

LUND UNIVERSITY

Non-Uniform Window Decoding Schedules for Spatially Coupled LDPC Codes

Ul Hassan, Najeeb; Pusane, Ali E.; Lentmaier, Michael; Fettweis, Gerhard P.; Costello, Daniel J. Jr.

Published in:
IEEE Transactions on Communications

DOI:
10.1109/TCOMM.2016.2633466

2017

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

Link to publication

Citation for published version (APA):
Ul Hassan, N., Pusane, A. E., Lentmaier, M., Fettweis, G. P., & Costello, D. J. J. (2017). Non-Uniform Window Decoding Schedules for Spatially Coupled LDPC Codes. *IEEE Transactions on Communications*, *65*(2), 501-510. Article 7762121. https://doi.org/10.1109/TCOMM.2016.2633466

Total number of authors:
5

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Download date: 22. Mar. 2026

Non-Uniform Window Decoding Schedules for Spatially Coupled LDPC Codes

Najeeb Ul Hassan, *Member, IEEE,* Ali E. Pusane, *Member, IEEE,* Michael Lentmaier, *Senior Member, IEEE,* Gerhard P. Fettweis, *Fellow, IEEE,* and Daniel J. Costello, Jr., *Life Fellow, IEEE*

***Abstract*—Spatially coupled low-density parity-check (SC-LDPC) codes can be decoded using a graph-based message passing algorithm applied across the total length of the coupled graph. However, considering practical constraints on decoding latency and complexity, a sliding window decoding approach is normally preferred. In order to reduce decoding complexity compared to standard parallel decoding schedules, serial schedules can be applied within a decoding window. However, uniform serial schedules within a window do not provide the expected reduction in complexity. Hence we propose non-uniform schedules (parallel and serial) based on measured improvements in the estimated bit error rate (BER). We show that these non-uniform schedules result in a significant reduction in complexity without any loss in performance. Further, based on observations made using density evolution, we propose a non-uniform pragmatic decoding schedule (parallel and serial) that does not require any additional calculations (e.g., BER estimates) within the decoding process.**

***Index Terms*—LDPC codes, LDPC convolutional codes, spatially coupled codes, iterative decoding, window decoding, decoding schedules.**

I. Introduction

Iterative decoding of low-density parity-check (LDPC) codes is very attractive in practice because decoding complexity grows only linearly with block length. However, due to the sub-optimality of iterative decoding, there is a gap between the (threshold) performance of iterative decoding and maximum a posteriori (MAP) probability decoding. Spatially coupled LDPC (SC-LDPC) codes, also known as LDPC convolutional codes [1], provide a way to close this gap: SC-LDPC codes with iterative decoding have been shown to achieve the MAP threshold of an underlying LDPC block code [2], making these codes attractive candidates for applications requiring near-capacity performance. One practical problem in achieving the MAP threshold with SC-LDPC codes is the requirement of a large coupling length L for the coupled graph. Applying *block decoding*, i.e., a message passing algorithm over the total length of the graph, results in large decoding latency and complexity. Hence a *sliding window decoder* was proposed in [3] that makes the latency and complexity independent of L. Recently, in [4] and [5], both binary and non-binary SC-LDPC codes have been compared with their LDPC block code counterparts under a latency constraint using a sliding window decoder, and it was shown that the coupled codes perform better than the block codes on an equal latency basis. In this paper, we propose new methods to reduce the decoding complexity of sliding window decoding compared to standard message passing window decoding schedules.

LDPC codes can be represented by a bipartite Tanner graph consisting of check and variable nodes. In the additive white Gaussian noise (AWGN) channel case, messages in the form of log-likelihood ratio (LLR) values are exchanged between the connected check and variable nodes in the graph during the decoding process. Since the absolute value of an LLR represents the strength of our belief that its associated variable node assumes a particular (binary) value, a decoder applying this message passing algorithm is known as a *belief propagation* (BP) decoder [6]. Since BP decoding is an iterative process, a *message passing schedule* within the Tanner graph is required. For applications requiring multi-Gb/s throughputs, the parallel (flooding) schedule, where all variable nodes in the graph are updated simultaneously followed by simultaneous check node updates, is the most common [7].

Alternatively, updates can be performed using a uniform serial schedule. In this case, before updating a check node, all its neighboring variable nodes are asked to produce messages along the edges connected to the check node[1]. Such a schedule ensures that the newly computed messages from the neighbors of a check node are used in the same decoding iteration and reduces the required number of iterations by approximately half compared to the uniform parallel schedule [8]. In both schedules, all the nodes in the graph are updated at every decoding iteration. Hence we refer to these schedules as the *uniform parallel* and *uniform serial* schedules, respectively. Some generalized serial schedules intended to further reduce the decoding complexity of LDPC block codes have been discussed in [8] and [9].

For SC-LDPC codes, applying a uniform serial schedule

N. Ul Hassan and G. Fettweis are with the Dresden University of Technology (TU Dresden), Vodafone Chair Mobile Communications Systems and SFB 912 HAEC, Dresden, Germany, najeeb_ul.hassan, gerhard.fettweis@tu-dresden.de. A. E. Pusane is with the Department of Electrical and Electronics Engineering, Bogazici University, Istanbul, Turkey, ali.pusane@boun.edu.tr. M. Lentmaier is with the Department of Electrical and Information Technology, Lund University, Lund, Sweden, michael.lentmaier@eit.lth.se. D. J. Costello, Jr. is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, Indiana, USA, costello.2@nd.edu.

This work was supported in part by the German Research Foundation (DFG) within the Collaborative Research Center SFB 912 HAEC, the European Social Fund in the framework of the Young Investigators Group 3DCSI, TÜBİTAK Grant 111E276, EU FP7 Marie Curie IRG Grant 268264, and NSF Grant CCF-1165714. The material in this paper was presented in part at the IEEE Information Theory Workshop, Sep. 2012, and at the IEEE Global Communications Conference, Dec. 2013. The authors are also grateful for the use of the high performance computing facilities of the ZIH at TU Dresden.

[1]Such a schedule is also known as an *on demand variable node update* schedule. Similarly, an *on-demand check node update* schedule can also be defined where, before updating a variable node, all its neighboring check nodes are asked to produce messages along the edges connected to the variable node.

within a window results in a reduction in decoding complexity compared to the uniform parallel schedule that is much less than the factor of two reduction achieved for LDPC block codes [8] (see Section II-C). Hence, in order to reduce decoding complexity for window decoding of SC-LDPC codes, we propose several *non-uniform window decoding schedules* (both parallel and serial) in Section III. A non-uniform schedule updates only a portion of the Tanner graph in one decoding iteration based on an *estimated bit error rate (BER) improvement*.

In [10], the authors proposed a *residual belief propagation* algorithm that dynamically changes the message passing schedule. Before exchanging messages from check to variable nodes, all messages (along the edges) are generated. For each message, the difference in magnitude (residual value) between the message generated in the current iteration and the message exchanged in the previous iteration is calculated. A message with the largest residual value is exchanged first followed by an update[2] of the variable node connected to the edge with largest residual value. Then the message with the next largest residual value is exchanged and the associated variable node is updated, and so on. The resultant algorithm is uniform in the sense that all nodes are updated in one iteration, but in a dynamic order. Inspired by the work in [10], the authors in [11] proposed a *node-wise scheduling* algorithm where, instead of exchanging only the message from a check node to a variable node with the largest residual value, all the messages from that check node are exchanged, i.e., a full check node update is performed. This results in certain nodes being updated more often than others, and hence the resultant schedule is non-uniform. Note that the node-wise scheduling algorithm proposed in [11] makes update decisions based on the notion of residual values. In contrast, the non-uniform schedules proposed in this work make update decisions based on an estimated BER improvement measure.

We demonstrate a reduction of up to 50% compared to uniform schedules using density evolution calculations in Section IV. Further, based on the density evolution observations, a simple non-uniform pragmatic decoding schedule is presented in Section V that provides more than a 50% reduction in complexity without any loss in BER performance. In Section VI, finite-length computer simulations are presented before concluding the work in Section VII.

II. Spatially Coupled LDPC Codes

A (J, K)-regular LDPC block code is characterized by a sparse parity-check matrix $\mathbf{H}$, containing exactly J ones in each column and K ones in each row. In order to describe spatial coupling, we consider transmitting a sequence of L code blocks $\mathbf{v}_t$, $t = 1, \ldots, L$, using a protograph-based LDPC block code. A protograph is a small bipartite graph consisting of n_c check nodes and n_v variable nodes and is represented by its $n_c \times n_v$ bi-adjacency matrix $\mathbf{B}$, called the *base matrix*. An LDPC block code is then obtained by applying a graph lifting procedure [12] that replaces each 1 in $\mathbf{B}$ by an $N \times N$ permutation matrix and each 0 by the $N \times N$ all-zero matrix[3]. An essential feature of SC-LDPC codes [1] [13] is that the code blocks transmitted at different time instants are interconnected, i.e., the code blocks $\mathbf{v}_t$ are *coupled* by the encoder to other time instants, thus introducing *memory*, so that the resulting code is *convolutional*. The largest distance between a pair of coupled code blocks defines the memory $m_{\rm cc}$ of the convolutional (coupled) code. The coupling of consecutive code blocks can be achieved by distributing the edges from the variable nodes at time t among the check nodes at times $t + i$, $i = 0, \ldots, m_{\rm cc}$, called an *edge-spreading* procedure [14], where the resulting edge connections to the check nodes at time $t + i$ are represented by the $n_c \times n_v$ bi-adjacency matrices $\mathbf{B}_i$, $i = 0, \ldots, m_{\rm cc}$. In order to maintain the degree distribution and structure of the original protograph, a valid edge-spreading must satisfy the condition $\sum_{i=0}^{m_{\rm cc}} \mathbf{B}_i = \mathbf{B}$. The edge-spreading procedure is illustrated in Fig. 1 for a $(3, 6)$-regular protograph with $n_v = 2$, $n_c = 1$, base matrix $\mathbf{B} = [3, 3]$, $m_{\rm cc} = 2$, and edge-spreading matrices $\mathbf{B}_0 = \mathbf{B}_1 = \mathbf{B}_2 = [1, 1]$. The corresponding sequence of coupled code blocks forms a codeword $\mathbf{v}_{[1,L]} = [\mathbf{v}_1, \ldots, \mathbf{v}_t, \ldots, \mathbf{v}_L]$ of a terminated convolutional code.

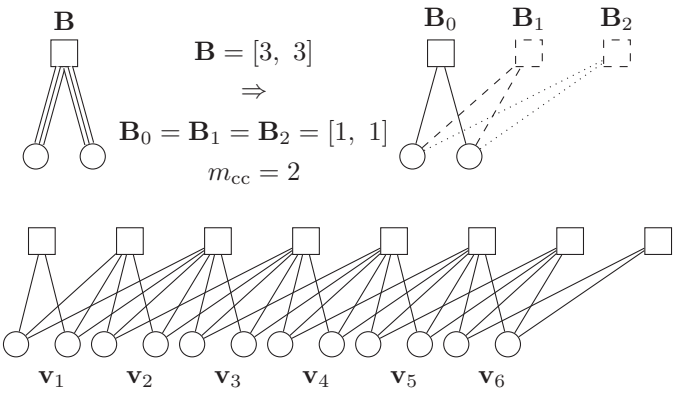


Fig. 1. Illustration of edge-spreading: the protograph of a $(3, 6)$-regular LDPC block code with base matrix $\mathbf{B}$ is repeated $L = 6$ times and the edges are spread over time according to the matrices $\mathbf{B}_0$, $\mathbf{B}_1$, and $\mathbf{B}_2$, resulting in a terminated convolutional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

A. Decoding of Spatially Coupled LDPC Codes

We assume transmission over the AWGN channel and characterize the channel by the standard deviation σ of the Gaussian noise. Decoding is performed using the BP algorithm, where the *extrinsic* LLRs passed from variable node v_k to check node c_j and from check node c_j to variable node v_k during a decoding iteration are denoted L_{v_k,c_j} and L_{c_j,v_k}, respectively, and are calculated as follows,

$$L_{v_k,c_j} = L_{\rm ch}(v_k) + \sum_{l \in \mathcal{N}(v_k) \backslash j} L_{c_l,v_k} \ , \tag{1}$$

$$L_{c_j,v_k} = 2 \tanh^{-1} \left(\prod_{l \in \mathcal{N}(c_j) \backslash k} \tanh \left(\frac{L_{v_l,c_j}}{2} \right) \right) \ . \tag{2}$$

[2]Note that 'update' refers to an operation where all outgoing messages from a node are generated and exchanged.

[3]In the case of a base matrix with integer values greater than 1, the lifting procedure replaces an integer b with the sum of b $N \times N$ permutation matrices.

TABLE I
REQUIRED NUMBER OF ITERATIONS FOR A BLOCK DECODER AND THE AVERAGE NUMBER OF NODE UPDATES U_{avg} FOR A WINDOW DECODER FOR VARIOUS VALUES OF L WHEN THE PARALLEL DECODING SCHEDULE IS APPLIED TO THE SC-LDPC CODE OF FIG. 1 WITH $I_{\max} = 1000$, $\sigma = 0.923$.

L	20	30	40	50	100
Block Decoder	125	181	248	314	646
Window Decoder ($W = 6$)	95	100	104	105	109
Window Decoder ($W = 8$)	118	133	140	145	154

Here $L_{\text{ch}}(v_k)$ denotes the channel LLR for variable node v_k at the input of the decoder and $\mathcal{N}(v_k)\backslash c_j$ represents the set of check nodes connected to v_k, excluding c_j. The exclusion of c_j precludes the information received by v_k from c_j from being reused to calculate the message L_{v_k,c_j}. Similarly, in (2) the variable node v_k is excluded while calculating the output message L_{c_j,v_k}. The output LLRs $L_{\text{out}}(k)$ for every variable node v_k are then computed at the end of each iteration as

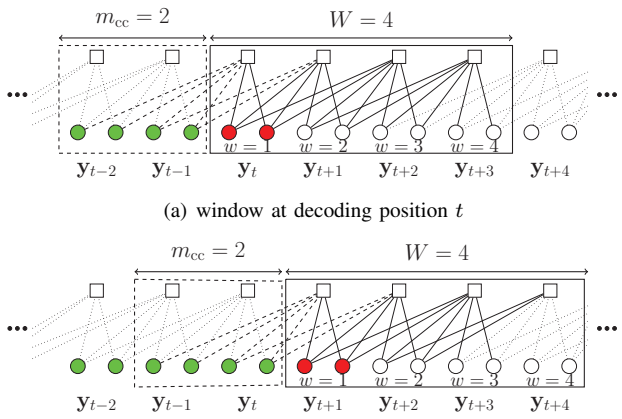
$$L_{\text{out}}(k) = L_{\text{ch}}(v_k) + \sum_{l \in \mathcal{N}(v_k)} L_{c_l,v_k} \quad . \tag{3}$$

The iterative message passing process continues until all the symbols are reliably decoded or some preset maximum number of iterations $I_{\max}$ is reached.

Decoding for a SC-LDPC code can be performed by running the BP algorithm over the L coupled code blocks $\mathbf{v}_{[1,L]}$. Since the coupled code is treated in this case as a block code, we refer to such a decoder as a *block decoder*. A major drawback of the parallel or serial block decoders (and also the classical pipeline decoder [1]) is that the number of required decoding iterations increases with the termination length L for channel parameters close to the threshold of the coupled ensemble, as shown in the first row of Table I [15] for the parallel schedule. Such a dramatic increase in decoding complexity can be avoided by employing efficient message passing schedules that make use of the convolutional structure of SC-LDPC codes [16]. An attractive practical implementation of such a schedule is the sliding window decoder first proposed in [3], for which both latency and complexity are independent of L. A two-sided version of such a window decoder was also considered for density evolution analysis in [17].

B. Window Decoding

Consider two code blocks $\mathbf{v}_t$ and $\mathbf{v}_{t'}$, where $t' \geq t+m_{\text{cc}}+1$ and $t, t' \in [1\ L]$. Due to the memory m_{cc} of the SC-LDPC code, these code blocks do not have any check nodes in common. This characteristic is exploited to define a latency constrained sliding window decoder of size $W \geq m_{\text{cc}} + 1$ code blocks. Figure 2 shows an example of a sliding window decoder with $W = 4$ and the edge-spreading used in Fig. 1. The window consists of W received code blocks, $\mathbf{y}_{t+w-1}, w = 1, \ldots, W$. At any decoding position t, only the nodes in $\mathbf{y}_t$ are decoded (represented in red), and hence they are termed *target nodes*. After the nodes in $\mathbf{y}_t$ are decoded or some maximum number of iterations $I_{\max}$ is performed, the window slides to the next decoding position, as shown in Fig. 2(b).



(a) window at decoding position t

(b) window at decoding position $t+1$

Fig. 2. Window decoder of size $W = 4$. The green variable nodes represent decoded nodes and the red nodes are the target nodes within the current decoding window. The dashed lines represent the required read access to the m_{cc} previously decoded code blocks.

In sliding window decoding, nodes at position t are updated in multiple decoding positions. In order to compare the decoding complexity for different window sizes and decoding schedules, we define the average number of node updates required to decode a node as [16]

$$U_{\text{avg}} = \frac{1}{L} \sum_{t=1}^{L} U_t \quad , \tag{4}$$

where U_t denotes the total number of times the variable nodes at position t are updated during the iterative decoding process. It can be observed from Table I that the decoding complexity increases with W, since a larger window size increases the number of times the nodes at each position are updated. However, the complexity is essentially independent of L for a fixed value of W and is well below the decoding complexity of a block decoder.

The choice of edge-spreading plays a role in determining the required W, and hence the decoding complexity. The edge-spreading considered in Fig. 1 results in degree-1 variable nodes at the right end of the window (see Fig. 2). As a result the output LLRs from these variable nodes, for every iteration, will just equal the channel LLRs L_{ch} (see (1)) at the input of the decoder with no update information from check nodes in the window. Hence, the edge-spreading $\mathbf{B}_0 = [2,\ 2], \mathbf{B}_1 = [1,\ 1]$, first introduced in [18], is considered a better choice for window decoding and is considered throughout the rest of the paper. This edge-spreading avoids degree-1 variable nodes at the right end of the window and thus achieves the desired performance with as small a W as possible. A more detailed set of rules on protograph design for windowed decoding can be found in [3].

C. Uniform Window Decoding Schedules

In uniform window decoding schedules (parallel and serial), all the nodes within the decoding window are updated in each decoding iteration. Note that these schedules are uniform only with respect to the active decoding window. Figure 3

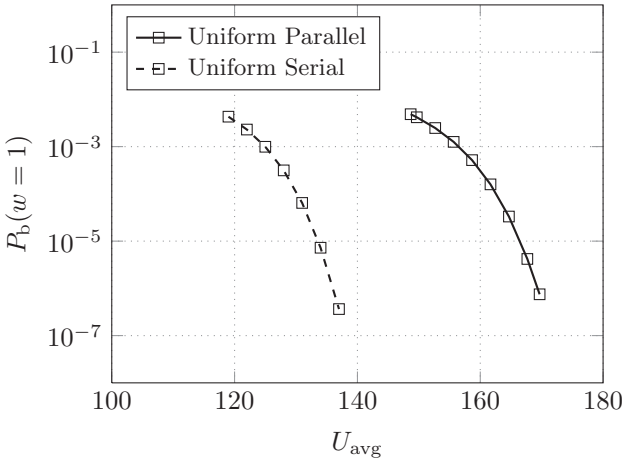


Fig. 3. Density evolution results for the target node error probability ($P_{\mathrm{b}}(w=1)$) as a function of the average number of node updates (U_{avg}) when the uniform parallel and uniform serial window decoding schedules are applied to a $(3,6)$-regular SC-LDPC code, $W=8$, $\sigma=0.923$.

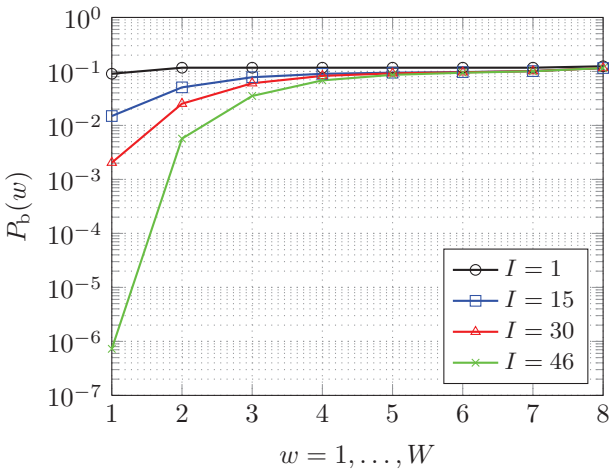


Fig. 4. Density evolution results for $P_{\mathrm{b}}(w), w=1,\ldots,W$, within the decoding window for the uniform parallel schedule with I iterations for a $(3,6)$-regular SC-LDPC code, $W=8$, $\sigma=0.923$.

shows the target node error probability $P_{\mathrm{b}}(w=1)$ as a function of U_{avg}[4] when the uniform parallel and uniform serial window decoding schedules with $W=8$ are applied for an AWGN channel with $\sigma=0.923$. For the uniform serial schedule, before updating the check nodes at position w, all the neighboring variable nodes are asked to produce messages along all the edges connected to check nodes at position w. The results were obtained using density evolution with $P_{\mathrm{b}}^{\max}=10^{-6}$ set as the maximum error probability for the target nodes. The iterative BP decoding algorithm continues until $P_{\mathrm{b}}(w=1)<P_{\mathrm{b}}^{\max}$, with $I_{\max}$ set to 1000. We see that the uniform serial window decoding schedule converges faster than the uniform parallel window decoding schedule. But the gain in decoding convergence speed is much less than a factor of two, whereas a factor of two in complexity reduction occurs when a uniform serial decoding schedule is applied to an LDPC block code or to a SC-LDPC code with block decoding [8].

Following the same setup as in Fig. 3, we now examine $P_{\mathrm{b}}(w), w=1,\ldots,W$, within a window for the uniform parallel window decoding schedule after $I=1,\ 15,\ 30$, and 46 decoding iterations are performed. Results are presented in Fig. 4. Here, after $I=46$ iterations the target node error probability is below the maximum target node error probability, i.e., $P_{\mathrm{b}}(w=1)<P_{\mathrm{b}}^{\max}$. It is observed that P_{b} for the nodes other than the target nodes ($w>1$) changes little with iterations, and similar behavior is observed for the uniform serial window decoding schedule (not shown here). This is due to the fact that the variable nodes on the left side of the window are connected to low-degree check nodes and hence are decoded faster than the other variable nodes in the window. This property of the window decoder motivates the examination of non-uniform window decoding schedules in the next section.

[4]Since the target nodes in the current decoding window were also involved in decoding the previous $W-1$ sets of target nodes, U_{avg} does not start at 0.

III. Non-Uniform Window Decoding Schedules

One possibility to reduce decoding complexity is to switch off the nodes not directly connected to the target nodes after the first few decoding iterations are performed. This avoids unnecessary updates and results in a significant complexity reduction compared to uniform window schedules. In this section, motivated by the results in Fig. 4, we propose several adaptive non-uniform window decoding schedules based on an estimated BER improvement that reduce complexity compared to the uniform window decoding schedules.

A. Estimated BER Improvement Based Non-Uniform Decoding Schedules

Here we propose a non-uniform window decoding schedule that tracks an estimated BER $\hat{P}_{\mathrm{b}}$ within the window and, in the subsequent iteration, updates only those nodes that have shown sufficient improvement.

1) Estimating Bit Error Rate: Before explaining the schedule, we present the method used to estimate the average BER $\hat{P}_{\mathrm{b}}(w)$ of the variable nodes at position w, called a *soft BER estimate* in [4]. The probability that the hard decision for the kth symbol at position w, with output LLR $L_{\mathrm{out}}(k)$, is in error can be estimated as

$$Z(k)=\frac{1}{1+e^{|L_{\mathrm{out}}(k)|}}\ ,$$

termed a *soft bit error indicator* in [19]. Once the soft bit error indicators for all the nodes at position w are available, the soft BER estimates $\hat{P}_{\mathrm{b}}(w)$ can be calculated as the expected value of the random variable Z, i.e.,

$$\hat{P}_{\mathrm{b}}(w)=\mathbb{E}[Z]=\frac{1}{Nn_v}\sum_{k=1}^{Nn_v}Z(k)\ , \tag{5}$$

where Nn_v is the number of variable nodes at position w, for $w=1,\ldots,W$.

2) Decoding Algorithm: Now let $\hat{P}_{\mathrm{b}}^{I}(w)$ denote the soft BER estimates for the nodes at position w within a window at the Ith iteration. Only the nodes for which $\hat{P}_{\mathrm{b}}^{I+1}(w) \leq \theta \cdot \hat{P}_{\mathrm{b}}^{I}(w)$ are updated in the subsequent iteration, where θ defines a pre-specified fractional improvement in estimated BER compared to the last iteration. The choice of θ is discussed in detail in Section IV. A detailed procedure for this non-uniform parallel window decoding schedule is given in Algorithm 1. The inputs to the algorithm are the window size W, the desired maximum error probability $P_{\mathrm{b}}^{\max}$ for the target nodes ($w = 1$), the maximum number of iterations $I_{\max}$ allowed within the window, and the improvement factor θ.

In the initialization phase, the algorithm computes the soft BER estimates $\hat{P}_{\mathrm{b}}(w), w = 1, \ldots, W$, for the nodes within the window using the output LLRs L_{out}. We define a Boolean vector **updateList** of size W so that the nodes at position w are updated in an iteration if **updateList[**w**]**=*true*. The vector **updateList** is initialized to *true* so that all the positions w within the window are updated in the first decoding iteration. The iterative process starts from line 4. The loops at lines 5 and 8 simultaneously update all the check and variable nodes at position w, respectively, for which the **updateList[**w**]** is *true*. The variable and check node updates are performed using (1) and (2), respectively.

Algorithm 1: Estimated BER Improvement based non-uniform parallel decoding schedule

Inputs: W, $P_{\mathrm{b}}^{\max}$, $I_{\max}$, θ

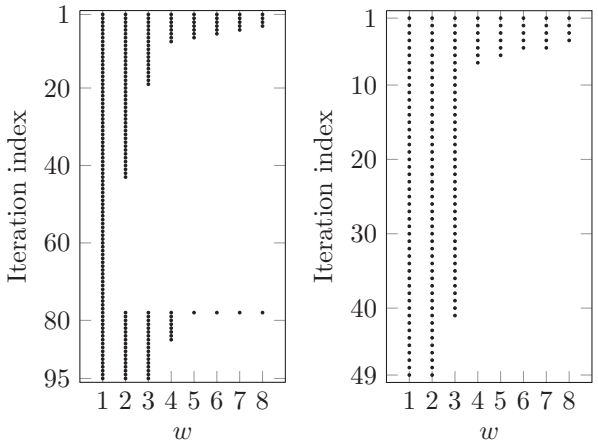
`/* initialization phase */`

1 **for** $w \leftarrow 1$ **to** W **do**
2 $\quad \hat{P}_{\mathrm{b,old}}(w) \leftarrow$ `CalculateSoftBER(`w`);`
3 $\quad$ updateList $[w] \leftarrow$ *true*;

`/* iterations start here */`

4 **for** $i \leftarrow 1$ **to** $I_{\max}$ **do**
5 $\quad$ **for** $w \leftarrow 1$ **to** W **do**
6 $\quad\quad$ **if** updateList $[w]$ **then**
7 $\quad\quad\quad$ `UpdateCheckNodesAt(`w`);`
8 $\quad$ **for** $w \leftarrow 1$ **to** W **do**
9 $\quad\quad$ **if** updateList $[w]$ **then**
10 $\quad\quad\quad$ `UpdateVariableNodesAt(`w`);`
11 $\quad\quad\quad$ `CalculateOutputLLR(`w`);`
12 $\quad\quad$ $\hat{P}_{\mathrm{b,new}}(w) \leftarrow$ `CalculateSoftBER(`w`);`
13 $\quad\quad$ **if** $\hat{P}_{\mathrm{b,new}}(w) \leq \theta \cdot \hat{P}_{\mathrm{b,old}}(w)$ **then**
14 $\quad\quad\quad$ updateList $[w] \leftarrow$ *true*;
15 $\quad\quad\quad$ $\hat{P}_{\mathrm{b,old}}(w) \leftarrow \hat{P}_{\mathrm{b,new}}(w)$;
16 $\quad\quad$ **else**
17 $\quad\quad\quad$ updateList $[w] \leftarrow$ *false*;
18 $\quad$ Reinitialize updateList to *true* if all elements are *false*;
19 $\quad$ *Break* if $\hat{P}_{\mathrm{b,new}}(1) \leq P_{\mathrm{b}}^{\max}$

The function `CalculateOutputLLR(`w`)` calculates L_{out} for the nodes at position w. These are used in `CalculateSoftBER(`w`)` to calculate the new soft BER estimates $\hat{P}_{\mathrm{b,new}}$ for the symbols at position w within the window. The vector **updateList[**w**]** for the next iteration is recalculated based on the new ($\hat{P}_{\mathrm{b,new}}$) and old ($\hat{P}_{\mathrm{b,old}}$) soft BER estimates. The window positions w for which the estimated BER improvement exceeds the parameter θ are updated in the next decoding iteration. The loop at line 4 is terminated if $\hat{P}_{\mathrm{b,new}}(w = 1)$ is less than $P_{\mathrm{b}}^{\max}$. Otherwise, the iterations continue until $I_{\max}$ is reached. Algorithm 1 gives the decoding procedure for the first decoding position. For the following decoding positions, the soft BER estimates in the initialization phase are calculated only for the new incoming nodes in the right side of the window.



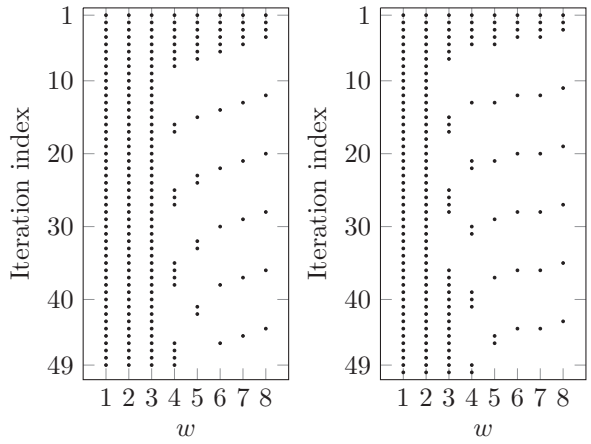
(a) Parallel (b) Serial

Fig. 5. Schedules adopted for estimated BER improvement based non-uniform parallel and serial decoding of a $(3, 6)$-regular SC-LDPC code, $W = 8$, $\sigma = 0.923$.

Figure 5 shows an example of which nodes are updated in each iteration, where all iterations until the soft BER estimate is less than $P_{\mathrm{b}}^{\max} = 10^{-6}$ are shown for both the parallel and serial schedules. In the first few iterations all the nodes show improvement, and hence are updated, but after a certain number of iterations the BER improvement for the nodes on the right side of the window is less than θ, and hence these nodes are not updated in the following iterations. This results in a non-uniform schedule, where the decoding complexity is reduced by eliminating unnecessary updates within the window.

Note that the decoding schedule in Fig. 5(a) has to be restarted at iteration index $I = 77$, i.e., all the nodes are updated at iteration index 77. This is because, at iteration index 76, $P_{\mathrm{b}}^{I=77}(w) \nleq \theta \cdot P_{\mathrm{b}}^{I=76}(w), \forall w = 1, \ldots, W$, while $P_{\mathrm{b}}^{I=77}(w = 1) > P_{\mathrm{b}}^{\max}$, and hence all the nodes must be updated to restart the process.

In general, a potential drawback of using a serial decoding schedule with a parallel implementation is that all the variable and check nodes within the decoding window cannot be updated simultaneously [7]. However, since the decision on updating in the proposed non-uniform serial decoding schedule is made on the basis of all Nn_v nodes at position w within the window, it is possible to update these nodes simultaneously, thus allowing partially parallel operation. Hence, the proposed



(a) Parallel (b) Serial

Fig. 6. Schedules adopted for estimated BER improvement based non-uniform paralel and serial decoding with force update parameter $F_{\mathrm{U}} = 8$ of a $(3, 6)$-regular SC-LDPC code, $W = 8$, $\sigma = 0.923$.

serial schedule is actually a hybrid parallel/serial schedule and can be implemented in a semi-parallel fashion.

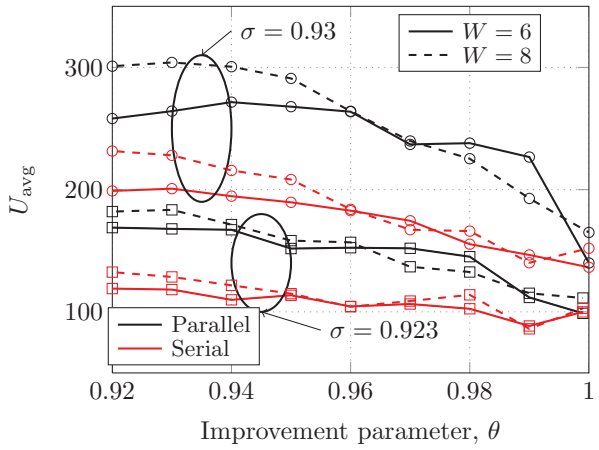
B. Further Complexity Reductions for Non-Uniform Window Decoding Schedules

The non-uniform schedules presented in Section III-A are pessimistic, in the sense that once a node is excluded from the **`updateList`**, it is never updated again (see Fig. 5). One consequence of this is that, in some cases, the iterative process must be restarted, because $\hat{P}_{\mathrm{b}}^{I+1}(w) \not\leq \theta \cdot \hat{P}_{\mathrm{b}}^{I}(w)$ (as in Fig 5(a) at $I = 76$) for all positions within the decoding window. Alternatively, we can force updates on any window positions w that have not been updated in some number F_{U} of previous iterations.

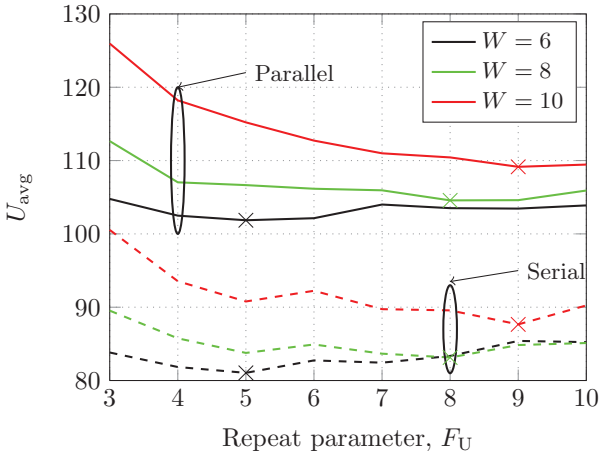
Figure 6 shows the schedules adopted when a force update parameter $F_{\mathrm{U}} = 8$ is applied with a window size $W = 8$, for both the parallel and serial schedules with $P_{\mathrm{b}}^{\max} = 10^{-6}$. It is observed that, for the parallel schedule, the number of iterations in a window is reduced by roughly half (see Fig. 6(a)) compared to those shown in Fig. 5(a), thus improving its decoding speed. On the other hand, the use of a force update parameter has less of an effect on the serial schedule (see Figs. 5(b) and 6(b)).

IV. Asymptotic Performance Evaluation

In this section, density evolution is used to evaluate the performance of the proposed non-uniform window decoding schedules presented in Section III with $P_{\mathrm{b}}^{\max} = 10^{-6}$ and $I_{\max} = 1000$ for an AWGN channel. The finite length computer simulation are presented in Section VI. Specifically, we investigate the effect of two parameters: 1) the fractional improvement parameter θ, and 2) the force update parameter F_{U}. We begin by determining the values of θ and F_{U} that minimize decoding complexity. Then, using the optimum choices of θ and F_{U}, the decoding complexity of the proposed non-uniform window schedules is compared to the uniform window schedules.



(a) For various values of θ, $W = 6,\ 8$, $\sigma = 0.923,\ 0.93$



(b) For various values of F_{U}, $W = 6,\ 8,\ 10$, $\sigma = 0.923$

Fig. 7. Decoding complexity for non-uniform parallel and serial decoding schedules for a $(3, 6)$-regular SC-LDPC code with $L = 100$.

A. Optimum Parameter Choices

1) Fractional Improvement Parameter θ: The decoding complexity of the estimated BER improvement based non-uniform decoding schedules depends on the fractional improvement parameter θ. To find an optimum choice of θ, we compare the decoding complexity for different values of θ for $W = 6$ and $W = 8$ in Fig. 7(a). We observe that choosing $\theta = 0.99$ works best for the serial schedule[5] for the different values of the channel parameter and the window size W. Hence we choose $\theta = 0.99$ for all subsequent evaluations.

2) Force Update Parameter F_{U}: Another parameter that governs the decoding complexity is the force update parameter F_{U}. Choosing F_{U} close to 0 will make the non-uniform decoding schedule behave similarly to the uniform schedule, since all nodes will be updated in nearly every decoding iteration. Similarly, a large value of F_{U} will limit the effect

[5]On the other hand, for the parallel schedule, values of θ closer to 1 may result in less complexity. Choosing $\theta = 0.99$ in this case may be suboptimal, but the difference will be significant only in the low E_b/N_0 (or high BER) region of the BER curve. Additionally, we have observed that choosing $\theta > 0.99$ results in large complexity compared to $\theta = 0.99$ in finite length computer simulation (results not shown here). Hence our choice of $\theta = 0.99$ is justified.

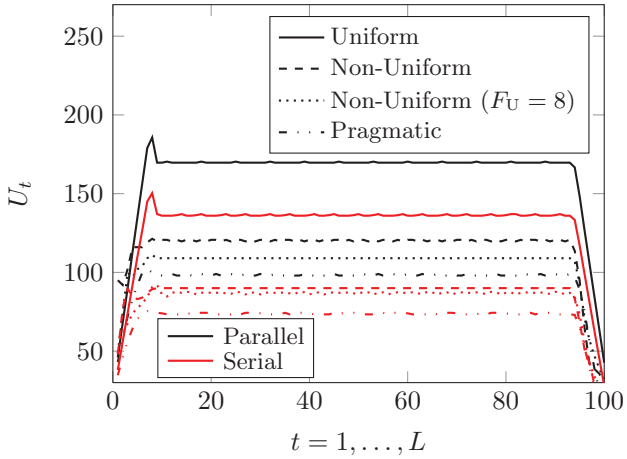


Fig. 8. Number of updates U_t for symbols at time $t = 1, \ldots, L$ for uniform and non-uniform window decoding schedules (both parallel and serial) for a $(3, 6)$-regular SC-LDPC code, $W = 8, \sigma = 0.923, L = 100$.

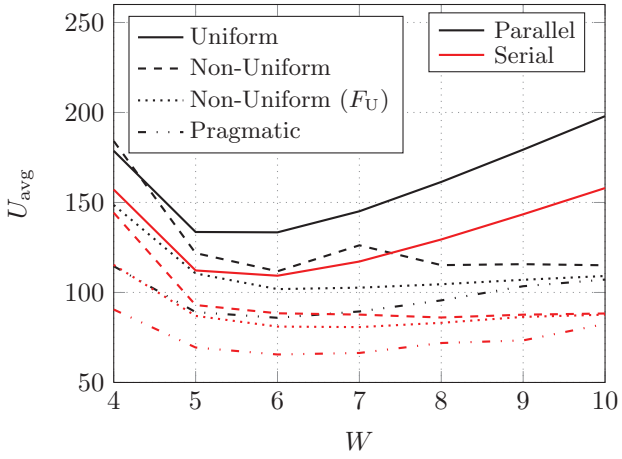


Fig. 9. Decoding complexity as a function of the window size W for a $(3, 6)$-regular SC-LDPC code, $\sigma = 0.923, L = 100$.

of forcing updates, and the decoding complexity will resemble that of the non-uniform schedules in Fig. 5. Figure 7(b) shows the decoding complexity as a function of F_{U} for the non-uniform parallel and serial schedules. The value of F_{U} that results in minimum decoding complexity is marked with an 'x' and is in most cases equal (or very close) to the window size. As an example, for the parallel schedule, for $W = 8$ the optimum value of F_{U} is 8, whereas for $W = 6$ and $W = 10$, the optimum values of F_{U} are 5 and 9, respectively, and the non-uniform serial schedule exhibits similar behavior. Hence we choose $F_{\mathrm{U}} = W$ for all subsequent evaluations.

B. Comparison of Uniform and Non-Uniform Schedules

1) Complexity as a Function of Time: Figure 8 compares the number of updates U_t as a function of time t for the uniform and non-uniform parallel and serial decoding schedules. It is observed that the deactivation of nodes within the window for the non-uniform schedules results in a reduction in decoding complexity compared to the uniform schedules. For the non-uniform parallel schedule, forcing updates on the nodes that had stopped showing BER improvement reduces the complexity by about 10% compared to the non-uniform parallel schedule without forced updates, a result of the fact that the number of required iterations was reduced by almost 50%. For the non-uniform serial schedule, the reduction in complexity with forced updates is less than 10%.

In general, the number of required iterations for the first decoding position ($t = 1$), for both the uniform and non-uniform schedules, is more than for the later positions, since previously calculated messages are reused when the window slides. Hence, for the uniform schedules, since all the nodes are updated at every iteration, a jump[6] in U_t is observed in Fig. 8 at $t = W = 8$. On the other hand, for the non-uniform schedules, since unnecessary updates (especially on the right side of the decoding window) are avoided, we see only a slight jump in complexity. Finally, comparing the non-uniform serial decoding schedule with forced updates to the uniform parallel decoding schedule, we observe that a complexity reduction of about 50% is obtained.

2) Complexity as a Function of Window Size: It has been shown that the BER performance of a window decoder improves as the window size increases [4]. However, for uniform window schedules (both parallel and serial), the decoding complexity increases with W (see Fig. 9). This is due to the overlapping nodes in successive window positions, which results in unnecessary updates within a window and hence increased decoding complexity. On the other hand, the complexity for the non-uniform decoding schedules remains nearly constant with W, since the adaptive nature of the non-uniform schedule avoids unnecessary updates within the window. Since W is a decoder parameter, it can be used to obtain a trade-off between decoding latency and BER performance. Compared to uniform window schedules, Fig. 9 suggests that this trade-off has a minimal effect on decoding complexity when the proposed non-uniform window schedules are used.

V. A Pragmatic Non-Uniform Decoding Schedule

The schedules presented in Section III require the calculation of a soft BER estimate after every decoding iteration. Keeping in mind the structure of the window decoder and the conclusions drawn from the performance evaluation results in Section IV, a non-uniform pragmatic decoding schedule (both parallel and serial) is proposed as follows:

- For a window size of W, define a periodic decoding schedule with an iteration period of $T = W$.
- Define the update limit $\mathsf{w}_{\max}(i)$ as the maximum position index within the window to be updated in iteration i, $i = 1, \ldots, T$.
- In the first iteration, update all the positions within the window, i.e., $\mathsf{w}_{\max}(1) = W$.
- In the succeeding $T - 1$ iterations, $\mathsf{w}_{\max}$ is decremented by 1 at each iteration, i.e., $\mathsf{w}_{\max}(i) = W - i + 1$, $i = 1, \ldots, T$.

[6]Since U_t accumulates the number of iterations performed only in the previous $W-1$ decoding positions, the effect of the larger number of iterations at the first decoding position lasts only until $t = W$.

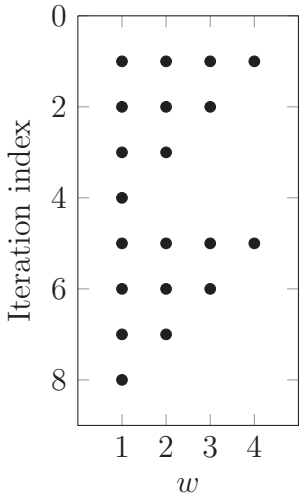


Fig. 10. Two periods of the non-uniform pragmatic schedule for $W = T = 4$.

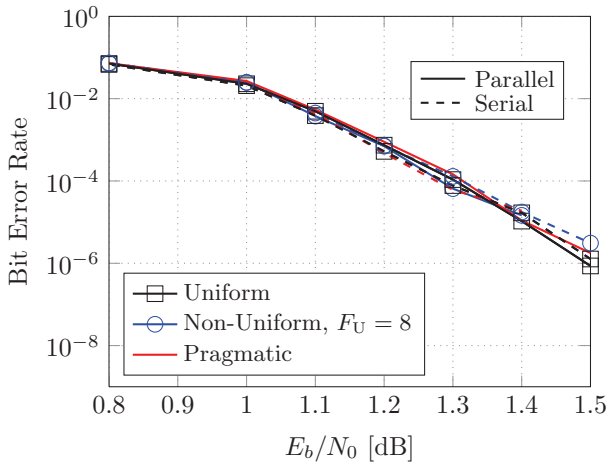
- After T iterations, $\mathsf{w}_{\max}$ is reinitialized to W and the same process continues until $I_{\max}$ is reached.

The decoding schedule is repeatedly applied within a window until $P_{\mathrm{b}}(w = 1) < P_{\mathrm{b}}^{\max}$. This update schedule does not require the calculation of soft BER estimates during the iterations and the decoding schedule is fixed over time. As an example, two periods of the non-uniform pragmatic schedule for $W = T = 4$ are shown in Fig. 10. Note that the node updates in the pragmatic schedule can be performed either in parallel or serially. Figures 8 and 9 show the density evolution results for both the pragmatic parallel and serial schedules. We observe that, for the same channel parameter, the pragmatic parallel and serial schedules result in the lowest complexity compared to all other parallel and serial schedules, respectively.

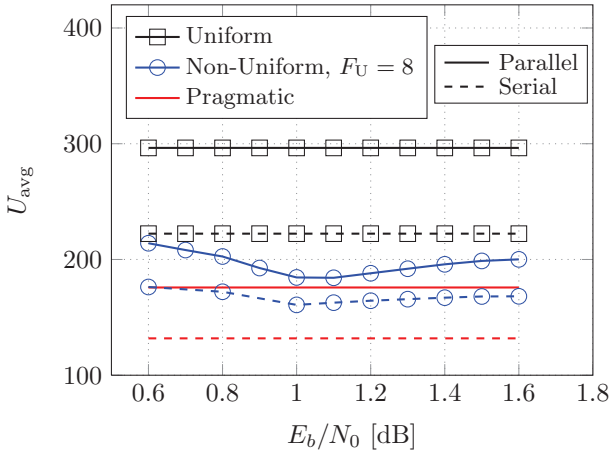
VI. Computer Simulation Performance Evaluation for Finite-Length Codes

In this section, the performance of an example finite-length SC-LDPC code on the AWGN channel is examined using computer simulation. The parity-check matrix with $L = 100$ is generated using the progressive edge growth (PEG) algorithm introduced in [20] with a lifting factor of $N = 500$ and such that cycles of length 4 are avoided.

We choose a window of size of $W = 8$ and apply both the uniform and non-uniform window decoding schedules described in the previous sections, where only non-uniform window schedules with force update parameter F_{U} are considered, since they achieve the smallest complexity. Number of iterations I is selected such that the decoded BER for the various schedules is similar. Since serial schedules converge faster than parallel schedules, we choose a larger value of I for the parallel schedules. Furthermore, we do not apply any stopping criterion for the iterations within a window, i.e., I iterations are always performed within a window. Figure 11(a) shows decoded BER curves for the uniform and non-uniform window decoding schedules. Comparing the uniform and estimated BER improvement based schedules, we see that, for the chosen values of I, there is no significant performance loss for the non-uniform schedules compared to the uniform schedules. The complexity of the decoding schedules is compared in Fig. 11(b), where we see that the non-uniform parallel decoding schedule provides a $35 - 40\%$ reduction in complexity compared to the uniform parallel decoding schedule and that the reduction can be increased to 50% by using the non-uniform serial decoding schedule.



(a) Bit error rate performance



(b) Average complexity

Fig. 11. BER simulation results and average number of node updates U_{avg} on an AWGN channel with $N = 500$ and $W = 8$ for a $(3, 6)$-regular SC-LDPC code. For uniform and pragmatic parallel window schedules, $I = 40$, $I = 30$ for uniform and serial pragmatic window schedules, $I = 50$ for non-uniform parallel window schedule, and $I = 40$ for non-uniform serial window schedule.

Figure 11(a) also shows the decoded BER when the pragmatic parallel and serial window schedules are applied. Both pragmatic schedules show similar performance to the uniform schedules. However, the average decoding complexity for the pragmatic parallel and serial schedules are only $U_{\mathrm{avg}} = 175$ and $U_{\mathrm{avg}} = 130$, respectively (see Fig. 11(b)). Finally, comparing the average decoding complexity of the uniform parallel ($U_{\mathrm{avg}} = 300$) and pragmatic serial ($U_{\mathrm{avg}} = 130$) schedules, a reduction of 55% is achieved, and there is no need to calculate soft BER estimates.

VII. Conclusion

We analyzed the reduction in decoding complexity achieved when non-uniform schedules instead of uniform schedules are applied to sliding window decoding of SC-LDPC codes. The decoding complexity of a latency constrained sliding window decoder is independent of the coupling length L of the code and is bounded above by a function that increases linearly with the window size W. Applying the uniform serial decoding schedule within a window results in only a modest reduction in decoding complexity compared to the uniform parallel decoding schedule. However, in the case of block decoding, serial schedules provide up to a 50% reduction in complexity. So, in order to reduce the window decoding complexity further, we proposed several non-uniform decoding schedules. For the estimated BER improvement based schedules, nodes within a window are switched off once they stop showing an improvement in their soft BER estimate. This results in up to a 50% reduction in complexity without any loss in performance. Furthermore, both parallel and serial non-uniform update schedules were proposed and the conclusions were supported by both density evolution results and computer simulations of finite-length codes. A pragmatic non-uniform decoding schedule was also proposed that does not require any calculations within the decoding process. We observe that the pragmatic serial window schedule reduces decoding complexity by 55% compared to the uniform parallel schedule.

Acknowledgment

The authors would like to thank the reviewers for their constructive comments to improve the presentation of the manuscript.

References

[1] A. Jimenez Felstrom and K. 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] 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.

[3] 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.

[4] N. Ul Hassan, M. Lentmaier, and G. Fettweis, "Comparison of LDPC block and LDPC convolutional codes based on their decoding latency," in *Proc. Int. Symp. on Turbo Codes & Iterative Inf. Proc.*, Aug. 2012.

[5] K. Huang, D. G. Mitchell, L. Wei, X. Ma, and D. J. Costello, "Performance comparison of non-binary LDPC block and spatially coupled codes," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, June 2014, pp. 876–880.

[6] R. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[7] M. Weiner and B. Nikolic, "A high-throughput, flexible LDPC decoder for multi-Gb/s wireless personal area networks," Master's thesis, EECS Department, University of California, Berkeley, Dec. 2010. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-177.html

[8] E. Sharon, N. Presman, and S. Litsyn, "Convergence analysis of generalized serial message-passing schedules," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 1013 –1024, Aug. 2009.

[9] A. Vila Casado, M. Griot, and R. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, June 2007, pp. 932 –937.

[10] I. M. G. Elidan and D. Koller, "Residual belief propagation: informed scheduling for asynchronous message passing," Jul. 2006.

[11] A. I. V. Casado, M. Griot, and R. D. Wesel, "LDPC decoders with informed dynamic scheduling," *IEEE Transactions on Communications*, vol. 58, no. 12, pp. 3470–3479, December 2010.

[12] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," *In IPN Progress Report 42-154, JPL*, Aug. 2003.

[13] 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, Sept. 2015.

[14] M. Lentmaier, G. Fettweis, K. Zigangirov, and D. Costello, "Approaching capacity with asymptotically regular LDPC codes," in *Proc. Inf. Theory and Applications Workshop*, Feb. 2009, pp. 173 –177.

[15] M. Lentmaier and G. Fettweis, "Coupled LDPC codes: Complexity aspects of threshold saturation," in *Proc. IEEE Inform. Theory Workshop (ITW)*, Oct. 2011, pp. 668 –672.

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

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

[18] M. Papaleo, A. Iyengar, P. Siegel, J. Wolf, and G. Corazza, "Windowed erasure decoding of LDPC convolutional codes," in *Proc. IEEE Inform. Theory Workshop (ITW)*, Jan. 2010, pp. 1 –5.

[19] P. A. Hoeher, I. Land, and U. Sorger, "Log-likelihood values and monte carlo simulation – some fundamental results," in *Proc. Int. Symp. on Turbo Codes & Iterative Inf. Proc.*, 2000, pp. 43 –46.

[20] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386 –398, Jan. 2005.



Najeeb Ul Hassan received the B.S. degree from NUCES Lahore, Pakistan in 2007, M.Sc. in Digital Communications from Christian-Albrechts-University of Kiel, Germany in 2010 and the Dipl.-Ing. degree in electrical engineering from Technical University of Dresden, Germany. For his Master's studies, he received an award for a best Master's Thesis in the Faculty and also a DAAD-Prize for the outstanding achievement as a foreign student. Currently he is working as a Post-Doctoral Research Associate at Vodafone Chair for Mobile Communications at the Technical University of Dresden, Germany. His main research interests include forward error correction codes with the focus on design and analysis of the iterative decoding algorithms.



Ali E. Pusane received the B.Sc. and M.Sc. degrees in electronics and communications engineering from Istanbul Technical University, Istanbul, Turkey, in 1999 and 2002, respectively, and the M.Sc. degree in electrical engineering, the M.Sc. degree in applied mathematics, and the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, in 2004, 2006, and 2008, respectively. He was a Visiting Assistant Professor at the Department of Electrical Engineering, University of Notre Dame, during 2008-2009, after which he joined the Department of Electrical and Electronics Engineering, Bogazici University, Istanbul, Turkey, as a faculty member. His research is in coding theory

and applications.



Michael Lentmaier received the Dipl.-Ing. degree in electrical engineering from University of Ulm, Germany in 1998, and the Ph.D. degree in telecommunication theory from Lund University, Sweden in 2003. He then worked as a Post-Doctoral Research Associate at University of Notre Dame, Indiana and at University of Ulm. From 2005 to 2007 he was with the Institute of Communications and Navigation of the German Aerospace Center (DLR) in Oberpfaffenhofen, where he worked on signal processing techniques in satellite navigation receivers. From 2008 to 2012 he was a senior researcher and lecturer at the Vodafone Chair Mobile Communications Systems at TU Dresden, where he was heading the Algorithms and Coding research group. Since January 2013 he is an Associate Professor at the Department of Electrical and Information Technology at Lund University. His research interests include design and analysis of coding systems, graph based iterative algorithms and Bayesian methods applied to decoding, detection and estimation in communication systems. He is a senior member of the IEEE and served as an editor for IEEE Communications Letters from 2010 to 2013 and IEEE Transactions on Communications since 2014. He was awarded the Communications Society & Information Theory Society Joint Paper Award (2012) for his paper "Iterative Decoding Threshold Analysis for LDPC Convolutional Codes".



Gerhard P. Fettweis earned his Ph.D. under H. Meyr's supervision from RWTH Aachen in 1990. After one year at IBM Research in San Jose, CA, he moved to TCSI Inc., Berkeley, CA. Since 1994 he is Vodafone Chair Professor at TU Dresden, Germany, with 20 companies from Asia/Europe/US sponsoring his research on wireless transmission and chip design. In 2012 he received the Honorary Doctorate from Tampere University. He coordinates 2 DFG centers at TU Dresden, namely cf**aed** and HAEC and the 5G Lab Germany.

Gerhard is IEEE Fellow, member of the German academy acatech, and his most recent award is the Stuart Meyer Memorial Award from IEEE VTS. In Dresden he has spun-out eleven start-ups, and setup funded projects in volume of close to EUR 1/2 billion. He has helped organizing IEEE conferences, most notably as TPC Chair of ICC 2009 and of TTM 2012, and as General Chair of VTC Spring 2013, DATE 2014 IEEE 5G Summit 2016.



Daniel J. Costello, Jr. received the M.S. and Ph.D. degrees in Electrical Engineering from the University of Notre Dame, Notre Dame, IN, in 1966 and 1969, respectively. Dr. Costello joined the faculty of the Illinois Institute of Technology, Chicago, IL, in 1969. In 1985 he became Professor of Electrical Engineering at the University of Notre Dame, Notre Dame, IN, and from 1989 to 1998 served as Chair of the Department of Electrical Engineering. In 2000, he was named the Leonard Bettex Professor of Electrical Engineering at Notre Dame, and in 2009 he became Bettex Professor Emeritus.

Dr. Costello has been a member of IEEE since 1969 and was elected Fellow in 1985. In 2009, he was co-recipient of the IEEE Donald G. Fink Prize Paper Award, which recognizes an outstanding survey, review, or tutorial paper in any IEEE publication issued during the previous calendar year. In 2012, he was a co-recipient of the joint IEEE Information Theory Society/Communications Society Prize Paper Award, which recognizes an outstanding research paper in the IT or COM Transactions during the previous two calendar years. In 2013, he received the Aaron D. Wyner Distinguished Service Award from the IEEE Information Theory Society, which recognizes outstanding leadership in and long standing exceptional service to the Information Theory community. In 2015 he received the IEEE Leon K. Kirchner Graduate Teaching Award, which recognizes inspirational teaching of graduate students in the IEEE fields of interest.

Dr. Costello's research interests are in digital communications, with special emphasis on error control coding and coded modulation. He has numerous technical publications in his field, and in 1983 he co-authored a textbook entitled "Error Control Coding: Fundamentals and Applications", the 2nd edition of which was published in 2004.