Improving LDPC Decoders via Informed Dynamic Scheduling

Andres I. Vila Casado, Miguel Griot and Richard D. Wesel
Department of Electrical Engineering, University of California, Los Angeles, CA 90095-1594
Email: avila@ee.ucla.edu, mgriot@ee.ucla.edu, wesel@ee.ucla.edu

Abstract— **Low-Density Parity-Check (LDPC) codes are usually decoded by running an iterative belief-propagation (BP), or message-passing, algorithm over the factor graph of the code. The message-passing schedule of the BP algorithm significantly affects the performance of the LDPC decoder. The authors recently presented a novel message-passing schedule, called Informed Dynamic Scheduling (IDS), that selects the message-passing schedule according to the observed rate of change of the messages. IDS yields a lower error-rate performance than traditional message-passing schedules (such as flooding and LBP) because it solves traditional trapping-set errors. However, for short-blocklength LDPC codes, IDS algorithms present non-trapping-set errors in the error floor region. This paper presents a careful analysis of those errors and proposes mixed scheduling strategies, combining LBP with IDS, that solve these non-trapping-set errors. Also, we will show that some lower-complexity techniques, such as mixed scheduling, perform close to the best IDS strategies for larger-blocklength codes.**

Index Terms— **Belief propagation, message-passing schedule, error-control codes, low-density parity-check codes.**

I. Introduction

LDPC codes are usually decoded using a message-passing algorithm, called Belief Propagation (BP), over a factor-graph representation of the code, as shown in [1] and [2]. Traditionally, the message-passing schedule updates of all the messages in the graph in every iteration. This update is either simultaneous (flooding scheduling) or sequential (layered belief propagation (LBP) [3] and [4]). A novel message-passing schedule was introduced in [5] where the current state of the messages in the graph is used to dynamically update the schedule, producing an Informed Dynamic Schedule (IDS).

Among the strategies presented in [5], Node-Wise Approximate-Residual Belief Propagation was shown to perform better than flooding and LBP even after a large number of iterations in the waterfall region of several blocklength-1944 LDPC codes. We will refer to this strategy as Approximate Node-wise Scheduling (ANS) in this paper. ANS outperforms traditional scheduling because it solves trapping sets that other scheduling strategies don't, which can greatly impact the error-floor performance of a code.

Trapping sets, or near-codewords, as defined in [6] and [7], are small variable-node sets such that the induced sub-graph has a small number of odd-degree neighbors. In [7], Richardson also mentions that the most troublesome trapping-set errors are those where the odd-degree neighbors have degree 1 (in the induced sub-graph), and the even-degree neighbors have degree 2 (in the induced sub-graph).

However, the error-floor region of short-blocklength LDPC codes decoded by ANS includes non-trapping-set errors that don't occur in flooding or LBP. The error-floor region of short-blocklength codes isn't dominated by trapping-set errors given that their minimum distance is so low that Maximum-Likelihood (ML) errors are in the order of the error floor. A careful study of the noise realizations that ANS cannot solve and flooding and LBP can solve, reveals that they are caused by the greedy nature of the algorithm. Furthermore, these decoding errors can be separated into two categories: non-ML undetected errors and myopic errors. Non-ML undetected errors happen when ANS forces the decoder to converge to a codeword that is farther away (in terms of squared Euclidian distance) from the received sequence than the codeword sent by the transmitter. Myopic errors occur when there are several bits in error and ANS updates only a few number of check nodes in a periodic fashion. Myopic errors only occur when the factor graph has several length-4 cycles.

We combine traditional scheduling strategies, such as LBP, with IDS strategies, such as ANS, to obtain decoders that can handle trapping-sets without incurring in the greedy errors of ANS. The proposed decoder uses LBP in the first iterations to avoid the greedy ANS errors and switches to ANS to solve trapping sets. The switch occurs after a pre-determined number of iterations, which we call fixed LBP/ANS, or after the number of unsatisfied check nodes is low, which we call adaptive LBP/ANS.

Since an ANS iteration is more complex than an LBP iteration, these mixed-scheduling strategies have the further benefit of having a lower complexity than using only ANS. Hence, mixed-scheduling strategies are also interesting for larger-blocklength codes. In order to lower the complexity, we also propose a simpler IDS named Low-Complexity ANS (LC-ANS). These lower-complexity strategies perform close to ANS.

This paper is organized as follows. Section II explains ANS and its relation with trapping sets. Section II-C analyzes the greedy ANS errors that occur in the error-floor region of short-blocklength LDPC codes. New IDS strategies, mixed scheduling and LC-ANS, are introduced in III. Simulation results of all the different message-passing schedules are compared and discussed in Section IV. Section V

This work was supported by the state of California and ST Microelectronics through UC discovery grant COM 03-10142.

delivers the conclusions.

II. ANS scheduling for LDPC decoding

A. LDPC decoding

The LDPC code graph is a bi-partite graph composed by N variable nodes v_j for $j \in \{1, \ldots, N\}$ that represent the codeword bits and M check nodes c_i for $i \in \{1, \ldots, M\}$ that represent the parity-check equations. The exchanged messages correspond to the Log-Likelihood Ratio (LLR) of the probabilities of the bits. The sign of the LLR indicates the most likely value of the bit and the absolute value of the LLR gives the reliability of the message. In this fashion, the channel information LLR of the variable node v_j is $C_{v_j} = \log\left(\frac{p(y_j|v_j=0)}{p(y_j|v_j=1)}\right)$, where y_j is the received signal. Then, for any c_i and v_j that are connected, the two message generating functions, are:

$$m_{v_j \to c_i} = \sum_{c_a \in \mathcal{N}(v_j) \backslash c_i} m_{c_a \to v_j} + C_{v_j}, \tag{1}$$

$$m_{c_i \to v_j} = 2 \times \operatorname{atanh}\left(\prod_{v_b \in \mathcal{N}(c_i) \backslash v_j} \tanh\left(\frac{m_{v_b \to c_i}}{2}\right)\right), \tag{2}$$

where $\mathcal{N}(v_j) \backslash c_i$ denotes the neighbors of v_j excluding c_i, and $\mathcal{N}(c_i) \backslash v_j$ denotes the neighbors of c_i excluding v_j.

B. Approximate Node-wise Scheduling (ANS)

The Residual Belief Propagation (RBP) algorithm was presented by Elidan et al. in [8]. RBP was proposed for general sequential message passing, not specifically for BP decoding. Several IDS strategies inspired by RBP were presented in [5] and Approximate Node-wise Scheduling (ANS), named Node-Wise ARBP in [5], was found to perform better than LBP across all iterations in the waterfall region of several LDPC codes.

In ANS scheduling, as well as in LBP, check nodes are updated sequentially using the most recent information available. LBP updates check nodes sequentially according to a predetermined schedule. ANS selects the next check node to be updated based on the current state of the messages in the graph. Specifically ANS selects the check node based on a metric α_c that measures how useful that check node update is to the decoding process.

For each check node, the metric α_c is the largest approximate residual of the check-to-variable messages that are generated in the check node. A residual is the norm (defined over the message space) of the difference between the values of the message before and after an update. When a residual is computed using the or min-sum check-node update equation, introduced in [9] and explained in [10], it is called an approximate residual. The performance degradation of using min-sum to compute the residuals is negligible as shown in [5]. ANS is formally described in Algorithm 1.

Fig. 1 shows an example of how ANS overcomes trapping sets. Updating the check node with the largest metric allows the decoding algorithm to focus on a part of the graph that hasn't converged yet. Thus, it is likely that ANS solves the variable nodes in error by sequentially updating the degree-1 check nodes connected to them. When a variable node in a trapping set is corrected, the induced sub-graph of the variable-nodes-in-error will change as follows. At least one check node that was degree-2 becomes degree-1 (in the induced sub-graph of variable-nodes in error) after the variable node correction. This check node is likely to be picked as the next check node to be updated by ANS because its messages will have large residuals. This update will probably correct another variable node in the trapping set.

We corroborated this analysis by Monte Carlo simulations. As an example, Fig. 2 shows the performance of the blocklength-2640 Margulis code, proposed in [11], using flooding, LBP and ANS. The FER of the blocklength-2640 Margulis code at high SNRs has been shown to be dominated by trapping-set errors in [6] and [7]. The ANS performance improvement with respect to both flooding and LBP shows that ANS can correct trapping sets that traditional scheduling strategies cannot.

Algorithm 1 ANS decoding for LDPC codes

```
Initialize all m_{c→v} = 0
Initialize all m_{v_j→c_i} = C_j
Compute all α_c
Find i = arg max_{u={1...N}} α_{c_u}
for every v_k ∈ N(c_i) do
    Generate and propagate m_{c_i→v_k}
    Set α_{c_i} = 0
    for every c_a ∈ N(v_k)\c_i do
        Generate and propagate m_{v_k→c_a}
        Compute α_{c_a}
    end for
end for
if Stopping rule is not satisfied then
    Position=4;
end if
```

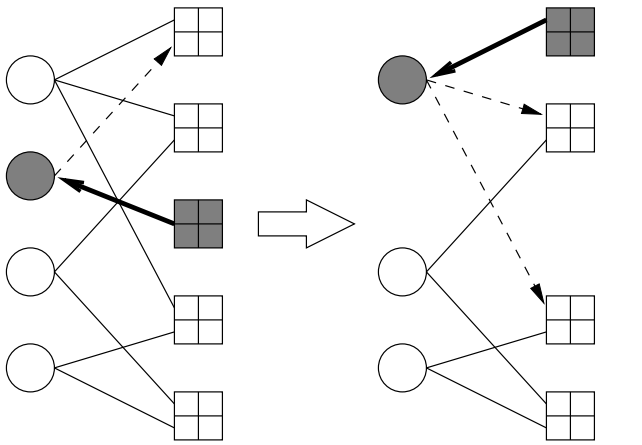

Fig. 1. Check-node update sequence that solves a trapping set. Dark nodes represent the check node that is updated and the variable node that is corrected.

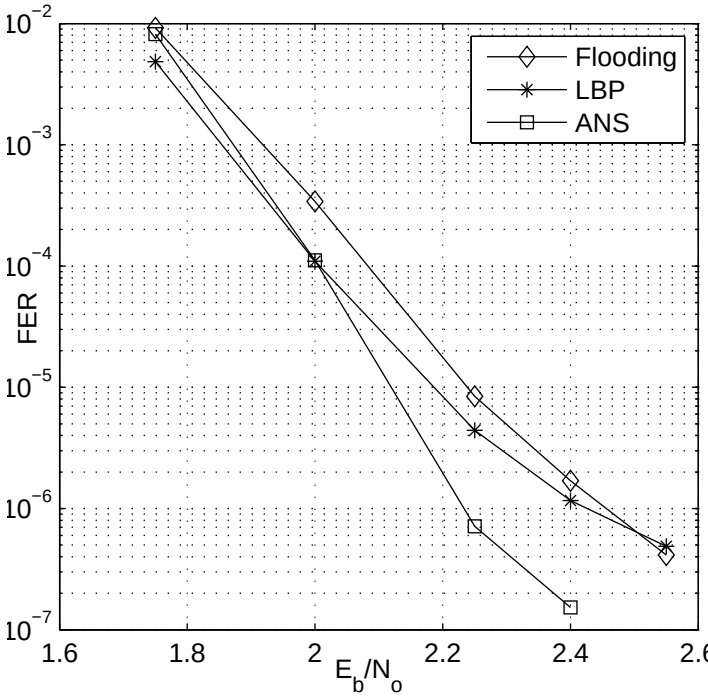

Fig. 2. AWGN performance of the blocklength-2640 Margulis code decoded by 3 different scheduling strategies: flooding, LBP and ANS. A maximum of 50 iterations was used.

C. Shortcomings of ANS decoding

ANS decoding, while better than traditional scheduling because it solves trapping sets, presents other types of errors that don't occur with LBP and flooding. They can be categorized into two classes: non-ML undetected errors and myopic errors.

We define non-ML undetected errors as undetected errors where the squared Euclidian distance between the decoded codeword and the received signal is larger than the squared Euclidian distance between the transmitted codeword and the received signal. This means that an ML decoder wouldn't make this mistake. Given its greedy nature, ANS makes more non-ML undetected errors than traditional scheduling strategies.

If there is a received signal that is near the border between two decoding regions (Voronoi regions), the initial BP iterations can take the decoder in any direction. ANS is more likely to make non-ML undetected errors than flooding or LBP because it can update only a part of the graph. This locally optimum approach is more likely to go in the wrong direction than the more global approach of LBP and flooding. The probability that ANS makes a non-ML undetected error decreases as the received signal is farther from the border. Thus, the negative effect of this behavior is more noticeable in the decoding of short-blocklength LDPC codes. Short-blocklength codes have a minimum Hamming distance small enough that the probability of receiving a signal near the border of two decoding regions is comparable to the probability of loopy-BP errors in high SNR regimes.

There is another type of error that results from the greediness of ANS. Myopic decoding errors happen when the decoder focuses on a small number of check nodes while there are many other bits in error to solve in a different part of the graph. These errors become significant when the graph has many length-4 cycles. If ANS updates one of the check nodes in a length-4 cycle sub-graph, it is likely that the next check node to be chosen is the other one in the cycle given that it receives two updated messages. Thus, if the code has graph structures that contain many length-4 cycles, such as the one shown in Fig. 3, it is likely for ANS to become stuck repeatedly updating the same small number of check nodes even if there are errors on other parts of the code. Simulations show that myopic errors are only significant for codes that present densely connected sub-graphs such as randomly constructed short-blocklength codes that allow length-4 cycles.

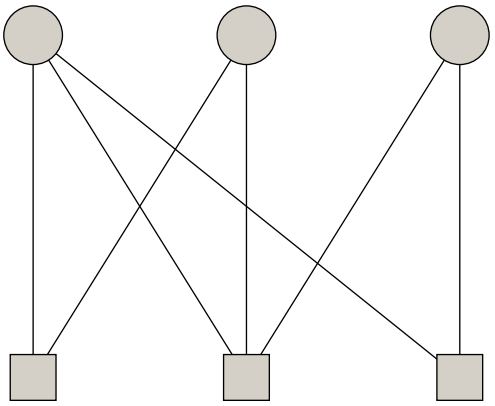

Fig. 3. Graph of a structure that can cause myopic errors

III. New IDS strategies

A. IDS strategies for short-blocklength LDPC codes

We propose mixed strategies that combine LBP and ANS iterations in order to correct trapping-set errors and avoid the greedy ANS errors. The decoder starts by performing LBP iterations and switches to ANS iterations. Fixed LBP/ANS (F-LBP/ANS) first does a pre-determined number of LBP iterations ξ and then switches to ANS.

Given that one of the main advantages of ANS is the fact that it solves trapping sets, we propose another mixed strategy that we call Adaptive LBP/ANS (A-LBP/ANS). In A-LBP/ANS the decoder switches from LBP to ANS when the number of unsatisfied check nodes is below a certain value ζ. This makes sense given that the dominant trapping sets are those that have a small number of unsatisfied check nodes [7]. Thus, LBP will decode until it hits a trapping set with a small number of unsatisfied check nodes where ANS, better equipped to solve trapping sets, takes over. Since an ANS iteration is more complex than an LBP iteration, these lower-complexity mixed strategies are also attractive for larger-blocklength codes because of their close error-rate performance to ANS. The optimal values

of ξ and ζ can be found trough Monte-Carlo simulations.

B. Lower-Complexity ANS (LC-ANS)

As mentioned in Section II, ANS selects the check node to be updated based on a metric α_c, which is the largest approximate residual of the check-to-variable messages that are generated in the check node. Thus, in order to generate α_{c_i} we must compute the approximate residuals of all the check-to-variable messages of check node c_i and find the largest one.

In order to reduce these computations we propose to infer which edges are more likely to have the larger residuals of each check node based on the following considerations. The largest α_{c_i} metric corresponds to the largest residual of all the check-to-variable messages in the graph. It is likely that the largest residual in the graph corresponds to a check-to-variable message that has a different sign before and after the update. It is also likely that among the check-to-variable messages that change their sign after the update, the largest residual corresponds to the message that has the largest reliability after the update.

Lower-Complexity ANS (LC-ANS) selects the check node to be updated based on a simplified check-node metric α_c^{LC} that focuses on the messages with the largest reliability after the update. The check-to-variable messages, generated in the same check node, with larger reliability correspond to the edges that have the variable-to-check messages with the smaller reliability. We define α_c^{LC} as the sum of the two residuals that correspond to the edges that have the two variable-to-check messages with the smallest reliability. Given that we use min-sum to compute the residuals, the two variable-to-check messages with the smallest reliability are known. Thus, in order to generate $\alpha_{c_i}^{LC}$, only two residuals are computed and then summed which is significantly less complex than generating α_{c_i}. Monte Carlo simulations, shown in Section IV, show that LC-ANS very close to ANS.

IV. Results

Table I shows the FER and Undetected FER (UFER) of 5 different rate-1/2 LDPC codes decoded using 5 different scheduling strategies. All the codes have blocklength 648 and have the same variable-node degree distribution. The UFER is defined as the total number of frames with undetected errors divided by the total number of frames simulated. The simulations correspond to an AWGN channel with $E_b/N_o = 3$ dB and a maximum number of 50 iterations was used.

Code A is a random code constructed using the ACE and SCC graph constraint algorithms proposed in [12] and [13] respectively. These algorithms were designed to avoid the presence of small stopping sets. However, this code allows the presence of length-4 cycles. Code B was randomly constructed while avoiding length-4 cycles. The ACE and the SCC algorithms were used to construct code C and length-4 cycles were avoided. Code D was also randomly constructed using the PEG algorithms first presented in and [14]. The PEG algorithm is design to locally maximize the girth of the graph as the matrix generation process goes on. This code has a girth of 6 thus it doesn't have any length-4 cycles either. Finally, code E is an LDPC code selected for the IEEE 802.11n standard [15].

Let us analyze the performance of the traditional scheduling strategies: flooding and LBP. We corroborated experimentally that the detected errors, which are the difference between their FER and UFER values, are mostly trapping-set errors. Also, as expected, LBP performs better than flooding.

ANS outperforms LBP for all the codes except for code A. This is the only code in the group that has length-4 cycles and we experimentally corroborated that myopic errors described in Section II-C dominate the performance of this code at this SNR. As further proof, code C was designed to keep the same ACE and SCC graph constraints as code A while avoiding length-4 cycles. Code C doesn't incur in any ANS myopic errors. This shows that myopic errors dominate the error performance when the graph has several length-4 cycles. Furthermore, we see that the values of UFER are larger than their corresponding UFER for flooding and LBP. This is due to an increase in the number of Non-ML undetected errors as explained in Section II-C. Table I clearly shows that the ANS FER performance of the last four codes is clearly dominated by the undetected errors given that the FER and UFER values are very close to each other.

Table I also shows the results of the mixed scheduling strategies. The fourth column shows the FER and UFER of F-LBP/ANS with $\xi = 35$. Hence, the decoder starts by performing 35 LBP iterations and finishes with 15 ANS iterations. The fifth column shows the FER and UFER of A-LBP/ANS with $\zeta = 5$. Hence, the decoder starts by performing LBP iterations until the number of unsatisfied check nodes is less than or equal to 5. The values of ξ and ζ were not optimized and a careful study of this optimization will be presented in the camera ready copy of this paper. Both mixed strategies correct the ANS myopic errors of code A and also have lower UFERs than ANS for all the codes.

Fig. 4 shows the performance of code A as the number of iterations increases. In the first iterations ANS presents good performance. However, it presents an error floor at 6×10^{-5}. As mentioned before, a careful analysis of these errors showed that they were myopic errors due to to the large number of length-4 cycles. No ANS myopic errors were observed for codes that don't have length-4 cycles. Furthermore, Fig. 4 shows that both mixed strategies perform very well when compared to LBP and flooding.

Fig. 5 shows the FER and UFER of code C for a maximum number of iterations equal to 50. The FER of the three IDS strategies closely approach their respective UFER for a high SNR. Also, while ANS presents a larger UFER than LBP and flooding at 3 dB, the mixed strategies' UFERs are as low as with LBP and flooding. This shows that the mixed strategies provide a good combination of harvesting the trapping-set correction capability of ANS while avoiding the errors generated by ANS's greed-

TABLE I

FER AND UFER OF 5 DIFFERENT LDPC CODES DECODED BY 5 DIFFERENT SCHEDULING STRATEGIES: FLOODING, LBP, ANS, F-LBP/ANS WITH $\xi = 35$ AND A-LBP/ANS WITH $\zeta = 5$. THE CHANNEL USED IS AWGN WITH $E_b/N_o = 3$ DB.

	Flooding		LBP		ANS		F-LBP/ANS		A-LBP/ANS	
Code	FER	UFER	FER	UFER	FER	UFER	FER	UFER	FER	UFER
A	4.2e-5	1.1e-6	1.7e-5	1.0e-6	5.8e-5	3.5e-6	3.9e-6	1.3e-6	3.9e-6	2.0e-6
B	1.6e-4	3.2e-6	1.1e-4	2.2e-6	3.4e-5	2.9e-5	2.0e-5	4.7e-6	1.5e-5	1.0e-5
C	3.4e-5	1.1e-6	1.6e-5	1.1e-6	5.1e-6	4.6e-6	3.0e-6	1.2e-6	2.6e-6	1.6e-6
D	4.4e-5	4.4e-6	3.0e-5	5.3e-6	1.9e-5	1.8e-5	1.2e-5	7.5e-6	1.1e-5	9.2e-6
E	2.2e-5	8.9e-7	6.5e-6	2.0e-6	5.8e-6	5.3e-6	3.3e-6	2.4e-6	4.2e-6	3.4e-6

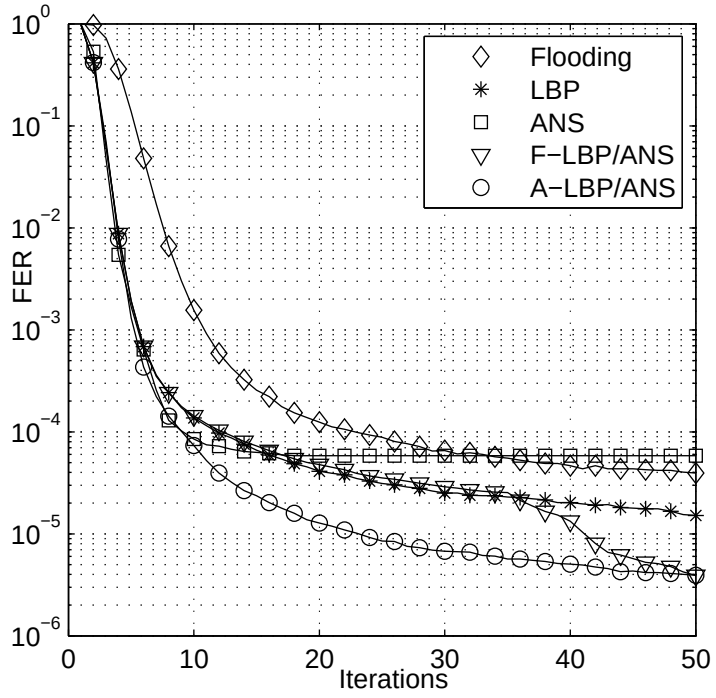

Fig. 4. AWGN Performance of code A vs. number of iterations for a fixed $E_b/N_o = 3$ dB. Results of 5 different scheduling strategies are presented: flooding, LBP, ANS, F-LBP/ANS with $\xi = 35$ and A-LBP/ANS with $\zeta = 5$.

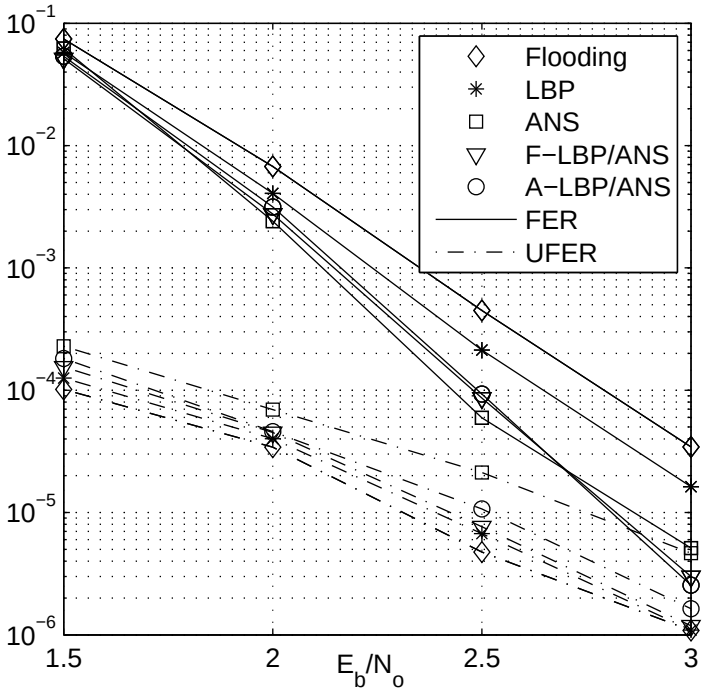

Fig. 5. AWGN performance of code C decoded by 5 different scheduling strategies: flooding, LBP, ANS, F-LBP/ANS with $\xi = 35$ and A-LBP/ANS with $\zeta = 5$.

iness. Mixed strategies are also less computationally demanding than pure ANS.

Fig. 6 shows the FER of a blocklength-1944 LDPC code decoded using 5 different scheduling strategies: flooding, LBP, ANS, LC-ANS and A-LBP/ANS with $\zeta = 5$. The code was designed to have no length-4 cycles and the maximum number of iterations was set to 50. Both A-LBP/ANS and LC-ANS perform closely to ANS while requiring a lower complexity. Furthermore, Fig. 7 shows that performance of LC-ANS is close to the performance of ANS for all iterations.

Also, Fig. 6 shows that the performance improvement of IDS strategies increases as the SNR increases. This is explained by the fact that as the SNR increases, trapping-set errors become dominant. This suggest that IDS strategies can significantly improve the error-floor of LDPC codes.

V. Conclusions

IDS strategies such as ANS have a better performance than traditional scheduling strategies such as flooding and LBP because they can solve trapping-set errors.

However, for short-blocklength codes there is an increase in the number of non-ML undetected errors that significantly affect the performance of ANS in high-SNR regimes. Also, for codes that have a large number of length-4 cycles ANS makes myopic errors that dominate the performance of the codes.

Mixing LBP and ANS iterations can solve trapping-set errors without incurring in the previously mentioned ANS greedy errors. We show experimentally that these strategies perform very well for 5 different short-blocklength codes.

Furthermore, mixed-scheduling strategies have a lower complexity than ANS since an LBP iteration is simpler than an ANS iteration. Thus, mixed strategies are a

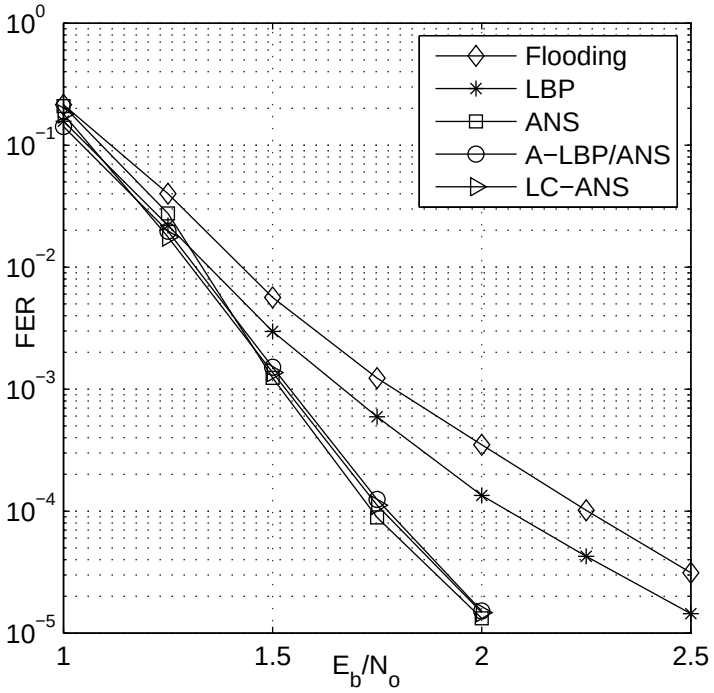

Fig. 6. AWGN performance of a blocklength-1944 LDPC code decoded by 5 different scheduling strategies: flooding, LBP, ANS, A-LBP/ANS with $\zeta = 5$ and LC-ANS.

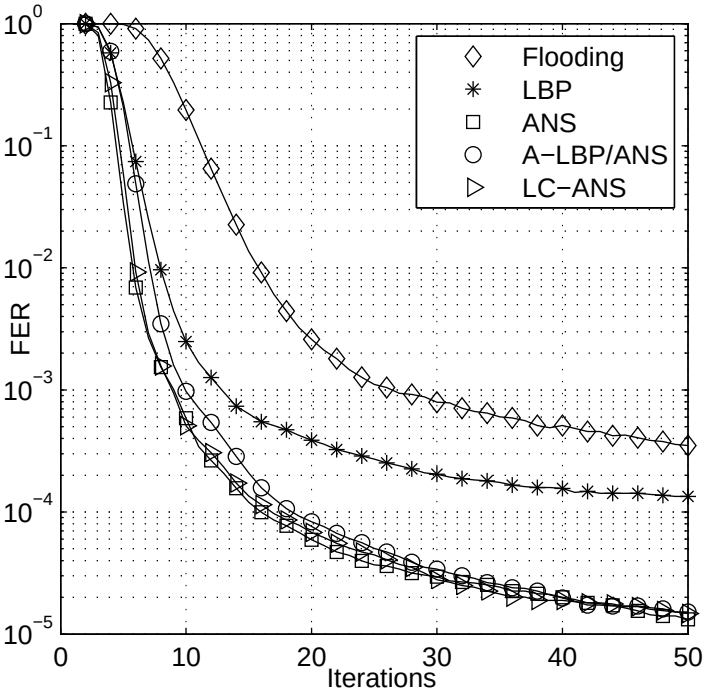

Fig. 7. AWGN Performance of a blocklength-1944 LDPC code vs. number of iterations for a fixed $E_b/N_o = 2$ dB. Results of 5 different scheduling strategies: flooding, LBP, ANS, A-LBP/ANS with $\zeta = 5$ and LC-ANS.

lower-complexity alternative to ANS given their similar performance. Also, we propose LC-ANS as another lower-complexity IDS strategy that also performs as well as ANS.

References

[1] R.J. McEliece, D.J.C. MacKay, and Jung-Fu Cheng. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16:140–152, February 1998.

[2] F. Kschischang, B. J. R. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. on Info. Th.*, 47(2):498–519, March 2001.

[3] M.M. Mansour and N.R. Shanbhag. High-throughput LDPC decoders. *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, 11:976–996, December 2003.

[4] D. Hocevar. A reduced complexity decoder architechture via layered decoding of LDPC codes. In *Proc. Signal Processing Systems SIPS 2004*, pages 107–112, October 2004.

[5] A. I. Vila Casado, M. Griot, and R. Wesel. Informed Dynamic Scheduling for Belief-Propagation Decoding of LDPC Codes. In *Proc. IEEE ICC 2007*, Glasgow, Scotland, June 2007.

[6] D. MacKay and M. Postol. Weaknesses of margulis and ramanujan-margulis low-density parity-check codes. *Electronic Notes in Theoretical Computer Science*, 74, 2003.

[7] T. Richardson. Error floors of LDPC codes. In *Proc. 41st Annual Allerton Conf. on Comm.*, Monticello, IL, 2003.

[8] G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: informed scheduling for asynchronous message passing. In *Proc. 22^{nd} Conference on Uncertainty in Artificial Intelligence*, MIT, Cambridge, MA, July 2006.

[9] N. Wiberg. Codes and decoding on general graphs. Ph.D. Dissertation, Department of Electrical Engineering, Linkoping University, Linkoping, Sweden. 1996.

[10] M. Fossorier, M. Mihaljevic, and H. Imai. Reduced complexity iterative decoding of low density parity check codes based on belief propagation. *IEEE Trans. on Comm.*, 47:673–680, May 1999.

[11] G. A. Margulis. Explicit constructions of graphs without short cycles and low-density codes. *Combinatorica 2*, 1:71–78, 1982.

[12] T. Tian, C. Jones, J. Villasenor, and R. Wesel. Avoidance of Cycles in Irregular LDPCC Construction. In *IEEE Transactions on Communications*, August 2004.

[13] A. Ramamoorthy and R. D. Wesel. Construction of Short Block Length Irregular LDPCCs. In *Proc. IEEE ICC 2004*, Paris, France, June 2004.

[14] Xiao Yu Hu, Evangelos Eleftherioua, and Dieter Michael Arnold. Progressive edge-growth tanner graphs. In *GLOBECOM, The Evolving Global Communications Network*, pages 995–1001, San Antonio, Texas, November 2001.

[15] IEEE P802.11n/D1.05 October 2006, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Enhancements for Higher Throughput (Draft).