3]$, where $\mathbf{B}$ is called the *base* matrix, and the entries in $\mathbf{B}$ denote that the constraint node in the graph is connected by three edges to each of the two variable nodes. Now form an M-fold *graph cover* by applying the *graph lifting* operation [6] to the protograph. This can be represented by placing a randomly selected permutation of size M (typically a large integer) on each edge of the protograph, thereby connecting M copies of each node and resulting in the Tanner graph of a (3, 6)-regular LDPC-BC with rate $R = b/c$ and block length $n = Mc$. The collection of all M-fold graph covers then represents a (3, 6)-regular LDPC-BC ensemble. The graph lifting operation is equivalent to replacing each entry in the base matrix B with the (modulo-2) sum of three randomly chosen (and distinct) $M \times M$ permutation matrices.[6]

In Fig. 6b, a bi-infinite chain of uncoupled block code protographs is shown, corresponding to block code transmission over time. Now consider spreading the edges of each protograph so that they connect to one or more nearby protographs in the chain in such a way that the degrees of all the nodes are preserved, as illustrated in Fig. 6c. This has the effect of coupling the different protographs together in a bi-infinite chain, which is equivalent to introducing memory into the code design, that is, transitioning from a rate $R = 1/2$ block code to a rate $R = 1/2$ convolutional code, where the syndrome former memory m_s, the coupling depth or number of nearby protographs connected to a given protograph, equals 2 in this case. Viewing Eq. 1 as the base matrix $\mathbf{B}_{\text{cc}}$ of the coupled *convolutional protograph*, this *edge-spreading* (unwrapping) technique is equivalent to splitting the block code base matrix $\mathbf{B}$ into $m_s + 1$ component submatrices $\mathbf{B}_0, \mathbf{B}_1, \dots, \mathbf{B}_{m_s}$, such that $\mathbf{B} = \mathbf{B}_0 + \mathbf{B}_1 + \dots + \mathbf{B}_{m_s}$, where in the most general time-varying case each row of (1) could start with a different base matrix (i.e., each protograph in the chain could be different), and the edge-spreading associated with each protograph could also vary. For the case shown in Fig. 6c, the coupled convolutional protograph is time-invariant, and the edge spreading is given by $\mathbf{B} = [3, 3] = \mathbf{B}_0 + \mathbf{B}_1 + \mathbf{B}_2 = [1, 1] + [1, 1] + [1, 1]$, so the bi-infinite convolutional base matrix becomes

$$\mathbf{B}_{cc} = \begin{bmatrix} & \ddots & & \ddots & & \ddots & & & & \\ 1 & 1 & 1 & 1 & 1 & 1 & & & & \\ & & 1 & 1 & 1 & 1 & 1 & 1 & & \\ & & & & 1 & 1 & 1 & 1 & 1 & 1 \\ & & & & & \ddots & & \ddots & & \ddots \end{bmatrix}. \qquad (2)$$

If the graph lifting operation is now applied to the convolutional protograph by placing randomly selected permutations of size M on each edge of the graph, an unterminated (3, 6)-regular LDPC-CC ensemble with constraint length $\nu_s = (m_s + 1)Mc$ results.

The coupled convolutional protograph can then be terminated, resulting in reduced constraint node degrees at both ends, as shown in Fig. 6d, and the $(L + 2) \times 2L$ terminated convolutional base matrix becomes

$$\mathbf{B}_{cc} = \begin{bmatrix} 1 & 1 & & & & & \\ 1 & 1 & 1 & 1 & & & \\ 1 & 1 & 1 & 1 & \ddots & & \\ & & 1 & 1 & \ddots & 1 & 1 \\ & & & & \ddots & 1 & 1 \\ & & & & & 1 & 1 \end{bmatrix}. \qquad (3)$$

Now applying the graph lifting operation results in a terminated (3, 6)-regular LDPC-CC ensemble, which can also be viewed as an LDPC-BC with block length $n = LMc$. Note that, because of the reduced constraint node degrees at each end, the graph is not quite (3, 6)-regular, and the code rate associated with the terminated LDPC-CC ensemble is less than the rate $R = 1/2$ of the underlying LDPC-BC. However, as the termination length $L \to \infty$, the terminated LDPC-CC ensemble becomes (3, 6)-regular and the associated code rate $R \to 1/2$.

Because the memory employed in the convolutional code design has the effect of coupling together several identical block code protographs, the above graphical construction of terminated LDPC-CC ensembles, also denoted

[6] *Depending on the permutations selected, the lifted parity-check matrix may contain some redundant rows, resulting in a code rate slightly larger than* b/c.

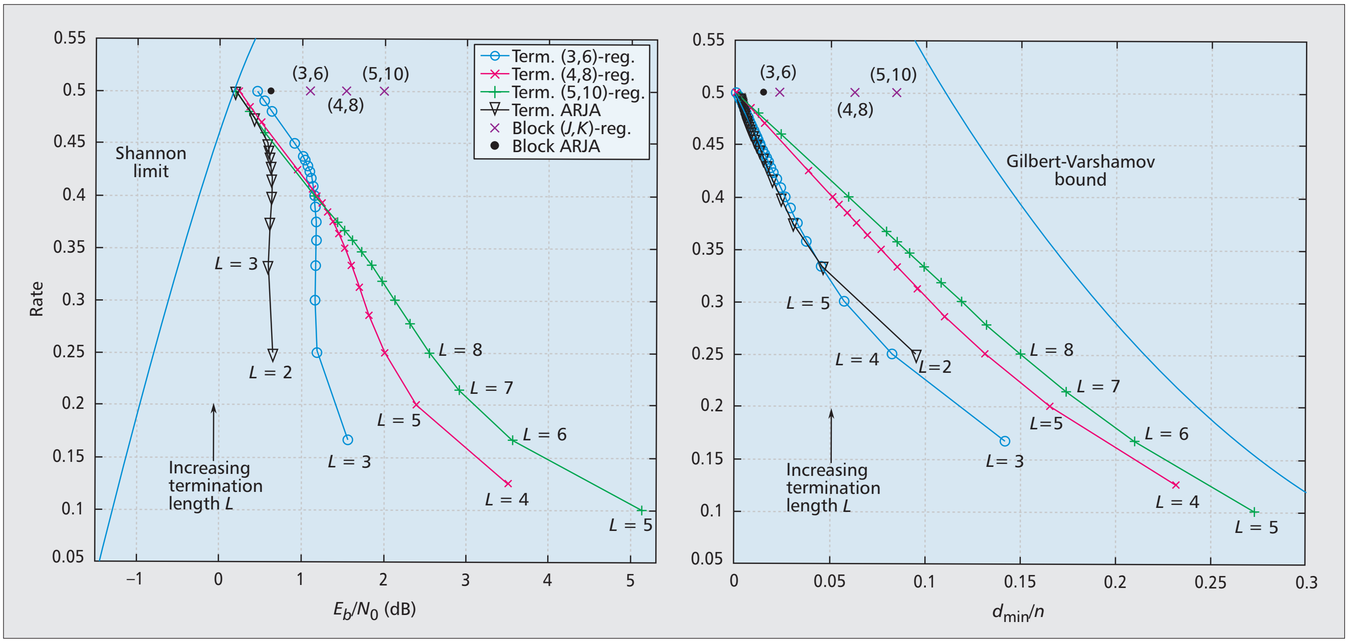

Figure 5. AWGNC BP decoding thresholds (left) and minimum distance growth rates (right) for families of terminated (J, $2J$)-regular LDPC-CC ensembles, terminated ARJA-based LDPC-CC ensembles, and the underlying LDPC-BC ensembles. The Shannon limit (lower bound on E_b/N_0) and the Gilbert-Varshamov bound (random coding bound on $d_{\min} = n$) are plotted for comparison.

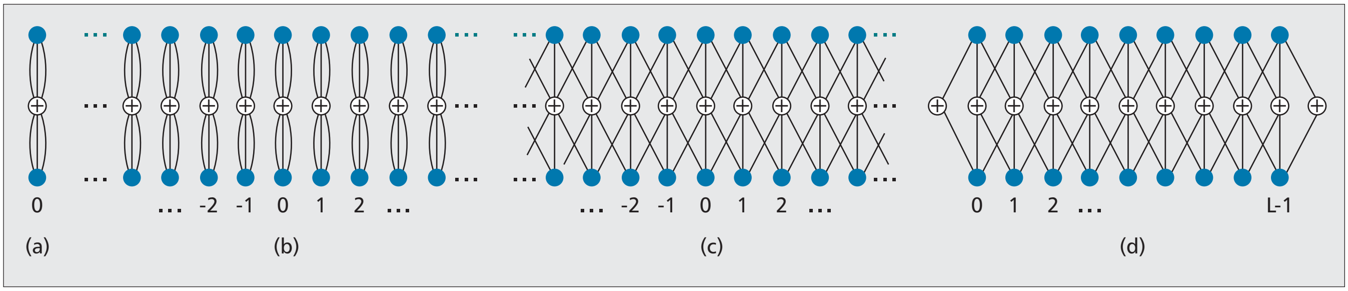

Figure 6. Tanner graphs associated with: a) a (3, 6)-regular block code protograph; b) a bi-infinite chain of uncoupled (3, 6)-regular protographs; c) a bi-infinite chain of coupled (3, 6)-regular protographs; d) a terminated chain of coupled (3, 6)-regular protographs.

as SC-LDPC code ensembles, is referred to as *spatial coupling* [8]. While it is the asymptotic threshold and minimum distance properties of SC-LDPC code ensembles, summarized above, that have generated so much interest in these codes, some basic questions having to do with how best to employ them for practical code lengths still must be solved before they can realize their exceptional promise as a robust, near-optimal solution to the channel coding problem. The following section describes some of these practical issues.

SC-LDPC Codes: Decoding and Practical Considerations

An important contribution of [5] was the introduction of a parallel, high-speed, pipeline-decoding architecture for LDPC-CCs based on the same iterative BP decoding algorithm used to decode LDPC-BCs. This is illustrated in terms of the convolutional protograph associated with an example (3, 6)-regular rate $R = b/c = 1/2$ LDPC-CC with $m_s = 2$ and $\nu_s=(m_s + 1)Mc = 6M$ in Fig. 7a. Given some fixed number I of decoding iterations, the pipeline decoding architecture employs I identical copies of a message-passing processor operating in parallel. Each processor covers a span of ν_s variable nodes, so that during a single decoding iteration messages are always passed within a single processor. As each new set of Mc (in general soft-valued) symbols (represented by M-ary vectors r_t^0 and r_t^1 in Fig. 7a) enters the decoder from the channel, Mc new LLRs are computed, and each processor updates (in parallel) exactly one set of Mc variable nodes and one set of $M(c - b)$ constraint nodes.[7] When the next set of Mc symbols arrives, the decoding window, containing I_{ν_s} variable nodes and I processors, shifts by one time unit (corresponding to a set of Mc received symbols) to the right, and another decoding iteration is performed. In this fashion, the decoder continuously accepts Mc new symbols from the channel and produces (with a delay of $I(m_s + 1)$ time units, or $I(m_s+1)Mc = I\nu_s$ received symbols) decoding estimates of Mc symbols (represented

[7] *Note that within each processor, it is possible to trade between high-speed (parallel) operation and lower-speed (serial) operation by varying the relation between code memory* m_s *and graph lifting factor* M. *For example, for a given constraint length* ν_s *(which determines the code strength), large* M *and small* m_s *result in high-speed processing, whereas the processing is slower for small* M *and large* m_s.

A topic of signficant current research interest involves a detailed performance/complexity comparison of SC-LDPC codes with LDPC-BCs. In particular, can SC-LDPCs with sliding window decoding achieve better performance than standard LDPC-BCs with less computational complexity?

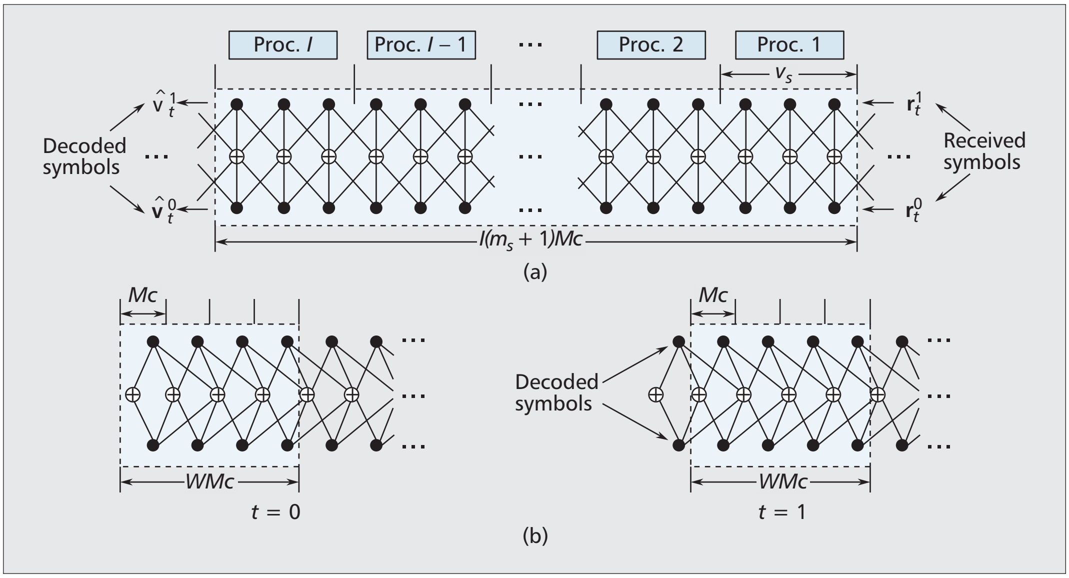

Figure 7. a) Example of a pipeline decoder operating on the protograph of a (3, 6)-regular $R = 1/2$ LDPC-CC; b) example of a sliding window decoder with window size $W = 4$ operating on the protograph of a (3, 6)-regular SC-LDPC code at times $t = 0$ (left), and $t = 1$ (right).

by M-ary vectors $\hat{\mathbf{v}}_t^0$ and $\hat{\mathbf{v}}_t^1$ in Fig. 7a) at each time unit.

As noted above, for the parallel pipeline decoder architecture illustrated in Fig. 7a, the hardware processor includes only one constraint length, that is, $\nu_s = (m_s + 1)Mc$ variable nodes. On the other hand, in the case of SC-LDPC codes (terminated LDPC-CCs), the standard LDPC-BC decoder architecture includes all the variables nodes in a block, which equals LMc, that is, the total block length (Fig. 6d). Since typically L >> m_s, the pipeline architecture achieves a large saving in processor size compared to the standard LDPC-BC architecture, while for the same number of iterations, the performance of the two decoders is identical.

The latency and memory requirements of the pipeline architecture, however, involve an additional factor of the number of iterations I, so that $I\nu_s = I(m_s + 1)Mc$ represents the total decoding latency in received symbols and the total number of soft received values that must be stored in the decoder memory at any given time. This equals the length of the decoding window in Fig. 7a, where the factor of M is included to account for the size of the permutation matrix. In some applications, since capacity-approaching performance can require a large number of iterations, these latency and storage requirements may be unacceptably high.

This fact has spurred interest in a modified *sliding window* decoding architecture for SC-LDPC codes with much reduced latency and memory requirements [12]. Rather than maintaining a full window size of $I\nu_s = I(m_s + 1)Mc$ symbols, like the pipeline decoder, the sliding window decoder uses a much smaller window, typically just a few constraint lengths. The concept of a sliding window decoder is illustrated in Fig. 7b. Assuming a window size of $W\,Mc$ symbols, where W is the number of protograph sections within the window, decoding iterations proceed until some stopping criterion is met or a fixed number of iterations has been performed, after which the window shifts and the Mc symbols shifted out of the window are decoded. The key feature of a sliding window decoder is that for $W << I(m_s + 1)$, its latency and memory requirements are much less than for the pipeline decoder. Values of W roughly 5 to 10 times as large as $(m_s + 1)$ have been shown [13] to result in significant savings in latency and memory with minimal performance degradation, compared to using the full window size $I(m_s + 1)$ for the (typically large) fixed number of iterations I needed to optimize the performance of the pipeline decoder.[8]

Because the initial few (depending on the coupling depth, or code memory) positions of the window include a part of the graph with reduced constraint node degrees (Fig. 7b), the information passed to variable nodes during the iterations associated with these initial window positions is highly reliable. The design of the sliding window decoder ensures that this highly reliable information then propagates down the graph as the window is shifted. This phenomenon is responsible for the threshold saturation effect associated with SC-LDPC codes. The same phenomenon manifests itself with the standard LDPC-BC or pipeline decoding architectures, but recent work shows that the propagation of reliable information through the graph occurs more efficiently with the sliding window architecture [13]. Thus, besides reducing latency and memory, another motivation for considering a sliding window decoder is to reduce the number of node updates (computational complexity) required to achieve a given level of performance.

Open Research Problems

Below is a partial list of open research problems related to the practical realization of SC-LDPC codes.

- A topic of signficant current research interest involves a detailed performance/complexity

[8] *Even for a sliding window decoder extending over a small number of constraint lengths, it may still be advantageous to employ the parallel pipeline architecture in the implementation of the decoding window.*

comparison of SC-LDPC codes with LDPC-BCs. In particular, can SC-LDPCs with sliding window decoding achieve better performance than standard LDPC-BCs with less computational complexity? A fair comparison must consider decoders with the same latency and memory requirements; that is, the sliding window decoder for an SC-LDPC code must be compared to an LDPC-BC whose block length is equal to WMc rather than LMc. In addition, the degree profiles of the two codes should be the same; for example, a (J, K)-regular SC-LDPC code should be compared to a (J, K)-regular LDPC-BC. Some of the factors to be considered include choosing the most efficient (message passing) *node update schedules* for decoding and designing appropriate *stopping rules* for deciding when enough iterations have been performed to allow reliable decoding. For example, practical stopping rules for the sliding window decoder could be based on a partial syndrome check, analogous to the stopping rule normally applied to the decoding of LDPC-BCs, or on the LLR statistics associated with the next set of symbols to be decoded, for example by using a threshold criterion.

•The results of [13] imply trade-offs favorable to SC-LDPC codes (assuming sliding window decoding) compared to LDPC-BCs in the waterfall region of the BER curve. There has been only limited investigation, however, regarding the error floors that might result from the use of SC-LDPC codes. Since error floor performance is a major factor in selecting codes for applications that require very low BERs, such as data storage and optical communication, it is important to consider those factors that contribute to decoding failures for SC-LDPC codes in the error floor. An important aspect of such an analysis involves establishing the precise connection between the problematic graphical substructures (trapping sets, absorbing sets) that can cause decoding failures and various decoding parameters, including scheduling choices (e.g., parallel or serial node updates), amount of quantization for the stored LLRs, and window size (for sliding window decoding).

•SC-LDPC codes are known to have excellent asymptotic properties, but there are many open questions regarding code design, in particular for short-to-moderate block lengths: scaling the lifting factor M and the termination length L to achieve the best possible performance for SC-LDPC codes of finite length; studying puncturing as a means of obtaining SC-LDPC codes with high rates in order to provide rate flexibility for standards applications; finding ways to mitigate the rate loss associated with short-to-moderate block length SC-LDPC codes, such as puncturing and partial termination, without affecting performance; and exploiting the potential of connecting together multiple SC-LDPC code chains. In addition, analyzing and designing more powerful SC-LDPC codes, such as nonbinary or generalized SC-LDPC codes, may be attractive in areas such as coded modulation and flash memories.

•Members of an LDPC code ensemble that are *quasi-cyclic* (QC) are of particular interest to code designers, since they can be encoded with low complexity using simple feedback shift-registers, and their structure leads to efficiencies in decoder design. The practical advantages of QC-LDPC-BC designs also carry over to the design of QC-SC-LDPC codes. It should be noted, however, that once the QC constraint is applied, the asymptotic ensemble average properties, as noted earlier, do not necessarily hold (since we are now choosing a code from a restricted subensemble), and particular QC-SC-LDPC codes must be carefully designed to ensure good performance.

Once the QC constraint is applied, the asymptotic ensemble average properties, as noted earlier, do not necessarily hold (since we are now choosing a code from a restricted sub-ensemble), and particular QC-SC-LDPC codes must be carefully designed to ensure good performance.

Concluding Remarks

In this article we have attempted to provide a brief tutorial overview of the exciting new field of spatially coupled low-density parity check codes. Capacity approaching iterative decoding thresholds and asymptotically good minimum distance properties make these codes potentially very attractive for future industry standard applications. We have traced the origins of SC-LDPC codes to the development of low-density parity check convolutional codes in [5], used a visually convenient protograph representation to describe their construction, discussed several issues related to practical decoder implementations, and summarized a few remaining open research problems.

Finally, it has recently been shown that the improved thresholds associated with spatial coupling apply generally regardless of the particular physical situation or communication channel [14, 15]. Furthermore, although space limitations do not allow us to provide details, we note that the concept of spatial coupling of a sequence of identical copies of a small structured graph (a protograph) is applicable to a wide variety of problems and has been shown to lead to improved system performance in areas as diverse as multiterminal source and channel coding, cooperative relaying, compressed sensing, secure communication, and statistical physics.

References

[1] C. E. Shannon, “A Mathematical Theory of Communication, ” *Bell Sys. Tech. J.*, vol. 27, Oct. 1948, pp. 379–423.

[2] R. G. Gallager, “Low-Density Parity-Check Codes, ” *IRE Trans. Info. Theory*, vol. 8, no. 1, Jan. 1962, pp. 21–28.

[3] M. G. Luby *et al.*, “Improved Low-Density Parity-Check Codes using Irregular Graphs, ” *IEEE Trans. Info. Theory*, vol. 47, no. 2, Feb. 2001, pp. 585–98.

[4] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*, Cambridge Univ. Press, 2008.

[5] A. Jiménez Felström and K. Sh. Zigangirov, “Time-Varying Periodic Convolutional Codes with Low-Density Parity-Check Matrices, ” *IEEE Trans. Info. Theory*, vol. 45, no. 6, Sept. 1999, pp. 2181–91.

[6] A. E. Pusane *et al.*, “Deriving Good LDPC Convolutional Codes from LDPC Block Codes, ” *IEEE Trans. Info. Theory*, vol. 57, no. 2, Feb. 2011, pp. 835–57.

[7] M. Lentmaier *et al.*, “Iterative Decoding Threshold Analysis for LDPC Convolutional Codes, ” *IEEE Trans. Inf. Theory*, vol. 56, no. 10, Oct. 2010, pp. 5274–89.

[8] S. Kudekar, T. J. Richardson, and R. L. Urbanke, “Threshold Saturation via Spatial Coupling: Why Convolutional LDPC Ensembles Perform So Well Over the BEC, ” *IEEE Trans. Info. Theory*, vol. 57, no. 2, Feb. 2011, pp. 803–34.

[9] D. Divsalar *et al.*, “Capacity-Approaching Protograph Codes, ” *IEEE JSAC*, vol. 27, no. 6, Aug. 2009, pp. 876–88.

[10] M. Lentmaier *et al.*, “Asymptotically Good LDPC Convolutional Codes with AWGN Channel Thresholds Close to the Shannon Limit, ” *Proc. 6th Int’l. Symp. Turbo Codes and Iterative Info. Processing*, Brest, France, Sept. 2010, pp. 334–38.

[11] J. Thorpe, “Low-Density Parity-Check (LDPC) Codes Constructed from Protographs, ” Jet Propulsion Lab, Pasadena, CA, INP Progress Report 42-154, Aug. 2003.

[12] A. R. Iyengar *et al.*, “Windowed Decoding of Protograph-Based LDPC Convolutional Codes over Erasure Channels, ” *IEEE Trans. Info. Theory*, vol. 58, no. 4, Apr. 2012, pp. 2303–20.

[13] M. Lentmaier, M. M. Prenda, and G. Fettweis, “Efficient Message Passing Scheduling for Terminated LDPC Convolutional Codes, ” *Proc. IEEE Int’l. Symp. Info. Theory*, St. Petersburg, Russia, Aug. 2011, pp. 1826–30.

[14] S. Kudekar, T. Richardson, and R. Urbanke, “Spatially Coupled Ensembles Universally Achieve Capacity under Belief Propagation, ” *IEEE Trans. Info. Theory*, vol. 59, no. 12, Dec. 2013, pp. 7761–7813.

[15] S. Kumar *et al.*, “A Proof of Threshold Saturation for Spatially-Coupled LDPC Codes on BMS Channels, ” *Proc. 50th Annual Allerton Conf.*, Monticello, IL, Oct. 2012, pp. 176–84.

BIOGRAPHIES

DANIEL J. COSTELLO, JR. [M’69, F’85] (costello.2@nd.edu) served three terms as a member of the IT Society Board of Governors from 1983 to 2010, and in 1986 he served as President. In 1999, he received a Humboldt Research Prize and an IT Society Third-Millennium Medal, and in 2000 he was named Bettex Professor of Electrical Engineering at Notre Dame. He was a co-recipient of the 2009 Fink Prize Paper Award and the 2012 joint IT/COM Prize Paper Award, and received the 2013 IT Society Wyner Distinguished Service Award.

LARA DOLECEK: [S’05,M’10,SM’13] (dolecek@ee.ucla.edu) is an assistant professor with the Electrical Engineering Department at the University of California at Los Angeles (UCLA). She holds a Ph.D. from the Electrical Engineering and Computer Science Department, UC Berkeley. She received the 2012 NSF CAREER Award and 2013 Okawa Research Grant, and was a 2013 Intel Fellow. Her research interests span coding and information theory, graphical models, statistical algorithms, and computational methods, with applications to emerging systems for data storage, processing, and communication.

THOMAS E. FUJA: [S’80,M’87,SM’97,F’04] (t.fuja@nd.edu) received his engineering education at the University of Michigan and Cornell University. Since 1998 he has been on the faculty of the University of Notre Dame, where he is currently chair of the Department of Electrical Engineering. Prior to that, he was on the faculty of the University of Maryland; he also served as an NSF program director in 1997–1998. His research interests are in coding theory and applications, and information theory.

JÖRG KLIEWER [S’97,M’99,SM’04] (joerg.kliewer@njit.edu) is an associate professor at the New Jersey Institute of Technology, Newark. He received his Dr.-Ing. degree in electrical engineering from the University of Kiel, Germany, in 1999. His research interests include error correcting codes, network coding, information theory, and communication networks. He has been an Associate Editor of the *IEEE Transactions on Communications* since 2008 and was the recipient of a German Research Foundation Fellowship Award in 2004.

DAVID G. M. MITCHELL [S’08,M’10] (david.mitchell@nd.edu) received his Ph.D. degree in electrical engineering from the University of Edinburgh, United Kingdom, in 2009. He is currently a visiting assistant professor in the Department of Electrical Engineering at the University of Notre Dame. His research interests include coding theory and practice, and information theory.

ROXANA SMARANDACHE [S’97,M’01,SM’14] (rsmarand@nd.edu) is an associate professor of mathematics and electrical engineering at the University of Notre Dame. She received her Ph.D. degree in mathematics from the University of Notre Dame in 2001. After spending 11 years on the faculty of San Diego State University, she joined the University of Notre Dame in 2012. Her research interests are in coding theory, combinatorics, graph theory, and network coding.