Two is Better than One: Reducing the Loss of the Window Decoder for SC-LDPC Codes

Alberto Tarable*, Marco Ferrari† and Luca Barletta‡
*CNR-IEIIT, Torino, Italy. Email: alberto.tarable@ieiit.cnr.it
†CNR-IEIIT, Milano, Italy. Email: marco.ferrari@ieiit.cnr.it
‡Politecnico di Milano, Milano, Italy. Email: luca.barletta@polimi.it

***Abstract*—In this paper, we consider spatially coupled LDPC codes derived from protographs. In particular, we analyze the performance of the window decoder (WD), which allows reducing the complexity, the memory requirements, and the latency of the flood belief-propagation decoder. We show that the performance degradation of WD is due to the fact that it exploits a single decoding wave instead of two. This has effect both in the ideal case of infinite code length, where it may imply a threshold loss, and in the case of finite length, where it affects the slope of the BER curve in the waterfall region. We show how a forward-backward decoder can reduce such problems at the price of a limited increase of average complexity.**

I. Introduction

In the last few years, spatially coupled LDPC (SC-LDPC) codes have been generally regarded by the scientific community as one of the most promising coding techniques, from several points of view. First, [1]-[2] have provided the proof of threshold saturation, thus showing that, under mild hypotheses, SC-LDPC codes are theoretically capacity-achieving. Second, the same analysis has actually revealed that the code design is greatly simplified with respect to block LDPC codes, as it is enough to consider regular ensembles, and no degree distribution optimization is anymore needed. Third, the introduction of the window decoder (WD) in [3], has paved the way for a relatively low-complexity decoder implementation, whose latency and memory requirements are independent of the block size, which is typically large for a high-performance SC-LDPC code.

In this paper, we plan to partially correct the picture, and to show that the three pros cited above cannot easily be achieved all together. In particular, our goal is to show that, at infinite as well as at finite length, the WD pays a performance penalty with respect to the flood belief-propagation (BP) decoder.[1]

- At infinite length, it is well known that the WD has a worse threshold than the flood BP decoder [3]. While for symmetric edge spreading, it is proved in [4] that, on the binary erasure channel (BEC), the WD threshold converges to the BP threshold exponentially fast in the window size W, we show that, for asymmetric SC-LDPC codes, this convergence is not guaranteed, a fact that is confirmed in the context of codes on GF(q) by [5]. We also give a simple justification of this lack of convergence.
- At finite length, more importantly, the performance loss takes the shape of a slope reduction in the waterfall region of the BER curve, that spoils under WD part of the *convolutional gain* [6] expected from SC-LDPC codes.

In both cases, we identify possible remedies to improve the WD performance. In particular, for the finite-length case, we introduce a new type of decoder, the forward-backward decoder, which performs, if needed, a backward sweep analogous to the forward sweep of the WD, in three variants, which correspond to different trade-off between complexity/memory requirements and performance.

In what follows, we will focus on protograph-based SC-LDPC codes. While our results do extend to other types of schemes, protograph-based SC-LDPC codes have become the standard *de facto* of SC-LDPC design ([7]), thus justifying the attention we reserve them. It is worth noting, however, that most of the theoretical results have been obtained for SC-LDPC code schemes that are not, strictly speaking, protograph-based.

A. Related work

The WD for SC-LDPC codes was introduced in [3] specifically to reduce the decoding latency in delay-constrained applications. Its most prominent precursor is the pipeline decoder, an efficient solution for a parallel implementation that was introduced in [8] and simplified in [9]. However, the pipeline decoder has a latency growing linearly with the number of iterations.

The analysis of the WD performance on the erasure channel for infinite length can be found in [3] and [7], where the WD threshold was defined. For the finite-length performance of WD, to our knowledge, there is a lack of analytic studies. In [10], the WD is studied together with other scheduling approaches from the complexity point-of-view.

Even the phenomenon of error propagation has received little attention. In [11], error propagation is faced as follows: for those variables that have already been processed in previous steps, the input at the current step is a convex combination of the raw channel output and of the previously computed a-posteriori information in the decoder output buffer. For the similar class of braided convolutional codes, [12] proposed a combination of techniques such as window extension and

[1] In the jargon of the SC-LDPC codes literature, the flood BP decoder is the standard BP decoder, where at each iteration all VNs and all CNs are activated simultaneously.

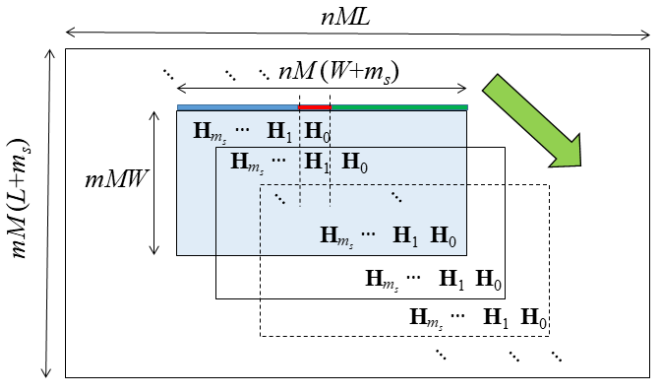

Fig. 1. Pictorial representation of the WD. Colors on top of the current window position represent different symbol types. Blue: memory symbols. Red: target symbols. Green: look-ahead symbols.

a retransmission strategy based on synchronization between TX and RX on a feedback channel. In [13], a method based on an adaptive number of iterations and window shifting is proposed to cope with error propagation in the WD decoder for SC-LDPC codes. Finally, in a couple of recent papers ([14]-[15]), Zhu *et al.* propose check node (CN) or variable node (VN) doping to alleviate error propagation. In particular, VN doping, consisting in inserting some known VN periodically in the frame, unlike the other solutions, does not require altering the WD. However, doping causes a rate loss.

The forward-backward decoder introduced in this paper shows some similarity with the *zig-zag* decoder of [16], which aims at improving the WD performance by decoding in the backward direction, interrupting the forward one, at suitable but unpredictable time instants. Also, the *replica shuffled* BP decoder of [17] is a sort of forward-backward decoder proposed in the context of uncoupled LDPC and turbo codes. In the replica decoder, the VNs are activated serially and in different orders, contrary to the WD where the CNs are activated serially.

II. SC-LDPC Codes and Window Decoder

A. Protograph-based SC-LDPC Codes

SC-LDPC codes are terminated LDPC convolutional codes. In the following, we will consider SC-LDPC codes derived from protographs. Let $\mathbf{B}$ be the $m \times n$ protograph of a regular (ℓ, r) LDPC code, whose rows correspond to weight-r CNs and columns to weight-ℓ VNs. It follows that $m = n\frac{\ell}{r}$. Entry (p, q) of $\mathbf{B}$ is a nonnegative integer, equal to the number of edges connecting the p-th CN to the q-th VN. Such base LDPC protograph is replicated L times, where L will be called hereafter the *coupling length*. Then, edge spreading is performed, i.e., some edges connected to VNs from the i-th replica, $i = 1, \ldots, L$, are detached from CNs at the same position and moved to CNs at positions $j \in [i+1, \ldots, i+m_s]$, where m_s is the *memory* (or coupling width)[2], with a time-invariant pattern. The resulting protograph of the (ℓ, r, m_s) SC-LDPC code ensemble will then be given by

$$\mathbf{B}_{[L]} = \begin{bmatrix} \mathbf{B}_0 & & \\ \mathbf{B}_1 & \ddots & \\ \vdots & \ddots & \mathbf{B}_0 \\ \mathbf{B}_{m_s} & & \mathbf{B}_1 \\ & \ddots & \vdots \\ & & \mathbf{B}_{m_s} \end{bmatrix}_{m(L+m_s)\times nL} \tag{1}$$

where $\mathbf{B}_0, \ldots, \mathbf{B}_{m_s}$ are $m \times n$ matrices of nonnegative integers, and $\sum_{j=0}^{m_s} \mathbf{B}_j = \mathbf{B}$. Because of the termination, the resulting rate of the SC-LDPC code is lower than that of the underlying block LDPC code, but the rate loss tends to zero as $L \to \infty$.

The parity-check matrix of the SC-LDPC code is then obtained by lifting the protograph matrix in (1). Namely, if M is the lifting factor, each zero of $\mathbf{B}_{[L]}$ is lifted to a $M \times M$ all-zero matrix. Instead, each entry of $\mathbf{B}_{[L]}$ equal to b (with $b > 0$) is lifted to the binary sum of b size-M permutation matrices. The resulting LDPC convolutional code is in general time-varying.

B. Window Decoder

We consider SC-LDPC codes on the AWGN channel. It has been proved in [1]-[2] that SC-LDPC codes decoded with flood BP decoding show the phenomenon of threshold saturation, i.e., for sufficiently large L and $M \to \infty$, the flood BP threshold tends to the MAP threshold of the base block LDPC code. However, for flood BP decoding, both memory requirements and latency are proportional to the coupling length L. Since L must be large in order to reduce the rate loss, such requirements may be too demanding for certain applications. The WD is a suboptimal scheduling of BP messages that allows to avoid such problems.

The WD is divided into stages. At the i-th stage, only the CNs that are at positions $i, \ldots, i+W-1$ and the VNs that are at positions $i-m_s, \ldots, i+W-1$ are activated. The VNs that are at positions i are to be decoded at stage i and are called the target symbols. The VNs at positions $i-m_s, \ldots, i-1$ have already been decoded and are called the memory symbols. Finally, the VNs at positions $i+1, \ldots, i+W$ at stage i are called look-ahead symbols.[3] After I iterations (or, after a given performance criterion has been satisfied) the window shifts right and down of one position and a new stage begins. Fig. 1 depicts the WD principles. Note that, because of its definition, $W \geq m_s + 1$.

III. Threshold of SC-LDPC Codes under WD

We consider in this section $M \to \infty$. Following [3], [7], we define the *WD threshold* $\rho(W, \mathcal{I}_{\min})$ on the AWGN channel as the infimum of E_s/N_0 values for which the mutual information of target symbols is at least $\mathcal{I}_{\min}$ after one decoding stage,

[2]CNs must be added at positions $L+1, \ldots, L+m_s$ in order to perform edge spreading at the right end of the block.

[3]All VNs at positions which are outside $[1, L]$ can be thought to be set to zero and perfectly known at the decoder.

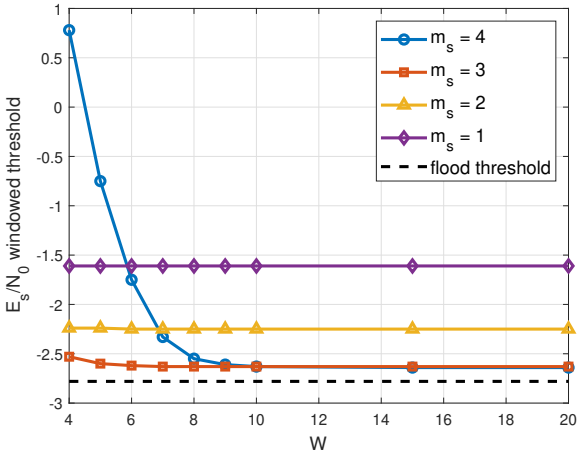

Fig. 2. WD threshold for rate-1/2 $(5, 10, m_s)$ SC-LDPC ensembles with different values of m_s on the BiAWGN channel.

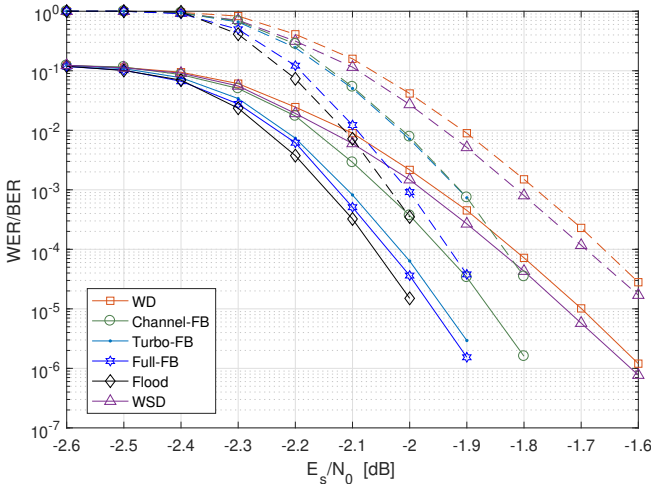

Fig. 3. Performance of the FB-WD decoders and comparison with the flood decoder and the WD: BER (solid lines) and WER (dashed lines) vs E_s/N_0.

given that, before decoding, the memory symbols have mutual information at least $\mathcal{I}_{\min}$ and the other symbols have mutual information at least $\mathcal{I}_{\text{ch}}$, the one received from the channel.

An analysis of the WD threshold for regular SC-LDPC codes on the BEC can be found in [4]. For a particular ensemble and a given residual erasure probability after decoding, it is shown in [4] that the WD threshold converges exponentially fast to the flood threshold when W increases.

However, numerical results are shown in the following, showing that the convergence is not guaranteed for all protograph-based SC-LDPC ensembles. Consider, for instance, the rate-1/2 regular ensemble $(5, 10, m_s)$ with $\mathbf{B}_0 = (5 - m_s)[1, 1]$ and $\mathbf{B}_i = [1, 1]$ for $i = 1, \ldots, m_s$. Fig. 2 shows the WD threshold on the AWGN channel for such ensembles, for $m_s = 1, \ldots, 4$, together with the flood threshold, which is essentially the same for all values of m_s. As it can be seen, while for $m_s = 3, 4$, the WD threshold converges to the flood threshold for W sufficiently large,[4] for $m_s = 1, 2$ even for $W = 20$ there is a relatively large gap between the two thresholds. The best scheme is the one with $m_s = 3$, which has a good performance also for small values of W, since it avoids degree-1 VNs within the window (see [5]).

The fact that, unlike for the symmetric ensemble of [4], the WD threshold does not always converge to the flood threshold for $W \to \infty$ was already observed in [5] for nonbinary SC-LDPC codes and can be explained as follows. We define on the AWGN channel the left (resp., right) activation threshold $\rho_L(\mathcal{I}_{\min})$ (resp., $\rho_R(\mathcal{I}_{\min})$) as the minimum E_s/N_0 needed to trigger the left (resp., right) decoding wave so as to achieve at least a mutual information $\mathcal{I}_{\min}$ for all symbols. Let $\rho_f(\mathcal{I}_{\min})$ be the flood threshold on the AWGN channel, then

$$\rho_f(\mathcal{I}_{\min}) = \min(\rho_L(\mathcal{I}_{\min}), \rho_R(\mathcal{I}_{\min})) \tag{2}$$

$$\lim_{W \to \infty} \rho(W, \mathcal{I}_{\min}) \geq \rho_L(\mathcal{I}_{\min}). \tag{3}$$

[4]The small gap is due to the fact that the WD threshold is actually an upper bound to the actual threshold.

Thus, if $\rho_L(\mathcal{I}_{\min}) > \rho_R(\mathcal{I}_{\min})$, the WD threshold does not tend to the flood threshold. It is worth noting that, to avoid this shortcoming, it suffices to "revert" the ensemble (i.e., to define the new ensemble with $\widetilde{\mathbf{B}}_i = \mathbf{B}_{m_s - i}$, $i = 0, \ldots, m_s$) so as to exchange the values of $\rho_L(\mathcal{I}_{\min})$ and $\rho_R(\mathcal{I}_{\min})$.

IV. Finite-length Performance of WD

For finite values of the lifting factor M, generally we observe that the waterfall region of the BER curve of the WD on the AWGN channel has a reduced slope compared to the flood BP decoder. As an example, for a girth-6 sample code of the rate-1/2 regular SC-LDPC ensemble $(5, 10, 4)$ with $\mathbf{B}_i = [1, 1]$ for all i, lifting factor $M = 300$ and coupling length $L = 100$, Fig. 3 shows the performance, in terms of BER/WER vs E_s/N_0, of the WD, with window size $W = 20$ and $I = 3$ iterations per window, versus the performance of the flood BP decoder after 300 iterations. We have chosen a large W in order to boost the WD performance. Increasing I yields a marginal gain, as for the flood BP decoder beyond 300 iterations. In both cases we apply *serial-C* decoding, where CNs are activated serially and VNs updated after each CN activation, to improve the decoding convergence with the iterations. As it can be seen, the BER curve for the WD is flatter than for the flood decoder.

In order to study the origin of such performance loss, we have investigated the statistics of the error events. We can distinguish two different E_s/N_0 regions. Whenever the E_s/N_0 value is larger than the BP threshold for the underlying block LDPC code, error events are typically short, because in this E_s/N_0 region each section is able to converge to the correct codeword even when the information coming from the neighboring sections is wrong or missing. However, when the E_s/N_0 value is between the MAP and the BP threshold for the underlying block LDPC code, each section relies exclusively on the "decoding waves" coming from the neighboring sections. Whenever these waves interrupt, it is unlikely that the section alone is able to converge. Because

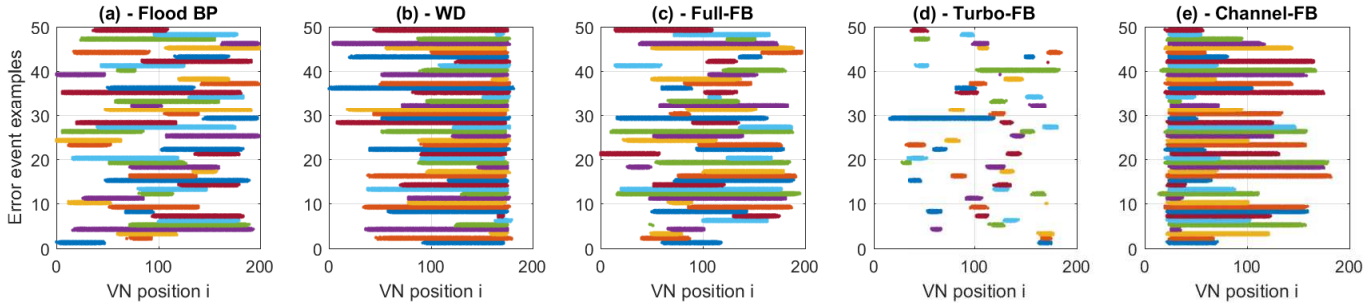

Fig. 4. Typical error events of various SC-LDPC decoders: flood BP (a), WD (b), Full-FB (c), Turbo-FB (d) and Channel-FB (e).

of that, error events are typically long and often in the WD, where there is a single decoding wave, after they have started, they cannot be stopped, giving rise to a sort of "avalanche effect." Fig. 4 (a)-(b) shows 50 typical error events collected by the flood decoder and by the WD, at E_s/N_0 values for which the SC-LDPC code is decoded on the AWGN channel with WER $\approx 10^{-3}$. The x-axis represents the VN position i, from 1 to $nL = 200$, while the y-axis counts codeword error events. The colored bars represent VN positions affected by the error event: the flood decoder errors (Fig. 4 (a)) start and end at different positions within the block, whereas the WD ones (Fig. 4 (b)) end close to the block extremity, but for their very end which are resolved thanks to the termination effect.

We interpret the BER loss of the WD w.r.t. the flood BP decoder as a consequence of the fact that, while in the latter there are two decoding waves, one moving forward from the beginning of the block, the other proceeding backward from the end, in the former there is only the forward wave. In the finite-length case, the steeper slope of the BER waterfall for the flood BP decoder denotes thus a veritable diversity gain w.r.t. the WD, that relies only on the forward wave for successful decoding.

V. The Forward-backward Decoder

In this section, we introduce a new decoder, called forward-backward WD (FB-WD), implemented in a few variants, which serves two purposes. The first is theoretical, since it allows to verify that the performance loss of the WD is actually due to the lack of diversity in the traveling wave, as postulated above. The second is practical, since the various FB-WD, lying so to speak in between the flood decoder and the WD, represent a way of trading off the two opposite exigences of performance and latency/memory requirements. The FB-WD consists of two successive runs: the first is exactly the same as the WD, while the second, starting when the window has reached the block end, consists in a similar block reprocessing where the window is shifted backward, up to the beginning. In principle, we could define a different window size for the two runs, which can be useful for asymmetric SC-LDPC ensembles. The backward processing run is activated only when the forward decoding is not successful, as can be easily ascertained by checking the parity-check equations. The variants of the FB-WD are described in the following.

- In the *Full* FB-WD, the backward processing uses the messages on each edge generated by the forward processing run.
- In the *Channel* FB-WD, the backward processing run, activates the CNs backward using as input only the messages coming from the channel, without using the messages generated by the forward processing run. The final decision for each VN is made by the backward processing run only, neglecting the forward WD results.
- In the *Turbo* FB-WD, the backward processing works as in the *Channel* FB-WD, but the final decision for each VN is made by adding to the channel LLRs both extrinsic LLRs of the two processing runs.

Fig. 3 shows the performance of the FB-WD, for window size $W = 20$ and $I = 3$ iterations per window. The slope of the BER waterfall for the Full FB-WD and Turbo FB-WD are very similar to the flood decoder. The Channel-FB curve slope is also better than for the WD, confirming the intuition of an increased diversity thanks to the FB-WD.

To stress the difference between the FB-WD and the decoders previously proposed to avoid burst-like error patterns as in [13], in Fig. 3 we also plot the performance curves of the WSD algorithm from [13]. In the WSD, whenever a stall is detected, the decoding window is shifted back by n_b blocks and decoding is rerun increasing the number I of iterations per window, up to a maximum value I_{max}. In our simulation, we set $I_{max} = 2I$, $n_b = W - 1$ and we use parity-check stall detection. However, this reprocessing is clearly not capable of exploiting the wave propagating in the opposite direction, which is what the FB decoders conversely attempt to do. In fact, albeit improving the error rate, the WSD shows performance curve slopes not different from the WD.

For a deeper understanding, in Fig. 4 we also plot examples of error events for the three FB-WD decoders. The variant *Full* FB-WD (c) exhibits error events similar to the flood decoder, whereas the variant *Turbo* FB-WD (d) exhibits many more, shorter error events compared to the flood decoder. The two processing runs of the Turbo FB-WD decoder cooperate in recovering most of the mistaken bits inside the error events of WD, but in some intermediate sections they fail to correct some portions of these events. This behavior is confirmed by the WER curves also shown as dashed lines in Fig. 3, which

are almost identical for the Turbo FB-WD and the Channel FB-WD decoders. The error events of the Channel FB-WD (e) have a symmetrical structure compared to WD, as obvious, since the final decisions are made by the backward WD only. Notwithstanding, the Channel FB-WD fails only when *both* processing runs fail, which yields part of the diversity gain of the flood decoder to its BER and WER curves.

From the practical point of view, all the FB-WD variants have processing complexity very similar to the WD, since at every step only those VNs and CNs that are within the window are active. More precisely, if P_W is the WER achieved by the WD, we have

$$\mathcal{Z}_{\mathrm{x-FB-WD}} \simeq (1 + P_W)\, \mathcal{Z}_{\mathrm{WD}}$$

where $\mathcal{Z}_{\mathrm{WD}}$ is the complexity of the WD and $\mathcal{Z}_{\mathrm{x-FB-WD}}$ is the complexity of any FB-WD variant. Since P_W is typically increasing with L, the processing complexity will show a weak dependence on L as well.

Memory requirements for the FB-WD can be much more demanding than for the WD. In the Full FB-WD, there is need to store the edge messages for the whole block, to be used during the backward sweep, thus the amount of memory needed is the same as for the flood decoder and grows with L. Only the scheduling is much more efficient. In the Turbo FB-WD, there is need to store the channel and the output LLRs only, in order to run the backward sweep and to combine the results at the end. Finally, in the Channel FB-WD, there is need to store the channel LLRs only, in order to feed the backward run should the forward run fail.

The FB-WD has also a larger maximum latency than the WD, particularly for the VNs that are at the beginning of the block. However, since the backward processing run is activated only in case of failure of the WD, which occurs in a very small fraction of the cases (see the WD WER curve in Fig. 3), the average FB-WD latency is very close to the WD latency. Precisely, for a WER equal to P_W, the average latency of any FB-WD variant can be computed as

$$\mathcal{L}_{\mathrm{x-FB-WD}} \simeq (1 - P_W)\, W + P_W\, (L + W) = W + P_W L.$$

Hence, average latency shows a dependence on L, which, however, tends to be negligible for $P_W \ll 1$. The concept of improving the WD performance by exploiting a backward decoding is not new, but can be found, e.g., in [16], where the so called *zig-zag decoder* is introduced. The zig-zag decoder detects the starting and the ending point of an error event in the forward sweep by computing partial syndromes. Upon detection of the complete error event, the decoder inverts its direction and starts decoding backward up to the beginning of the wrong path. The decoding direction can change until the error is corrected. While the zig-zag decoder has a good performance, it presents some problem from the hardware implementation point-of-view due to the unpredictable change of decoding direction and the detection of the error events. In comparison, the FB-WD decoder is easier to implement in hardware and can still achieve a good performance improvement over the standard WD.

VI. Conclusions

In this paper, we have compared the performance of the WD for SC-LDPC codes with the flood BP decoder. In the infinite-length case, the phenomenon of lack of convergence of the threshold of the WD to that of the flood BP decoder is observed and explained, while a simple possible method to avoid it is proposed. In the finite-length case, we show simulation results indicating that the slope of the BER threshold in the waterfall region is lower for the WD than for the flood BP decoder, a phenomenon explained as a lack of diversity. To counteract this problem, the FB-WD is introduced, in several variants, that, with different memory requirements, are able to improve the BER slope without compromising the feasibility of the decoder.

References

[1] S. Kudekar, T. Richardson, and R. 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.

[2] ——, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 7761–7813, Dec. 2013.

[3] A. Iyengar, M. Papaleo, P. Siegel, J. Wolf, A. Vanelli-Coralli, and G. 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.

[4] A. R. Iyengar, P. H. Siegel, R. Urbanke, and J. K. Wolf, "Windowed decoding of spatially coupled codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2277–2292, April 2013.

[5] L. Wei, D. Mitchell, T. E. Fuja, and D. J. Costello, "Design of spatially coupled LDPC codes over GF(q) for windowed decoding," *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 4781–4800, Sep. 2016.

[6] D. Costello, L. Dolecek, T. E. Fuja, J. Kliewer, D. Mitchell, and R. Smarandache, "Spatially coupled sparse codes on graphs: Theory and practice," *IEEE Commun. Mag.*, vol. 52, pp. 168–176, July 2014.

[7] D. 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. J. Felstrom and K. S. 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.

[9] A. E. Pusane, A. J. Feltstrom, A. Sridharan, M. Lentmaier, K. S. Zigangirov, and D. J. Costello, "Implementation aspects of LDPC convolutional codes," *IEEE Trans. Commun.*, vol. 56, no. 7, pp. 1060–1069, 2008.

[10] M. Lentmaier, M. Prenda, and G. P. Fettweis, "Efficient message passing scheduling for terminated LDPC convolutional codes," in *Proc. IEEE Int. Symp. Inform. Theory*, July 2011, pp. 1826–1830.

[11] I. E. Bocharova, B. D. Kudryashov, and R. Johannesson, "LDPC convolutional codes versus QC LDPC block codes in communication standard scenarios," in *Proc. IEEE ISIT*, 2014, pp. 2774–2778.

[12] M. Zhu, D. Mitchell, M. Lentmaier, D. J. Costello, and B. Bai, "Combating error propagation in window decoding of braided convolutional codes," in *Proc. IEEE Int. Symp. Inform. Theory*, 2018, pp. 1380–1384.

[13] K. Klaiber, S. Cammerer, L. Schmalen, and S. t. Brink, "Avoiding burst-like error patterns in windowed decoding of spatially coupled LDPC codes," in *Proc. IEEE Int. Symp. Turbo Codes Iterative Inf. Proc.*, 2018.

[14] M. Zhu, D. Mitchell, M. Lentmaier, and D. J. Costello, "A novel design of spatially coupled LDPC codes for sliding window decoding," in *Proc. IEEE Int. Symp. Inform. Theory*, 2020, pp. 473–478.

[15] ——, "Decoder error propagation mitigation for spatially coupled LDPC codes," *arXiv preprint arXiv:2004.08530*, 2020.

[16] S. Abu-Surra, E. Pisek, and R. Taori, "Spatially-coupled low-density parity check codes: Zigzag-window decoding and code-family design considerations," in *Proc. ITA*, Feb 2015, pp. 275–281.

[17] J. Zhang, Y. Wang, M. P. Fossorier, and J. S. Yedidia, "Iterative decoding with replicas," *IEEE Trans. Inf. Theory*, vol. 53, no. 5, pp. 1644–1663, 2007.