A Novel Design of Spatially Coupled LDPC Codes for Sliding Window Decoding

Zhu, Min; Mitchell, David G. M.; Lentmaier, Michael; Costello, Jr., Daniel J.

Published in:
2020 IEEE International Symposium on Information Theory, ISIT 2020

DOI:
10.1109/ISIT44484.2020.9173961

2020

Document Version:
Peer reviewed version (aka post-print)

A Novel Design of Spatially Coupled LDPC Codes for Sliding Window Decoding

Min Zhu*, David G. M. Mitchell†, Michael Lentmaier‡, and Daniel J. Costello, Jr.§
*State Key Laboratory of ISN, Xidian University, Xi'an, P. R. China, zhunanzhumin@gmail.com
†Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM, USA, dgmm@nmsu.edu
‡Department of Electrical and Information Technology, Lund University, Lund, Sweden, michael.lentmaier@eit.lth.se
§Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, USA, dcostel1@nd.edu

***Abstract*—We introduce a novel design of spatially coupled low density parity check codes in order to reduce the effects of error propagation in low-latency sliding window decoding for large frame lengths or streaming applications. Specifically, we employ reduced-degree check nodes spaced throughout the coupling chain, which have the effect of allowing the decoder to recover from error bursts. A simplified analysis of the block error rate (BLER) of the proposed codes is presented that allows us to predict the effect of different placements of reduced-degree checks in the coupling chain. Simulation results supporting the beneficial effect of the new code design on the overall BLER performance are included.**

I. Introduction

Spatially coupled low density parity check (SC-LDPC) codes, the *convolutional* counterpart of low density parity check block codes (LDPC-BCs), were first introduced in [1]. Subsequently, it was shown in [2]–[5] that SC-LDPC codes derived from *regular* LDPC-BC codes exhibit *threshold saturation*, i. e., the *suboptimal* belief propagation (BP) iterative decoding threshold of SC-LDPC code ensembles over memoryless binary-input symmetric-output channels coincides with the *optimal* MAP threshold of their underlying LDPC-BC ensembles.

Without loss of generality, we consider an example of (3,6)-regular SC-LDPC codes, constructed by means of protographs [6]. Fig. 1(a) shows an independent (uncoupled) sequence of (3,6)-regular LDPC-BC protographs with *base matrix* $\mathbf{B} = [3\ 3]$, while Fig. 1(b) shows a (3,6)-regular SC-LDPC code chain obtained by applying *edge spreading* to the uncoupled protographs in Fig. 1(a) [7]. The edge spreading is defined by a set of *component base matrices* $\mathbf{B}_0 = \mathbf{B}_1 = \mathbf{B}_2 = [1\ 1]$ satisfying the condition $\mathbf{B} = \sum_{i=1}^{m} \mathbf{B}_i$, where the *coupling width* is $m = 2$.[1] Applying the *graph lifting factor* M to the SC-LDPC protograph of Fig. 1(b) results in an ensemble of (3,6)-regular SC-LDPC codes in which each time unit represents a *block* of $2M$ coded symbols. An important feature of the SC-LDPC protograph in Fig. 1(b) is the *structured irregularity* at the beginning, comprised of a degree-2 check followed by a degree-4 check before the graph assumes its regular structure, that accounts for the threshold saturation phenomenon noted above. For *terminated* (finite-length) protographs, a similar set of reduced-degree checks also appears at the end [7].

Sliding window decoding (SWD) of SC-LDPC codes [8], as opposed to using a standard *flooding decoding schedule* over an entire terminated graph, can be employed to reduce decoding latency (and decoding complexity) for long code chains

[1]The coupling width m is also referred to as the *syndrome former memory* [1].

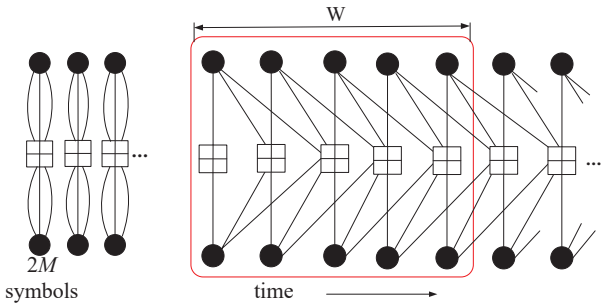


Fig. 1. A (3,6)-regular SC-LDPC code protograph obtained from an underlying LDPC-BC protograph with base matrix $B = [3, 3]$. The black circles represent variable nodes, and the "plus" squares represent check nodes. (a) A chain of independent (uncoupled) protographs; (b) Spreading edges to the $m = 2$ nearest neighbors.

and for *unterminated* streaming (continuous transmission) applications. For example, in Fig. 1(b), the red rectangular box represents a decoding window of *size* $W = 5$ (blocks). To decode (1) a BP flooding schedule is applied to all the nodes in the window for some fixed number of iterations, or until some *stopping criterion* is met, (2) the block of $2M$ *target symbols* in the first window position is decoded according to the signs of their *log-likelihood ratios* (LLRs), (3) the window shifts one time unit (block) to the right, and (4) decoding continues in the same fashion until the entire chain is decoded, where the *decoding latency* in symbols is given by $2MW$.

In order to reduce the decoding latency (and decoding complexity), the window size W should be chosen as small as possible. Defining the *decoding constraint length* as $\eta = m+1$, experimental results have shown that good performance can typically be maintained as long as $W \geq 4\eta$ [9]. However, for smaller values of W (reduced latency), or for very long code chains, infrequent but severe decoder *error propagation* sometimes occurs. Error propagation can be triggered when, after a block decoding error occurs, the decoding of the subsequent block is affected, which in turn causes a continuous string of block errors, resulting in an unacceptable loss in performance. Techniques that involve adapting the number of decoder iterations and shifting the window position have recently been proposed by Klaiber et al. [10] to combat the error propagation problem. Also, for the related class of braided convolutional codes (BCCs) [11] with SWD [12], error propagation mitigation methods that involve extending the size of the window, resynchronization, and retransmission have been proposed [13].

In this paper, we introduce a novel code design that strategically places graph irregularities, viz., reduced-degree

check nodes, throughout the coupled chain, thus truncating the error propagation without any changes to the decoder, at a cost of some modest rate loss. In Section II, we describe how error propagation is triggered in SWD of SC-LDPC codes and present a simplified analysis of the *block error rate* (BLER) performance that illustrates how the severity of the problem depends on the length of the code chain. In Section III, we introduce the new code design and use the analysis from Section II to predict its performance, showing how the effect of error propagation can be limited. Section IV presents numerical results obtained from computer simulations that confirm the ability of the new code design to limit error propagation and improve performance, and some concluding remarks are presented in Section V.

II. Decoder Error Propagation in SWD

In SWD of SC-LDPC codes, when a block of target symbols at time t is decoded, the window shifts to include the most recent block of received symbols at time $t+W$, and decoding commences on the block of target symbols at time $t+1$. During the decoding of the time $t+1$ block, for a coupling width of m, the final LLRs of the m past decoded blocks, from time $t-m+1$ to time t, remain involved in the decoding process, although these LLRs are no longer updated.

Under normal operation, decoding proceeds with correctly decoded blocks until such time as a block of target symbols contains one or more LLRs with incorrect sign when the window shifts, thus resulting in a *block error*. Typically, if only a few symbols have incorrect LLRs and most of the correct symbols have strong LLRs, the LLRs of the incorrectly decoded block will have only a small effect, and the decoder will recover and continue to correctly decode subsequent blocks, assuming most of the symbols in the window have large and correct LLRs. This type of operation results in randomly distributed error blocks.

However, if an error block contains a large number of incorrect LLRs, particularly if they have large absolute values and many of the LLRs associated with the correct symbols are small, those "bad" LLRs may negatively affect the decoding of the next block of target symbols, causing a block error that would not have occurred under normal operating conditions. This in turn could trigger additional block errors, resulting in an *error propagation* effect, i.e., a continuous sequence of incorrectly decoded blocks.

In an application where information is transmitted in *frames* of a fixed length L time units, with graph termination (reduced check node degrees for m time units) following the last block of transmitted variable nodes, any error propagation will be limited and decoding will start fresh with the next frame. However, if L is large, a significant number of blocks could be affected by error propagation, thus severely degrading performance. In a streaming application, with no termination, the situation could be catastrophic, resulting in a BLER that asymptotically tends to 1.

Before proceeding with a discussion of methods to circumvent decoder error propagation, it is instructive to consider the difficulty of assessing the extent of the problem using conventional Monte Carlo simulation techniques. Normally, due to the constraints imposed by system operators on the allowable time duration of submitted jobs, decoded BLERs are determined by simulating a large number N of frames, each of length L, such that the total number of simulated code symbols $nMLN$ is large enough to gather reliable error statistics, where n is the number of variable nodes in the uncoupled protograph ($n = 2$ in Fig. 1). Under normal decoder operating conditions, when block errors occur randomly, this process works perfectly well, but when the decoder experiences error propagation, BLER statistics can be severely affected by the particular combination of L and N chosen for the simulation. Below, we give a simplified analysis of how error propagation can affect the accuracy of simulated BLER statistics.

Assume that, in any given frame, the decoder operates in one of two states: (1) a *random error state* S_{re} in which block errors occur independently with probability p, and (2) an *error propagation state* S_{ep} in which block errors occur with probability 1. Also assume that, at each time unit $t = 1, 2, 3, \ldots, L$ the decoder transitions from state S_{re} to state S_{ep} independently with probability q (typically, $q \ll p$) and that, once in state S_{ep}, the decoder remains there for the rest of the frame.[2] A state diagram describing this situation is shown in Fig. 2.

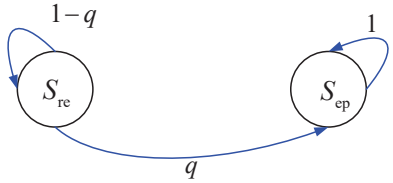


Fig. 2. The state diagram describing the operation of a decoder subject to error propagation.

Now consider that n, M, and $B \triangleq LN$, the total number of blocks to be simulated, are fixed, and let $\Lambda = \alpha L$ and $\Omega = N/\alpha$, so that $B = LN = \Lambda\Omega$, where the *frame length parameter* $\alpha \geq 1$. We can view $\alpha = 1$ as corresponding to simulating N frames of length L and $\alpha > 1$ as corresponding to simulating a smaller number Ω of frames of length $\Lambda > L$, while maintaining the same total number of simulated symbols nMB.[3] Under normal (random error) decoder operating conditions, the simulated BLER should be independent of the chosen value of α. When decoder error propagation is possible, however, we now show that the value of α can affect the simulated BLER.

For a frame of length Λ, we express the probability that the decoder first enters state S_{ep} at time $t = \tau$ (and thus stays in state S_{ep} until time $t = \Lambda$) as

$$P_\tau\left(S_{\text{ep}}, t = [\tau : \Lambda]\right) = q(1-q)^{\tau-1}, \tau = 1, 2, \ldots, \Lambda, \quad (1)$$

where the notation $t = [t_1 : t_2]$ denotes the set of time units from t_1 to t_2. Similarly, we can write the probability that the decoder stays in state S_{re} throughout the entire frame as

$$\begin{aligned} P\left(S_{\text{re}}, t = [1 : \Lambda]\right) &= 1 - \sum_{\tau=1}^{\Lambda} P_\tau\left(S_{\text{ep}}, t = [\tau : \Lambda]\right) \\ &= (1-q)^{\Lambda}. \end{aligned} \quad (2)$$

Now, given that a frame enters state S_{ep} at time $t = \tau$, we can express the average BLER as

$$P_{\text{BL}}\left(\tau = 1\right) = 1, \quad (3a)$$

[2] A given frame can (1) operate entirely in state S_{re}, where error propagation never occurs, (2) start in state S_{re} and then at some time transition to state S_{ep}, or (3) operate entirely in state S_{ep}, where the very first block is decoded incorrectly and block errors continue throughout the rest of the frame.

[3] We assume α is chosen so that Ω and Λ are integers.

$$P_{\mathrm{BL}}(\tau) = [p \cdot (\tau - 2) + 0 \cdot 1 + 1 \cdot (\Lambda - \tau + 1)]/\Lambda$$
$$= [p \cdot (\tau - 2) + \Lambda - \tau + 1]/\Lambda, \tau = 2, \ldots, \Lambda, \quad (3b)$$

where we note that state S_{ep} must be preceded by at least one correctly decoded block. Finally, we can write the overall average BLER as

$$P_{\mathrm{BL}} = \sum_{\tau=1}^{\Lambda} P_{\mathrm{BL}}(\tau) \cdot P_{\tau}(S_{\mathrm{ep}}, t = [\tau : \Lambda]) + p \cdot P(S_{\mathrm{re}}, t = [1 : \Lambda]) \quad (4)$$
$$= \sum_{\tau=1}^{\Lambda} P_{\mathrm{BL}}(\tau) \cdot q(1-q)^{\tau-1} + p \cdot (1-q)^{\Lambda}.$$

Looking at (4), it is clear that, if $q = 0$, i.e., we never enter state S_{ep}, then $P_{\mathrm{BL}} = p$, independent of the frame length Λ. This is the normal condition under which Monte Carlo simulations are conducted. However, under error propagation conditions, the simulated BLER will increase as a function of the frame length, as the following example illustrates.

Example 1: Assume $q = 10^{-4}$, $p = 10^{-2}$, and $L = 100$.[4]

Case 1: For $\alpha = 1$, i.e., we simulate $\Omega = N$ frames, each of length $\Lambda = L = 100$. Using (3) and (4) we obtain $P_{\mathrm{BL}} = 1.4982 \cdot 10^{-2}$.[5]

Case 2: For $\alpha = 10$, i.e., we simulate $\Omega = N/10$ frames, each of length $\Lambda = 10L = 1000$, we obtain $P_{\mathrm{BL}} = 5.7939 \cdot 10^{-2}$.

Case 3: For $\alpha = 100$, i.e., we simulate $\Omega = N/100$ frames, each of length $\Lambda = 100L = 10,000$, we obtain $P_{\mathrm{BL}} = 3.74244 \cdot 10^{-1}$. ■

From the example, we clearly see that, under error propagation conditions, the simulated BLER depends on the frame length $\Lambda = \alpha L$, becoming worse as α increases. We now illustrate this effect using actual simulated data rather than the above simplified analysis, where P_{BL} is plotted as a function of E_b/N_0 and the frame length parameter α.

Example 2: The BLER performance of SWD of the (3,6)-regular SC-LDPC codes of Fig. 1(b) is shown in Fig. 3, with $W = 18$, $M = 2000$, $L = 250$, $N = 20,000$, frame length $\Lambda = \alpha L$, number of frames $\Omega = N/\alpha$, and frame length parameter α =1, 2, and 4, where we note that, in each case, the total number of simulated blocks is $B = LN = \Lambda\Omega = 5 \times 10^6$ and the total number of simulated code symbols is $nMB = 2 \times 10^{10}$. From the figure, we observe that, with increasing α, the BLER performance becomes worse, even though there are relatively few error-propagation frames overall, thus confirming the results of the above analysis.[6] ■

III. A Novel Construction of SC-LDPC Codes

In order to combat the problem of error propagation in SWD, we now introduce a novel code design based on occasionally inserting additional check nodes into the protograph of a regular SC-LDPC code. This structured irregularity, which we refer to as *check node doping*, slightly reduces the code rate but has the beneficial effect of partitioning a long or continuous graph into shorter "sections", thus limiting the effects of error propagation.[7] Because check node doping effectively divides a long frame into shorter sub-frames, we can apply the simplified analysis of the previous section to demonstrate the improved BLER performance compared to a standard regular code design. To illustrate the graph doping process, we consider the (3,6)-regular SC-LDPC codes of Fig. 1(b) as an example, while noting that the design can be applied in the same way to any (J, K)-regular SC-LDPC code.

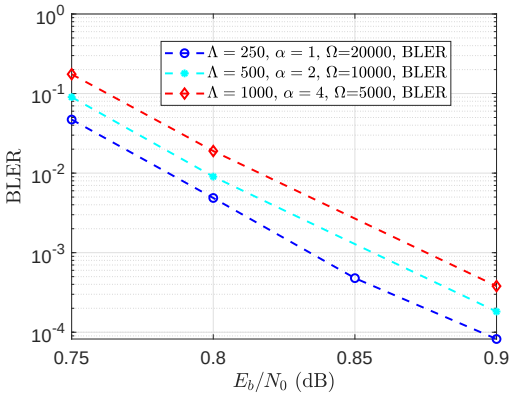


Fig. 3. SWD BLER performance of the (3,6)-regular SC-LDPC codes of Fig. 1(b) for three different frame lengths but the same number of simulated blocks.

A. Code Construction

In the (3,6)-regular SC-LDPC code design shown in Fig. 1, all the variable nodes (VNs) have degree 3, while all the check nodes (CNs), except the ones at the graph boundaries, have degree 6. As noted previously, the reliable information generated by the reduced-degree CNs at the boundaries is responsible for the threshold saturation property of SC-LDPC codes. Motivated by this fact and our desire to limit the effects of error propagation in long frames, the CN doped code design introduces similar reduced-degree CNs at different points in the coupled chain, as shown in Fig. 4, where the VNs at time $t = \tau_1$ (colored red) spread their three edges to the CNs at times $t = \tau_1 + 1$, $t = \tau_1 + 2$, and $t = \tau_1 + 3$; the red VNs at time $t = \tau_2$ spread their three edges to the CNs at times $t = \tau_2 + 2$, $t = \tau_2 + 3$, and $t = \tau_2 + 4$, and so on, i.e., if the red VNs at time $t = \tau_j$ are chosen as the jth *doping point*, their edges are connected to the CNs at times $\tau_j + j$, $\tau_j + j + 1$, and $\tau_j + j + 2$. Additionally, the VNs between doping points (colored black) are coupled in the same way as the preceding red VN pair. This construction has the effect of inserting an additional CN at each doping point, and as a result three degree 4 CNs (colored green) are generated. At a cost of a slight rate loss, these reduced degree CNs at the doping points result in stronger "local" codes compared to the standard (3,6)-regular design, which facilitates the ability of a SWD to truncate error propagation. The base matrix corresponding to these CN doped SC-LDPC codes is shown in Fig. 5(a).

To decode a CN doped (3,6)-regular SC-LDPC code, SWD can be applied to the doped chain, with a slight difference in the way the window shifts compared to standard (3,6)-regular SWD. The window-shifting schedule for check node doped (3,6)-regular SC-LDPC codes is illustrated in Fig. 4

[4]Here p and q are chosen arbitrarily for purposes of illustration. In practice, p and q would be functions of the signal-to-noise (SNR) E_b/N_0 and W and could be determined experimentally.

[5]We note from (4) that the calculation of P_{BL} depends only on the frame length Λ and not on the number of simulated frames Ω.

[6]Fig. 3 represents only a narrow range of SNRs, below the threshold (1.11 dB) of the underlying LDPC-BC, where error propagation presents a significant problem. For larger values of E_b/N_0 and/or W, SWD typically recovers from an error burst and returns to the random error state.

[7]The basic concept of doping was first introduced by ten Brink in [14].

(protograph view) and Fig. 5(b) (base matrix view) For a window of size W, the block of $2M$ symbols at the earliest time (leftmost position in the window) are the target symbols. Flooding BP decoding is performed within the window, either using a fixed number of iterations or with a stopping rule. After a block of target symbols is decoded, the window (VNs and CNs) shifts by one time unit. When a doping point (red VN pair) becomes the target block, the window shifts by one VN time unit to include one new block of VNs, as before, but it shifts by two CN time units to include two new blocks of CNs (and thus still including the same total number of CNs).[8] After the red target symbols are decoded, the window (VNs and CNs) again shifts by one time unit until the next doping point is reached, and the same process is repeated. Generally, for the VN block of (red) target symbols at doping point τ_j, the decoding window covers the VNs from times τ_j to τ_j+W-1 and the CNs from time τ_j+j to $\tau_j+j+W-1$.

B. BLER Analysis

We now make use of the simplified analysis of the BLER of SWD of SC-LDPC codes presented in Sec. II to illustrate the advantage of CN doped SC-LDPC codes. Let $\lambda_j, j \in [0,1,\ldots,l-1]$ denote the number of blocks in each sub-frame of the graph (between doping points), where l represents the number of sub-frames in a frame. Referring to Fig. 4, for a frame of length Λ, it follows that $\sum_{j=0}^{l-1}\lambda_j=\Lambda$, where $\lambda_j = \tau_{j+1}-\tau_j$, $\tau_0 \triangleq 0$, and $\tau_l \triangleq \Lambda$. We now make the assumption that, if error propagation occurs in the ith sub-frame, the decoder state S_{ep} will be truncated at the end of that sub-frame.[9] This allows us to apply the analysis of Sec. II independently to each sub-frame.

Since, there are λ_j blocks, $j \in [0,1,\ldots,l-1]$, in the jth sub-frame, according to (4) we can write the average BLER of this sub-frame as

$$P_{\text{BL},\lambda_j} = \sum_{\tau=1}^{\lambda_j} P_{\text{BL},\lambda_j}(\tau)\cdot q(1-q)^{\tau-1} + p(1-q)^{\lambda_j}, \quad (5)$$

where $P_{\text{BL},\lambda_j}(\tau)$ is the average BLER given that the jth sub-frame enters state S_{ep} at time $t=\tau$. Now using (3) we can express $P_{\text{BL},\lambda_j}(\tau)$ as

$$P_{BL,\lambda_j}(\tau=1) = 1, \quad (6a)$$

$$P_{\text{BL},\lambda_j}(\tau) = [p\cdot(\tau-2)+\lambda_j-\tau+1]/\lambda_j, \tau=2,\ldots,\lambda_j. \quad (6b)$$

Finally, we can write the overall average BLER as

$$P_{\text{BL,doped}} = \frac{\sum_{j=0}^{l-1} P_{\text{BL},\lambda_j}\cdot\lambda_j}{\Lambda}. \quad (7)$$

To determine the best doping points for a fixed number of sub-frames l, (7) can be treated as an optimization problem with respect to the λ_j parameters. Here we consider only the important special case that the doped CNs are spaced uniformly in the coupling chain (that is $\lambda_j=\lambda$, $j\in[0,1,\ldots,l-1]$), in which case (7) can be written as

$$P_{\text{BL,doped}} = P_{\text{BL},\lambda_j}|_{\lambda_j=\lambda}, \quad j=0,1,\ldots,l-1,$$
$$= \sum_{\tau=1}^{\lambda} P_{\text{BL},\lambda}(\tau)\cdot q(1-q)^{\tau-1} + p(1-q)^{\lambda}. \quad (8)$$

Example 3: Assume $q=10^{-4}$, $p=10^{-2}$, and $\Lambda=1000$.

Case 1: For the original (3,6)-regular SC-LDPC codes, we obtain $P_{BL}=5.7939\cdot10^{-2}$ (same as case 2 in Example 1).

Case 2: If one doped CN is spaced in the middle of the chain, i.e., $\lambda_0=\lambda_1=\lambda=500$, then according to (8) we obtain $P_{\text{BL,doped}}=3.4391\cdot10^{-2}$.

Case 3: If three doped CNs are spaced uniformly, i.e., $\lambda_0=\lambda_1=\lambda_2=\lambda_3=\lambda=250$, then according to (8) we obtain $P_{\text{BL,doped}}=2.2321\cdot10^{-2}$. ■

From the example, we see that the CN doped (3,6)-regular SC-LDPC code has improved BLER performance compared to the original (3,6)-regular SC-LDPC code, at a cost of a slight rate loss due to the additional CNs, and that adding more doping points further improves the BLER performance while increasing the rate loss.

C. Rate Loss

Let n_c and n_v denote the total number of CNs and the total number of VNs in CN doped SC-LDPC codes, respectively. The design rate of the CN doped SC-LDPC codes with frame length Λ and γ doped CNs is given as

$$R_{\Lambda,\text{doped}} = 1-\frac{n_c}{n_v} = 1-\left(\frac{\Lambda+m}{\Lambda}\right)(1-R)-\frac{\gamma}{\Lambda\eta_v}, \quad (9)$$

where $R=1-\eta_c/\eta_v$ is the design rate of the uncoupled protograph [7].

Compared to the design rate $R_\Lambda = 1-\left(\frac{\Lambda+m}{\Lambda}\right)(1-R)$ of the original SC-LDPC codes [7], we see from (9) that the design rate of CN doped SC-LDPC codes is smaller, i.e., CN doping results in some *rate loss*. However, we note that *full termination* at the doping points, which also truncates error propagation, results in a larger rate loss. Below, the design rates of the three cases considered in Example 3 are calculated.
Case 1: $\gamma=0$, $R_{\Lambda=1000}=1-\left(\frac{1000+2}{1000}\right)(1-0.5)=0.499$.
Case 2: $\gamma=1$, $R_{\Lambda=1000,\text{doped}}=1-\left(\frac{1000+2}{1000}\right)(1-0.5)-\frac{1}{1000\times2}=0.4985$. For full termination, $R_{\Lambda=500}=1-\left(\frac{500+2}{500}\right)(1-0.5)=0.498$.
Case 3: $\gamma=3$, $R_{\Lambda=1000,\text{doped}}=1-\left(\frac{1000+2}{1000}\right)(1-0.5)-\frac{3}{1000\times2}=0.4975$. For full termination, $R_{\Lambda=250}=1-\left(\frac{250+2}{250}\right)(1-0.5)=0.496$.
From these results, we note that, even though CN doping results in some rate loss, the rate loss of full termination is greater.

IV. Numerical Results

In order to verify the effectiveness of the new code design, the bit error rate (BER) distribution per block of a typical error-propagation frame in SWD of both CN doped (Fig. 4) and standard (Fig. 1(b)) (3,6)-regular SC-LDPC codes sent over an AWGN channel with BPSK signaling is plotted in Fig. 6, where $M=2000$, $L=250$, and $W=18$. Two examples of CN doping are included, one with a single doping point at time $t=125$, and one with two doping points at $t=83$ and 166. From the figure, we can clearly see that CN doping effectively truncates the error propagation and that adding more doped check nodes truncates the error propagation earlier.

[8] We note here that the time scales differ for VNs and CNs, since the CN doping interrupts the regular pattern of exactly one CN for every two VNs.

[9] This assumption is supported by extensive simulations showing that CN doping is effective in limiting error propagation, as will be demonstrated in the next section.

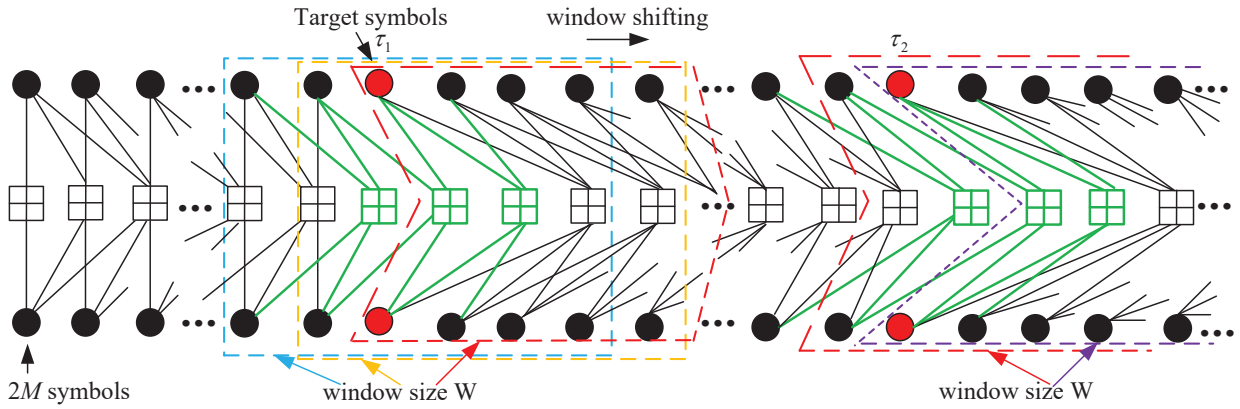
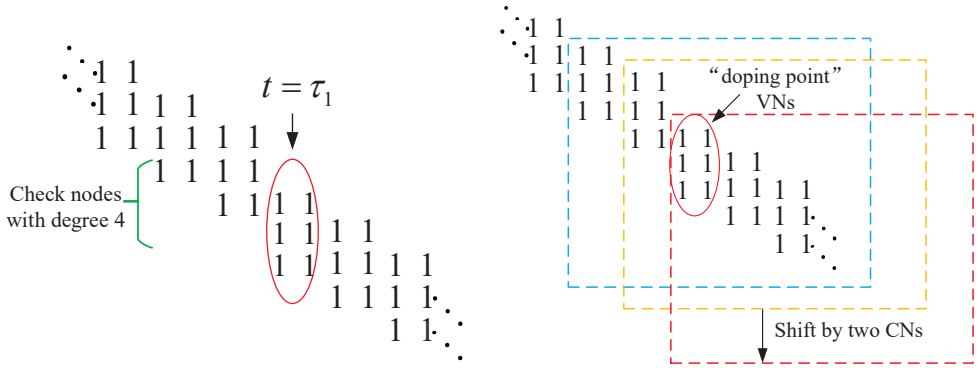


Fig. 4. The protograph and SWD schedule for a CN doped (3,6)-regular SC-LDPC code with occasional structured irregularities (CNs of reduced degree) spaced throughout the chain.



(a) Construction (b) SWD schedule

Fig. 5. The base matrix view of the construction and SWD schedule for CN doped (3,6)-regular SC-LDPC codes.

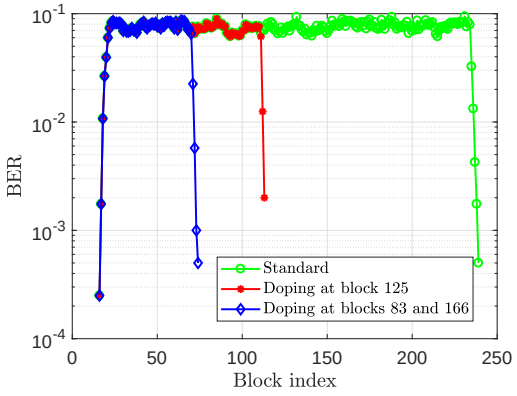


Fig. 6. Bit error rate distribution per block.

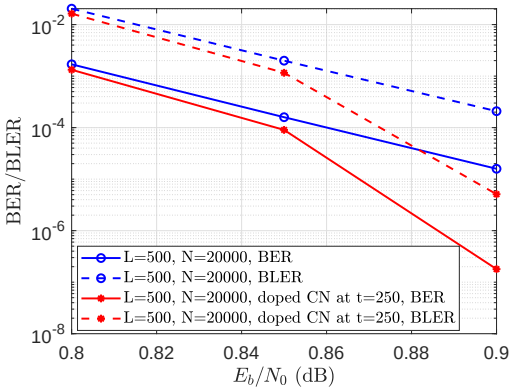


Fig. 7. Performance comparison of doped and undoped codes.

Next, for a termination length $L = 500$ and window size $W = 18$, Fig. 7 shows the BER and BLER performance comparison between the standard (3,6)-regular SC-LDPC code with rate $R = 0.498$ and a CN doped (3,6)-regular SC-LDPC code with rate $R = 0.497$. We note that the CN doped code gains up to two orders of magnitude in BER and more one order of magnitude in BLER compared to the standard code at SNR operating points of interest.

V. Conclusion

In this paper, we presented the concept of check node doped (J, K)-regular SC-LDPC codes to limit the effects of error propagation in low latency SWD for large frame length or streaming applications. The novel code design takes advantage of a structured irregularity in the code graph introduced by occasionally adding check nodes (doping) that allow SWD to recover from error propagation. A simplified analysis of the BLER indicates that improved performance is obtained by increasing the number of doping points, at a cost of some slight rate loss, and simulation results were used to further illustrate the beneficial effect of the new code design.

Acknowledgment

This work was supported in part by NSFC under Grant 61701368 and by the National Science Foundation under Grant Nos. ECCS-1710920 and OIA-1757207.

References

[1] A. J. Felström, and K. Sh. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2181–2191, Sep. 1999.

[2] M. Lentmaier, A. Sridharan, D. J. Costello, Jr., and K. Sh. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.

[3] 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. Inf. Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.

[4] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 7761–7813, Dec. 2013.

[5] S. Kumar, A. J. Young, N. Macris, and H. D. Pfister, "A proof of threshold saturation for spatially-coupled LDPC codes on BMS channels," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, Oct. 2012, pp. 176–184.

[6] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," *Jet Propuls. Lab.*, Pasadena, CA, USA, INP Prog. Rep. 42-154, Aug. 2003.

[7] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, "Spatially coupled LDPC codes constructed from protographs," *IEEE Trans. Inf. Theory*, vol. 61, no. 9, pp. 4866–4889, Sep. 2015.

[8] A. R. Iyengar, M. Papaleo, P. H. Siegel, J. K. Wolf, A. Vanelli-Coralli, and G. E. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2303–2320, Apr. 2012.

[9] K. Huang, D. G. M. Mitchell, L. Wei, X. Ma, and D. J. Costello, "Performance comparison of LDPC block and spatially coupled codes over GF (q)," *IEEE Transactions on Communications*, vol. 63, no. 3, pp. 592-604, Mar. 2015.

[10] K. Klaiber, S. Cammerer, L. Schmalen, and S. ten Brink, "Avoiding burst-like error patterns in windowed decoding of spatially coupled LDPC codes," in *Proc. IEEE 10th Int. Symp. on Turbo Codes & Iterative Inf. Processing (ISTC)*, Hong Kong, China, 2018, pp. 1-5.

[11] W. Zhang, M. Lentmaier, K. Sh. Zigangirov, and D. J. Costello, Jr., "Braided convolutional codes: a new class of turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 316-331, Jan. 2010.

[12] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, Jr., and B. Bai, "Braided convolutional codes with sliding window decoding," *IEEE Trans. on Communications*, vol. 65, no. 9, pp. 3645-3658, Sept. 2017.

[13] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, Jr., and B. Bai, "Combating error propagation in window decoding of braided convolutional codes," in *Proc. IEEE Int. Symp. Information Theory*, Vail, CO, USA, June 17-22, 2018, pp. 1380-1384.

[14] S. ten Brink, "Code doping for triggering iterative decoding convergence," in *Proc. IEEE Int. Symp. Information Theory*, Washington, DC, USA, June 24-29, 2001, pp. 235.