Efficient Message Passing Scheduling for Terminated LDPC Convolutional Codes

Lentmaier, Michael; Mellado Prenda, Maria; Fettweis, Gerhard

Published in:
2011 IEEE International Symposium on Information Theory Proceedings

DOI:
10.1109/ISIT.2011.6033865

2011

Efficient Message Passing Scheduling for Terminated LDPC Convolutional Codes

Michael Lentmaier, Maria Mellado Prenda, and Gerhard P. Fettweis
Vodafone Chair Mobile Communications Systems
Dresden University of Technology (TU Dresden), 01062 Dresden, Germany
Email: michael.lentmaier@ifn.et.tu-dresden.de

***Abstract*—Message passing schedules that reduce the decoding complexity of terminated LDPC convolutional code ensembles are analyzed. Considering the AWGN channel, various schedules are compared by means of density evolution. The results of the analysis together with computer simulations for some (3,6)-regular codes confirm that sliding window decoding is an attractive practical solution for low-latency and low-complexity decoding.**

I. Introduction

It can be observed that terminated LDPC convolutional code ensembles have better belief propagation (BP) decoding thresholds than their tailbiting versions or the block codes they are constructed from [1], [2], [3]. However, if a standard BP block decoder is applied to these ensembles the decoding complexity per symbol increases linearly with the number of consecutively encoded blocks, which we denote as termination factor L. Small values of L, on the other hand, result in a large rate loss that is undesirable.

At the same time it is widely known that the special structure of convolutional codes is well-suited for efficient pipeline decoding [4], [5], [6], allowing for a continuous windowed transmission without any termination or, more practically, a termination after an arbitrarily large number L of consecutively encoded blocks. A different type of sliding window decoding was introduced in [7] for proving that the threshold improvement of LDPC convolutional codes does not vanish as $L \to \infty$ (see also [2]). This type of window decoding was also investigated in [8] from a latency perspective.

In this paper, considering the AWGN channel, we analyze and compare various message passing schedules (MPS) by means of a density evolution analysis. We demonstrate that the window decoder considered in [8], [7] and [2] can be interpreted as a natural and easily implementable approximation of a special MPS in which superfluous node updates are omitted. Simulation results for some (3,6)-regular code examples show that with the sliding window decoder LDPC convolutional codes can outperform their block code counterparts without increasing latency and complexity.

II. Approaching Capacity with Asymptotically Regular LDPC Ensembles

Consider an ensemble of LDPC block codes, defined by a protograph with n_c check nodes and n_v variable nodes and its bi-adjacency matrix $\mathbf{B}$, called *base matrix*. An individual code of this ensemble, with an $Nn_c \times Nn_v$ parity-check matrix $\mathbf{H}$, can be derived from this protograph by a lifting procedure that replaces each 1 in $\mathbf{B}$ by an $N \times N$ permutation matrix and each 0 by an $N \times N$ all-zero matrix[1]. Assume now that we want to transmit a sequence of codewords $\mathbf{v}_t$, $t = 1, \dots, L$. In conventional block coding, each of these codewords of length $n_t = Nn_c$ is encoded independently by means of a code of the ensemble. The achievable performance depends on the lifting factor N and the structure of the protograph. In convolutional coding, on the other hand, the blocks $\mathbf{v}_t$ are *coupled* by the encoder over various time instants t. The memory $m_{\rm cc}$ of the convolutional code determines the maximal distance between a pair of coupled blocks. Starting from the base matrix $\mathbf{B}$ of a block code ensemble, the coupling of consecutive blocks can be achieved by an *edge spreading* procedure [9] that divides the edges from variable nodes at time t among equivalent check nodes at times $t + i$, $i = 0, \dots, m_{\rm cc}$. This procedure is illustrated in Fig. 1 for a (3,6)-regular ensemble with $\mathbf{B} = [3, 3]$. The resulting sequence of coupled code blocks forms a codeword $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_t \dots \mathbf{v}_L]$ of a terminated LDPC convolutional code. The corresponding ensemble can be described by means of a *convolutional protograph* with base matrix

$$\mathbf{B}_{[1,L]} = \begin{bmatrix} \mathbf{B}_0 & & \\ \vdots & \ddots & \\ \mathbf{B}_{m_{\rm cc}} & & \mathbf{B}_0 \\ & \ddots & \vdots \\ & & \mathbf{B}_{m_{\rm cc}} \end{bmatrix}_{(L+m_{\rm cc})n_c \times Ln_v} . \quad (1)$$

The block coding ensemble with disconnected protographs corresponds to the special case $m_{\rm cc} = 0$ and $\mathbf{B}_0 = \mathbf{B}$.

In order to maintain the degree distribution and the structure of the original ensemble, a valid edge spreading should satisfy the condition

$$\sum_{i=0}^{m_{\rm cc}} \mathbf{B}_i = \mathbf{B} \ . \quad (2)$$

This condition ensures that the entries of $\mathbf{B}$ are divided among the matrices $\mathbf{B}_i$ in such a way that the sums over the columns and rows of $\mathbf{B}_{[1,L]}$ are equal to those of $\mathbf{B}$. The only exception are the first and last $m_{\rm cc}n_c$ rows of $\mathbf{B}_{[1,L]}$,

[1] Integer entries larger than one, representing multiple edges between a pair of nodes, are replaced by a sum of permutation matrices.

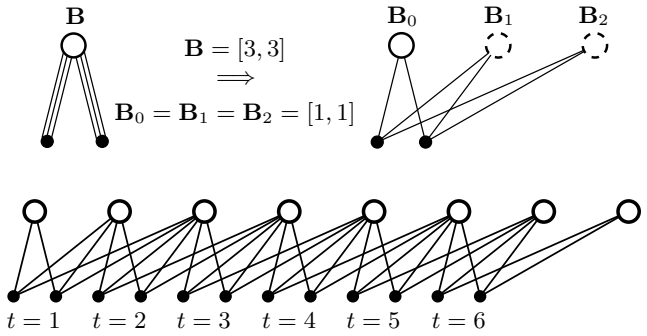

Fig. 1. Illustration of edge spreading: the protograph of a (3,6)-regular block code with base matrix $\mathbf{B}$ is repeated $L = 6$ times and the edges are spread over time according to the component base matrices $\mathbf{B}_0$, $\mathbf{B}_1$, and $\mathbf{B}_2$, resulting in a terminated LDPC convolutional code.

whose weights are reduced as a result of the termination at the ends of the convolutional ensemble. The corresponding check nodes at the start and end of the protograph have lower degrees (see also Fig. 1), resulting in a slight irregularity with stronger protection of the symbols associated with the connected variable nodes. As $L \to \infty$, the fraction of lower degree check nodes vanishes and the degree distribution of the coupled ensemble $\mathbf{B}_{[1,L]}$ converges to that of the original block code ensemble $\mathbf{B}$. As a consequence, the ensemble of terminated LDPC convolutional codes considered in Fig. 1 (which we subsequently refer to as *Ensemble A*) forms an example of *asymptotically regular* LDPC codes. Despite of the vanishing fraction of stronger check nodes, it turns out that the coupled ensembles have a substantially better BP decoding threshold than the block ensembles they are constructed from. In particular, as $L \to \infty$ the BP decoding threshold of the coupled ensembles converges to the optimal MAP decoding threshold of the underlying block ensemble. For regular LDPC codes this *threshold saturation* phenomenon has been proven analytically for the BEC in [3], and it can be observed empirically for the AWGN channel as well. The slight *structured irregularity* of the coupled ensembles leads to BP decoding thresholds that approach the Shannon limit as the node degrees increase.

III. Complexity Analyis for Different Message Passing Schedules

We assume that transmission takes place over the AWGN channel and consider BP decoding with log-likelihood ratios (LLRs) acting as messages. Let $L_{\mathrm{ch}}(i)$ denote the channel LLR of code symbol i and $L_c(e)$ and $L_v(e)$ denote the messages passed from a check node and a variable node along an edge e, respectively. The edges connected to a variable node i or a check node m are labeled by $e^v_{i,j}$ or $e^c_{m,k}$. The j-th edge of variable node i is connected to the k-th edge of check node m if $e^v_{i,j} = e^c_{m,k}$. Before decoding, all messages $L_v(e)$ are initialized with the incoming channel LLRs and all messages $L_c(e)$ are set to zero.

A. Conventional Flooding Schedule (MPS-I)

The message passing schedule that is most frequently used for BP decoding of LDPC codes is the *flooding schedule*.

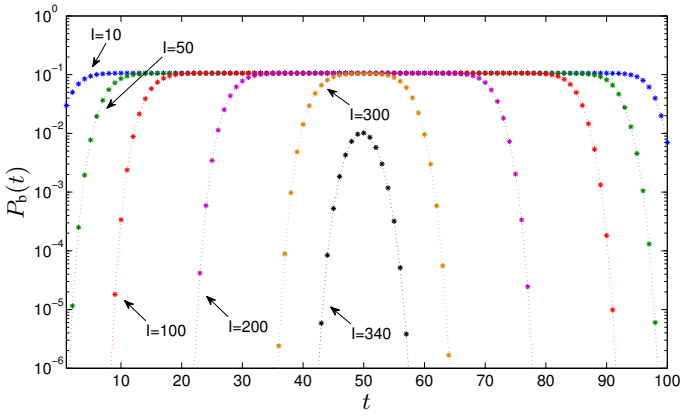

Fig. 2. Ensemble A, $L = 100$, $\sigma = 0.923$: Density evolution bit error probability $P_{\mathrm{b}}(t)$, corresponding to symbols at time instants $t = 1, \ldots, L$, after different numbers of iterations I.

In each decoding iteration first all check nodes and then all variables nodes are updated. Since within each decoding iteration the check node updates and variable node updates of all nodes can be performed in parallel it is also called *parallel schedule*.

Message Passing Schedule I (MPS-I)

1) Check node update:
   For all check nodes $m = 1, \ldots, LNn_c$
   For all edges $e^c_{m,k}$ update the outgoing messages according to

$$L_c(e^c_{m,k}) = 2\tanh^{-1}\left(\prod_{k' \neq k} \tanh\left(\frac{L_v(e^c_{m,k'})}{2}\right)\right)$$

2) Variable node update:
   For all variable nodes $i = 1, \ldots, LNn_v$
   For all edges $e^v_{i,j}$ update the outgoing messages according to

$$L_v(e^v_{i,j}) = L_{\mathrm{ch}}(i) + \sum_{j' \neq j} L_c(e^v_{i,j'})$$

For a code with sufficiently large girth, the probability distribution of the messages $L_c(e)$ and $L_v(e)$ exchanged during the iterations can be computed by density evolution within the protograph. We have used the discretized density evolution technique by Chung [10] for a convergence analysis of coupled codes from Ensemble A. For a decoder with MPS-I the bit error probability $P_{\mathrm{b}}(t)$ corresponding to messages $L_v(e)$ from variable nodes at time t is shown in Fig. 2 for different numbers of iterations. As expected, for all iterations the symbols at the start and end of the protograph are better protected due to the lower check node degrees from the termination. Furthermore, the curves show that this improved performance propagates through toward the center as the number of iterations increases until, eventually, a low $P_{\mathrm{b}}(t)$ can be observed for all t.

B. Target Probability Based Schedule (MPS-II)

The results in Fig. 2 show that the number of iterations required to guarantee a particular target error probability

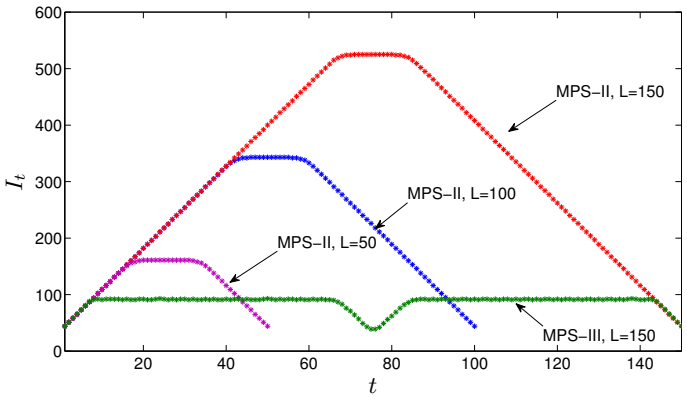

Fig. 3. Ensemble A, $\sigma = 0.923$: Number of iterations I_t for symbols at time instants $t = 1, \ldots, L$, for different message passing schedules.

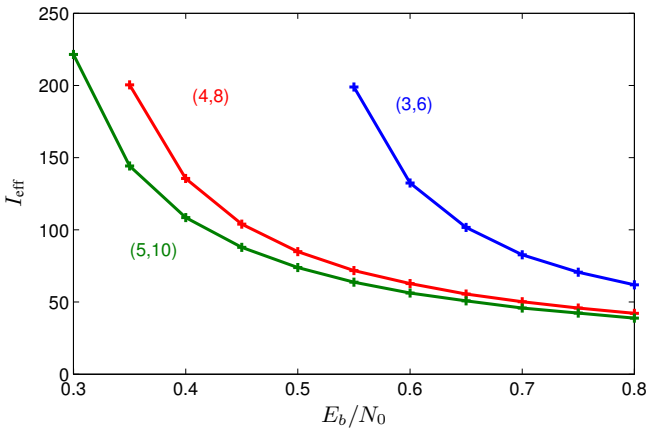

Fig. 4. Effective number of iterations I_{eff} as function of E_b/N_0 for different regular LDPC convolutional ensembles with message passing schedule MPS-III (considering the limit $L \to \infty$).

$P_{\text{b}}^{\max}$ depends on the position t within the protograph. Due to the special structure of the coupled ensemble, while a $P_{\text{b}}(t) < 10^{-5}$ is achieved already after $I = 50$ iterations at position $t = 1$, more than $I = 300$ iterations are required closer to the center at $t = 40$. The following schedule makes use of this behavior by deactivating updates of nodes at positions t where the target probability already has been achieved.

Message Passing Schedule II (MPS-II)

1) For all t such that $P_{\text{b}}(t) < P_{\text{b}}^{\max}$:
   Exclude check nodes m and variable nodes i at position t from the set of updated nodes.
2) Perform check node and variable node updates for the remaining nodes according to the rules of MPS-I.

Let I_t denote the number of times the variable and check nodes at position t are updated during the iteration procedure. The average decoding complexity per symbol is then proportional to the *effective* number of iterations I_{eff}, defined as $I_{\text{eff}} = 1/L \sum_{t=1}^{L} I_t$. While for MPS-I I_t is equal to the total number of iterations for all t, i.e., $I_{\text{eff}} = I_t = I$, for MPS-II I_t depends on the position in the protograph, as depicted in Fig. 3 for $P_{\text{b}}^{\max} = 10^{-5}$. For $L = 100$ the maximal value of I_t is equal to the total number of iterations I required in Fig. 2. It can be observed that I_t increases linearly with the distance from the ends of the protograph, with a slope that is independent of the termination factor L. In the neighborhood of the center a flattening of the curves to a constant value can be observed, which can be prescribed to the influence of messages propagating from the other end of the graph. The width of the flat region as well as the slope of the curve depend on $P_{\text{b}}^{\max}$ and on the standard deviation σ of the channel noise, but not on L. Note that MPS-II clearly reduces I_{eff} and, hence, the average complexity per symbol compared to MPS-I.

C. Improvement Based Schedule (MPS-III)

According to Fig. 3 the nodes in the center of the protograph require the largest number of updates. On the other hand, from Fig. 2 we can conclude that little influence from the stronger check nodes can be expected within the first decoding iterations. In order to quantify the actual relevance of an update, let us introduce the fractional bit error improvement $\Delta P_{\text{b}}(t) = (P_{\text{b}}^{\text{old}}(t) - P_{\text{b}}^{\text{new}}(t))/P_{\text{b}}^{\text{old}}(t)$ resulting from an update of all nodes at position t. The following schedule updates only nodes with a minimum improvement θ, $0 \leq \theta \leq 1$.

Message Passing Schedule III (MPS-III)

1) For all t such that $P_{\text{b}}(t) < P_{\text{b}}^{\max}$:
   Exclude check nodes m and variable nodes i at position t from the set of updated nodes.
2) For all t such that $\Delta P_{\text{b}}(t) < \theta$:
   Exclude variable nodes i at position t from the set of updated nodes.
3) For all check nodes m at positions t that are not connected to variable nodes to be updated:
   Exclude check nodes m at position t from the set of updated nodes.
4) Perform check node and variable node updates for the remaining nodes according to the rules of MPS-I.

The values I_t for this schedule are also depicted in Fig. 3 for $\theta = 0.01$. It turns out that the proposed deactivation of nodes in the center leads to a dramatic reduction in complexity. For all $t = 1, \ldots, L$ the required number of updates is reduced to a value that previously was only sufficient for the first and last few positions in the protograph. The maximal I_t depends on $P_{\text{b}}^{\max}$, on σ, and on θ, but not on L. As a remarkable consequence, in contrast to the other schedules, for MPS-III the effective number of iterations per symbol I_{eff} does no longer increase with the termination factor L (see Fig. 4 and Fig. 5). Although we limit ourselves to regular LDPC ensembles of rate $R = 1/2$ in this paper, the same principle behavior can be observed for arbitrary regular and irregular base protographs and edge spreadings [9].

D. Window Decoding Schedule (MPS-IV)

The complexity reduction in the MPS-III schedule was obtained by omitting irrelevant node updates at the ends and

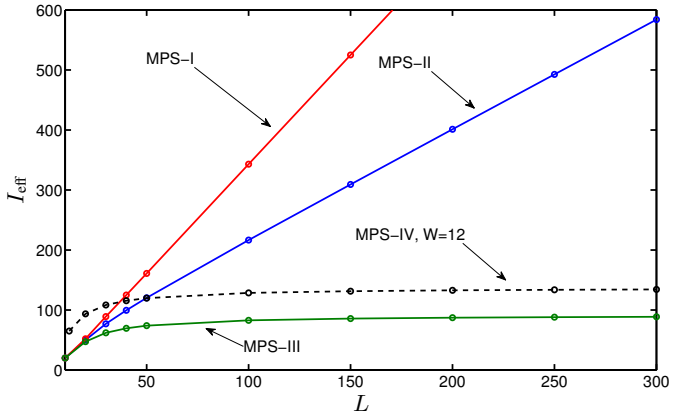

Fig. 5. Ensemble A, $\sigma = 0.923$: Effective number of iterations I_{eff}, as function of the termination factor L, for different message passing schedules.

in the center of the protograph. Due to the structure of the coupled ensemble, during the decoding process the remaining active nodes move from both ends of the protograph toward the center. A natural approximation of this schedule can be easily implemented by means of a sliding window decoder.

Message Passing Schedule IV (MPS-IV) [8] [7] [2]

For all window positions $p = 1, \ldots, L$:

For a number I_p^{win} of iterations:

a) Let all variable nodes i at positions $t = p, \ldots, p + W - 1$ form the set of updated variable nodes.

b) Let all check nodes m at position $t = p, \ldots, p + W - 1$ form the set of updated variable nodes.

c) Perform check node and variable node updates for nodes in these sets according to the rules of MPS-I.

The number of iterations I_p^{win} and the window size W should be chosen in such a way that the target error probability is achieved at position $t = p$. For a given W it can be observed that the probabilities $P_{\text{b}}(t)$, $t = p, \ldots, p + W - 1$ within the window converge to a fix point as $I_t \to \infty$. This behavior is illustrated in Fig. 6 for different window sizes W. The figure also shows another ensemble (Ensemble B), defined by another valid edge spreading $\mathbf{B}_0 = [2, 2]$ and $\mathbf{B}_1 = [1, 1]$ of $\mathbf{B} = [3, 3]$. This ensemble, which has been proposed in [8], achieves the target error probability with a smaller W. Figure 7 shows the protograph of Ensemble B and illustrates the ranges of updated nodes covered by the sliding window.

The values I_t for Ensemble B with windowed schedule MPS-IV are shown in Fig. 8 for window sizes $W = 4$, 6, and 12. Schedule MPS-II is also shown for comparison. The number of iterations I_p^{win} at window position p is chosen adaptively, so that the window is shifted after the target probability $P_{\text{b}}^{\max} = 10^{-5}$ has been achieved. The maximal I_t increases with the values I_p^{win} and W, where I_p^{win} is also influenced by W. The values I_p^{win} are smallest for large windows, when the slope of I_t at the start of the protograph approaches that of MPS-II. In terms of average complexity the optimal window size in the example is $W = 6$. A comparison with the

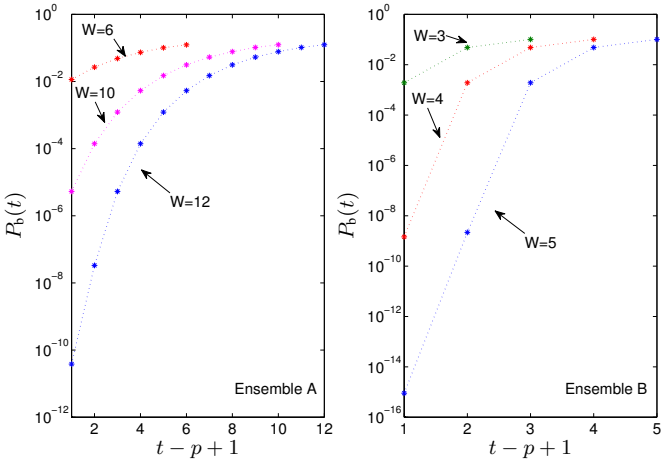

Fig. 6. MPS-IV (windowed schedule), $\sigma = 0.923$: Bit error probability $P_{\text{b}}(t)$ of symbols at positions $t = p, \ldots, p + W - 1$ within the decoding window after $I = 1000$ iterations. Symbols to the left of the window are assumed to be perfectly known.

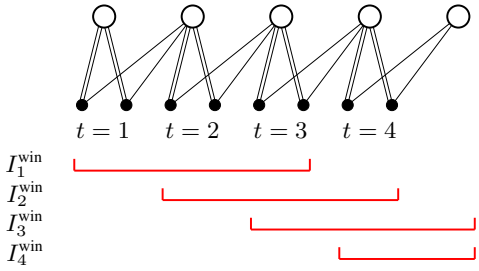

Fig. 7. Illustration of the ranges of updated nodes within the decoding window, moving along the protograph of Ensemble B for $L = 4$. I_p^{win} iterations are performed while the left end of the window is at position p, $p = 1, \ldots, L$.

behavior of the low-complexity MPS-III schedule considered in the previous subsection shows that the window decoder efficiently reduces the number of irrelevant node updates for a given target error probability. A further practical advantage of the window decoder is that the latency is determined by the window size, so that both decoding complexity and latency are independent of the termination factor L.

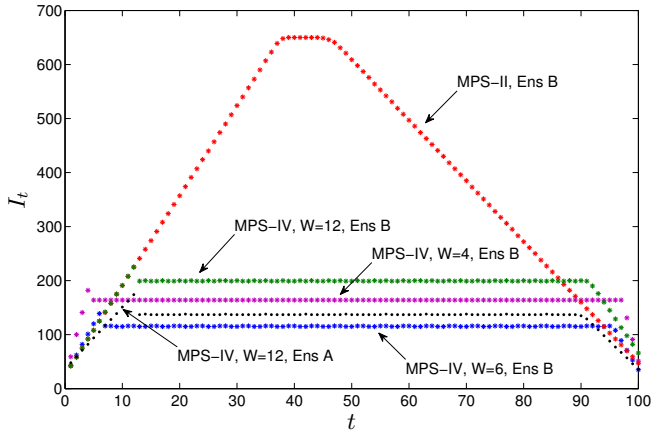

Fig. 8. $L = 100$, $\sigma = 0.923$: Influence of window size W on the complexity I_t of MPS-IV (windowed schedule). MPS-II is shown for comparison.

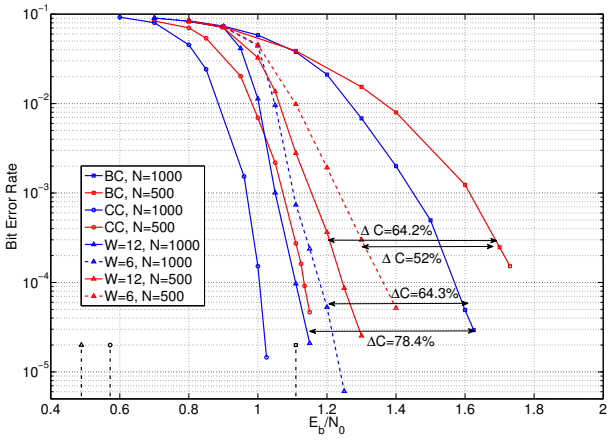

Fig. 9. Performance of terminated (3,6)-regular LDPC convolutional codes with lifting factors $N = 500$ (red) and $N = 1000$ (blue). The horizontal arrows are labeled with the complexity advantage ΔC of the convolutional codes compared to block codes of equal lifting factor N.

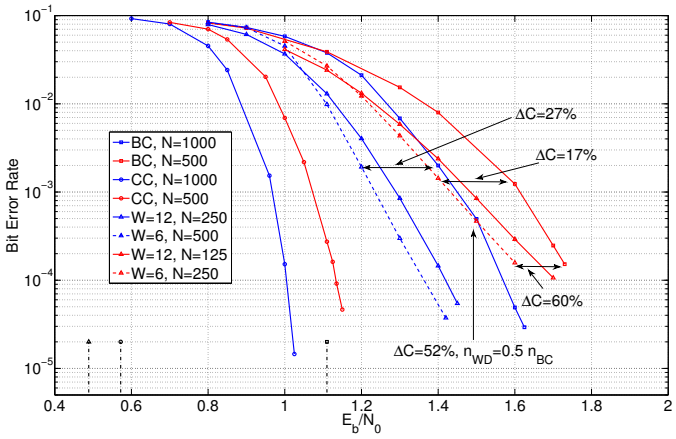

Fig. 10. Performance of terminated (3,6)-regular LDPC convolutional codes compared to block codes of lifting factor $N = 500$ (red) and $N = 1000$ (blue). The lifting factors of the convolutional codes are chosen such that the latency is equal to that of the block codes.

IV. Performance Evaluation of Window Decoding

In this section we evaluate the performance of terminated (3,6)-regular LDPC convolutional codes from Ensemble B by means of computer simulations. Window decoding scheduling (MPS-IV) for $L = 1000$ and $W = 6$ and $W = 12$ is compared to MPS-II schedule of convolutional codes with $L = 100$ (CC) and block codes (BC). The corresponding thresholds are shown as vertical lines. All simulated codes where picked randomly from the corresponding ensembles without any cycle reduction procedure. For the MPS-II schedule (CC) and the block codes (BC) the maximal iteration number was set to $I_{\max} = 10000$. For the window decoder $I_1^{\text{win}} = 501$ iterations have been performed at window position $p = 1$. For all other positions $p > 1$ the number of iterations I_p^{win} was chosen adaptively, with a maximal number of $I_{p,\max}^{\text{win}} = 501$. For fair complexity comparison, in terms of the average effective number of required iterations I_{eff}, a stopping rule was applied to the block codes, which interrupts the iterations as soon as a valid codeword is achieved.

Figure 9 shows the results of these simulations for lifting factors $N = 500$ and $N = 1000$. The value ΔC in the figure denotes the complexity advantage of the convolutional codes compared to the block codes. Compared to the MPS-II decoder, the window decoder reduces the decoding complexity at the cost of a performance degradation. At the same time, the window decoder outperforms the corresponding block codes with a lower average decoding complexity.

In Fig. 10 the lifting factors are chosen in such a way that the latency $n_{\text{WD}} = 2WN$ of the window decoder is equal to the length of the block codes $n_{\text{BC}} = 6N$. Even in this case the window decoder still outperforms the block codes both in terms of complexity and bit error rate, although the advantage is now smaller than in Fig. 9. At 1.5 dB we can see an example that achieves equal performance as a block code with window decoding at half the latency and half the complexity.

The authors are grateful for the use of the high performance computing facilities of the ZIH at TU Dresden.

References

[1] A. Sridharan, M. Lentmaier, D. J. Costello, Jr., and K. Sh. Zigangirov, "Convergence analysis of a class of LDPC convolutional codes for the erasure channel," in *Proceedings of the 42nd Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, 2004.

[2] M. Lentmaier, A. Sridharan, D.J. Costello, Jr., and K.Sh. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inform. 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. Inform. Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.

[4] A. Jiménez Feltström and K.Sh. Zigangirov, "Periodic time-varying convolutional codes with low-density parity-check matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 2181–2190, Sept. 1999.

[5] A. E. Pusane, A. Jiménez Feltström, A. Sridharan, M. Lentmaier, K. Sh. Zigangirov, and D. J. Costello, Jr., "Implementation aspects of LDPC convolutional codes," *IEEE Transactions on Communications*, vol. 56, no. 7, pp. 1060–1069, July 2008.

[6] E. Matúš, M.B.S. Tavares, M. Bimberg, and G.P. Fettweis, "Towards a GBit/s programmable decoder for LDPC convolutional codes," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2007.

[7] M. Lentmaier, A. Sridharan, K. Sh. Zigangirov, and D. J. Costello, Jr., "Terminated LDPC convolutional codes with thresholds close to capacity," in *Proc. IEEE International Symposium on Information Theory*, Adelaide, Australia, Sept. 2005, pp. 1372–1376.

[8] M. Papaleo, A. Iyengar, P. Siegel, J. Wolf, and G. Corazza, "Windowed erasure decoding of LDPC convolutional codes," in *Proc. 2010 IEEE Information Theory Workshop*, Cairo, Egypt, Jan. 2010.

[9] M. Lentmaier, G.P. Fettweis, K.Sh. Zigangirov, and D.J. Costello, Jr., "Approaching capacity with asymptotically regular LDPC codes," in *Proc. Information Theory and Applications Workshop*, San Diego, USA, Feb. 2009.

[10] Sae-Young Chung, Jr. Forney, G.D., T.J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the Shannon limit," *Communications Letters, IEEE*, vol. 5, no. 2, pp. 58–60, Feb 2001.