Layered BP Decoding for Rate-Compatible Punctured LDPC Codes

Jeongseok Ha, Demijan Klinc, Jini Kwon, and Steven W. McLaughlin

Abstract— Rate-compatible punctured LDPC codes have shown to perform well over a wide variety of code rates, both theoretically and practically. However it has been reported that the belief propagation (BP) decoding for these codes converges slower than for unpunctured codes. Layered BP algorithm is a modified BP algorithm that accelerates the decoding convergence by means of sequential scheduling of check node updates. In this letter, we propose an efficient scheduling of check node updates for rate-compatible punctured LDPC codes that performs well. We show that the convergence speed of the proposed scheduling outperforms conventional (random) scheduling and conventional BP decoding. Performance improvements become more distinctive with the growing fraction of punctured bits.

Index Terms— Rate-compatible punctured LDPC codes, fast decoding, layered BP decoding, check node scheduling/layering.

I. INTRODUCTION

RATE-COMPATIBLE punctured low-density parity-check (RCP-LDPC) codes have been gaining increasing attention in the recent years because of their good performance over a range of rates [2] [3]. At a relatively small cost in implementation complexity they offer high flexibility in terms of code rate, which enables the communication system to increase throughput over time-varying channels [1]. The details of how RCP-LDPC codes should be punctured to achieve good performance over a wide range of code rates are already well understood [2] [3]. However, it was observed in [4] that RCP-LDPC codes require a higher number of iterations for successful decoding than unpunctured codes, especially when a high fraction of bits is punctured. Hence, if the decoder implementation limits the maximum number of iterations to a low number, the bit error rate (BER) performance of RCP-LDPC codes can be degraded significantly.

In this letter we show how the required number of iterations for RCP-LDPC codes can be reduced by using the layered belief propagation (BP) decoding algorithm [7]–[9]. Unlike a conventional BP decoder, a layered BP decoder divides check nodes into small subgroups called *layers* and breaks each iteration into multiple *subiterations*. In each subiteration one layer of check nodes and their neighboring variable nodes are processed. We propose a simple and efficient layering strategy

J. Ha and J. Kwon are with the School of Electrical and Computer Engineering, Information and Communications University, 119, Munjiro, Daejeon, 305-732, Korea (email: {jsha, jini2474}@icu.ac.kr).

D. Klinc and S. W. McLaughlin are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA (email: demi, swm@ece.gatech.edu).

for the check nodes that significantly increases decoding speed of RCP-LDPC codes as compared to when the check nodes are layered conventionally, that is randomly. It should be noted that it is also possible to layer variable nodes [10], but in this paper we only consider check node layering.

The next section summarizes layered BP decoding and introduces the proposed layering of the check nodes. Section III gives some simulation results, where the proposed layering is compared with the conventional layering techniques and the last section briefly concludes the presented work.

II. LAYERED BP DECODING AND PROPOSED LAYERING

Layered BP decoding divides the Tanner graph of an LDPC code into smaller subgraphs, such that each subgraph consists of a set of check nodes and all their neighboring variable nodes. Each check node appears in exactly one subgraph, while variable nodes can appear in multiple subgraphs. A conventional decoding iteration is split into multiple subiterations, such that in each subiteration the check node and variable node updates are calculated in one subgraph. The decoding then progresses sequentially through subgraphs by performing message updates subiteration by subiteration. A parity check test over the entire codeword is performed at the end of each subiteration.

Layered BP decoding yields faster decoding convergence, but it does not improve the BER when the maximum number of iterations is large enough. In practical communication systems, however, the maximum number of iterations is limited, in which case faster decoding convergence results in better BER performance.

In [9], "random" and "bimodal" layering of check nodes is studied for unpunctured regular LDPC codes. The performance differences between the layerings were shown to exist, but they are only marginal. On the other hand, if an LDPC code is punctured, significant performance gains can be achieved by a careful layering selection. In the following, we focus on how a good layering selection can be obtained and explain the underlying ideas.

At the start of the decoding process, the log-likelihood ratio of the punctured bits is set to 0, since the punctured bits are not transmitted and a decoder does not have any information about them from the channel. We say that a variable node corresponding to a punctured bit, call it punctured variable node, is *recovered* when it receives a non-zero message from at least one of its neighboring check nodes[1] for the first time. A check node that provides the first non-zero message is called the *survived check node*. For classification purposes we denote a punctured variable node as k-step recoverable (k-SR) [3], if

---

[1] The non-zero message does not have to imply the correct bit value.

it is recovered in the k-th iteration under conventional BP decoding. Equivalently, we say that its *level of recoverability* is k. Note that a punctured variable node can have multiple survived check nodes, while a check node can be a survived check node to at most one punctured variable node. Consequently, no two punctured variable nodes have a common survived check node. All unpunctured variable nodes are defined to be 0-SR and thus have a level of recoverability 0. For any given punctured LDPC code, the levels of recoverability of punctured variable nodes and identification of their survived check nodes can be determined using the framework introduced in [3]. In the following, we assume that all punctured variable nodes are recovered in a finite number of iterations, or in other words that no set of punctured variable nodes constitutes a stopping-set.

We propose a layering of the check nodes in accordance with the level of recoverability, that is such that the punctured variable nodes are recovered as quickly as possible. More precisely, if a check node is a survived check node to a k-SR punctured variable node, it is assigned to the layer L_k, where $1 \leq k \leq K$. By upper bounding k with K we assume that under the conventional BP algorithm all punctured variable nodes are recovered after the K-th iteration. All remaining check nodes, i.e. those that do not recover any punctured variable node, are assigned to the last layer L_{K+1}. Note that in cases when a high proportion of variable nodes are punctured, the layer L_{K+1} may not contain any check nodes.

If the layers from L_1 to L_{K+1} are processed sequentially in ascending order, all punctured variable nodes will be recovered after the first iteration. Hence, it is ensured that the punctured variable nodes get involved in the decoding process very quickly, which results in faster convergence. What follows is a short algorithm for determining the layer of each check node, where the integer M denotes the number of rows in a parity-check matrix, or equivalently the number of check nodes.

for $i = 1$ **to** M **do**
    let p be the highest level of recoverability
    among the variable nodes in the i-th row
    if $(1 \leq p \leq K)$
      the i-th check node is assigned to L_p
    else
      the i-th check node is assigned to L_{K+1}
end

Note that the layering should be performed by observing levels of recoverability of punctured variable nodes at the highest considered code rate. For all intermediate code rates, the layering remains unchanged.

III. SIMULATION RESULTS

To verify the effectiveness of the proposed layering scheme we construct an LDPC mother code with the degree distribution (from [11])

$$\lambda(x) = 0.30780x + 0.27287x^2 + 0.41933x^6 \text{ and}$$
$$\rho(x) = 0.4x^5 + 0.6x^6$$

with a code rate 0.5 and block length 2000. The mother code is punctured using the algorithm from [3] to obtain code rates

TABLE I
THE DISTRIBUTION OF CHECK NODES IN LAYERS. THE HIGHEST LEVEL OF RECOVERABILITY AMONG THE PUNCTURED VARIABLE NODES IS 5.

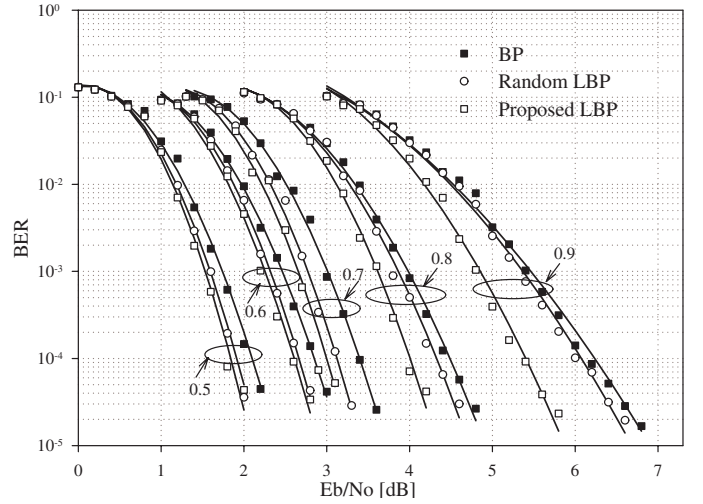|	Proposed	Random1	Random2
Layer 1	222	200	222
Layer 2	244	200	244
Layer 3	178	200	178
Layer 4	178	200	178
Layer 5	178	200	178

Fig. 1. BERs of punctured LDPC codes at code rates 0.5, 0.6, 0.7, 0.8 and 0.9 over an AWGN channel with the maximum number of iterations set to 15. The filled squares, unfilled circles, and unfilled squares represent the BERs of layered BP decoding with the proposed layering, random layering and the conventional BP decoding, respectively.

0.6, 0.7, 0.8, and 0.9. Detailed information about the check node layering is shown in Table I. Notice that the highest level of recoverability among the punctured variable nodes is 5 and each check node serves as a survived check node to one punctured variable node, hence there is no layer L_6.

We compare the proposed layering with the case when check nodes are layered randomly. In one case we create a random layering such that the layer sizes are uniform and in the other, the layer sizes match those from the proposed layering. The performance of these two random layerings is virtually indistinguishable, therefore they are represented by a single curve in the following figures.

Fig. 1 shows the simulation results over an AWGN channel with the maximum number of iterations set to 15. The performance improvement of the proposed over random layering grows with the number of punctured bits, or equivalently with the growing code rate. This behavior is expected, for as the number of punctured bits grows higher, a carefully selected layering has an increasing effect on the speed of their recovery. At the code rate 0.9 the E_b/N_0 gain over layered BP decoding with random layering and conventional BP decoding is around 0.7 dB and 0.8 dB at the BER of 10^{-4}, respectively. Notice that the proposed layering outperforms random layering at all considered code rates, including the unpunctured case (mother code). As a result, the check node layering can stay unchanged over all code rates which further decreases the implementation complexity.
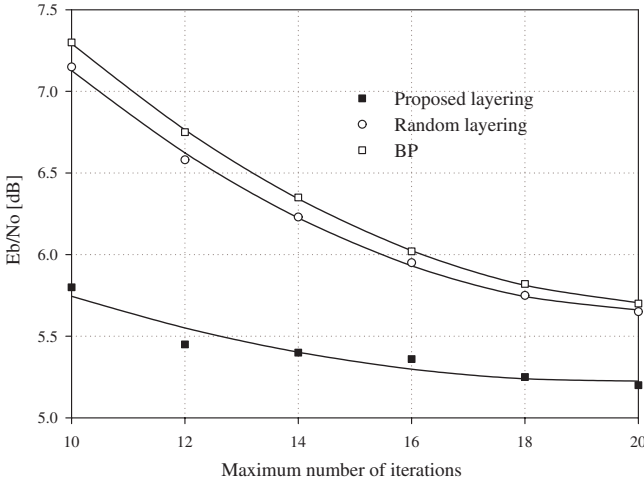
Fig. 2. The required E_b/N_0 values for a BER of 10^{-4} at the code rate 0.9 with a range of the maximum number of iterations which starts from 10 to 20 by an increment of 2, where the LDPC codes are the ones in Fig. 1.
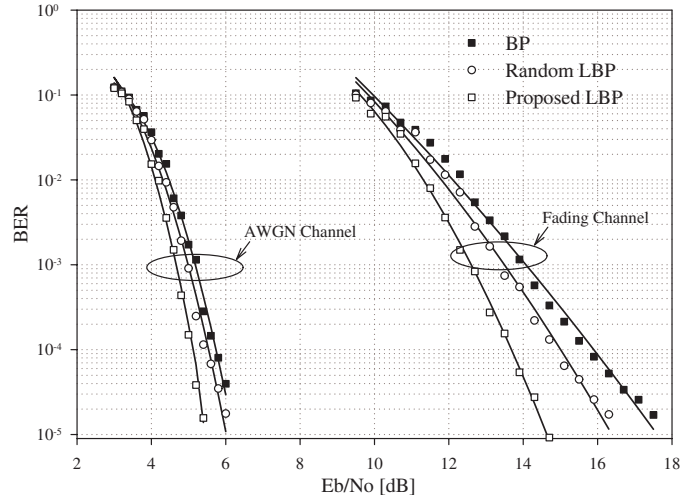

Fig. 3. BER performances of punctured LDPC codes at the code rate 0.88 over an AWGN and a fading channels whose base LDPC code is from 3GPP-LTE proposed by ZTE. The filled squares, unfilled circles, and unfilled squares represent the BERs of layered BP decoding with the proposed layering, the random layering and the conventional BP decoding, respectively.

On the other hand, we are also interested in the performance gain at different values for the maximum number of iterations. In Fig. 2 we compare the required E_b/N_0 values for achieving BER of 10^{-4} at the code rate 0.9 with the maximum number of iterations ranging from 10 to 20 in increments of 2. When the maximum number of iterations is set to 10 the gain of the proposed layering is 1.3 dB and 1.5 dB over layered BP decoding with random layering and conventional BP decoding, respectively. The performance improvements reduce to about 0.5 dB when the maximum number of iterations is set to 20.

Finally, we verify the proposed layering in a more realistic scenario, i.e. with an LDPC code that has been proposed for 3GPP-LTE (The 3rd Generation Partnership Project, Long-Term Evolution). More specifically, we test our idea on RCP-LDPC codes with a structured mother code of length 2080 and code rate 0.5 that were proposed by ZTE in [12]. We analyze their punctured code at the code rate 0.88 with the algorithm in [3] and layer the survived check nodes as previously proposed.

The BER performance of the punctured LDPC code is evaluated over two channels: an AWGN channel and a fading channel, where on the fading channel each coded bit is subject to independent fading with a unit channel gain. The simulation results are depicted in Fig. 3. We see that the performance improvement due to the proposed layering becomes even more distinctive over the fading channel. The gain over the random layering and conventional BP decoding is 1 dB and 2 dB at the BER of 10^{-4}, respectively.

IV. CONCLUSION

We propose a layering scheme for layered BP decoding of RCP-LDPC codes. The proposed layering is an answer to a drawback associated with RCP-LDPC codes as presented in [4] – slow decoding convergence. Our idea is particularly useful when the maximum number of iterations is limited due to implementation complexity or latency issues.

The proposed layering yields faster decoding convergence than both layered BP decoding with random layering and con-ventional BP decoding. This claim is supported by simulation

results over AWGN and fading channels. We show that the proposed layering is superior at all considered code rates with random and structured LDPC codes. Moreover, significant improvements in BER performance can be achieved if the maximum number of iterations is kept low, which is usually the case in practical systems.

REFERENCES

[1] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389–400, Apr. 1988.

[2] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. IT-50, pp. 2824–2836, Nov. 2004.

[3] J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *IEEE Trans. Inf. Theory*, vol. IT-52, pp. 728–738, Feb. 2006.

[4] D. Klinc, J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible punctured low-density parity-check codes for Ultra Wide Band systems," in *Proc. IEEE GLOBECOM*, St. Louis, U.S.A., Nov. 2005, pp. 3856–3860.

[5] G. Yue, X. Wang, and M. Madihian, "Design of rate-compatible irregular repeat accumulate codes," *IEEE Trans. Commun. preprint*.

[6] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, "Design of efficiently-encodable rate-compatible irregular LDPC codes," in *Proc. IEEE Int. Conf. Commun. 2006*.

[7] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. SIPS 2004*, pp. 107–112.

[8] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, "High throughput low-density parity-check decoder architectures," in *IEEE Int. Conf. Commun. 2001*, pp. 3019–3024.

[9] S. H. Kim, "Analysis of quasi-cyclic low-density parity-check codes and protograph codes," Ph.D. dissertation, Seoul National University, Seoul, Korea, 2006.

[10] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb. 2005.

[11] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, pp. 657–670, Feb. 2001.

[12] 3GPP, R1-061019, ZTE, "Structured LDPC Coding with Rate Match-ing," TSG RAN1#44bis, Mar. 2006.