Efficient Puncturing Scheme for Irregular LDPC Codes Based on Serial Schedules

Hua Li and Linhua Zheng

***Abstract*—In practice, low-density parity-check (LDPC) codes are usually decoded by serial schedules. However, the existing schemes for rate-compatible puncturing LDPC codes are proposed on the assumption that the flooding schedules are employed in decoding algorithm. In this letter, an efficient puncturing scheme is proposed for irregular LDPC codes based on serial schedule. The scheme selects variable nodes to be punctured one at a time. At each time, a check node is selected firstly based on the idea that the punctured variable nodes are allocated evenly to all the check nodes. Then the puncturing variable node is selected among its neighbors. Furthermore, the order of the check nodes in the selection serves as the updating order in the serial schedule, which ensures that all the punctured codes can be recovered at the first iteration. Simulation results demonstrate that the scheme performs better for a wide range of code rates compared with the existing puncturing scheme, especially for high-rate punctured codes.**

***Index Terms*—Rate-compatible punctured LDPC codes, high-rate punctured codes, serial schedules, approximate cycle extrinsic message degree (ACE).**

I. Introduction

RATE compatible codes, which allow coding across a range of rates using a common encoder/decoder infrastructure, are essential for many communication applications operating over time variant channels. The most common way to generate rate-compatible codes is puncturing, i.e., starting with a low-rate mother code and then selectively discarding some of the coded bit to arrive at higher rate codes. The puncturing technique was firstly applied to convolutional codes by Hagenauer [1].

Rate-compatible punctured low-density parity-check (LDPC) codes have been gained increasing attention in recent years because of their good performance over a range of rate [2]–[11]. Theoretically, The puncturing distributions for punctured LDPC codes were investigated and their theoretical performance was analyzed [3]. In [4], the maximum achievable rate of punctured LDPC codes was identified. For practical issues, the punctured LDPC codes with finite length was studied in [5], where the locations of punctured nodes were selected randomly. Nevertheless, randomly punctured LDPC codes may suffer severe performance loss. To overcome the performance loss, the popular grouping/sorting algorithm was proposed by Ha *et al.* [6]. This algorithm punctured finite-length LDPC codes based on the classification of variable nodes. Later in [7], an efficient puncturing algorithm was presented for rate-compatible LDPC codes based on the proposed cost function. Both the algorithms yielded better performance than the random puncturing in their simulation results. To further improve the performance of punctured LDPC codes, the distances of punctured variable nodes away from each other were regarded as the criteria to select punctured nodes [8]. In [9], the number of short cycles with low approximate cycle extrinsic message degree (ACE) played an important role in selecting punctured nodes. These schemes performed well over a wide range of code rates. However, they were proposed on the assumption that the belief propagation (BP) algorithm employs flooding schedules. When the decoder of punctured codes utilizes serial schedules, they are not optimal. In [10], [11], the LDPC codes were designed to be well-suited for puncturing and encoding, which were referred as efficiently-encodable rate-compatible LDPC codes. Although good performance was obtained in high-rate punctured codes, these methods can not be generalized to other types of LDPC codes.

In this letter, we propose an efficient puncturing scheme for irregular LDPC codes. The punctured codes obtained by this scheme have faster convergence in the decoding process than these obtained by existing schemes. In practice, the maximal number of the allowed decoding iterations is usually limited due to latency or throughput constraints. Hence, faster convergence means better error rate performance. This scheme selects variable nodes to be punctured one at a time. Before selecting each puncturing node, the scheme selects a check node firstly. In this way, the proposed scheme not only selects the set of the punctured variable nodes but also provides check-nodes order for updating in the serial schedule. Although serial schedules can be operated by serially updating variable-node messages and check-node messages, in this letter the serial schedules are specifically referred to as the latter. Besides, there are two criteria to be considered: (1) minimizing the degree of the punctured nodes, (2) minimizing the number of punctured variable nodes involved in short cycles with low ACE. Criterion (1) only works in the former selection of puncturing codes. While criterion (2) plays an important role in the later selection.

II. LDPC Codes and Serial Schedules

A LDPC code, which is defined by a sparse parity-check matrix H with $M \times N$ size, can be equivalently expressed by a Tanner graph $G = \{C \cup V, E\}$, where $C = \{c_1, c_2, \cdots, c_M\}$ and $V = \{v_1, v_2, \cdots, v_N\}$ are the sets of the variable nodes and the check nodes, respectively, and E is the set of the edges. For an arbitrary node w, the neighbors set $N(w)$ is defined as the set

Manuscript received April 13, 2015; revised June 22, 2015; accepted July 2, 2015. Date of publication July 8, 2015; date of current version September 4, 2015. The associate editor coordinating the review of this paper and approving it for publication was H. Saeedi.

The authors are with the School of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China (e-mail: hjhylihua@126.com; lhzheng@nudt.edu.cn).

Digital Object Identifier 10.1109/LCOMM.2015.2453314

of nodes that can be reached from w by traversing only one edge. The cardinality of $N(w)$ is called the degree of w, denoted as $d(w)$.

ACE is proposed as the metric to measure the cycle connectivity in Tanner graph of LDPC codes and has an important effect on the error-floor performance [12]. Consider the ACE of a $2l$-length cycle. If no variable nodes in the cycle share check nodes outside of the cycle, then the ACE of the cycle is $\sum_{i=1}^{l}(d(v_i)-2)$, where $d(v_i)$ is the degree of the ith variable node in this cycle. Since short cycles with low ACE lead to performance degradation [12], it is desirable to avoid puncturing the variable nodes of such cycles. To evaluate the extent in which different variable nodes participate in short cycles with low ACE values, similar to [9], the ACE score spectrum of a variable node is defined as follows: Let $C_v(l)$ denote the set of the length $2l$ cycles in which variable node v participates. $S^v_{\eta_{ACE}}(l)$ represents the number of cycles with the ACE values more than η_{ACE} in set $C_v(l)$. The ACE score spectrum $\mathbf{S}^v_{\eta_{ACE}}(l_{max})$ of variable node v is then defined as the $(l_{max}-1)$-tuple $\{S^v_{\eta_{ACE}}(2), S^v_{\eta_{ACE}}(3), \cdots, S^v_{\eta_{ACE}}(l_{max})\}$.

LDPC codes can be efficiently decoded by iterative message-passing decoding algorithm, such as belief propagation (BP) algorithm. These algorithms operate on the Tanner graph representation of the code by iteratively exchanging messages between the variable and check nodes along the edges of the graph. The order of passing messages along the graph edges is referred to as a schedule. Conventionally, flooding schedule is widely used in BP algorithm, but serial schedule is proved to be a more efficient schedule because the propagation of messages on the graph is much faster [13]. In this letter serial schedules are specifically referred to the serial schedules which are done by serially updating check-node messages. Thus, serial schedule can be defined by a bijection:$\{c_1, c_2, \cdots, c_M\} \rightarrow \{1, 2, \cdots, M\}$. Besides, serial schedule can be partially parallelized, where the $\{c_1, c_2, \cdots, c_M\}$ is divided into the subsets, such that each check node appears in exactly one subset and no two check nodes in a subset are connected to the same variable node. The proposed puncturing scheme can be also performed with partially serial schedules. However, in the part III and part IV, we will mainly concentrate on the serial schedule.

III. The Proposed Puncturing Scheme

Consider a LDPC mother code with rate r_0 and the parity-check matrix H with size $M \times N$. To obtain the sequence of rate-compatible LDPC codes with rates $r_m > \cdots > r_1 > r_0$, the required number of punctured nodes is given by

$$np_k = \left\lfloor \frac{N(r_k - r_0)}{r_k} \right\rfloor \tag{1}$$

where $1 \le k \le m$. Let P_k denote the set of punctured nodes to obtain punctured code with rate r_k. The proposed puncturing scheme starts by puncturing np_1 variable nodes to obtain the first punctured code with rate r_1 in the sequence. P_1 contains the indices of the np_1 punctured nodes. It then continues by puncturing $np_2 - np_1$ more variable nodes, for a total of np_2 punctured nodes, to obtain the second punctured code with rate r_2 in the sequence. The indices of the np_2 punctured nodes are contained in P_2. This process continues until np_m variable nodes for the code with rate r_m are punctured. P_m contains the indices of the np_m punctured nodes.

The proposed scheme assumes that the decoding algorithm of the punctured LDPC codes uses serial schedule. The updating order of check nodes is denoted as the set $O = \{c^1, c^2, \cdots, c^M\}$, which is provided by the proposed puncturing scheme. At the begining O is an empty set. In the puncturing process, we select a check node to the set O before selecting each puncturing node. To select the i-th puncturing node, we first select the i-th check node c^i from the set $C \setminus O$. c^i is the check node with minmum $p(c)$ among the set $C \setminus O$, where $p(c)$ is denoted as the cardinality of the set $N(c) \cap P_k$ $(1 \le k \le m)$. Then we find the set $V_i = N(c^i) \setminus P_k$ $(1 \le k \le m)$ and select a puncturing node from it. If $|V_i| = d(c^i)$, the variable node v with minimum $d(v)$ is selected to P_k from V_i. If $|V_i| < d(c^i)$, find the ACE score spectrum $\mathbf{S}^v_{\eta_{ACE}}(l_{max})$, $v \in V_i$, for given values of η_{ACE} and l_{max}. The variable node v with the minimum number of short cycles with low ACE is selected to P_k from V_i. We continue the process until np_m variable nodes are selected. In the selection, if there are more than one variable node with minimum $d(v)$ or ACE score spectrum, select one at random.

The Proposed Scheme:

Step 0.0 [Inputs] Parity-check matrix H with size $M \times N$; sequence of rates $r_0, r_1, \cdots, r_m$; η_{ACE}; l_{max}.

Step 1.0 [Initialization] $P_0 = P_1 = P_2 = \cdots = P_m = \emptyset$, $O = \emptyset$, $k = 0, j = 0$.

Step 2.0 [Iteration] $k = k + 1$, calculate np_k by (1), $P_k = P_{k-1}$, $\delta np = np_k - |P_{k-1}|$.

Step 2.1.0 $j = j + 1$. Select c^j from $C \setminus O$, such that $P(c^j) \le p(c)$ for all $c \in (C \setminus O)$. If there are multiple such nodes, select one randomly as c^j. $O = O \cup \{c^j\}$.

Step 2.1.1 Make $V_j = N(c^j) \setminus P_k$, if $|V_j| = |N(c^j)|$, go to step 2.1.2, else go to 2.1.3.

Step 2.1.2 Pick the node v^* from V_j such that $d(v^*) \le d(v)$ for all $v \in V_j$. If there are multiple such nodes, pick one randomly as v^*, then go to step 2.1.4.

Step 2.1.3 Set $l = 2$ and $\Lambda = \emptyset$;

(a) $\Lambda = \{v' \in V_j | S^{v'}_{\eta_{ACE}}(l) \le S^v_{\eta_{ACE}}(l),\ \forall v \in V_j\}$.

(b) If $|\Lambda| = 1$, set $v^* = \Lambda$ and go to step 2.1.4.

(c) If $l < l_{max}$, set $l = l + 1$ and go to (a); else pick v^* from Λ randomly and go to step 2.1.4.

Step 2.1.4 $P_k = P_k \cup \{v^*\}$, $\delta np = \delta np - 1$, if $\delta np \neq 0$, go to step 2.1.0, else go to step 2.2.

Step 2.2 If $k = m$, stop the iteration, else go to step 2.0.

In the proposed scheme, O will contain row indices of the check nodes whose order would be used as the updating order in serial schedule. As the size of O is np_m at the end of the scheme. Thus the rest of the check nodes $(M - np_m)$ are randomly located in the set O. In Step 2.0, P_k inherits the column indices from P_{k-1}, and δnp accounts for the number of additional nodes which are needed to puncture besides the ones in P_{k-1}. The loop between Step 2.1.0 and Step 2.1.4 continues until $\delta np = 0$. In Step 2.1.0, the number j is the index of the puncturing order of the punctured nodes. In Step 2.1.3, for $v \in V_j$, calculate the ACE score spectrum $\mathbf{S}^v_{\eta_{ACE}}(l_{max})$ for given values of η_{ACE} and

l_{max}. Select the variable node with the smallest ACE scores $S^v_{\eta_{ACE}}(2)$. If there are more than one nodes with the same $S^v_{\eta_{ACE}}(2)$, select the one with the smallest $S^v_{\eta_{ACE}}(3)$. Continue the process by going through the components of the ACE score spectrum in the increasing order until a variable node is selected. If there are still a number of variable nodes with the same $S^v_{\eta_{ACE}}(l)$ when $l = l_{max}$, select one randomly. When k increases to m ($k = m$ in Step 2.2), the algorithm stops.

IV. Simulation Results

In this section, we present the results on puncturing two irregular LDPC mother codes of rate $r_0 = 1/2$ with the proposed scheme described in Section III. The one is randomly constructed by progressive edge-growth (PEG) algorithm with block length $N = 1008$ and the other is structured LDPC code specified in 802.16e systems with block length $N = 1152$. Simulations are conducted with binary phase shift keying (BPSK) modulation over additive white Gaussian noise (AWGN) channel. Notations E_b and N_0 denote the average energy per information bit and the one-sided power spectral density of the AWGN respectively. To guarantee statistical confidence, the BER performances are measured after observing at least 100 erroneous code bits at each E_b/N_0 value. The BP algorithms with serial schedules are used for decoding the rate-compatible punctured codes. The updating order of check nodes is $\{c_1, c_2, \cdots, c_M\}$ ($M = 504$ and 576 in the two examples respectively), which is provided by the proposed puncturing scheme.

We firstly implement the randomly constructed LDPC code (1008, 504) defined in [14] and use the proposed scheme to obtain a rate-compatible sequence of codes with rates 0.5, 0.6, 0.7, 0.8 and 0.9. At the proposed scheme, we select punctured variable nodes one at a time along the order of parity-check nodes in serial schedule. Then the locations of punctured bits are determined by two criteria: $d(c)$ and ACE score spectrum $\mathbf{S}^v_{\eta_{ACE}}(l_{max})$. Fig. 1 compares the proposed scheme with the scheme in [9]. In the both scheme, we set $l_{max} = 10$ and $\eta_{ACE} = 4$ (referred to as η_{max} in [9]). The maximum number of iterations is set to 15. It is obvious that the punctured codes derived from the proposed scheme outperform those from the existing scheme at all the simulated rates. At the rate 0.6, the number of punctured variable nodes is 168 and the proposed scheme performs better with a margin of 0.25 dB at the BER of 10^{-4}. As the rate increases, the performance gap between the two schemes grows steadily. At the rate 0.9, the number of punctured variable nodes is 448 and the gap reaches 0.8 dB at the BER of 10^{-4}.

In the second example, we implement a mother code defined in 802.16 e systems to compare the performance gain of the proposed scheme and the scheme of [9]. First, set $l_{max} = 8$ and $\eta_{ACE} = 4$ (referred to as η_{max} in [9]) for the two schemes. The required E_b/E_0 values for achieving BER of 10^{-4} at the code rate 0.8 and 0.9 are shown in Fig. 2. The maximum number of iterations range from 10 to 20. When the maximum number of iterations is set to 10, the performance gains of the proposed scheme are 1.2 dB and 1.8 dB over the scheme of [9] at code rate 0.8 and 0.9 respectively. These gains reduce to 0.5 dB and 0.7 dB when the maximum number of iterations increase to 20.

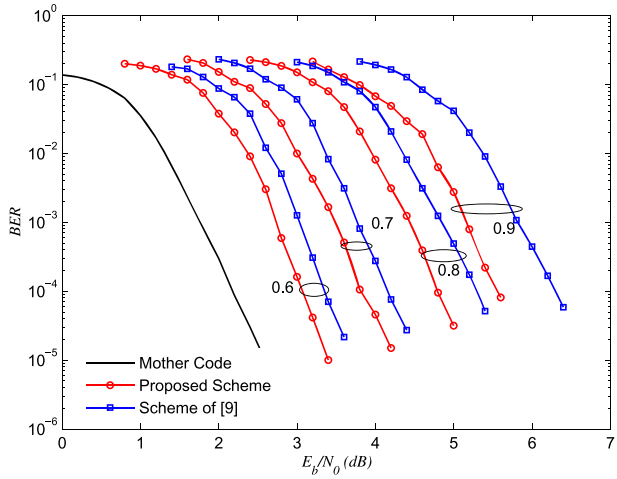

Fig. 1. BER performance of punctured LDPC codes with rate 0.6, 0.7, 0.8 and 0.9 derived from proposed puncturing scheme and the scheme of [9]; the mother LDPC codes is (1008, 504).

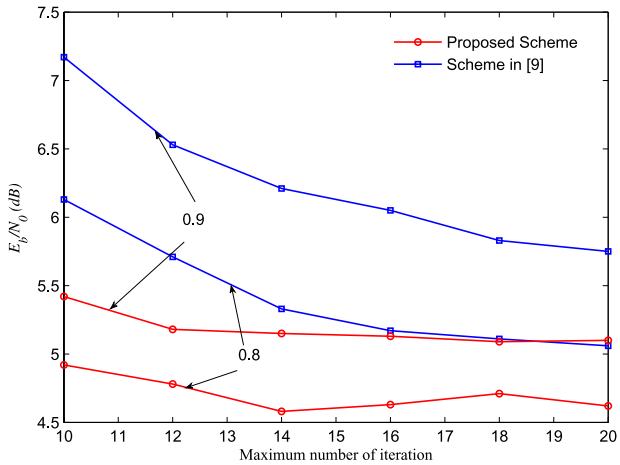

Fig. 2. The required E_b/N_0 values for BER of 10^{-4} with code rate 0.8 and 0.9 for given maximum number of iterations; the punctured LDPC codes derived from proposed scheme and the scheme of [9] with mother codes (1152, 576).

V. Conclusion

An efficient punctured scheme is proposed for irregular LDPC codes with serial schedule. The updating order of check nodes in the serial schedule is provided by the puncturing scheme, which accelerates the convergence rate of the decoding process and improves the performance of the punctured codes. Besides, the degree and the ACE score spectrum are used to minimize the performance loss of the punctured codes in decoding process. The proposed scheme performs very well for a wide range of code rates, especially in the high-rate scenarios.

References

[1] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes)," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389–400, Apr. 1988.

[2] D. Elkouss, J. Martinez-Mateo, and V. Martin, "Untainted puncturing for irregular low-density parity-check codes," *IEEE Wireless Commun.*, vol. 1, no. 6, pp. 585–588. Dec. 2012.

[3] J. Ha, J. Kim and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2824–2836, Nov. 2004.
[4] H. Pishro-Nik and F. Fekri, "Results on punctured low-density parity-check codes and improved iterative decoding techniques," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 599–613, Feb. 2007.
[5] M. R. Yazdani and A. H. Banihashemi, "On construction of rate-compatible low-density parity-check codes," *IEEE Commun. Lett.*, vol. 8, no. 3, pp. 159–161, Mar. 2004.
[6] 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. 52, no. 2, pp. 728–738, Feb. 2006.
[7] H. Y. Park, J. W. Kang, K. S. Kim, and K. C. Whang, "Efficient puncturing method for rate-compatible low-density parity-check codes," *IEEE Trans. Wireless Commun.*, vol. 6, no. 11, pp. 3914–3919. Nov. 2007.
[8] B. N. Vellambi and F. Fekri, "Finite-length rate-compatible LDPC codes: A novel puncturing scheme," *IEEE Trans. Commun.*, vol. 57, no. 2, pp. 297–302, Feb. 2009.
[9] R. Asvadi and A. H. Banihashemi, "A rate-compatible puncturing scheme for finite-length LDPC codes," *IEEE Commun. Lett.*, vol. 17, no. 1, pp. 147–150, Jan. 2013.
[10] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, "The design of efficiently-encodable rate-compatible LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 2, pp. 365–375, Feb. 2009.
[11] C. Shi and A. Ramamoorthy, "Design and analysis of E^2RC codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 889–898, Aug. 2009.
[12] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC codes construction," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1242–1247. Aug. 2004.
[13] E. Sharon, S. Litsyn, and J. Goldberger, "Efficient serial message-passing schedules for LDPC decoding," *IEEE Trans. Inf. Theory*, vol. 53, no. 11, pp. 4076–4091, Nov. 2007.
[14] X. Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graph," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.