

Fast Decoding of Rate-Compatible Punctured LDPC Codes

Jini Kwon* and Demijan Klinc[†]

Jeongseok Ha* and Steven W. McLaughlin[†]

*School of Electrical and Computer Engineering, Information and Communications University
119, Munjiro, Daejeon, 305-732, Korea

Email: {jini2474, jsha}@icu.ac.kr

[†]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA

Email: {demi, swm}@ece.gatech.edu

Abstract—While rate-compatible punctured low-density parity-check (RCP-LDPC) codes offer high flexibility in terms of code rate at a relatively low cost in implementation complexity, they are reported to require more decoding iterations than unpunctured LDPC codes. In this paper¹ we consider layered belief propagation decoding and propose efficient check node layering that significantly accelerates the decoding convergence of RCP-LDPC codes. We show that the proposed layering outperforms both random layering and conventional BP decoding. The performance improvements become more distinctive at high rates and they come at no additional implementation cost.

I. INTRODUCTION

Low-Density Parity-Check (LDPC) [1] codes have been intensively studied since their rediscovery [2]. Some recent work has been focused on rate-compatible LDPC codes since they offer high flexibility in terms of the code rate at a relatively small cost in implementation complexity.

Rate-compatible punctured convolutional codes were introduced in [3]. Ha *et al.* showed that puncturing can be successfully applied to LDPC codes by showing that it is possible to design capacity-approaching rate-compatible punctured LDPC (RCP-LDPC) codes at very long block lengths [4], [5]. Soon afterward puncturing methods for RCP-LDPC codes at short block lengths were presented in [6], [7]. These results served as a foundation for the work in [9]–[11], where the focus was on improving the performance of codes at high rates and designing codes with linear-time encoders. One issue of importance, common to all types of RCP-LDPC codes, is slow convergence of the belief-propagation (BP) decoding [8]. In other words, the decoder generally requires more iterations to find the transmitted code word. The reason is quite intuitive. Since some of the bits are punctured and hence not transmitted at all, the decoder is delayed by the fact that it needs to derive an estimate of the punctured bits first. In real applications this can pose a serious problem as slower convergence can result in higher power consumption and/or longer latency. To the best of our knowledge, no solution to this problem has been published so far.

¹The work of Ha and Kwon was supported by Electronics and Telecommunications Research Institute (ETRI) and the work of Klinc and McLaughlin was supported in part by Samsung Electronics Co. Ltd., Korea.

In this paper, we show how convergence of RCP-LDPC codes can be significantly accelerated using the *layered BP* algorithm. Proposed in [12]–[14], the layered BP algorithm is a modification of the conventional BP algorithm, where the check nodes are divided in subgroups called *layers* and each iteration is broken into multiple *subiterations*. In each subiteration one layer of check nodes and their neighboring variable nodes are processed. Kim *et al.* indicated in [14] that smart layering of check nodes can be important for achieving faster convergence. However, [14] only deals with unpunctured LDPC codes, where different layering schemes result in only marginal performance alternations. On the contrary, we show that the performance of the layered BP decoding with RCP-LDPC codes depends strongly on the chosen layering of check nodes. We propose a simple and efficient layering algorithm that significantly increases decoding speed of RCP-LDPC codes as compared to when the check nodes are layered randomly, or when conventional BP algorithm is used.

The remainder of this paper is organized as follows. Section II briefly summarizes the layered BP algorithm. Subsequently, Section III introduces the ideas behind the proposed layering of check nodes and gives a short algorithm for determining the layer of each check node. Section IV compares the performance of the proposed layering with random layering and with conventional BP decoding. Last section briefly concludes the presented work.

II. LAYERED BP ALGORITHM

An LDPC code is usually specified by a parity-check matrix or by a Tanner graph [16] as exemplified in Fig. 1. The check nodes (squares) and variable nodes (circles) in the Tanner graph represent the rows and columns of the parity-check matrix, respectively. The non-zero terms in the parity-check matrix define the connections between the check nodes and variable nodes in the Tanner graph. Let $C = \{c_1, c_2, \dots, c_M\}$ be the set of all check nodes and let $V = \{v_1, v_2, \dots, v_N\}$ be the set of all variable nodes. For an arbitrary node x , we say that the neighbors of x are all nodes that can be reached from x by traversing exactly one edge in the Tanner graph. The neighbors of x are denoted by \mathcal{N}_x .

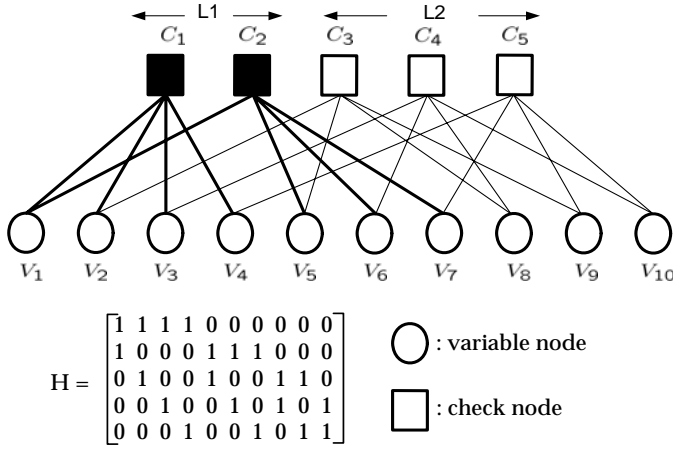


Fig. 1. A parity-check matrix and the corresponding bipartite graph; L_1 and L_2 are check node layers.

The layered BP decoding divides the Tanner graph of an LDPC code into smaller subgraphs, called layers, 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.

More specifically, for an LDPC code defined by an $M \times N$ parity-check matrix, the layered BP algorithm is defined as follows:

• *Initialization:*

- 1) Group the check nodes into $K + 1 \geq 1$ subgroups L_k for $k = 1, \dots, (K+1)^2$ such that $\bigcup_{k=1}^{K+1} L_k = C$ and $L_i \cap L_j = \emptyset$ for all $i \neq j$. Each subgroup L_i is called a layer. In Fig. 2, there are two layers and $L_1 = \{c_1, c_2\}$, $L_2 = \{c_3, c_4, c_5\}$ and $K + 1 = 2$.
- 2) The log-likelihood ratio (LLR) q_i , $1 \leq i \leq N$ of each variable node is initialized to the channel LLR which equals

$$L_i = \log \frac{\Pr[v_i = 0 | \mathbf{r}]}{\Pr[v_i = 1 | \mathbf{r}]}$$

where $\mathbf{r} = (r_1, r_2, \dots, r_N)$ is a received vector of length N . We assume binary antipodal signaling of transmitted bits such that $0 \in \text{GF}(2) \leftrightarrow +1 \in \mathbb{R}$ and $1 \in \text{GF}(2) \leftrightarrow -1 \in \mathbb{R}$.

- 3) All check node messages $r_{ji}^{(0,\xi)}$ for $c_j \in L_\xi$, $\xi = 1, \dots, K + 1$, and $v_i \in \mathcal{N}_{c_j}$ are initialized to 0, respectively.
- 4) $\ell = 1$ and $\xi = 1$.

²We use $K + 1$ instead of K for consistency with our proposed layering algorithm which will be explained shortly.

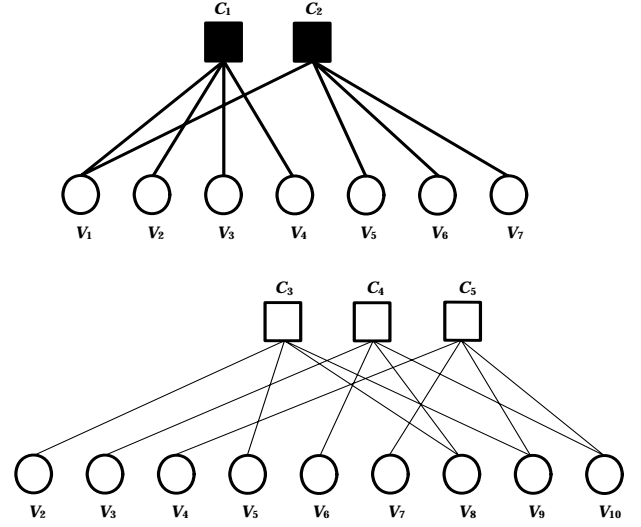


Fig. 2. Two layers (L_1 and L_2) of the Tanner graph in Fig. 1 and their corresponding subgraphs.

• *Iterative Processing:*

- 1) **Check Node Update:** For each check node $c_j \in L_\xi$, compute the updated message to its neighboring variable nodes $v_i \in \mathcal{N}_{c_j}$

$$r_{ji}^{(\ell,\xi)} = 2 \tanh^{-1} \left(\prod_{k \in \mathcal{N}_{c_j} \setminus i} \tanh \left(\frac{q_k - r_{jk}^{(\ell-1,\xi)}}{2} \right) \right).$$

- 2) **Variable Node Update:** For each variable node $v_i \in \mathcal{N}_{c_j}$, $c_j \in L_\xi$, update its LLR

$$q_i = q_i + \sum_{k \in \mathcal{N}_{v_i}} r_{ki}^{(\ell,\xi)} - r_{ki}^{(\ell-1,\xi)}.$$

- 3) **Parity-Check Test:**

- (a) Create $\hat{\mathbf{v}} = (\hat{v}_1, \hat{v}_2, \dots, \hat{v}_n)$ such that $\hat{v}_i = 1$ if $q_i < 0$ and $\hat{v}_i = 0$ if $q_i \geq 0$.
- (b) If $\mathbf{H}\hat{\mathbf{v}} = \mathbf{0}$, then the decoding algorithm halts, and $\hat{\mathbf{v}}$ is declared to be the valid codeword.
- (c) $\xi = \xi + 1$ and if ξ is smaller than or equal to $K + 1$, repeat the algorithm from Check Node Update.
- (d) $\ell = \ell + 1$ and $\xi = 1$, and if ℓ is bigger than the maximum number of iteration, it declares decoding failure and halts.
- (d) repeat the algorithm from Check Node Update.

Under this framework, the conventional BP algorithm can be seen as layered BP with a single layer containing all check nodes. The layer BP decoding yields faster decoding convergence, but it does not improve the bit-error rate (BER) performance when the maximum number of iteration is big

enough. However, in practical systems the maximum number of iterations is usually limited and therefore faster convergence can significantly improve the BER performance. We note that it is also possible to layer variable nodes [15], but in this paper we only consider check node layering.

III. PROPOSED LAYERING

In [14], “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 the differences were relatively small. 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 LLR of the punctured bits is set to 0, since they 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 nodes³ 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) [7], if it is recovered in the k -th iteration under conventional BP decoding. Equivalently, we say that its level of recoverability is k .

Notice 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 [7]. 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 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 nodes, are assigned to the last layer L_{K+1} . In cases when a high proportion of variable nodes is punctured, the layer L_{K+1} may not contain any check nodes.

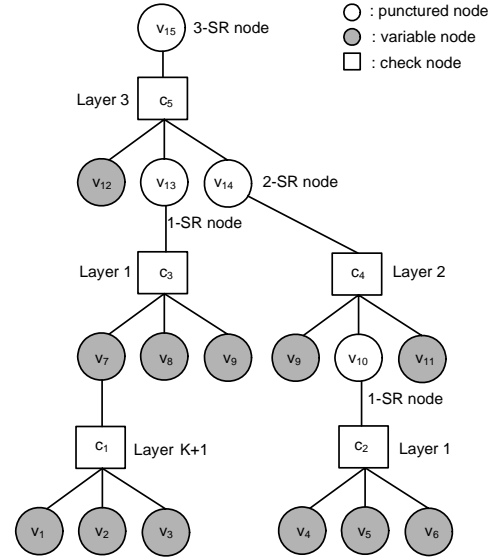


Fig. 3. Recovery tree of a 3-SR node where $c_2, c_3 \in L_1$, $c_4 \in L_2$, $c_5 \in L_3$, $c_1 \in L_{K+1}$, and $K \geq 3$.

Let us observe on an example in Fig. 3 how a 3-SR node is recovered under layered BP decoding if the proposed layering of check nodes is used. In the first subiteration the check nodes c_2 and c_3 (both in L_1) recover 1-SR variable nodes v_{10} and v_{13} . Subsequently in the second subiteration, the check node c_4 recovers 2-SR variable node v_{14} . Now all neighbors of c_5 have non-zero LLRs (except v_{15}), therefore in the next, third subiteration, v_{15} will be recovered by c_5 .

If the layers from L_1 to L_{K+1} are processed sequentially in the ascending order, all punctured variable nodes will be recovered after the first iteration, or more exactly after the first K subiterations. Hence, it is ensured that the punctured variable nodes get involved in the decoding process very quickly, which results in faster convergence. 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, is given as follows

```

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.

IV. SIMULATION RESULTS

In this section, we show that the proposed layering (compared to random layering) requires less iterations for the same

³The non-zero message does not have to imply the correct bit value.

TABLE I

THE NUMBER OF PUNCTURED BITS IN EACH LAYER WHERE THE RANDOM LAYERING TAKES EQUALLY DIVIDED 1000 ROWS AND THE PROPOSED LAYERING HAS SURVIVED CHECKS FROM 0-SR TO 5-SR NODES.

	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

BER performance or gives a better BER performance for a given maximum number of iterations. Our claim is verified by comparing the performance of the conventional BP and the layered BP algorithms using random and proposed layering.

We design an LDPC mother code of rate 0.5 and block length 2000 whose degree distribution pair from the edge perspective [17] is

$$\begin{aligned}\lambda(x) &= 0.30780x + 0.27287x^2 + 0.41933x^6 \text{ and} \\ \rho(x) &= 0.4x^5 + 0.6x^6.\end{aligned}\quad (1)$$

This code is punctured using the algorithm in [7] to obtain a set of RCP-LDPC codes at rates 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.

The simulation results over an AWGN channel are shown in Fig. 4 – the maximum number of iteration was set to 15. Clearly, improvements in performance due to the proposed layering become more distinctive with the increasing code rate. For instance, at the code rate 0.9 the E_b/N_0 gain over the 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. As a result, the check node layering can stay unchanged over all code rates which further decreases the implementation complexity.

We also wanted to see how the BER performance changes at different values for the maximum number of iterations. In Fig. 5 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 the 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.

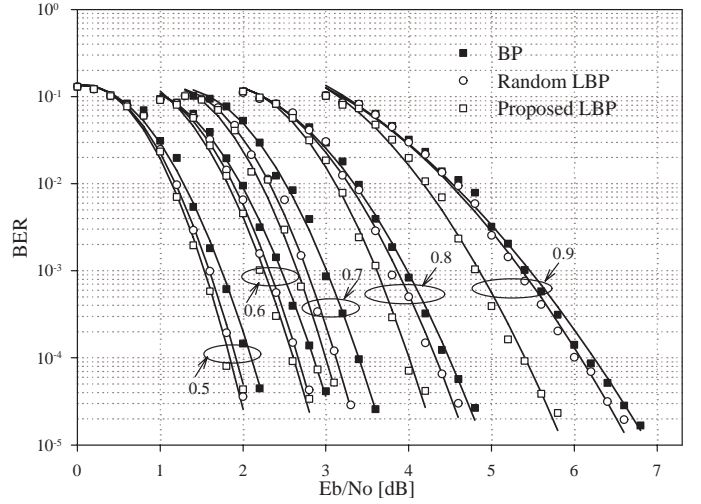


Fig. 4. BERs of punctured LDPC codes at rates 0.5, 0.6, 0.7, 0.8 and 0.9 over an AWGN channel with the maximum number of iteration set to 15. The filled squares, unfilled circles, and unfilled squares represent the BERs of the layered BP decoding with the proposed layering, the random layering and the conventional BP decoding, respectively.

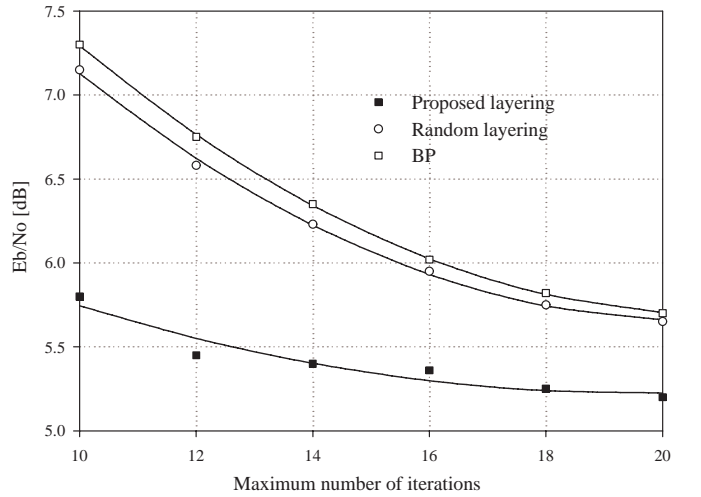


Fig. 5. The required E_b/N_0 values for a BER of 10^{-4} at rate 0.9 with a range of the maximum number of iterations which starts from 10 to 20 by an increment of 2, where the LDPC codes are the ones in Fig. 4.

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 [18]. We analyze their punctured code at the code rate 0.88 with the algorithm in [7] 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 the unit channel gain.

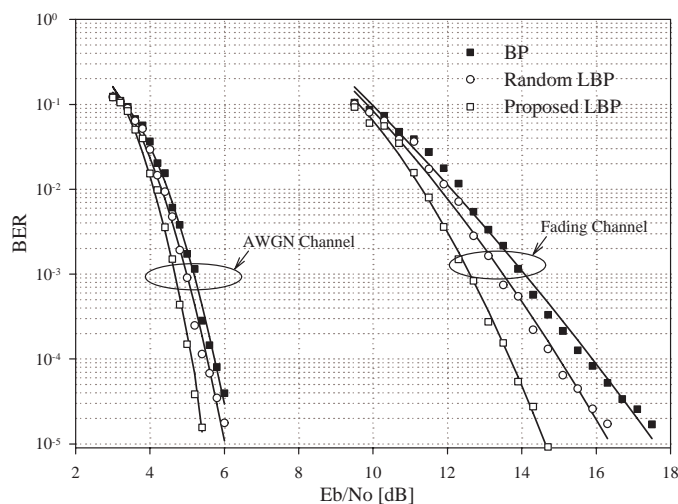


Fig. 6. BER performances of punctured LDPC codes at rate 0.88 over an AWGN and a fading channels whose base LDPC code is from 3GPP-LTE proposed by ZTE. The filled squares, unfilled circles, and unfilled squares represent the BERs of the layered BP decoding with the proposed layering, the random layering and the conventional BP decoding, respectively.

The simulation results are depicted in Fig. 6. 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.

V. CONCLUSION

This paper discusses a solution to an inherent weakness of RCP-LDPC codes – their slow decoding convergence [8]. We propose a simple and efficient algorithm that finds good check node layerings under the layered BP decoding. The check nodes are layered such that in each subiteration all punctured variable nodes with the same level of recoverability are recovered. That is, in the first subiteration all 1-SR nodes are recovered, in the second subiteration all 2-SR node are recovered, etc. Consequently, the punctured variable nodes get involved in the decoding process much quicker than with random layering selections or under conventional BP decoding.

The proposed layering yields faster decoding convergence than both layered BP decoding with random layering and conventional 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.

The proposed idea is particularly useful when the maximum number of iterations of LDPC decoders is limited due to practical reasons.

REFERENCES

- [1] R. G. Gallager, "Low-density parity check codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–22, Jan. 1962.
- [2] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. IT-45, pp. 399–431, Mar. 1999.
- [3] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCP codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389–400, Apr. 1988.
- [4] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. IT-50, pp. 2824–2836, Nov. 2004.
- [5] J. Ha and S. W. McLaughlin, "Optimal puncturing of irregular low-density parity-check codes," in *IEEE Int. Conf. Commun.*, Anchorage, Alaska, May 2003, pp. 3110–3114.
- [6] J. Ha, J. Kim, and S. W. McLaughlin, "Puncturing for finite length low-density parity-check codes," in *Proc. Int. Symp. Information Theory*, Chicago, USA, 2004, p. 151.
- [7] J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *IEEE Trans. Inform. Theory*, vol. IT-52, pp. 728 – 738, Feb. 2006.
- [8] 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.
- [9] E. Choi, S.-B. Suh, and J. Kim, "Rate-compatible puncturing for low-density parity-check codes with dual-diagonal parity structure," in *16th Int. Symp. on Personal Indoor and Mobile Radio Commun. (PIMRC)*, Berlin, Germany, 2005.
- [10] G. Yue, X. Wang, and M. Madhian, "Design of rate-compatible irregular repeat accumulate codes," *IEEE Trans. Commun.*
- [11] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, "Design of efficiently-encodable rate-compatible irregular LDPC codes," in *IEEE Int. Conf. Commun.*, 2006.
- [12] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *SIPS*, 2004, pp. 107–112.
- [13] 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.
- [14] S. H. Kim, "Analysis of quasi-cyclic low-density parity-check codes and protograph codes," Ph.D. dissertation, Seoul National University, Seoul, Korea, 2006.
- [15] J. Zhang and M. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209 – 213, Feb. 2005.
- [16] M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.
- [17] 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. Inform. Theory*, vol. IT-47, pp. 657–670, Feb. 2001.
- [18] 3GPP, R1-061019, ZTE, "Structured LDPC Coding with Rate Matching," TSG RAN1#44bis, Mar. 2006.