Scaling design parameters on the overall performance of Spatially-Coupled LDPC codes

Alexandre Felipe, André L.N. Souza, Juliano R.F. Oliveira and Jacklyn D. Reis

***Abstract*— This work evaluates the effects of scaling the syndrome former memory, lifting factor and window size on the performance of spatially-coupled LDPC codes generated from protographs decoded using a sliding window. The results give a good insight on how these design parameters alter the trade-off between computational complexity and error-correcting capacity.**

***Keywords*— Forward Error Correction, Spatially-Coupled Low-density Parity-Check Code, Parity-check codes.**

I. Introduction

The role of forward error correction (FEC) has become of critical importance in fiber optic communications. More specifically, soft-decision based FEC is considered an important means to improve the reach performance of beyond 400 Gbit/s coherent communication system. Low-Density Parity-Check block codes (LDPC-BC) were proposed by Gallager in 1962 [1]. At the time, their potential remained undiscovered due to the lack of computational power to perform simulations. At early 1990s, the first class of codes having bit error rates performance near Shannon Limit, called Turbo Codes, was proposed[2]. After that, the advances of iterative decoding algorithms [3], [4], [5], now referred to as Belief Propagation (BP) algorithms, revived interest in Gallager's codes. In 2001, a rate-$1/2$ irregular LDPC with frame length of 10^7 achieved a bit error rate of 10^{-6}, at 0.04 dB of the Shannon Limit for that rate [6]. In the last years, by comparing ASIC implementations of Turbo decoders [7] and LDPC decoders [8], it is evident that LDPC is the most promising soft-decision based FEC for high-capacity coherent optical transmission systems beyond 400 Gbit/s.

The convolutional correspondent of LDPC block codes were first proposed in [9]. LDPC convolutional codes are also defined by sparse parity-check matrices, which allow them to be decoded using the same iterative message-passing algorithms as LDPC-BC. Convolutional LDPC codes with small syndrome former memory and large sub-matrices are often referred to as spatially-coupled LDPC (SC-LDPC). Spatial coupling is a very general concept and can be applied to virtually any code with both hard and soft decision. SC-LDPC codes have attracted significant attention because of their good characteristics (thresholds approaching capacity with BP decoding and low error floors). They have been successfully applied in many areas of communications and signal processing, such as, for example, relay [10], multiple access [11] and broadcast channels [12], channels with memory [13] and is getting increasing attention on optical communication. LDPC convolutional codes have been shown to be suitable for practical implementation in many different scenarios, including continuous transmission using left-terminated parity check matrices, and block transmission using terminated parity-check matrices, respectively [14]. They are also known for their encoding simplicity, since code construction methods yield a shift-register based systematic encoder for real time encoding of continuous data. The structure of SC-LDPC parity-check matrix allow the use of a hardware efficient windowed decoding scheme [15] and is also a good candidate for developing rate adaptive codes for future elastic optical networks [16], [17].

LDPC block and spatially-coupled codes are known to achieve capacity when the codeword is sufficiently large. It is therefore natural to compare LDPC block codes with its convolutional counterpart. In practice, however, the intricate relation between performance, computational complexity and latency has to be considered. Comparisons between block and spatially coupled codes were performed in terms of computational and hardware complexity, decoding delay, memory and very-large-scale integration (VLSI) implementation requirements in [18], [19] and showed that SC-LDPC codes perform better than their block counterparts in various scenarios. Studies of the finite length scaling properties of SC-LDPC over the binary erasure channel were performed in [20], [21].

Focusing on the implementation of SC-LDPC codes, this work is based on the study and discussion of the effects of some construction parameters (syndrome former memory, lifting factor and window size) on the overall performance, decoding latency and computational complexity of regular left-terminated spatially-coupled LDPC codes over AWGN channel.

The rest of the paper is organized as follows. Section II presents the mathematical notations used throughout this work and the concept of LDPC codes as an introduction to SC-LDPC codes presented in sec. III. Section IV depict the relation between the design parameters and computational complexity. Lastly, sec. V describes the simulations present the comparison results between different constructions of left-terminated SC-LDPC codes. Section VI concludes the paper.

II. Low-Density Parity Check Codes

A binary code is a subset of $\{0, 1\}^n$ and is classified as linear if it forms a vector space in $GF(2)$, thus a linear code can be described by a set of parity check equations which in

Alexandre Felipe, André L.N. Souza, Juliano R.F. Oliveira and Jacklyn D. Reis are with CPqD, Division of Optical Technologies, Campinas - SP, 13086-902, Brazil, E-mails: afelipe, aluizs, jrfo and jacklyn@cpqd.com.br.

turn can be represented as the nullspace of a matrix in $GF(2)$, called Parity-Check Matrix, i.e for any codeword v, $H \cdot v^T$ is satisfied. Low-Density Parity Check codes proposed by Robert G. Gallager in [1] have very sparse parity check matrices H. For regular codes with rate $R = 1 - d_v/d_c$ described by a full rank parity-check matrix, each column of ($\mathbf{H}$) contains exactly d_v non-zero entries and each row contains d_c non-zero entries, and they are called column and row weights (or variable and check node degrees). For performance improvement, the weight of each column can be optimized for a specific system and the code becomes irregular [6], and this is not covered in this work.

Another way to represent a linear code is by a Tanner graph. It is a bipartite in which parity equations are represented by check nodes, input bits (or matrix columns) are represented by variable nodes, and each nonzero term of the parity check matrix is represented by an edge connecting the corresponding variable node and corresponding check node [22].

To better understand the code structure, we present a toy example: consider an LDPC-BC with codewords (v) that contain $K = 5$ information and $N - K = 3$ parity bits leading to a code rate $R = K/N = 5/8$ and an overhead (OH) of $1/R - 1$. The bits in each block are submitted to the following restrictions:

$$\begin{aligned} u^{(1)} \oplus u^{(3)} \oplus u^{(5)} \oplus p^{(1)} = 0 \\ u^{(2)} \oplus u^{(4)} \oplus u^{(5)} \oplus p^{(2)} = 0 \\ u^{(1)} \oplus u^{(2)} \oplus u^{(4)} \oplus p^{(3)} = 0 \end{aligned} \tag{1}$$

where $\mathbf{u} = \left[u^{(1)}, u^{(2)}, ..., u^{(5)}\right]$ are information and $\mathbf{p} = \left[p^{(1)}, p^{(2)}, p^{(3)}\right]$ are check bits. As stated above, these restrictions can be represented by a matrix $\mathbf{H}$ such that $\mathbf{v} \cdot \mathbf{H^T} = 0$, where $\mathbf{v} = [\mathbf{u}|\mathbf{p}]$ and $\mathbf{H^T}$ is given by (2).

$$\mathbf{H^T} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

1) LDPC codes generated from protographs: Increasing the block length while maintaining the code structure (rate, node and variable degrees) improves the performance of LDPC codes [1]. A good way of constructing large, structured parity-check matrices is by applying a graph lifting operation to *protographs* [23] as follows.

A protograph with design rate $R = 1 - d_v/d_c$ is a small parity-check matrix. The Tanner graph representing a protograph-based LDPC-BC is obtained by $\mathcal{M}$-lifting the protograph. Graph lifting corresponds to replacing each non-zero entry in a protograph (B) by a sum of $\prod_{ij}$ random permutation matrices of size $\mathcal{M} \times \mathcal{M}$ and each zero entry by the $\mathcal{M} \times \mathcal{M}$ all zero matrix. The resulting parity-check matrix H is $\mathcal{M}$ times larger than the protograph and has the same rate, degree distribution, and computation graphs as the protograph. The process of $\mathcal{M}$-lifting a ($d_v = 6$, $d_c = 3$) protograph $\mathbf{B}$ in (3) results in the matrix shown on (4).

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{3}$$

$$\mathbf{H^T} = \begin{pmatrix} \prod_{11} & \prod_{12} & \prod_{13} & \prod_{14} & \prod_{15} & \prod_{16} \\ \prod_{21} & \prod_{22} & \prod_{23} & \prod_{24} & \prod_{25} & \prod_{26} \\ \prod_{31} & \prod_{32} & \prod_{33} & \prod_{34} & \prod_{35} & \prod_{36} \end{pmatrix} \tag{4}$$

III. Spatially Coupled Low-Density Parity Check Codes

Increasing the codeword size of a LDPC block code tends to improve the error correction performance. By the other hand it increases the complexity of ASIC implementation, in a fully parallel architecture, the decoder area tends to increase linearly with the number of nodes of the tanner graph, and the connecting each of these nodes is difficult task. The encoder tends is straight forward since it not required to deal with probabilities iteratively, the main limitation to LDPC block encoding is that the parity check matrix must have some structure in order to have linear complexity. One emerging solution to address these limitations are SC-LDPC codes. Instead of transmitting independent sequential blocks, spatially coupling creates a virtually infinite codeword by adding a dependence between a block and its neighbors.

2) Construction of SC-LDPC codes generated from protographs: A rate $R = K/N$ binary, left-terminated SC-LDPC code with syndrome former memory m_s can be constructed from a semi-infinite parity-check matrix $\mathbf{H_{SC}}$ such that $\mathbf{v_{SC}} \cdot \mathbf{H^T_{SC}} = 0$, where this time $\mathbf{v_{SC}} = [\mathbf{v_0}, \mathbf{v_1}, \mathbf{v_2}, \ldots]$ is the concatenation of codewords $\mathbf{v_t}$ in time.

$$\mathbf{H_{SC}} = \begin{pmatrix} \mathbf{H_0} & 0 & \cdots & & \\ \mathbf{H_1} & \mathbf{H_0} & \ddots & & \\ \vdots & \mathbf{H_1} & \mathbf{H_0} & & \\ \mathbf{H_{m_s}} & \vdots & \mathbf{H_1} & \ddots & \\ 0 & \mathbf{H_{m_s}} & \vdots & \ddots & \\ \vdots & \ddots & \mathbf{H_{m_s}} & \ddots & \\ & \ddots & & \ddots & \end{pmatrix} \tag{5}$$

where $dim(\mathbf{H_n}) = M \times N$ are sparse binary parity-check matrices generated from single- line protographs with $d_v = \lceil 1/OH + 1 \rceil$ columns. Specific code rates can be achieved via puncturing or shortening the parity-check matrix [16], [17].

3) Windowed decoding of SC-LDPC codes: The structure of spatially coupled codes imposes a constraint on the variable nodes connected to the same parity-check equations, i.e. the staircase pattern of the matrix leads to the fact that each parity equation depends on a portion of the transmitted sequence of length $N \cdot (m_s + 1)$.

Thus, it is possible to derive a local decoding algorithm, that in each iteration decides a block of N consecutive bits. The information of every decided bits are updated for posterior window steps. Gallager already noticed that using log-likelihood ratios (LLR) would be more convenient for decoding LDPC codes [1]. An advantage of using LLR instead of probabilities is that the belief propagation can be implemented using the so called Min-Sum algorithm [5], more suitable for hardware implementation since it does not require multiplications or large lookup tables [24], [25].

Algorithm 1 is used for decoding SC-LDPC codes using a LDPC block decoder. The matrix H_{BC} is not left-terminated, but the same result is achieved by adding zeros before the codeword in line 3. Each iteration represents a window step, the bits already decoded are left-shifted in line 7 and line 9 put a copy of the received symbols LLRs in the rest of vector S. Line 12 uses a black box *soft-input soft-output* LDPC block decoder, where H_{BC} is a $W \cdot N$ submatrix of H_{SC}, and includes the rows of H_{SC} having no ones out of that columns and n_{it} is the maximum number of iterations performed by the LDPC block decoder. Finally, each output bit is decided based on the probability estimated by the previous decoding steps.

Algorithm 1 SC-LDPC Decoding algorithm

1: **procedure** DecodeLDPCSC
2: $\quad LLR_i = log\left(\frac{P(y_i|x_i=0)}{P(y_i|x_i=1)}\right) \forall i \in \mathrm{N}$
3: $\quad S \leftarrow \mathbf{0}$ ▷ Emulate left termination
4: $\quad$ **for** $i \leftarrow 1$ **to** ∞ **do**
5: $\quad\quad$ **for** $j \leftarrow 0$ **to** $W \cdot M$ **do**
6: $\quad\quad\quad$ **if** $j < M \cdot m_s$ **then**
7: $\quad\quad\quad\quad S_j \leftarrow S_{j+M}$
8: $\quad\quad\quad$ **else**
9: $\quad\quad\quad\quad S_j \leftarrow LLR_{i \cdot M+(j-M \cdot m_s)}$
10: $\quad\quad\quad$ **end if**
11: $\quad\quad$ **end for**
12: $\quad\quad S = DecodeLDPC(S, H_{BC}, n_{it})$
13: $\quad\quad$ **for** $j \leftarrow 0$ **to** M **do**
14: $\quad\quad\quad$ **if** $S_j < 0$ **then**
15: $\quad\quad\quad\quad$ **output**$_{M \cdot i+j} \leftarrow 1$
16: $\quad\quad\quad$ **else**
17: $\quad\quad\quad\quad$ **output**$_{M \cdot i+j} \leftarrow 0$
18: $\quad\quad\quad$ **end if**
19: $\quad\quad$ **end for**
20: $\quad$ **end for**
21: **end procedure**

IV. Analysis

By analyzing the methodology described in III to construct regular left-terminated SC-LDPC codes, the crucial design parameters are the syndrome former memory m_s, the lifting factor M and the decoding window size W. The encoding complexity per bit is only affected by the rate and m_s. The decoder complexity depends on the number of ones in the parity check matrix. By inspecting H_{BC} in Alg. 1, it has $(N - m_s) \cdot (m_s + 1)$ ones. The decoding latency is the time required to receive all bits of the decoding window, $W \cdot N$ bits, plus the decoding time.

4) Effect of windowed decoding: Another concern is about what information is lost when using a small window for decoding. Let d_g be the distance in the tanner graph between two variable nodes, and d_t the distance of the corresponding pair of bits in the transmission sequence. From the structure

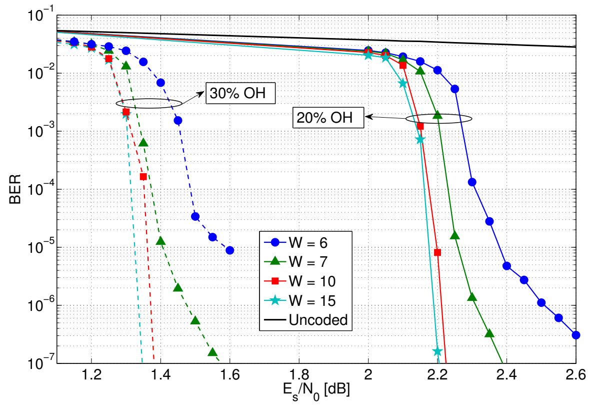

Fig. 1. Post-FEC BER *versus* SNR for N equal to 30000 and variable window size.

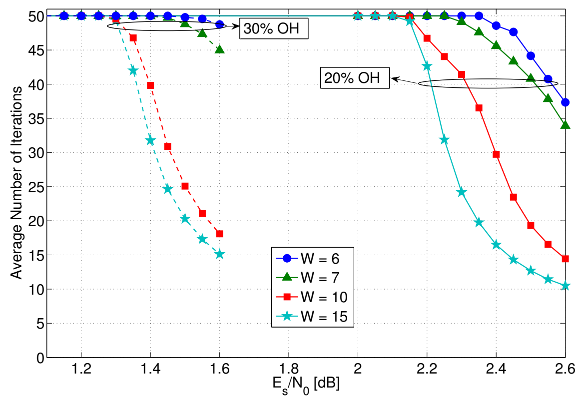

Fig. 2. Average number of decoding iterations per window step *versus* SNR for N equal to 30000 and variable window size.

of H_{SC}, we have the following relations:

$$\lfloor l_t/N \rfloor > m_s \Rightarrow l_g = 2 \tag{6}$$

$$\lfloor l_t/N \rfloor > m_s \Rightarrow l_g > 2 \tag{7}$$

Notice that the distance between two variable nodes is an even number, since every edge in the graph connects a variable node to a check node. The decoding window includes $N \cdot m_s$ already decoded bits, N to be decoded in the current window step, and $(W - m_s - 1) \cdot N$ not required to be corrected in the current window step. Thus the separation between a bit of interest and an ignored bit is more than $l_t > (W - m_s - 1) \cdot N$. Relation (8) is obtained by induction using (6) and (7). It represents the distance between any bit of interest and any bit ignored at a window step.

$$l_g \leq 2 \cdot \lfloor (W - m_s - 1)/m_s \rfloor \tag{8}$$

V. Results

We performed simulations in order to give an insight on how m_s, M and W alter the overall performance, latency and computational complexity of the resulting code. A BPSK modulated signal is transmitted over an AWGN channel, and iterative decoding is performed using the Min-Sum algorithm

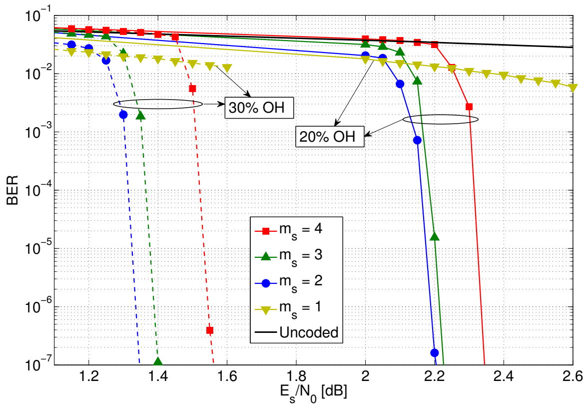

Fig. 3. Post-FEC BER *versus* SNR for N equal to 30000, window size equal to 15 and variable syndrome former memory.

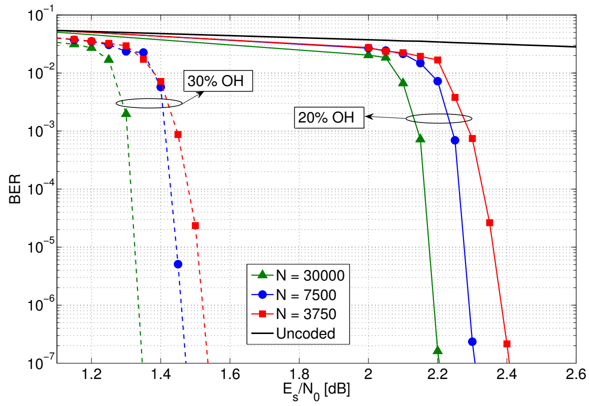

Fig. 5. Post-FEC BER *versus* SNR for syndrome former memory equal to 2, window size equal to 15 and variable N.

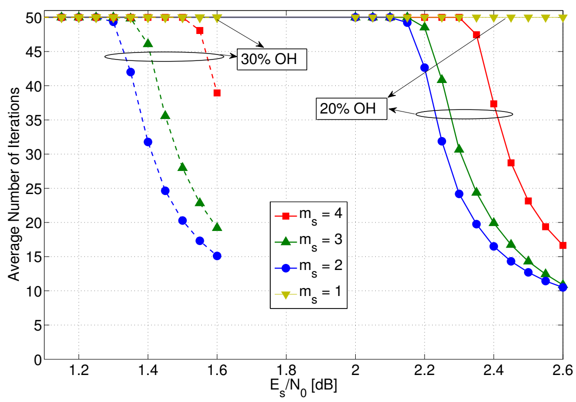

Fig. 4. Average number of decoding iterations per window step *versus* SNR for N equal to 30000, window size equal to 15 and variable syndrome former memory.

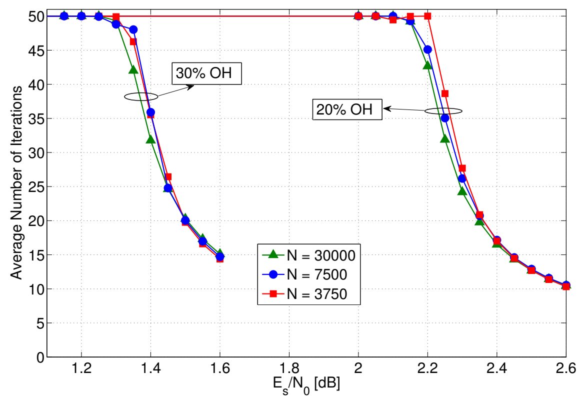

Fig. 6. Average number of decoding iterations per window step *versus* SNR for syndrome former memory equal to 2, window size equal to 15 and variable N.

[5]. The decoder finishes one window step after 50 iterations or when all the parity equations involving the bits of interest are satisfied. The default LDPC-SC code is constructed with a lifting factor that results in $N = 30000$, syndrome former memory $m_s = 2$, and window size $W = 15$, and at each simulation scenario one of the parameters M, m_s, W was varied. Bit error rate (BER) and the average number of iterations executed by the decoder at each window step are presented for codes with 20% (solid lines) and 30% (dashed lines) overheads.

5) Variable Window Size: The window size W does not affect the parity-check matrix defined in (5), and is a clear example of how the decoding of LDPC-SC can be performed on a flexible way. Figs 1 and 2 present, respectively, the post-FEC BER and average number of iterations when decoding with different window sizes. High error floors are observed for both code rates when decoded with small window size. Reducing W from 15 to 7 appears to have an impact similar to reducing from 7 to 6, as can be seen from the inequality (8), a decoder using $W = 7$ and $m_s = 2$ ensures that every variable node at distance $l_t \leq 4$ in the tanner graph is taken into account, while $W = 6$ do not ensure that. Another noticeable result is that, $W = 10$ and $W = 15$ have almost the same waterfall behavior. However, observing the required iteration count in Fig. 2 at 2.4 dB for 20% overhead code, the average number of iterations for $W = 10$ is almost twice as large as the number of iterations required when decoding for $W = 15$, meaning that less computational effort is required by decoding with $W = 15$ than $W = 10$. On the other hand, using $W = 15$ would result in larger latency, meaning that a system employing SC-LDPC can vary the window size according to the required number of iterations in order to save power.

6) Variable Syndrome Former Memory: The decoding complexity of the constructed code is tightly connected to the syndrome former memory, since it determines the number of ones the Parity-Check Matrix has by column. Figure 4 shows the average number of iterations executed by the underlying LDPC block decoder, that is directly related to the distance of the waterfall regions in these cases. Figure 3 shows the post-FEC BER with different syndrome former memory values. We see that increasing m_s does not necessarily improve the performance of this coding scheme. Syndrome former memory equal to one leads to a very poor performance because bit participates only in two parity check equations. The curves for $m_s = 2, 3, 4$ have the same waterfall behavior, but the lowest threshold at $BER = 10^{-7}$ is achieved with $m_s = 2$ and the difference between $m_s = 2$ and $m_s = 4$ is higher than 0.15 dB. This behavior again can be explained by the inequality (8) that indicates how far are the bits ignored from the bits decided. This shows that increasing the lifting factor in order to achieve better results is a costly option, since in addition to increasing the parity-check matrix density it requires increasing the decoding window length.

7) Variable Lifting Factor: Both dimensions of parity-check sub-matrices $\mathbf{H_n}$ are directly proportional to the lifting factor. Increasing M also increases the decoding latency and complexity of a decoding iteration. Figures 5 and 6 shows the bit error rate and average number of iterations executed by the decoding algorithm per window step, respectively. It is noticeable that reducing N by factor of eight results in a coding gain difference of 0.2 dB at $BER = 10^{-7}$, however the number of required iterations to achieve such BER is near to 50 for $N = 30000$, and reduces to 15 for $N = 3750$. If reducing the coding gain is allowed, lowering the lifting factor is a feasible implementation, since it does not produce high error floors, and simultaneously reduces code latency, decoder iteration complexity, and number of iterations required to achieve the same post-FEC BER.

VI. CONCLUSION

We performed simulations to asses the effects of scaling the syndrome former memory, window size and lifting factor on the performance of spatially coupled LDPC codes with 20% and 30% overheads over the AWGN channel. From the results we can observe how the parameters of SC-LDPC codes affect the post-FEC BER, obtained using a particular code construction and decoding algorithm.

We conclude that the same code (and thus the same encoder) provides some flexibility concerning the decoder window size that can be exploited by flexible decoders to reduce power consumption by dynamically selecting the optimal window size.

Among the design parameters it is preferable to optimize the lifting factor and keep the syndrome former memory low, e.g. $m_s = 2$. Higher values of lifting factor increase the complexity (power consumption) for both encoding and decoding. In Addition, it requires a larger window size in order to ensure that the same neighborhood of the variable nodes of interest in the tanner graph is considered by the decoder, increasing the number of terms participating in each parity check equation. Higher values of lifting factor M increases both the encoder and decoder complexity linearly, because it preserves the column and row degrees. Reducing the window size leads to higher error floors, requiring the concatenation with a more complex hard-decision FEC scheme to achieve error-free operation.

ACKNOWLEDGMENTS

This work was supported by the Brazilian Ministry of Communications under FUNTTEL/FINEP.

REFERENCES

[1] Robert G Gallager. Low-density parity-check codes. *IRE T. Inform. Theor.*, 8(1):21–28, 1962.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Proc. of ICC '93*, volume 2, pages 1064–1070 vol.2, May 1993.

[3] D. J. C. MacKay and R. M. Neal. Near shannon limit performance of low density parity check codes. *Electron. Lett.*, 32(18):1645–, Aug 1996.

[4] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE T. Inform. Theory*, 42(2):429–445, Mar 1996.

[5] M. P. C. Fossorier, M. Mihaljevic, and H. Imai. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *IEEE T. Commun.*, 47(5):673–680, May 1999.

[6] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE T. Inform. Theory*, 47(2):619–637, Feb 2001.

[7] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang. Design and implementation of a parallel turbo-decoder ASIC for 3GPP-LTE. *IEEE J. Solid-St. Circ.*, 46(1):8–17, Jan 2011.

[8] N. Onizawa, T. Hanyu, and V. C. Gaudet. Design of high-throughput fully parallel LDPC decoders based on wire partitioning. *IEEE T. VLSI Syst.*, 18(3):482–489, March 2010.

[9] A. Jimenez Felstrom and K. S. Zigangirov. Time-varying periodic convolutional codes with low-density parity-check matrix. *IEEE T. Inform. Theory*, 45(6):2181–2191, Sep 1999.

[10] Zhongwei Si, R. Thobaben, and M. Skoglund. Bilayer LDPC convolutional codes for half-duplex relay channels. In *Proc. of ISIT'11*, pages 1464–1468, July 2011.

[11] Shrinivas Kudekar and Kenta Kasai. Spatially coupled codes over the multiple access channel. In *Proc. of ISIT'11*, pages 2816–2820. IEEE, 2011.

[12] Z. Si, R. Thobaben, M. Skoglund, and T. J. Oechtering. Bidirectional broadcasting by using multi-edge type LDPC convolutional codes. In *Proc. of ISTC'12*, pages 91–95, Aug 2012.

[13] Shrinivas Kudekar and Kenta Kasai. Threshold saturation on channels with memory via spatial coupling. In *Proc. of ISIT'11*, pages 2562–2566. IEEE, 2011.

[14] A. Leven and L. Schmalen. Status and recent advances on forward error correction technologies for lightwave systems. *J. Lightwave Technol.*, 32(16):2735–2750, Aug 2014.

[15] 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 T. Inform. Theory*, 58(4):2303–2320, April 2012.

[16] Yequn Zhang and Ivan B. Djordjevic. Staircase rate-adaptive LDPC-Coded modulation for high-speed intelligent optical transmission. In *Proc. of OFC'14*, page M3A.6. Optical Society of America, 2014.

[17] Tao Tian and Christopher R. Jones. Construction of rate-compatible LDPC codes utilizing information shortening and parity puncturing. *EURASIP J. Wirel. Comm.*, 2005(5):1–7, 2005.

[18] Daniel J. Costello, Ali Emre Pusane, Stephen Bates, and Kamil Sh. Zigangirov. A comparison between LDPC block and convolutional codes. In *Proc. Inform. Theory and App. Workshop*, 2006.

[19] N. ul Hassan, M. Lentmaier, and G. P. Fettweis. Comparison of LDPC block and LDPC convolutional codes based on their decoding latency. In *Proc. of ISTC'12*, pages 225–229, Aug 2012.

[20] P. M. Olmos and R. Urbanke. Scaling behavior of convolutional LDPC ensembles over the BEC. In *Proc. of ISIT'11*, pages 1816–1820, July 2011.

[21] P. M. Olmos and R. Urbanke. A closed-form scaling law for convolutional LDPC codes over the BEC. In *Proc. of ITW'13*, pages 1–5, Sept 2013.

[22] R. Tanner. A recursive approach to low complexity codes. *IEEE T. Inform. Theory*, 27(5):533–547, Sep 1981.

[23] J Thorpe. Low-density parity-check (LDPC) codes constructed from protographs. *Interplanetary Network Progress Report*, 154:1, 2003.

[24] E. Liao, Engling Yeo, and B. Nikolic. Low-density parity-check code constructions for hardware implementation. In *Proc. of ICC '04*, volume 5, pages 2573–2577 Vol.5, June 2004.

[25] H. Zhong, W. Xu, N. Xie, and T. Zhang. Area-efficient min-sum decoder design for high-rate quasi-cyclic low-density parity-check codes in magnetic recording. *IEEE T. Magn.*, 43(12):4117–4122, Dec 2007.