Adaptive Mixed Scheduling for Correcting Errors in Trapping Sets of LDPC Codes

Saejoon Kim, *Member, IEEE*, Seunghyuk Lee, Jun Heo, *Member, IEEE*, and Chong S. Rim

***Abstract*—Trapping sets of low-density parity-check codes are the problematic substructures that contribute to the error floor behavior under iterative decoding. Adoption of mixed scheduling for belief propagation decoding has shown to be very effective in solving some of the trapping sets. In this letter, we propose an *adaptive* mixed scheduling scheme that provides significantly improved code performance compared to existing mixed scheduling schemes.**

***Index Terms*—LDPC codes, trapping sets, mixed scheduling.**

I. Introduction

THE error floor behavior of finite-length low-density parity-check (LDPC) codes over the additive white Gaussian noise (AWGN) channel under iterative decoding is attributed to the bad substructures known as trapping sets [7] present in the code. It was recently shown that a way to correct the errors present in the trapping sets is through modifying the decoder's scheduling scheme [1] [2]. In a related work, [1] [6] and references therein have shown that sending messages in a sequence from subsets of variable and check nodes, broadly called the *sequential* scheduling scheme, can improve the convergence speed and the asymptotic code performance compared to the flooding scheme. In particular, a class of sequential decoding schemes known as *informed dynamic scheduling* (IDS) [1] [2] has shown to yield excellent code performance partially by correcting errors contained in trapping sets.

While IDS schemes are able to correct errors contained in trapping sets of LDPC codes, these scheduling strategies incur non-trapping set errors due to their greedy nature. Specifically, these non-trapping set errors do not arise when decoded by the flooding or a non-greedy sequential scheduling scheme. To this end, to correct errors present in trapping sets while avoiding non-trapping set errors as well, Casado *et al.* [2] proposed a mixed scheduling scheme in which a non-greedy sequential scheduling scheme is performed first to take care of the non-trapping set errors and afterward an IDS scheme is performed to solve errors in trapping sets. This mixed scheduling scheme yielded improved performance for some codes compared to an IDS scheme processed alone. However the condition for the switch from a non-greedy sequential scheduling scheme to an IDS scheme remained as a topic for further investigation.

We address the issue of the switch condition from the first to the second scheduling schemes in this letter. We propose an *adaptive* switch condition where the switch iteration dynamically changes as decoding progresses and it will be shown that our proposed scheme significantly improves on the previous mixed scheduling schemes by nearly an order of magnitude in terms of word-error-rate (WER) for some signal-to-noise ratios (SNRs).

II. Mixed Scheduling Schemes

Sequential belief propagation decoding refers to updating messages in a sequential order where the order can either be predetermined or change dynamically according to the messages generated during the decoding process. Layered belief propagation (LBP) decoding [4] belongs to the former type of sequential decoding in which subsets of check-to-variable node messages and the corresponding variable-to-check node messages are sequentially updated in a predetermined order. LBP decoding is known to converge faster than and yield code performance that is superior to flooding.

Nodewise residual belief propagation (NWRBP) [1] is an IDS scheme that belongs to the latter type of sequential decoding. It schedules the message updates according to the value of the *residual* of the message [3]. In NWRBP, only the check-to-variable node messages associated with the largest residual and the corresponding variable-to-check node messages are selected for each update. Thus, the current state of messages in the graph is used to dynamically update only parts of the graph at a time to increase the decoding convergence speed. As mentioned before, this dynamical updating structure also contributes to correcting errors contained in trapping sets which become dominant as SNR increases. For these reasons, NWRBP offer both better convergence speed and asymptotic performance than the flooding scheme.

On the other hand, while the greedy nature of NWRBP scheme enables correcting errors contained in trapping sets, it comes at a cost of incurring non-trapping set errors that do not arise when decoded by the flooding or a non-greedy sequential scheduling scheme. To remedy this drawback of NWRBP, *i.e.*, correct errors present in trapping sets while avoiding non-trapping set errors as well, Casado *et al.* [2] have proposed a mixed scheduling scheme in which a non-greedy sequential scheduling scheme is first processed to take care of the non-trapping set errors while NWRBP is processed afterwards to solve errors contained in trapping sets in the error-floor region. In [2], LBP was used as the non-greedy sequential

Manuscript received December 16, 2009. The associate editor coordinating the review of this letter and approving it for publication was V. Stankovic.

S. Kim, S. Lee, and C. S. Rim are with the Department of Computer Science and Engineering, Sogang University, Seoul, Korea (e-mail: saejoon@sogang.ac.kr).

J. Heo is with the School of Electrical Engineering, Korea University, Seoul, Korea.

The work of S. Kim was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education, Science and Technology (2009-0074611), and by the Special Research Grant of Sogang University 200911057.01. The work of J. Heo was supported by a Korea University Grant 2007.

Digital Object Identifier 10.1109/LCOMM.2010.07.092417

scheduling scheme and various switch conditions from the first to the second scheduling schemes were proposed that provided improved asymptotic code performance for codes that do contain non-trapping set errors. Switch conditions can be categorized into fixed and adaptive where in the former, switch occurs after a predetermined number of decoding iterations, and in the latter, it occurs when a given condition is satisfied. Since the fixed type needs to be modified for better performance according to the type of code used and the SNR environments, we focus here on the adaptive type that generalize better.

III. Proposed Adaptive Mixed Scheduling Scheme

Our proposed scheduling scheme is a version of the mixed scheduling scheme of [2] in which the switch iteration from LBP to NWRBP *adaptively* changes. The only previously known adaptive switch condition has been presented in [2] and has used the number of unsatisfied check nodes as the threshold measure. The number of unsatisfied check nodes, which we denote by $\bar{S}$, is the number of check nodes whose neighbors (after hard-decision decoding) do not sum to 0 modulo 2. Specifically, in [2] the switch from the first to the second scheduling scheme is activated if $\bar{S}$ is below a predetermined threshold or if the iteration number hits 30 whichever comes earlier. The use of $\bar{S}$ as the threshold is appropriate since it is known that the most problematic trapping sets have small values of $\bar{S}$ [7].

On the other hand, experiments have indicated that the minimum value of $\bar{S}$ can vary significantly for different SNRs and simulation runs, and the value of $\bar{S}$ can fluctuate frequently with respect to the number of iterations in a given simulation run. Therefore, it is difficult to set a fixed number as the threshold below which a switch should occur. Furthermore, as will be shown in the next section, while appropriate the use of the minimum value of $\bar{S}$ for the switch is not optimal in terms of minimizing the WER.

As a related measure to $\bar{S}$, consider the following

$$L^* = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{LLR}(i)|}{\deg(i)}$$

where $\mathrm{LLR}(i)$ and $\deg(i)$ are the LLR value and the degree of the variable node i, respectively, $|\cdot|$ is the absolute value sign, and n is the number of variable nodes. This expression is designed to represent an average reliability of variable nodes since LLR values of correctly decoded variable nodes, in general, tend to be larger than those of incorrectly decoded ones. The degree of the variable node factor was included for fair weighing of each variable node's LLR value. Thus, the larger the value of L^* is, the higher the average reliability of variable nodes is. In particular, as $\bar{S}$ is an approximate measure of the amount of incorrectly decoded variable nodes, the values of $\bar{S}$ and L^* possess trend that are approximately inversely proportional as decoding progresses. A positive example of this trend will be demonstrated in the next section. Attractive feature of L^* which suits for our purpose is that its value changes more gradually with respect to decoding iterations than $\bar{S}$. Specifically, in a given window of iterations, while $\bar{S}$ can have multiple local minima that take on a wide range of values, L^* normally has only a few of local maxima probably since it is the *average* LLR value over all variable nodes. Thus, it is more computationally feasible to move to the decoding iteration that achieves a large value of L^*, by going one decoding iteration backwards, than that achieves a small value of $\bar{S}$ as decoding progresses.

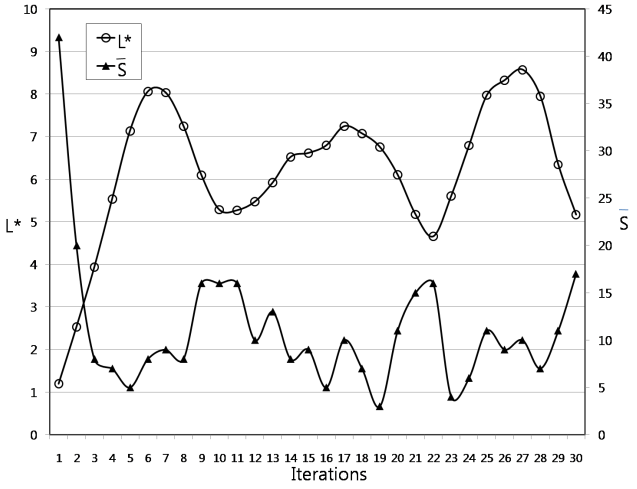

Fig. 1. Values of L^* (left y-axis) and $\bar{S}$ (right y-axis) during LBP decoding.

In our proposed adaptive mixed scheduling, the switch from LBP to NWRBP is activated at the decoding iteration that achieves the first or the second local maximum of L^* or if the iteration number hits a given number, *e.g.*, 30, whichever comes earlier. Decoding finishes whenever $\bar{S} = 0$, and if the switch is made, NWRBP is processed with the messages inherited from LBP. Simulation results indicate that switch at the first local maximum yielded the best performance for a moderate number of iterations while that at the second local maximum yielded the best asymptotic performance. We can deduce that in the former case, the switch is made prematurely before LBP's capability to correct non-trapping set errors is fully executed. Switch at later local maxima did not perform as well, and we believe LLRs of some of variable nodes in error that were generated from LBP became too large to be corrected by NWRBP in this case.

Computational cost of our scheme in comparison to LBP or NWRBP is the additional need, until the switch is made, to calculate for L^* after each decoding iteration which takes only a linear amount of time and to store the values of check-to-variable node messages of the previous decoding iteration for use by NWRBP in case the switch is made.

IV. Results

In this section, we show simulation results of various belief propagation decoding algorithms of LDPC codes used over the AWGN channel. In all simulations, we used a rate-$\frac{1}{2}$, length 500 code that was generated from the progressive-edge growth (PEG) algorithm [5] using an optimal degree distribution.

Figure 1 shows a typical example of L^* (left y-axis) and $\bar{S}$ (right y-axis) curves with respect to decoding iterations (x-axis) when LBP decoding is processed. The two curves are approximately inversely proportional to each other as described before, and clearly, it is much easier to identify

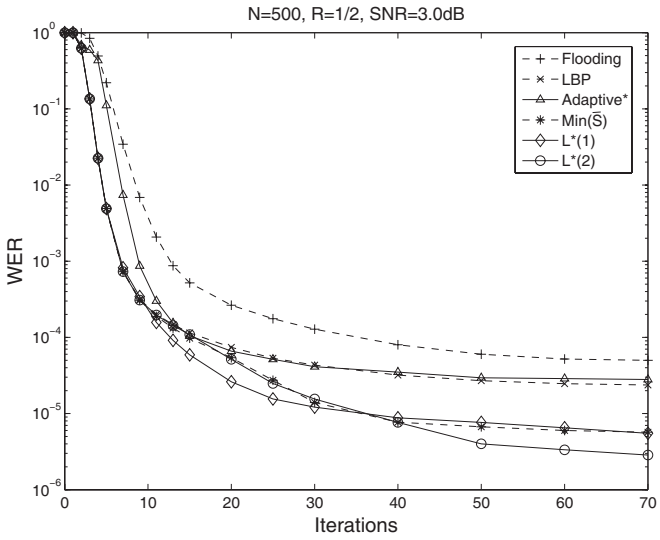

Fig. 2. WER performance of belief propagation decoding algorithms for $N = 500, R = \frac{1}{2}$ PEG algorithm-generated LDPC code at 3.0dB.

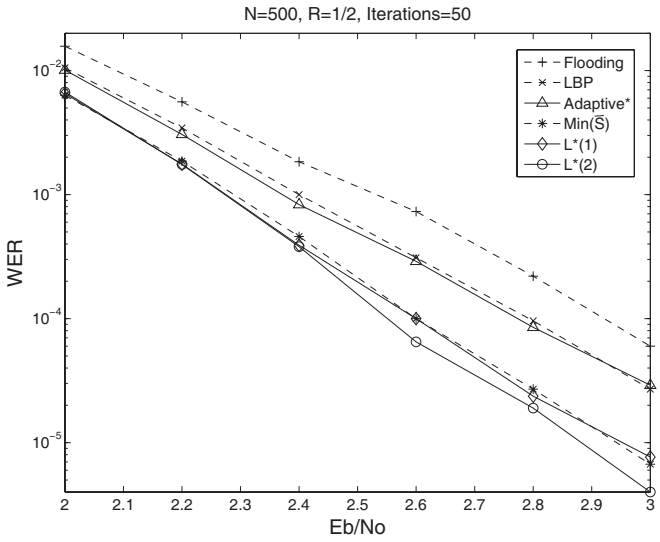

Fig. 3. WER performance of belief propagation decoding algorithms for $N = 500, R = \frac{1}{2}$ PEG algorithm-generated LDPC code after 50 iterations.

the maximum of L^* curve than to identify the minimum of $\bar{S}$ curve in a window of decoding iterations. We note that we follow the changes of L^* to select the iteration for switch and not its absolute values which can vary significantly from one simulation run to another.

Figure 2 shows the WER performance of the flooding ("Flooding"), LBP ("LBP"), adaptive mixed scheduling of [2] ("Adaptive*"), an optimal mixed scheduling scheme using $\bar{S}$ as the switch condition ("Min($\bar{S}$)"), and our proposed mixed scheduling schemes ("$L^*(1)$" and "$L^*(2)$") with respect to decoding iterations at 3.0 dB. Threshold value used for "Adaptive*" was 11 which was optimized through exhaustive search from 5 to 20. "$L^*(1)$" and "$L^*(2)$" represent our proposed scheduling schemes using the first and the second local maxima as the switch condition, respectively.

Optimal mixed scheduling when $\bar{S}$ is used as the switch condition represents the scheduling scheme that processes the switch when $\bar{S}$ (> 0) is the minimum. We have included this practically unrealistic scheme to illustrate the limit up to which code performance using $\bar{S}$ as the threshold measure can achieve. In our example, $\bar{S}$ value of the first 30 decoding iterations were observed to select the switch iteration.

The figure clearly illustrates that all sequential and mixed scheduling schemes perform noticeably better than the flooding scheme. In particular, it shows that our proposed schemes significantly outperform "Adaptive*." Furthermore, "$L^*(1)$" outperforms "Min($\bar{S}$)" for a moderate number of decoding iterations, and "$L^*(2)$" outperforms all other scheduling schemes asymptotically. Note "Min($\bar{S}$)" performs much better than "Adaptive*" due to the previously mentioned fact that the minimum value of $\bar{S}$ can vary significantly for different simulation runs.

Figure 3 shows the WER performance of the six scheduling schemes shown in Fig. 2 with respect to SNR after 50 decoding iterations. For the range of SNRs shown in the figure, "$L^*(2)$" performs the best and "$L^*(1)$" performs almost identical to "Min($\bar{S}$)" which represents the performance of using the minimum value of $\bar{S}$ for the switch. This sheds us with the information that L^* can serve as a better threshold for the switch than $\bar{S}$ in a mixed scheduling scheme. Note that our proposed schemes perform increasingly better than "LBP" as SNR increases or trapping set errors become more dominant in determining code performance.

V. Conclusion

In this letter, we have proposed a novel adaptive mixed scheduling scheme that can correct errors contained in trapping sets of LDPC codes without incurring many non-trapping set errors. In particular, the proposed scheme improved on the previous mixed scheduling scheme of [2] by nearly an order of magnitude in terms of WER. Simulation results have shown us that the average magnitude of LLRs of variable nodes normalized by their respective degrees can serve as a better threshold for the switch from a non-greedy scheduling scheme to an IDS scheme than the number of unsatisfied check nodes.

References

[1] A. Casado, M. Griot, and R. D. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," *IEEE International Conference on Communications*, 2007.

[2] A. Casado, M. Griot, and R. D. Wesel, "Improving LDPC decoders via informed dynamic scheduling," *IEEE Information Theory Workshop*, 2007.

[3] G. Elidan, I. McGraw, and D. Koller, "Residual belief propagation: informed scheduling for asynchronous message passing," in *Proc. Conference on Uncertainty in Artificial Intelligence*, MIT, USA, July 2006.

[4] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. Signal Processing Systems 2004*, pp. 107-112, Oct. 2004.

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

[6] H. Kfir and I. Kanter, "Parallel versus sequential updating for belief propagation decoding," *Physica A*, vol. 330, pp. 259-270, Sep. 2003.

[7] T. Richardson, "Error floors of LDPC codes," in *Proc. Allerton Conf. on Communication, Control, and Computing*, 2003.