Improved Sliding Window Decoding of Spatially Coupled Low-Density Parity-Check Codes

Shiyuan Mo, Li Chen

School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China, 510006
Email: moshiy@mail2.sysu.edu.cn, chenli55@mail.sysu.edu.cn

***Abstract*—Spatially coupled low-density parity-check (SC-LDPC) codes can achieve capacity approaching performance with a small message recovery latency due to the sliding window decoding (SWD). Using a partial Tanner graph, the SWD performs iterative message passing until the average error probability $\overline{\mathcal{P}}_e$ of the target symbols falls below a threshold or the maximum iteration number is reached. However, $\overline{\mathcal{P}}_e$ does not decrease monotonically as iteration progresses. This implies the symbol likelihoods that were yielded when the decoding terminates may not be optimal for making decisions. Therefore, this paper proposes an improved SWD (ISWD) for SC-LDPC codes. The proposal monitors the achievable minimum of $\overline{\mathcal{P}}_e$ and stores its associated likelihoods, so that when the decoding terminates the target symbols will be estimated based on the stored likelihoods. Our research shows the ISWD is able to enhance the decoding performance, especially in the waterfall region. It exhibits an asymptotic convergence to the SWD performance. A complexity reducing variant of the ISWD is also proposed to facilitate the decoding but at the cost of error-correction performance.**

***Index Terms*—Belief propagation, LDPC codes, protographs, spatially coupled codes, sliding window decoding.**

I. Introduction

Spatially coupled low-density parity-check (SC-LDPC) codes [1] [2] have galvanized recent interest by its capacity-approaching performance and low message recovery latency. It can be constructed from a block protograph by the so called *graph coupling* and *lifting* processes [3] [4]. Its parity-check matrix has the same structure as LDPC convolutional codes [5] such that the code has memory and can be unterminated. For SC-LDPC codes, the iterative belief propagation (BP) decoding threshold can reach the maximum *a posteriori* (MAP) decoding threshold when the codeword length is sufficiently large [1] [6]. Moreover, the code's minimum distance grows linearly with the codeword length contributing to the removal of error floor [7].

Parity-check matrix of SC-LDPC codes exhibits a diagonal band of nonzero entries, giving way to perform the sliding window decoding (SWD) [6] [8]. After receiving a portion of the entire codeword, BP decoding can start by using the code's partial Tanner graph. The decoding window slides along the diagonal band, estimating codeword symbols set-by-set and featuring a low message recovery latency. The set of symbols that are estimated by a decoding window are called the *target symbols* of the window. A decoding window terminates once the average error probability $\overline{\mathcal{P}}_e$ of the target symbols falls below a threshold or the maximum iteration number is reached. In order to reduce the SWD complexity, non-uniform schedules for BP updates have been proposed in [9] where symbols that show little sign of their soft bit error rate (BER) [10] improvement will be skipped for updates. So far, the existing SWD techniques estimate the target symbols based on the final likelihoods that are yielded when the decoding terminates. However, our recent research has shown that the average error probability $\overline{\mathcal{P}}_e$ does not decrease monotonically with the BP iterations. It implies when a window terminates by reaching the maximum iteration number, $\overline{\mathcal{P}}_e$ may not converge to a minimum value. Consequently, the decisions that are made based on those final likelihoods may not be optimal.

Motivated by this decoding phenomenon, we introduce an improved SWD (ISWD) that monitors the achievable minimum of the average error probability $\overline{\mathcal{P}}_e$. Symbol likelihoods that produce the minimum value will be stored by the decoding system. When the maximum iteration number is reached, the target symbols will be estimated based on their memorized likelihoods. Since a set of symbols will be monitored once it is included by a window, the likelihoods are sampled over several decoding windows. Intuitively, they can yield a more accurate estimation. Our simulation results show the ISWD outperforms the SWD, especially in the BER curve's waterfall region. It converges to the SWD performance as the signal-to-noise ratio (SNR) increases. A complexity reducing variant of the ISWD is also proposed to facilitate the decoding.

II. Preliminaries

This section introduces the preliminaries of the paper, including the construction of SC-LDPC codes and the SWD.

A. SC-LDPC Codes

The construction of an SC-LDPC code can start from a block protograph [4] [11]. Given the block protograph of the (ρ, κ) regular LDPC codes, where ρ and κ are the column weight and row weight, respectively, we can first replicate and label it in a finite series with time instants $t = 0, 1, 2, \ldots, L-1$. For example, Fig. 1(a) shows the block protograph of the (3, 6) regular LDPC codes and Fig. 1(b) shows the series of L labeled block protographs. *Edge spreading* among the L block protographs is then performed to generate a coupled protograph. In particular, edges of the block protograph at time instant t are emanated from its variable nodes and connected to the check nodes of block protographs at time instants t, $t+1$,

$\ldots$, $t+w$, where w is called the *coupling width*. Fig. 1(c) shows this process with $w=2$. Applying this edge spreading to all of the L block protographs, we can obtain a coupled protograph that is shown in Fig. 1(d), where L can now be called the *coupling length*. In the coupled protograph, extra w check nodes are required for terminating the spreading edges.

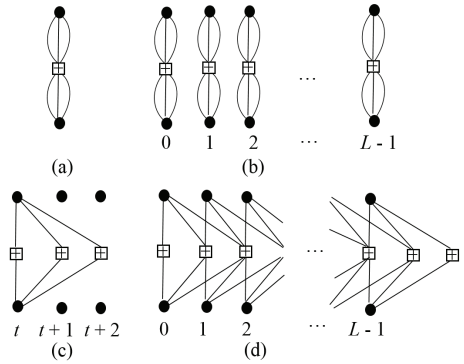

Fig. 1. (a) Block protograph of the (3, 6) regular LDPC codes, (b) a series of L block protographs, (c) edge spreading with $w=2$ at time t, and (d) coupled protograph of the terminated SC-LDPC codes with $w=2$.

A block protograph with α check nodes and β variable nodes can be represented by a base matrix $\mathbf{B}=[\mathbf{B}(i,j)]_{\alpha\times\beta}$, where $1\le i\le\alpha$, $1\le j\le\beta$ and entry $\mathbf{B}(i,j)$ indicates the number of edges connecting check node i and variable node j. For example, the block protograph of Fig. 1(a) can be represented by matrix $\mathbf{B}=[3\ 3]$. Consequently, spreading edges among $w+1$ block protographs can be described by decomposing $\mathbf{B}$ into $w+1$ submatrices of the same size and

$$\mathbf{B}(i,j)=\sum_{\mu=0}^{w}\mathbf{B}_{\mu}(i,j). \tag{1}$$

As a result, the coupled protograph can be represented by the so called SC base matrix

$$\mathbf{B}_{\text{SC}}=\begin{bmatrix}
\mathbf{B}_0 & & & & & & \\
\mathbf{B}_1 & \mathbf{B}_0 & & & & & \\
\vdots & \vdots & \ddots & & & & \\
\mathbf{B}_w & \mathbf{B}_{w-1} & \cdots & \mathbf{B}_0 & & & \\
 & \ddots & \ddots & \ddots & \ddots & & \\
 & & \mathbf{B}_w & \mathbf{B}_{w-1} & \cdots & \mathbf{B}_0 & \\
 & & & \ddots & \vdots & \vdots & \\
 & & & & \mathbf{B}_w & \mathbf{B}_{w-1} & \\
 & & & & & \mathbf{B}_w &
\end{bmatrix}. \tag{2}$$

It defines a terminated SC-LDPC code ensemble. The above $\mathbf{B}_{\text{SC}}$ shows the first and last $w\alpha$ check nodes have reduced degrees, which can also be seen in the coupled protograph of Fig. 1(d). This is an important feature to realize the excellent BP decoding thresholds of SC-LDPC codes [4] [6].

Let $\mathbf{B}_{\text{SC}}=[\mathbf{B}_{\text{SC}}(i,j)]_{(L+w)\alpha\times L\beta}$, where $1\le i\le(L+w)\alpha$ and $1\le j\le L\beta$. Parity-check matrix $\mathbf{H}_{\text{SC}}$ of an SC-LDPC code can be further obtained by lifting $\mathbf{B}_{\text{SC}}$ with a factor of M. In particular, if $\mathbf{B}_{\text{SC}}(i,j)=0$, it will be replaced by a $M\times M$ all zero matrix. Otherwise, it will be replaced by a summation of $\mathbf{B}_{\text{SC}}(i,j)$ permutation matrices of size $M\times M$. As a result, the size of $\mathbf{H}_{\text{SC}}$ is $(L+w)M\alpha\times LM\beta$. Design rate of the SC-LDPC code is $R_{\text{SC}}=1-\frac{(L+w)M\alpha}{LM\beta}=1-\frac{(L+w)\alpha}{L\beta}$. Similar to $\mathbf{B}_{\text{SC}}$, $\mathbf{H}_{\text{SC}}$ exhibits a diagonal band of nonzero entries. The maximum width of this diagonal band is $(w+1)M\beta$, which is called the code's *constraint length*.

B. Sliding Window Decoding

Assume that codeword $\underline{c}=(c_1,c_2,\ldots,c_N)$ is transmitted over a discrete memoryless channel using BPSK modulation, where $N=LM\beta$. Let $\underline{y}=(y_1,y_2,\ldots,y_N)$ denote the received symbol sequence. With y_j, the log-likelihood ratio (LLR) of each codeword symbol c_j can be initialized by

$$\gamma_j=\ln\left(\frac{\Pr\left(y_j\mid c_j=0\right)}{\Pr\left(y_j\mid c_j=1\right)}\right), \tag{3}$$

where $\Pr\left(y_j\mid c_j=0\right)$ and $\Pr\left(y_j\mid c_j=1\right)$ are the channel observations.

Let $\mathbf{H}_{\text{SC}}=[\mathbf{H}_{\text{SC}}(i,j)]_{(L+w)M\alpha\times LM\beta}$, where $1\le i\le(L+w)M\alpha$ and $1\le j\le LM\beta$. Subsequently, $I(j)$ and $J(i)$ are defined as $I(j)=\{i\mid\mathbf{H}_{\text{SC}}(i,j)=1,\ \forall\ i\}$ and $J(i)=\{j\mid\mathbf{H}_{\text{SC}}(i,j)=1,\ \forall\ j\}$, respectively. In order to specify the iterative BP decoding output, let $\gamma_j(u)$ denote the LLR of c_j that is yielded by iteration u. At the beginning, they are initialized as

$$\gamma_j(0)=\gamma_j,\ \forall\ j. \tag{4}$$

At iteration u ($u>0$), the extrinsic LLR w.r.t. channel input is determined by [12]

$$\Phi_j(u)=\sum_{i\in I(j)}2\tanh^{-1}\left(\prod_{j'\in J(i)\backslash j}\tanh\left(\frac{\gamma_{j'}(u-1)}{2}\right)\right). \tag{5}$$

The *a posteriori* LLR of c_j can be further determined by

$$\gamma_j(u)=\gamma_j+\Phi_j(u). \tag{6}$$

They can be transformed into the *a posteriori* probabilities (APPs) as

$$\mathcal{P}_{j,0}(u)=\frac{1}{1+e^{-\gamma_j(u)}},\mathcal{P}_{j,1}(u)=\frac{1}{1+e^{\gamma_j(u)}}, \tag{7}$$

which denote the APPs of c_j being 0 and 1, respectively. Decision will be made based on the APPs such as $\hat{c}_j=0$ if $\mathcal{P}_{j,0}(u)>\mathcal{P}_{j,1}(u)$, or $\hat{c}_j=1$ otherwise. Since the SWD estimates a portion of the entire codeword at a time, we cannot use $\mathbf{H}_{\text{SC}}$ to validate the decoding estimation so as to terminate the decoding. The following describes the SWD.

Let W denote the size of decoding window which contains W block protographs, where $w+1\leqslant W\leqslant L$. In $\mathbf{H}_{\text{SC}}$, the decoding window covers $WM\alpha$ rows and $WM\beta$ columns as shown in Fig. 2 which illustrates the SWD mechanism. The SWD is performed based on the partial Tanner graph framed by the window. That says in calculating the extrinsic LLR of (5), only the check nodes that are included by the window will be considered. The first $M\beta$ symbols of the window are its target symbols. They are associated with the first block

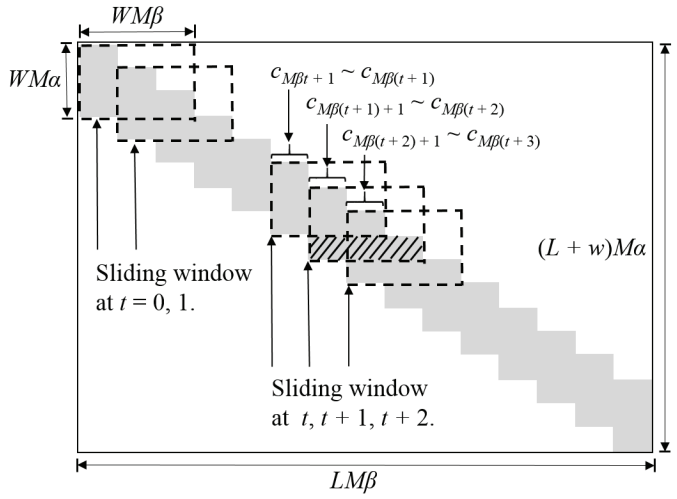

Fig. 2. SWD of an SC-LDPC code with $w = 2$ and $L = 15$. Window size $W = 3$ and the gray area indicates the nonzero diagonal band in the $\mathbf{H}_{\text{SC}}$.

protograph in the window. After iteration u, error probability of c_j is determined by

$$\mathcal{P}_{e,j}(u) = \min\left\{\mathcal{P}_{j,0}(u), \mathcal{P}_{j,1}(u)\right\}. \tag{8}$$

Hence, the average error probability $\overline{\mathcal{P}}_{e,t}(u)$ of the target symbols will be

$$\overline{\mathcal{P}}_{e,t}(u) = \frac{1}{M\beta}\sum_{j=M\beta t+1}^{M\beta(t+1)} \mathcal{P}_{e,j}(u), \tag{9}$$

where t identifies the set of target symbols and $0 \le t \le L-1$. Note that the decoding window slides along the coupled protograph block-by-block. Time instant of a block protograph also indicates the time instant of a decoding window. For the SWD, a decoding window will be terminated once $\overline{\mathcal{P}}_{e,t}(u)$ falls below a predetermined threshold or the maximum iteration number is reached. Estimation of the target symbols $c_{M\beta t+1}, c_{M\beta t+2}, \ldots, c_{M\beta(t+1)}$ will be made based on their likelihoods when the decoding terminates. Afterwards, the window will slide to decode the next $M\beta$ symbols with its time instant updated by $t = t+1$. With $\mathbf{H}_{\text{SC}}$, this process can be seen as sliding along its nonzero diagonal band, which is also demonstrated by Fig. 2. Notice in this paper, we employ the flooding schedule for BP updates.

III. The Improved Sliding Window Decodings

This section introduces the ISWD and its complexity reducing variant.

A. The ISWD

In the ISWD, the BP decoding outputs of a set of symbols will be monitored once they are included by a window. Given a window size W, the BP decoding at time instant t calculates the LLRs of W sets of codeword symbols. Let

$$\mathcal{S}_\tau = \{c_{M\beta\tau+1}, c_{M\beta\tau+2}, \ldots, c_{M\beta(\tau+1)}\} \tag{10}$$

denote a particular symbol set in the window, where $\tau = t, t+1, \ldots, t+W-1$. Among them, $\mathcal{S}_t$ is the target symbol set. The ISWD monitors the achievable minimums of

$$\overline{\mathcal{P}}_{e,t}(u), \overline{\mathcal{P}}_{e,t+1}(u), \ldots, \overline{\mathcal{P}}_{e,t+W-1}(u), \tag{11}$$

respectively. The symbol likelihoods that are associated with the minimums will also be stored. When the maximum iteration number is reached, decision of the target symbols will be made based on the stored likelihoods. Note that when $t \ge W-1$, each symbol set will be included by W sliding windows. Hence, its average error probability will be monitored over W window decoding events. This distinguishes the proposal with the SWD in which $\overline{\mathcal{P}}_{e,t}(u)$ will only be monitored by the window at time instant t.

Let $\mathcal{P}^*_{e,\tau}$ denote the minimum of the average error probability $\overline{\mathcal{P}}_{e,\tau}(u)$ that is monitored during the period when $\mathcal{S}_\tau$ is included by a decoding window. Once $\mathcal{S}_\tau$ is included by a decoding window, it will be initialized as $\mathcal{P}^*_{e,\tau} = 1$. At time instant t, the average error probability $\overline{\mathcal{P}}_{e,\tau}(u)$ of the W symbol sets will be monitored by (8) and (9). If

$$\overline{\mathcal{P}}_{e,\tau}(u) < \mathcal{P}^*_{e,\tau}, \tag{12}$$

$\mathcal{P}^*_{e,\tau}$ will be updated by

$$\mathcal{P}^*_{e,\tau} = \overline{\mathcal{P}}_{e,\tau}(u) \tag{13}$$

and the BP decoding likelihoods of symbol set $\mathcal{S}_\tau$ will also be stored. Since the current window aims to estimate symbol set $\mathcal{S}_t$, the decoder will only assess whether $\mathcal{P}^*_{e,t}$ falls below a predetermined error probability threshold (denoted as Ξ). If so, the current window will terminate. Otherwise, the BP iteration continues. The decoding window will also be terminated once the maximal iteration number (denoted as Υ) is reached. When the decoding terminates, estimation on the target symbols $c_{M\beta t+1}, c_{M\beta t+2}, \ldots, c_{M\beta(t+1)}$ will be made based on their stored likelihoods. The window will then slide to decode the next set of target symbols by updating the time instant as $t = t+1$. The above mentioned ISWD process is summarized as in Algorithm 1.

Like the SWD, the ISWD starts after receiving the first $WM\beta$ symbols. When the window slides to the next time instant t $(t > 0)$, symbol set $\mathcal{S}_{t+W-1}$ is newly included and their LLRs will be initialized as in (3) and (4). Meanwhile, LLRs of the other symbol sets $\mathcal{S}_t$, $\mathcal{S}_{t+1}$, …, $\mathcal{S}_{t+W-2}$ will be inherited from the previous window decoding outcomes. Compared to the SWD algorithm, the above ISWD algorithm will not incur extra iterations in each decoding window. But it requires extra memory since the decoder needs to store likelihoods of $WM\beta$ codeword symbols.

B. A Complexity Reducing Variant

The above description shows that for a symbol set $\mathcal{S}_t$ where $t \ge W-1$, they will start to be monitored by the decoding window at time instant $t-W+1$. This implies once they become the target symbols at a later time instant, their minimum average error probability $\mathcal{P}^*_{e,t}$ and the associated likelihoods are already available. The decoding complexity can be reduced by assessing whether $\mathcal{P}^*_{e,t} < \Xi$ before the BP

Algorithm 1 The ISWD Algorithm

Parameters: Ξ, Υ and W;

Initialize $t = 0$ and $u = 0$;
While $t < L$ **do**
  While $u < \Upsilon$ **do**
    Perform the BP decoding as in (5) and (6);
    Update $u = u + 1$;
    For $\tau = t$ to $t + W - 1$ **do**
      Determine $\overline{\mathcal{P}}_{e,\tau}(u)$ as in (8) and (9);
      If $\overline{\mathcal{P}}_{e,\tau}(u) < \mathcal{P}^*_{e,\tau}$, update $\mathcal{P}^*_{e,\tau}$ as in (13) and store the likelihoods of $\mathcal{S}_\tau$;
    End for
    If $\mathcal{P}^*_{e,t} < \Xi$, terminate the decoding window and **goto** 12;
  End while
  Estimate $\mathcal{S}_t$ based on the stored likelihoods;
  Update $t = t + 1$ and restore $u = 0$;
End while

iterations start. If so, the window will further slide to decode the next symbol set $\mathcal{S}_{t+1}$ without any dedicated BP operations for set $\mathcal{S}_t$. As a result, the average iteration number in each decoding window can be reduced.

However, it should be pointed out that skipping BP operation of a certain decoding window might affect the iterative convergence of the later windows, especially when the window size is too small. For example, Fig. 2 shows the sliding window decoding with $w = 2$ and $W = 3$. If $\mathcal{P}^*_{e,t+1} < \Xi$ at the beginning of the window decoding at time instant $t + 1$, the window will further slide into time instant $t+2$, skipping any BP operations at $t + 1$. As a result, the stored likelihoods of $\mathcal{S}_{t+1}$ will be calculated without considering the check equations in the shadow area. Those check nodes have not been updated either. This will affect the reliability of the decoding likelihoods, which would in turn incur more iterations for the later windows. Therefore, a sufficiently large window size should be chosen in order to realize this complexity reduction of ISWD. In the following section, numerical results will be provided to demonstrate this requirement.

IV. Simulation Results and Discussions

This section presents our simulation results and discussions of the ISWD algorithm and its complexity reducing variant. Decoding performances were obtained over the additive white Gaussian noise (AWGN) channel using BPSK modulation. The simulated SC-LDPC codes were constructed from the $\mathbf{B} = [3\ 3]$ base matrix with $L = 50$ and $w = 2$. The maximum BP iteration number is 100 for each decoding window and the termination threshold is 1×10^{-6}. In this session, we refer the ISWD algorithm and its complexity reducing variant as the ISWD-I and the ISWD-II algorithms, respectively.

Figs. 3 and 4 show the BER performance of two $(3, 6)$ SC-LDPC codes that were constructed with a lifting factor of 100 and 500, respectively. Their codeword length are 10000 and 50000, with an actual code rate 0.48 for both. They show

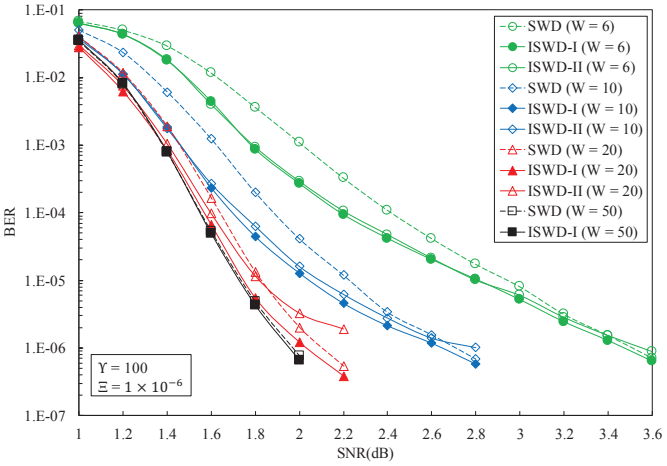

Fig. 3. Performance of the (3, 6) SC-LDPC code with $N = 10000$.

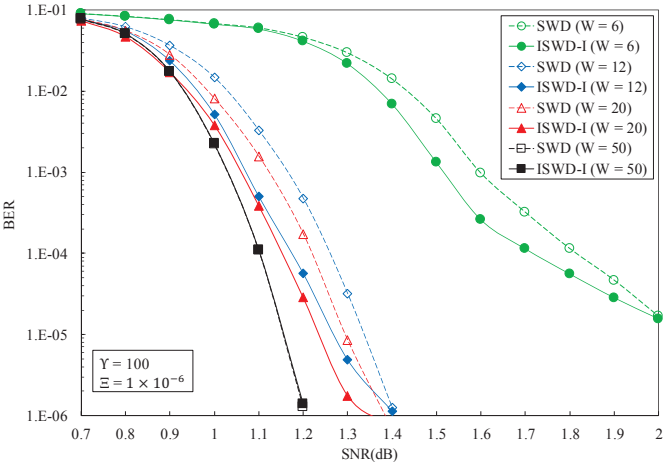

Fig. 4. Performance of the (3, 6) SC-LDPC code with $N = 50000$.

the ISWD-I algorithm outperforms the conventional SWD algorithm with various window sizes W. The performance improvement is especially significant in the BER curves' waterfall region. Note that the ISWD-I algorithm's advantage is realized when the maximum BP iteration number is reached such that estimation of the target symbols will be made based on potentially more reliable LLRs. This happens more often in the waterfall region. As the SNR increases, the ISWD-I performance converges to the SWD performance since most of the window decoding events will terminate without reaching the maximum iteration number. In this case, the two decodings are identical. Note that by increasing the window size W, the ISWD-I algorithm achieves a less significant performance improvement. This is because with a larger window size, the SWD yields a better performance, making it harder to be improved. With $W = L$, performance difference between the ISWD-I and SWD diminishes. As for the complexity reducing ISWD-II algorithm, Fig. 3 shows it also outperforms the SWD algorithm in the waterfall region but suffers a severer error floor. Overall, the two proposals do not show an asymptotic performance advantage over the SWD algorithm.

TABLE I
COMPLEXITY COMPARISON BETWEEN THE ISWD-I AND ISWD-II ALGORITHMS

Parameters	$w=2, W=3$							$w=2, W=10$						
SNR (dB)	3.5	4.0	4.5	5.0	5.5	6.0	6.5	1.6	1.8	2.0	2.2	2.4	2.6	2.8
ISWD-I (Ω)	286	242	126	45	15	6	4	50	23	16	13	12	11	10
ISWD-II (Ω)	286	242	126	46	21	12	6	50	23	15	12	10	9	8

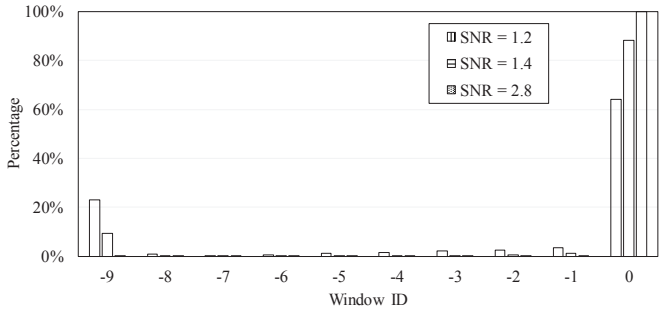

Fig. 5. Decoding statistics of the ISWD-I algorithm. Window IDs of $0 \sim -9$ represent the decoding windows at time instants $t \sim t-9$, respectively.

We now provide more insights of the improved SWDs. Fig. 5 shows the decoding statistics on the percentage of how many sets of stored LLRs (used for decision) were produced by different decoding windows in the ISWD-I algorithm. It was obtained by simulating the above SC-LDPC code with $N = 10000$ using $W = 10$. Therefore, besides the first two symbol sets, the others will be monitored by 10 decoding windows. Fig. 5 shows that most of the stored LLRs are either produced when the symbol set is targeted (window ID of 0) or when they start to be monitored (window ID of -9). When the SNR = 1.2dB, 64% of the stored LLRs are produced when the corresponding symbols are targeted and 23% are produced when they start to be monitored. This verifies the ISWD-I algorithm's performance advantage shown in the waterfall region. Note that in low SNRs the received information is unreliable. Performing more BP iterations will not help improve the decoding estimation. This explains why most of the intermediate windows cannot produce the stored LLRs. As the SNR increases, most of the stored LLRs are produced when the corresponding symbols are targeted. In this case, the ISWD-I and SWD become the same and their performances converge. This implies to reduce the monitoring effort, we can monitor a symbol set when they are included by the first and last decoding windows.

Table I further compares the decoding complexity between the ISWD-I and the ISWD-II algorithms. Their decoding complexity is reflected by the average iteration number (denoted as Ω) for decoding each symbol set. Again, this statistics was obtained by simulating the above SC-LDPC code. It shows as the SNR increases, the decoding converges earlier. It also shows that the ISWD-II algorithm cannot reduce the average iteration number with $W = 3$. This is because when $W = 3$, skipping BP operations of a certain decoding window affects the convergence of the later windows.

V. CONCLUSIONS

This paper has proposed two improved SWD algorithms that are featured by monitoring the average error probability $\overline{\mathcal{P}}_{e,t}(u)$ of a symbol set $\mathcal{S}_t$ when it is included by a number of decoding windows. Symbols of $\mathcal{S}_t$ are estimated based on the stored LLRs that produce the minimum value of $\overline{\mathcal{P}}_{e,t}(u)$. Our simulation results show the ISWD algorithm outperforms the conventional SWD algorithm, especially in the BER curve's waterfall region. It exhibits an asymptotic convergence to the SWD performance. The complexity reducing variant of ISWD algorithm can also outperform the SWD algorithm in the waterfall region, but suffers a severer error floor.

VI. ACKNOWLEDGMENT

This is supported by the National Natural Science Foundation of China (NSFC) under grants 61372079 and 61671486.

REFERENCES

[1] S. Kudekar, T. Richardson and R. Urbanke, "Threshold saturation via spatial coupling: why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.

[2] D. Costello, Jr., L. Dolecek, T. Fuja, J. Kliewer, D. Mithchell and R. Smarandache, "Spatially coupled sparse codes on graphs: Theory and Practice," *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 168–176, Jul. 2014.

[3] D. Divsalar, S. Dolinar. C. Jones and K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009.

[4] D. Mitchell, M. Lentmaier and D. Costello, Jr., "Spatially coupled LDPC codes constructed from protographs," *IEEE Trans. Inf. Theory*, vol. 61, no. 9, pp. 4866–4889, Sept. 2015.

[5] A. 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, Sept. 1999.

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

[7] D. Mitchell, A. Pusane and D. Costello, Jr., "Minimum distance and trapping set analysis of protograph-based LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 254–281, Jan. 2013.

[8] A. Iyengar, M. Papaleo, P. Siegel, J. Wolf, A. Vaneli-Corali and G. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2303–2320, Apr. 2012.

[9] N. ul Hassan, A. Pusane, M. Lentmaier, G. Fettweis and D. Costello, Jr., "Non-uniform window decoding schedules for spatially coupled LDPC codes," *IEEE Trans. Commun.*, vol. 65, no. 2, pp. 501–510, Feb. 2017.

[10] N. ul Hassan, M. Lentmaier and G. Fettweis, "Comparision of LDPC block and LDPC convolutional codes based on their decoding latency," in *Proc. Int. Symp. Turbo Codes & Iter. Inf.*, Gothenburg, Sweden, Aug. 2012.

[11] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," *Jet Propulsion Laboratory, INP Progress Report*, pp. 42–154, Aug. 2003.

[12] F. Kschischang, B. Frez and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.