Low-Complexity Sliding Window Decoding of Spatially-Coupled LDPC Codes Constructed From Short 5G NR LDPC

Abdoul-Hadi Konfé, Pasteur Poda, Raphaël Le Bidan

Low-Complexity Sliding Window Decoding of Spatially-Coupled LDPC Codes Constructed From Short 5G NR LDPC

Abdoul-Hadi KONFÉ
Laboratoire LAMDI
École Polytechnique de Ouagadougou
Ouagadougou, Burkina Faso
ahkonfe@epo.gov.bf

Pasteur PODA
Laboratoire LAMDI
Université Nazi BONI
Bobo-Dioulasso, Burkina Faso
pasteur.poda@u-naziboni.bf

Raphaël LE BIDAN
IMT Atlantique, Dept. MEE
Lab-STICC UMR CNRS 6285
Brest, France
raphael.lebidan@imt-atlantique.fr

***Abstract*—We investigate in this paper how to optimize the number of iterations required for sliding window decoding of low-density parity-check (LDPC) codes. The aim of this study is to reduce decoding complexity and overcome the phenomenon of decoder blocking. We propose a new design of SC-LDPC codes based on some 5G short LDPC codes to reduce decoding complexity. Simulation results demonstrate a potential reduction in decoding computational complexity of the order of 30% without significant degradation in decoding performance. In addition to taking care of complexity, this mitigates the problem of sliding window decoder blocking by reducing the probability of observing this phenomenon.**

***Index Terms*—B5G, complexity, decoder error propagation, short block-length, sliding window (SW) decoding, spatially-coupled low-density parity-check (SC-LDPC) code.**

I. Introduction

The implementation of efficient decoding methods for spatially-coupled low-density parity-check (SC-LDPC) codes using the belief propagation (BP) algorithm is quite known when using block low-density parity-check (LDPC) codes. This method would certainly lead to a significant amount of iterations, resulting in complex decoding due to excessive node updates. In addition, the resulting decoder lacks two essential elements of the convolutional structure. Continuous decoding can be carried out without waiting for a complete block to be received, as the distance between two variable nodes connected to the same control node is also restricted by the code's memory capacity.
The implementation of a sliding window decoder appears as a solution to the above mentioned method. It decreases the iteration count and improves decoding efficiency. Recently, sliding window (SW) decoders have been used in satellite communications [1], automatic speech recognition [2] and broadcasting [3]. Although it is widely used in new cutting-edge technologies design, SW decoding also has a few shortcomings which is good to be aware of. The first is the need to perform a large number of window updates, resulting in high decoding complexity. And the second is the possibility of the decoder to be disrupted by sporadic error patterns, leading to blocking.
Numerous research works have tackled the problem of high decoding complexity and sporadic error patterns. In works [4],[5], the authors have focused on solving the decoding blockage problem. Some of them [4], [6] have proposed solutions for predicting the occurrence of decoder blocking in order to counter it. This solution reduces decoding complexity by preventing decoder blocking through prediction. Other related techniques [7], [8] have been established to automate blocking prediction. However, it should be noted that a shortcoming of this approach is that prediction generates additional computations which, even smaller, contribute to increasing decoding complexity. It is therefore necessary to evaluate and compare these levels of complexity in order to determine precisely the proportion of complexity saved. In work [8], strategies for occasionally reducing or inserting the number of degree control nodes are proposed to counter the blocking problem. They reduce decoding blocking and are innovative, being the first of their kind. However, this technique entails a loss of efficiency due to the manipulations carried out at node level. The use of the *tailbiting* [9] technique is a solution to keep this performance loss as low as possible. In studies [10], [11] and [12], three design techniques are suggested to mitigate the impact of blocking. They concern firstly an extension of the window size, then resynchronization and finally retransmission. A performance improvement is observed with a reduction in computational

979-8-3503-7786-6/24/$31.00 ©2024 IEEE

complexity. The problem with this design is the loss of performance caused by termination. The technique used is *braided*, which is not known to be effective in reducing rate-loss. The use of the *tailbiting* technique to replace the *braided* [13] technique could reduce blocking while maintaining negligible rate-loss.

Also, the computational complexity of using resynchronization and retransmission in the process needs to be evaluated to check whether they are really negligible, as any retransmission requires additional operations. In [14], the authors propose a low-complexity design using the three strategies of window size, window update strategy and early termination based on parity checking. This design turns out to be particularly better when it comes to a hardware implementation aimed at reducing complexity through a judicious choice from existing methods in the literature. In reality, it does not introduce any new design techniques. It would be interesting to apply this method to codes commonly used in mobile communications systems, for example, in order to determine how far this complexity reduction can go in real-life situations. In general, several contributions have been suggested to enhance the decoding of SC-LDPC codes through the implementation of the SW decoding method. Some of them aim to reduce the effects of error propagation, while others seek to reduce computational complexity and decoding latency.

In this paper, we present a method that will significantly reduce computational complexity with the aim of further improving the decoding of the relatively short codes used in modern communications systems.

Our article is divided into four parts. The second section is an introduction to SW decoding of LDPC codes and a presentation of decoding algorithms using matrices and protographs. The third section focuses on the proposed decoding method's principle. It also includes an analysis of the simulation results. The final section presents a summary of the main conclusions.

II. Sliding window decoding of LDPC codes

A. Introduction to window decoding methods

Another way of decoding SC-LDPC codes is to use a variant of the BP decoding algorithm called sliding window decoding [15], [16]. The term sliding window means that windows move along the Tanner graph to update messages across branches. In other words, the BP algorithm is applied only to the control and variable nodes included in a predefined window. An advantage of SW decoding is the reduction of decoding latency through window-adaptive scheduling techniques. This is important in 5G communications, as latency must be less than one millisecond for delay-sensitive scenarios.

To exploit the convolutional properties of the error control matrix, it is possible to implement a continuous SW iterative decoder. This decoder operates on a window of variable nodes of size $w = I.v_s$, where I represents the number of decoding iterations to be performed and v_s designates the constraint length of the code. This decoder consists in dividing the sequence to be decoded into windows of fixed size w, over which the BP decoding algorithm is executed with a constant number of iterations. The window is then shifted by one layer and the process repeated. In this way, for each window, layers already decoded will help subsequent layers to be decoded more easily. The latter, once decoded, will in turn help subsequent layers to decode properly when the window is moved. In a similar way, SW decoding makes use of the fact that the bit error rate or error rate per block per position converges in a wave-like manner [4]. Put differently, decoding of subsequent blocks is contingent upon the successful decoding of preceding blocks. Thus, updating nodes in a few positions and shifting the active decoding window suffice whenever a specific block is successfully decoded or when reaching the maximum iteration limit. Moreover, unlike the standard BP algorithm, the SW decoding algorithm requires knowledge the positions engaged during the decoding process. Undulatory convergence means that the Bit Error Rate (BER), or the error rate per block and position, tends to fluctuate smoothly with the iteration count performed on each window. This indicates a dependency between successive decoding operations.

B. Standard BP versus sliding window decoding

There are subtle nuances between these two reference decoding methods when it comes to efficient decoding of LDPC codes. Compared with conventional BP codes [17], SC-LDPC codes [18]-[21] can be iteratively decoded using conventional decoding algorithms. Reliable messages coming from the endpoints are then propagated across the graph to the center over the course of iterations. SC-LDPC have recently been used to construct LDPC codes for 5G with improved performance [22]. The BP algorithm is an decoding technique based on the exchange of information between the nodes of a graph, where nodes communicate with each other by propagating "beliefs" about bit values.

With SW decoding, decoding only updates the nodes of a window that has been localized, and then the window moves through the graph. It is important to note that SW decoding doesn't update all nodes simultaneously. At each window location, a target symbol block is decoded, then the window is moved to the right. In other words, this technique divides the graph into smaller windows and updates only the nodes available in the current window. Approaches offering bidirectional window movement have recently emerged [4], [23].

In summary, the main difference between the BP decoding technique and the SW technique lies in the way information is propagated across the graph branches and updated at node level. The BP algorithm works globally on the whole graph,

while the SW technique focuses on parts of the graph at a time. This can have an impact on the complexity of the algorithm and, above all, on its efficiency.

C. Sliding window decoding using protographs

We present the key points of the SW decoding algorithm, using a graphical method based on protographs [24], [25], which proves very effective for visualizing the decoding steps. L being the coupling factor, at each window position $p \in \{1, L\}$ and for a defined number of iterations I_p^{wind}, we identify three major steps in the decoding process. First, let's assume that all variable nodes c_n at positions $p \in \{p, p+W-1\}$ constitute the set of variable nodes to be updated. W represents the maximum conceivable window size and is defined as $L+w$. At position $t \in \{p, p+W-1\}$, consider that all control nodes p_m form the set of updated variable nodes. Secondly, updates of control nodes and variable nodes for these sets, according to BP decoding rules with flood scheduling, are performed. We recall that in this scheduling scheme, at each iteration of the decoding process, the control links are firstly updated, then the variable links. For variable node updating, we establish relationships (1), (2), and (3) for all c_n variable nodes and for all $n \longrightarrow m$ branches.

$$T_{n\to m} = I_n + \sum_{\substack{m\neq m' \\ m'\in\mathcal{M}(n)}} E_{m'\to m} \tag{1}$$

$$T_n = T_n - E_{m\longrightarrow n} \tag{2}$$

$$T_n = \sum_{m'\in\mathcal{M}(n)} E_{m'\to m} \tag{3}$$

In the case of updating control nodes for all control nodes p_m and for all branches from m to n we derive the following equations (4) and (5):

$$E_{m\to n} = -2\tanh^{-1}\left(\prod_{\substack{n\neq n' \\ n'\in\mathcal{N}(n)}} \tanh\left(-\frac{T_{n'\to m}}{2}\right)\right) \tag{4}$$

$$E_{m\to n} = \prod_n \text{sign}(-T)\varphi\left(\sum_n \varphi(x_n)\right), \tag{5}$$

avec $\varphi(x) = ln(\frac{e^{|x|}+1}{e^{|x|}-1})$

The iteration number I_p^{wind} and the size of the window W must be selected so that the probability of target error is reached at location $t = p$(where p is the window position). For a given value of W, we can observe that the likelihoods $P_b(t)$, for $t \in \{p, p+W-1\}$, inside the window converge towards a fixed point when I_t tends towards infinity (a fixed point being a value which, when used as input for an iterative function, produces the same value on output). The benefit of the SW decoder lies in the fact that latency is defined via the size of W. Thus, both the decoding complexity and latency remain unaffected by the coupling factor L. Fig. 1 illustrates the principle of SC-LDPC code decoding using a SW. Notation explanations are:

- $T_{n\to m}$: represents a variable that describes the update from one variable node c_n to another variable node c_m.
- I_n: represents the initial information associated with variable node n.
- $M(n)$: the set of branches starting from node n, these neighboring nodes are those directly connected to n.
- $E_{m'\to m}$: amount of information from all variable nodes m that are connected to n and are not equal to m.
- $sign(x)$: this function returns -1 if x is negative, 0 if x is zero, and 1 if x is positive.
- $\sum_{\substack{m\neq m' \\ m'\in\mathcal{M}(n)}} E_{m'\to m}$: this is a summation over all neighboring nodes m of n, with the exception of node m.

D. Sliding window decoding using matrices

Here's how the SW decoding algorithm can be described for the basic matrix method [19], [26].

- Initially, the window described using the spatially coupled base matrix (B_{SC}), has a fixed size measured in symbols, equivalent to $(d_2 - d_1)W \times (d_2 - d_1)W$. It moves from instant $t = 1$ to instant $t = L$. W, the window size, corresponds to the total number of column blocks of size d_2 included in the window.
- Secondly, at each instantaneous position per window, the BP scheduler executes until a predetermined total number of iterations has occurred, or an early-stop condition has been satisfied. Recall that the size of the base matrices of the B_i components is $(d_2 - d_1) \times d_2$.
- Next, the window offsets the d_2-column symbols, and the d_2-column symbols offset out of the window are finally decoded. Target symbols are the first d_2 column blocks inside a window.
- In addition, it is assumed here that all variable and control nodes in a window will be updated at every iteration, and messages will be passed to the window from previously decoded symbols where appropriate. Moreover, it should be noted that the biggest feasible window size W is equal to position $(L+w)$, where the entire parity control matrix is spanned. SW decoding is therefore the same as BP decoding with *flooding* scheduling.
- Finally, the smallest possible window size W is at position $(w+1)$, which means that during SC-LDPC code decoding, the window length should be at least equal to the restraint length.

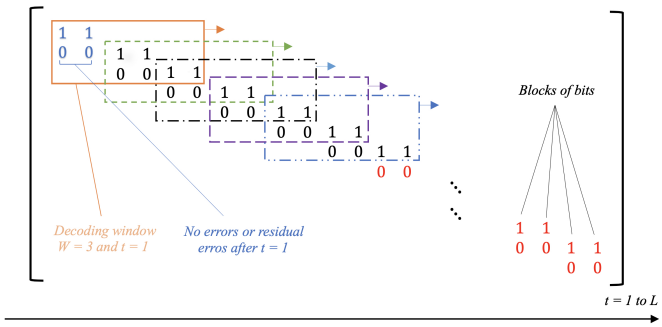

Fig. 1. Illustration of sliding window decoding of SC-LDPC codes.

III. Reduced decoding complexity

A. Method and principle of the proposed solution

In this section, we propose a method for decoding SC-LDPC codes constructed on the basis of some short 5G LDPC codes. The SW decoding method is used because of its low latency and reduced decoding complexity compared to BP decoding.

This method could alleviate the multiple shortcomings of using a standard SW decoder, mainly catastrophic block errors, which will naturally reduce decoding latency and complexity. To further reduce decoding complexity and latency, we define a stopping criterion based on the calculation of the Average Number of Iterations (NIM) required to correctly decode a codeword. This number will enable us to know, for a given Signal to Noise Ratio (SNR), over a given decoding window, how to adjust the iteration count to decrease complexity of SW decoding without significantly compromising decoding performance. The NIM is computed during simulations, considering the ratio of iterations used in a particular decoding step to the total iterations, and for a specified SNR. In essence, our decoder, IAID (Incrementally Adaptative Iterations Decoder), is based on the concept of progressively reducing the number of iterations (Fig. 2) to avoid decoder deadlocks, which can lead to serious errors and considerably impair the performance of the SW decoder. In practice, we find that fewer iterations are needed to correctly decode a codeword at high SNRs, while more iterations are required at low SNR values.

As far as we know, there hasn't been any assessment in the literature regarding incrementally adaptive iterations or a stop criterion strategy determined by averaging the number of iterations. We therefore carried out our simulations and compared our results with existing methods in the literature to see if we gain in computational complexity. For our simulations, we'll be using spatially-terminated coupled codes, which are codes used in practical systems and result in superior decoding performance. To gain complexity, we'll consider that the decoding window in SW decoding moves unidirectionally. While proposals exist for specialized SW decoding schemes, where the window moves bidirectionally to prevent decoder stagnation, these designs do come at the cost of increased decoding operations. In addition, several efficient window decoding strategies have been proposed in the literature to reduce decoding complexity by avoiding message passing or unnecessary iterations [27]. Similarly, as error correction capabilities improve with small memory m_s and large block size n (but not very large to stay within the framework of relatively short 5G LDPC codes), we opt to work with unit-memory SC-LDPC codes (i.e. $m_s = 1$).

In a broad sense, we can identify three primary factors that dictate the decoding efficiency and computational intricacy of unidirectional SW decoding:

- Adjusting the size of the decoding window regulates the volume of information processed simultaneously.
- Modifying the window update strategy governs the sections of the decoding window that undergo updates.
- the stop criterion to interrupt decoding before the upper limit of permissible iterations is reached, thus reducing computational complexity.

In our work, we focus exclusively on two aspects, namely incrementally adaptive iteration (based on strong and weak SNRs) and the stopping criterion (based on predetermining the NIM). It's worth mentioning that in the existing literature, most of the stopping criteria used to reduce decoding complexity rely on the reliability of code symbols, i.e. approaches based on the log likelihood ratio (LLR) [28]. References [14] implement a stopping criterion wherein the decoding window shifts to the subsequent position once all parity-check equations (PCE) within the window are fulfilled. We will also observe whether the uniform distribution of "1" in the matrices when designing our SC-LDPC codes (uniform coupling) or the non-uniform distribution (non-uniform coupling) will further reduce decoding complexity while maintaining acceptable decoding performance.

The PCE method investigates the satisfaction of the PCE linked to the actual window. In parallel, the LLR-based method generally approximates the mean soft BER as a function of the LLR output of every variable. We use the approach based on PCE, as it is generally less complex than the LLR-based approach. Once all PCE within the window are fulfilled, the decoding window transitions to the next position.

B. Estimation of average number of iterations

The average number of iterations (NIM) in SW decoding of SC-LDPC codes depends on several factors, including SNR, the specific code used, relative position in the decoding window, coding rate and channel type. Determining the NIM makes it possible to optimize decoding parameters according to the performance requirements of a specific need, while reducing complexity.

To determine the NIM, we create simulation scenarios that

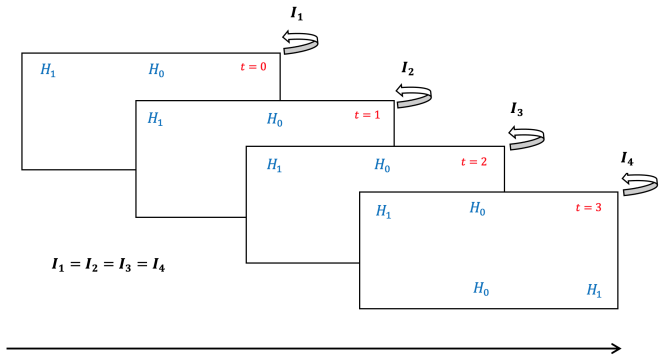

Fig. 2. Standard iterations in sliding window decoding.

take into account the above-mentioned parameters. Next, we perform several iterations for each scenario, which will decode the code words received and count the number of iterations required until the solution converges, i.e. no errors detected. Finally, by averaging the simulation results obtained over a large number of scenarios, we obtain an estimate of the NIM. For obvious pratical considerations, a fractional NIM is replaced by the nearest integer value.

Adapting the number of decoding iterations according to position in a SW decoding system is an effective strategy for optimizing decoding performance while minimizing computational complexity. This adaptation is useful in the context of SC-LDPC codes, where the code structure allows for a gradual improvement in bit error probability as decoding progresses along the chain. To adapt the number of decoding iterations and reduce the probability of a blocking error occurring, convergence monitoring is used. The method involves tracking how well the decoding process is converging at each stage and modifying the count of iterations accordingly. If the decoder satisfies a convergence criterion (a zero-syndrome condition), it reduces the number of iterations for subsequent positions. After each iteration, the syndrome is calculated. If it is zero, the block is correctly decoded, and we can move on to the next block with potentially fewer iterations.

Adapting the number of iterations could play an important role in reducing the probability of a blocking error occurring. If a fixed number of iterations is used for all decoding, there is a risk that this number may be insufficient to converge on a correct solution in certain cases, particularly when the SNR is low. In such cases, the decoder may converge on an incorrect solution, leading to a blocking error. Adapting the number of iterations based on SNR or position in the decoding window can enhance the decoder's convergence on the correct solution. For example, low SNR requires more iterations to correct errors, while high SNR needs fewer. A pseudo-code for NIM estimation is proposed below:

Average number of iterations

```
SNR_values, number_simulations, maximum_iterations
Estimate_NIM (SNR_values, number_simulations, maximum_iterations)
for each SNR in SNR_values do
    total_iterations = 0
    for i from 1 to number_simulations do
        total_iterations = decode (SNR, maximum_iterations) +1
    end for
    results [SNR] = total_iterations / number_simulations
end for
return results
decode (SNR, maximum_iterations)
initialize (SNR)
for iteration from 1 to maximum_iterations do
    if iteration_successful() then
        return iteration
    end if
end for
return maximum_iterations
main
NIM_results = Estimate_NIM (SNR_values, number_simulations, maximum_iterations)
display (NIM_results)
```

C. *Different simulation scenarios and results*

In this subsection, we first present the simulation scenarios and the results obtained. The results are then analyzed and followed by a discussion. The principles of classical decoding [16] and of our proposed version are summarized in Fig. 1 (in the standard approach, we usually have $I_1 = I_2 = I_3 = I_4$, whereas in the one we advocate, $I_1 \geq I_2 \geq I_3 \geq I_4$ with $I_1 = NIM_1$, $I_2 = NIM_2$, $I_3 = NIM_3$ and $I_4 = NIM_4$). Below, we illustrate in Fig. 2 the average number of iterations for decoding SC-LDPC codes constructed on the basis of some short 5G LDPC codes.

The SW decoder with code length 128 and coding rates of $1/2$, $1/4$, $4/5$, $2/3$ and $3/4$ is used. For an SC-LDPC code of length 128, coding rate $3/4$ and 200 iterations, we give some of the NIM values obtained. Simulation is carried out using BPSK (Binary Phase Shift Keying) modulation on the AWGN (Additive White Gaussian Noise) channel. For SNR equal to 0.5, 1.7 and 4.4 we have respectively NIM equal to 191.6429, 74.67257 and 2.203119.

Simulation scenarios: for an SC-LDPC code of length 128 with coding rates $1/2$, $1/4$, $2/3$, $3/4$ and $4/5$ and a maximum of 200 iterations. SNR were tested over the interval $[-1 : 0.2 : 6]$. Results and details are shown in Fig. 3.

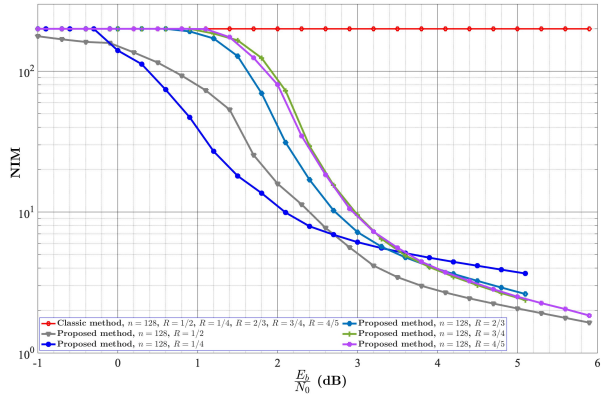

Fig. 3. Reducing number of iterations of SC-LDPC codes with $n = 128$.

The various curves in Fig. 3 show a clear decrease in the mean iteration number as the SNR increases. This results in a decrease in decoding complexity, which correlates with the iteration count. Using (1) and on the basis of the data provided by the various curves, we are able to evaluate the mean decoding intricacy of the decoder using an adaptive incremental SW iteration method. This approach achieves a complexity reduction of around 30% compared with a standard decoder with SW and 200 iterations.

We show in Fig. 4 that our proposed optimization in terms of iteration reduction does not degrade Bit Error Rate (BER) or Frame Error Rate (FER) performances at SNR between $-1dB$ and $2dB$, and only slightly degrades performance at SNR above $2dB$, compared with our decoder using a constant number of iterations. As the reduction we bring about is significant (30%), the small performance degradation after an SNR greater than $2dB$ may be acceptable.

This is comparable to the 34% average decoding complexity reduction reported in [4] (relative to a standard SW decoder with 4 fixed iterations).

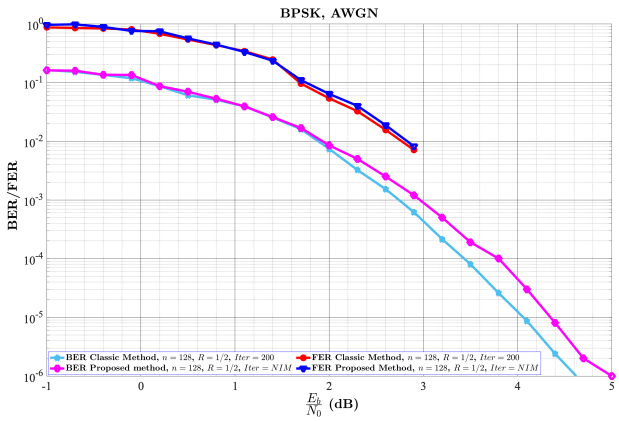

Fig. 4. BER performance graph for our proposed optimization.

Conclusion

In this work, we've formulated an effective method for mitigating the complexity of sliding window decoding. This approach adaptively reduces the number of iterations as a function of Signal to Noise Ratio, values, in order to reduce the computational complexity when using sliding window decoding. By controlling computational complexity in this way, the proposed solution also makes it possible to reduce the decoder's blocking frequency. The proposed algorithm for optimizing the number of iterations to be performed during sliding window decoding has been evaluated in the context of codes constructed on the basis of some short 5G low-density parity-check codes, and has demonstrated a potential reduction in decoding computational complexity of the order of 30%. In addition to addressing complexity, this method mitigates the phenomenon of sliding window decoder blocking by reducing the probability of its occurrence.

References

[1] Y. Zhang, J. Jiao, Y. Wang, Y. Wang, R. Lu and Q. Zhang, "Soft OSD-Sliding Window Decoding for Staircase LDPC Codes in Deep Space Communications," 2023 IEEE/CIC International Conference on Communications in China (ICCC), Dalian, China, 2023, pp. 1-6.

[2] Y. Zhuo, X. Zhou and Q. Wang, "A Sliding Window Method for BATS Code in Space Communication," 2022 International Symposium on Networks, Computers and Communications (ISNCC), Shenzhen, China, 2022, pp. 1-5.

[3] Z. He, K. Peng, J. Song and Y. Zhang, "Efficient Sliding Window Decoding of Spatially Coupled LDPC Codes for Broadcasting," 2020 IEEE International Conference on Electrical Engineering and Photonics (EExPolytech), St. Petersburg, Russia, 2020, pp. 102-105.

[4] K. Klaiber, S. Cammerer, L. Schmalen and S. t. Brink, "Avoiding Burst-like Error Patterns in Windowed Decoding of Spatially Coupled LDPC Codes," 2018 IEEE 10th International Symposium on Turbo Codes Iterative Information Processing (ISTC), Hong Kong, 2018, pp. 1-5.

[5] M. Schlüter, N. U. Hassan and G. P. Fettweis, "On the construction of protograph based SC-LDPC codes for windowed decoding," 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, 2018, pp. 1-6.

[6] M. Zhu, D. G. M. Mitchell, M. Lentmaier and D. J. Costello, "Modeling a Sliding Window Decoder for Spatially Coupled LDPC Codes," 2021 IEEE Globecom Workshops (GC Wkshps), 2021, pp. 1-6.

[7] M. Zhu, D. G. M. Mitchell, M. Lentmaier and D. J. Costello, "Adaptive Doping of Spatially Coupled LDPC Codes," 2020 IEEE Information Theory Workshop (ITW), 2021, pp. 1-5.

[8] M. Zhu, D. G. M. Mitchell, M. Lentmaier and D. J. Costello, "A Novel Design of Spatially Coupled LDPC Codes for Sliding Window Decoding," IEEE ISIT 2020.

[9] S. Cammerer, V. Aref, L. Schmalen, and S. T. Brink, "Triggering wave-like convergence of tail-biting spatially coupled LDPC codes," in Proc. Ann. Conf. Inf. Sci. Syst. (CISS), Mar. 2016, pp. 93–98.

[10] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello and B. Bai, "Combating Error Propagation in Window Decoding of Braided Convolutional Codes," 2018 IEEE International Symposium on Information Theory (ISIT), Vail, CO, 2018, pp. 1380-1384.

[11] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, Jr., and B. Bai, "Combating error propagation in window decoding of braided convolutional codes,"(2018) available at http://arxiv.org/abs/1801.03235.

[12] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello and B. Bai, "Error Propagation Mitigation in Sliding Window Decoding of Braided Convolutional Codes," in IEEE Transactions on Communications, 2020.
[13] W. Zhang, M. Lentmaier, K. Sh. Zigangirov, and D. J. Costello, Jr., "Braided convolutional codes : a new class of turbo-like codes," IEEE Trans. Inf. Theory, vol. 56, no. 1, pp. 316-331, Jan. 2010.
[14] J. Frenzel, S. Muller-Weinfurtner, J. B. Huber and R. R. Muller. "Comparison of windowed-decoder configurations for spatially coupled LDPC codes under equal-complexity constraints," arXiv preprint:2004.12973 (2020).
[15] W. Zhang, M. Lentmaier, K. Sh. Zigangirov, and D. J. Costello, Jr., "Braided convolutional codes : a new class of turbo-like codes," IEEE Trans. Inf. Theory, vol. 56, no. 1, pp. 316-331, Jan. 2010.
[16] A. R. Iyengar, P. H. Siegel, R. Urbanke, and J. K. Wolf, "Windowed decoding of spatially coupled codes," IEEE transactions on Information Theory, vol. 59, no. 4, pp. 2277–2292, 2013.
[17] Dariush Divsalar, Sam Dolinar, Christopher R. Jones, Kenneth Andrews, "Capacity-approaching protograph codes, " IEEE Journal on Selected Areas in Communications, vol. 27, no. 6, august 2009.
[18] A. J. Felström and K. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix,", IEEE Trans. lifo. Theory, vol. 45, no. 6, pp. 2181-2191, Sep. 1999.
[19] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, Jr., "LDPC block and convolutional codes based on circulant matrices," IEEE Trans. Inf. Theory, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.
[20] A. Leven, L. Schmalen, "Status and recent advances on forward error correction technologies for lightwave systems," J. Lightwave Technol. 32(16), 2735–2750 (2014).
[21] D.J. Costello, Jr., L. Dolecek, T.E. Fuja, J. Kliewer, D.G.M. Mitchell, and R. Smarandache, "Spatially coupled sparse codes on graphs: Theory and practice," IEEE Commun. Mag., pp. 168–176, July 2014.
[22] A. -H. Konfé, P. Poda and R. L. Bidan, "Enhancing 5G Forward Error Correction Codes for URLLC by Spatial Coupling," 2022 11th International Symposium on Signal, Image, Video and Communications (ISIVC), El Jadida, Morocco, 2022, pp. 1-6.
[23] S. Abu-Surra, E.Pisek, R.Taori, "Spatially-coupled low-density parity check codes : Zigzag-window decoding and code-family design considerations, " in: Proc. 2015 IEEE Inf. Theory Appl. Work. (ITA), San Diego, CA, USA, 2015, pp. 275–281.
[24] J. Thorpe, "Low-Density Parity-Check (LDPC) Codes Constructed from Protographs," Jet Propulsion Lab, Pasadena, CA, INP Progress Report 42-154, Aug. 2003.
[25] M. Lentmaier, M. M. Prenda and G. P. Fettweis, "Efficient message passing scheduling for terminated LDPC convolutional codes," 2011 IEEE International Symposium on Information Theory Proceedings, St. Petersburg, Russia, 2011, pp. 1826-1830.
[26] L. Wei, D. G. M. Mitchell, T. E. Fuja and D. J. Costello, "Design of Spatially Coupled LDPC Codes Over GF (q) for Windowed Decoding," in IEEE Transactions on Information Theory, vol. 62, no. 9, pp. 4781-4800, Sept. 201.
[27] Q. Wu, L. Lv, Y. Yao and S. Wu, "A Parallel and Memory-Efficient Decoding for Spatially-Coupled LDPC Codes," 2020 IEEE/CIC International Conference on Communications in China (ICCC), Chongqing, China, 2020, pp. 1016-1021.
[28] C. Tang, M. Jiang, C. Zhao and H. Shen, "Design of Protograph-Based LDPC Codes with Limited Decoding Complexity," in IEEE Communications Letters, vol. 21, no. 12, pp. 2570-2573, Dec. 2017.