Layered Decoding for Protograph-Based Low-Density Parity-Check Hadamard Codes

Peng W. Zhang, Francis C.M. Lau, *Fellow, IEEE,* and Chiu-W. Sham, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a layered decoding algorithm for protograph-based low-density parity-check Hadamard codes (PLDPC-HCs), which have been shown to be ultimate-Shannon-limit approaching. Compared with the standard decoding algorithm, the layered decoding algorithm improves the convergence rate by about two times. At a bit error rate of 2.0×10^{-5}, the layered decoder using 20 decoding iterations shows a very small degradation of 0.03 dB compared with the standard decoder using 40 decoding iterations. Moreover, the layered decoder using 21 decoding iterations shows the same error performance as the standard decoder using 41 decoding iterations.**

***Index Terms*—convergence rate, layered decoding, protograph-based LDPC-Hadamard codes, ultimate Shannon limit**

I. Introduction

Turbo codes [1] and low-density parity-check (LDPC) codes [2] are well-known error correction codes. Besides being used in wireless communication systems (3G/4G/5G, WiFi) and satellite communications, they have been used together with Hadamard codes to form ultimate-Shannon-limit-approaching codes, i.e., low bit error rate (BER) even approaching $E_b/N_0 = -1.59$ dB. In [3], a low-rate turbo-Hadamard code, in which the turbo concept is used in concatenation with Hadamard codes, has been proposed. In [4], a generalized LDPC code called LDPC-Hadamard code (LDPC-HC) has been proposed. It is formed by replacing the single-parity checks of an LDPC code with Hadamard codes accompanied by additional degree-1 Hadamard check bits.

To find a good LDPC-HC, an extrinsic information transfer (EXIT) chart method is used to optimize the degree distributions of the code in [5]. However, the method cannot analyze codes with degree-1 or punctured variable nodes (VNs). In [6], [7], [8], a protograph-based EXIT (PEXIT) method has been proposed to analyze and design protograph-based LDPC-Hadamard codes (PLDPC-HCs). The technique enables analyzing LDPC-HCs with degree-1 and/or punctured variable nodes. The rate-0.0494 PLDPC-HC in [6], [7] has shown a theoretical threshold of -1.42 dB. Moreover, simulation results show that with a maximum of 300 standard belief-propagation (BP) decoding iterations, the code can achieve a BER of 10^{-5} at $E_b/N_0 = -1.19$ dB.

The work described in this paper was partially supported by a grant from the RGC of the Hong Kong SAR, China (Project No. PolyU 152170/18E).

P.W. Zhang and F.C.M. Lau are with the Future Wireless Networks and IoT Focusing Area, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: pengwei.zhang@connect.polyu.hk and francis-cm.lau@polyu.edu.hk).

C.-W. Sham is with the Department of Computer Science, The University of Auckland, New Zealand (e-mail: b.sham@auckland.ac.nz).

It is well known that using layered BP decoding for LDPC codes can accelerate the convergence and to reduce the hardware requirements compared with using standard BP decoding [9]. In [10], an efficient check-node-update scheduling has been proposed for rate-compatible punctured LDPC codes, and is shown to outperform conventional scheduling and conventional BP decoding in terms of convergence speed. In [11], an efficient dynamic scheduling scheme has been proposed to speed up the convergence rate of LDPC decoders at medium to high signal-to-noise (SNR) region. In [12], a safe early termination strategy has been developed for layered LDPC decoding in order to help saving resources such as power and processing time. Yet layered decoding algorithms for generalized LDPC codes such as PLDPC-HCs are lacking.

In this paper, we propose a layered decoding strategy for PLDPC-HCs with an aim of improving the convergence rate. The organization of the paper is as follows. Section II briefly reviews the PLDPC-HC. Section III presents the standard decoding algorithm and our proposed layered decoding for PLDPC-HCs. We present the BER results for the standard and layered decoders in Section IV and make some conclusions in Section V.

II. Protograph-Based LDPC-Hadamard Codes

Hadamard codes are formed by the column (or equivalently row) vectors of a Hadamard matrix. An order-r Hadamard matrix can be recursively constructed by

$$\pm\boldsymbol{H}_q = \{\pm\boldsymbol{h}_j\} = \begin{bmatrix} \pm\boldsymbol{H}_{q/2} & \pm\boldsymbol{H}_{q/2} \\ \pm\boldsymbol{H}_{q/2} & \mp\boldsymbol{H}_{q/2} \end{bmatrix},$$

where $q = 2^r$ is the Hadamard code length; $\pm\boldsymbol{h}_j$ $(j = 0, 1, \ldots, q-1)$ denotes the j-th column of $\pm\boldsymbol{H}_q$; and $\pm\boldsymbol{H}_1 = [\pm 1]$. Fig. 1 shows the order-4 positive Hadamard matrix $+\boldsymbol{H}_{16}$. The corresponding negative matrix $-\boldsymbol{H}_{16}$ can be derived by multiplying $+\boldsymbol{H}_{16}$ with -1. Supposing $+1$ is mapped to bit "0" and -1 is mapped to bit "1", each column $\pm\boldsymbol{h}_j$ $(j = 0, 1, \ldots, q-1)$ corresponds to a Hadamard codeword. We further denote the i-th bit $(i = 0, 1, \ldots, q-1)$ of a Hadamard codeword by c_i^H and consider the case where r is even[1]. It has been shown that [5], [7]

$$\left[c_0^H \oplus c_1^H \oplus c_2^H \oplus \cdots \oplus\ c_{2^{k-1}}^H \oplus \cdots \oplus\ c_{2^{r-1}}^H\right] \oplus c_{2^r-1}^H = 0$$

where $\oplus$ represents the XOR operator. In other words, $(r+2)$ code bits at specific locations of an even-order Hadamard codeword always form a single-parity-check (SPC) code. In the example shown in Fig. 1 where $r = 4$, the bits at the

[1]Detailed discussion can be found in [7] when r is odd

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2021.3057717, IEEE Communications Letters

$$+\boldsymbol{H}_{16} = \begin{array}{c} \begin{matrix} +\boldsymbol{h}_0 & +\boldsymbol{h}_1 & +\boldsymbol{h}_2 & +\boldsymbol{h}_3 & +\boldsymbol{h}_4 & +\boldsymbol{h}_5 & +\boldsymbol{h}_6 & +\boldsymbol{h}_7 & +\boldsymbol{h}_8 & +\boldsymbol{h}_9 & +\boldsymbol{h}_{10} & +\boldsymbol{h}_{11} & +\boldsymbol{h}_{12} & +\boldsymbol{h}_{13} & +\boldsymbol{h}_{14} & +\boldsymbol{h}_{15} \end{matrix} \\ \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \\ +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 \end{bmatrix} \begin{matrix} +\boldsymbol{h}_0^{\mathrm{T}} \\ +\boldsymbol{h}_1^{\mathrm{T}} \\ +\boldsymbol{h}_2^{\mathrm{T}} \\ +\boldsymbol{h}_3^{\mathrm{T}} \\ +\boldsymbol{h}_4^{\mathrm{T}} \\ +\boldsymbol{h}_5^{\mathrm{T}} \\ +\boldsymbol{h}_6^{\mathrm{T}} \\ +\boldsymbol{h}_7^{\mathrm{T}} \\ +\boldsymbol{h}_8^{\mathrm{T}} \\ +\boldsymbol{h}_9^{\mathrm{T}} \\ +\boldsymbol{h}_{10}^{\mathrm{T}} \\ +\boldsymbol{h}_{11}^{\mathrm{T}} \\ +\boldsymbol{h}_{12}^{\mathrm{T}} \\ +\boldsymbol{h}_{13}^{\mathrm{T}} \\ +\boldsymbol{h}_{14}^{\mathrm{T}} \\ +\boldsymbol{h}_{15}^{\mathrm{T}} \end{matrix} \end{array}$$

Fig. 1. The order $r = 4$ positive Hadamard matrix $+\boldsymbol{H}_{16}$. $+\boldsymbol{h}_j^T$ is the transpose of $+\boldsymbol{h}_j$.

0th, 1st, 2nd, 4th, 8th and 15th positions of a Hadamard code always satisfies the SPC constraint. Thus, when $r + 2$ bits that satisfy the SPC constraint are used as inputs to encode a Hadamard code, a Hadamard code with order-r can be used together with systematic encoding. Moreover, $2^r - r - 2$ Hadamard parity bits are generated.

The protograph of a PLDPC-HC consists of a set of protograph variable nodes (PVNs) and a set of Hadamard check nodes (HCNs) with (possibly parallel) edges connected between these two sets of nodes. We denote the number of PVNs as n and the number of HCNs as m. The protograph can be denoted by a base matrix $\boldsymbol{B}_{m\times n}$ of size $m \times n$ (see Fig. 2). Each PVN corresponds to one column of $\boldsymbol{B}_{m\times n}$ and each HCN corresponds to one row of $\boldsymbol{B}_{m\times n}$. In particular, the (i, j)-th entry of the base matrix is allowed to be greater than 1 and represents the number of edges between the i-th PVN and j-th HCN in the protograph. The connections corresponding to each PVN form a repeat code whereas those corresponding to each HCN form a SPC code. Each SPC corresponding to a HCN is further encoded into an Hadamard code such that the protograph of a PLDPC-HC can be obtained [6], [7]. Moreover, the Hadamard parity bits derived are denoted as degree-1 Hadamard variable nodes (D1H-VN) and are appended to the HCN in the protograph shown in Fig. 2.

Assuming that each HCN is connected to d PVNs, the order of the Hadamard code would be $r = d-2$. We further assume that r is an even number, and hence each HCN connects $2^r - r - 2$ D1H-VNs (when r is odd, the number of D1H-VNs connected to each HCN is increased to $2^r - 2$ [5], [7]). Fig. 2 shows the base matrix (of size 7×11) and the corresponding protograph for an LDPC-HC. In addition to fulfilling the SPC constraint, all edges connected to each HCN (i.e., all edges from PVNs and D1H-VNs) need to satisfy a Hadamard constraint. Note that all rows in the base matrix in Fig. 2 have a weight of $d = 6$. Thus the order of the Hadamard code equals $r = d - 2 = 4$ and hence all HCNs have an additional $2^r - r - 2 = 10$ D1H-VNs attached. Fig. 3 shows an example in which a $(6, 5)$ SPC codeword is encoded into a length-16 (order $r = 4$) Hadamard codeword.

Once a protograph is obtained, we use the copy-and-permute operation to lift $\boldsymbol{B}_{m\times n}$ twice with factors z_1 and z_2, respectively. During the first lifting process, the (i, j)-th entry of $\boldsymbol{B}_{m\times n}$, denoted by $\boldsymbol{B}_{m\times n}(i, j)$, is replaced by the sum of $\boldsymbol{B}_{m\times n}(i, j)$ different $z_1 \times z_1$ permutation matrices when $\boldsymbol{B}_{m\times n}(i, j) \geq 1$; otherwise it is replaced by a $z_1 \times z_1$ zero matrix. The lifted matrix, denoted by $\boldsymbol{H}_1$, has a size of $mz_1 \times nz_1$. The second lifting further replaces each entry "1" in $\boldsymbol{H}_1$ with a $z_2 \times z_2$ circulant permutation matrix (CPM), and each entry "0" with a $z_2 \times z_2$ zero matrix. The overall quasi-cyclic matrix, denoted by $\boldsymbol{H}_{M\times N}$, has a size of $M \times N$ where $N = nz_1z_2$ and $M = mz_1z_2$. It has the same degree distribution properties as the base matrix [2] and represents the connections between the PVNs and HCNs in a PLDPC-HC. The PLDPC-HC so formed has N PVNs and $M(2^r - r - 2)$ D1H-VNs. The bits corresponding to PVNs and D1H-VNs will be transmitted through a channel. Thus, the code length l equals the summation of the numbers of PVNs and D1H-VNs, that is, $l = N + M(2^r - r - 2)$. Moreover, the code rate of PLDPC-HC equals $R = \frac{N-M}{l} = \frac{n-m}{n+m(2^r-r-2)}$ when r is even.

III. Decoding Algorithms

The receiver obtains the channel log-likelihood-ratio (LLR) values of the PVNs and D1H-VNs, based on which the transmitted PLDPC-HC is decoded. We denote

- $L_{ch}^{\mathrm{PVN}}(\beta)$ as the channel LLR value of the β-th PVN ($\beta = 1, 2, \ldots, N$);
- $\boldsymbol{L}_{ch}^{\mathrm{D1H}(\alpha)}$ as a vector consisting of the channel LLR values of the D1H-VNs connected to the α-th HCN ($\alpha = 1, 2, \ldots, M$);
- $L_{app}^{\mathrm{PVN}}(\beta)$ as the *a posteriori* (APP) LLR value of the β-th PVN ($\beta = 1, 2, \ldots, N$);
- $L_{ex}^{\mathrm{PVN}}(\alpha, \beta)$ as the extrinsic LLR sent from the β-th PVN to the α-th HCN ($\alpha = 1, 2, \ldots, M$; $\beta = 1, 2, \ldots, N$);
- $L_{app}^{\mathrm{H}}(\alpha, \beta)$ as the APP LLR computed by the α-th HCN for the β-th PVN ($\alpha = 1, 2, \ldots, M$; $\beta = 1, 2, \ldots, N$);
- $L_{ex}^{\mathrm{H}}(\alpha, \beta)$ as the extrinsic LLR sent from the α-th HCN to the β-th PVN ($\alpha = 1, 2, \ldots, M$; $\beta = 1, 2, \ldots, N$).

We also denote

- $\mathcal{P}(\alpha)$ as the set of PVNs connected to the α-th HCN;
- $\mathcal{H}(\beta)$ as the set of HCNs connected to the β-th PVN.

The standard PLDPC-Hadamard decoder [6], [7] consists of two component decoders, i.e., repeat decoder and Hadamard decoder, which are shown in Fig. 4 and Fig. 5, respectively. The repeat decoder is the same as the variable-node processor used in an LDPC decoder. The check node processor used in an LDPC decoder is replaced by a symbol-by-symbol maximum a posteriori probability (MAP) Hadamard decoder. Referring to Fig. 5, the symbol-MAP Hadamard decoder receives $d = r + 2$ inputs from the repeat decoders and $2^r - r - 2$ inputs from the D1H-VNs, and computes d outputs and feeds them back to the repeat decoders.

[2]Our lifting method keeps the code rate unchanged while other lifting methods may result in a slightly larger code rate [13].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2021.3057717, IEEE Communications Letters

$$\boldsymbol{B}_{7\times 11} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 & 1 \\ 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 2 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 \\ 3 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 2 & 0 \end{bmatrix}$$

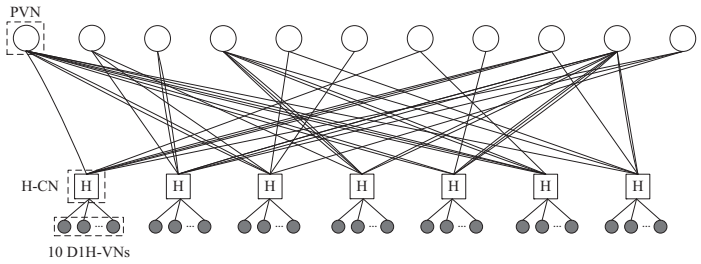

Fig. 2. The base matrix and corresponding protograph of a PLDPC-Hadamard code. A circle denotes a protograph variable node (PVN), a square with "H" denotes a Hadamard check node (HCN), and a filled circle denotes a degree-1 Hadamard variable node (D1H-VN). Row weight $d = 6$, $r = d - 2 = 4$ and $2^r - r - 2 = 10$ D1H-VNs are attached to each HCN. Code rate $R = 0.0494$.

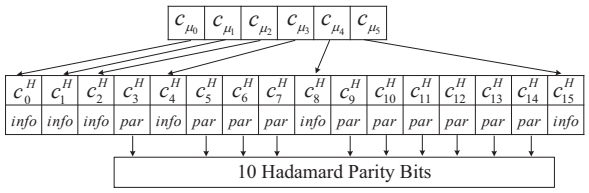

Fig. 3. Example of encoding a length-6 SPC codeword $[c_{\mu_0}\ c_{\mu_1}\ \cdots\ c_{\mu_5}]$ into a length-16 (order $r = 4$) Hadamard codeword $[c_0^H\ c_1^H\ \cdots\ c_{15}^H]$, where $c_0^H = c_{\mu_0}$, $c_1^H = c_{\mu_1}$, $c_2^H = c_{\mu_2}$, $c_4^H = c_{\mu_3}$, $c_8^H = c_{\mu_4}$, $c_{15}^H = c_{\mu_5}$ and 10 remaining coded bits are Hadamard parity bits.

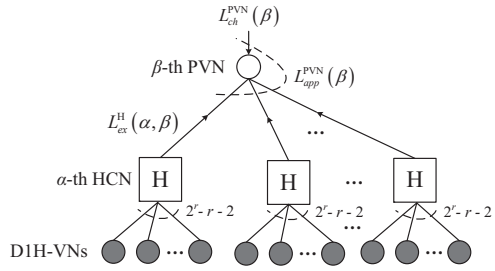

Fig. 4. A repeat decoder for PVN message processing.

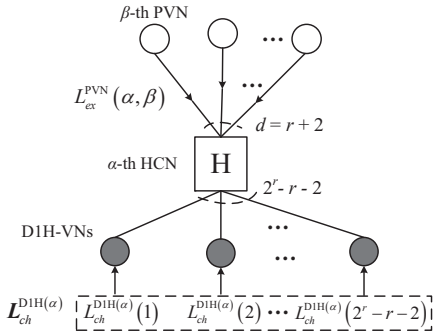

Fig. 5. A symbol-MAP Hadamard decoder for HCN message processing.

A. Standard Decoding Algorithm

The standard decoding method is described as follows.

1) Initialization: Set $L_{ex}^{\mathrm{PVN}}(\alpha,\beta) = L_{ch}^{\mathrm{PVN}}(\beta)$, $\forall\alpha = 1,2,\ldots,M$ and $\beta = 1,2,\ldots,N$.
2) Symbol-MAP Hadamard decoder: For the α-th HCN ($\alpha = 1,2,\ldots,M$), compute the following.
   a) Compute $L_{app}^{\mathrm{H}}(\alpha,\beta)$ for the β-th PVN ($\beta \in \mathcal{P}(\alpha)$) using

$$\begin{aligned}\boldsymbol{L}_{app}^{\mathrm{H}}(\alpha) &= \{L_{app}^{\mathrm{H}}(\alpha,\beta) : \beta \in \mathcal{P}(\alpha)\} \\ &= \mathcal{T}\left[\{L_{ex}^{\mathrm{PVN}}(\alpha,\beta) : \beta \in \mathcal{P}(\alpha)\}, \boldsymbol{L}_{ch}^{\mathrm{D1H}(\alpha)}\right] \quad (1)\end{aligned}$$

   where $\mathcal{T}$ is a transformation involving the fast Hadamard transform (FHT) and the dual FHT (DFHT) operations [5], [7]. Fig. 6 illustrates the arrangement of the $2^r = 16$ inputs when they are fed to the FHT block in a symbol-MAP Hadamard decoder for the case $r = 4$. (The $r + 2$ extrinsic LLR values from PVNs are assigned to the 1st, 2nd,...,$(2^{k-1}+1)$-th, ..., $(2^{r-1}+1)$-th and 2^r-th positions; while the channel LLR values of the D1H-VNs are assigned to the remaining $2^r - r - 2$ positions [7].)
   b) Compute $L_{ex}^{\mathrm{H}}(\alpha,\beta)$ by subtracting $L_{ex}^{\mathrm{PVN}}(\alpha,\beta)$ from $L_{app}^{\mathrm{H}}(\alpha,\beta)$, i.e.,

$$L_{ex}^{\mathrm{H}}(\alpha,\beta) = L_{app}^{\mathrm{H}}(\alpha,\beta) - L_{ex}^{\mathrm{PVN}}(\alpha,\beta);\ \ \forall\beta \in \mathcal{P}(\alpha). \quad (2)$$

3) Repeat decoder: For the β-th PVN ($\beta = 1,2,\ldots,N$), compute the following.
   a) Compute $L_{app}^{\mathrm{PVN}}(\beta)$ for the β-th PVN using

$$L_{app}^{\mathrm{PVN}}(\beta) = \sum_{\alpha\in\mathcal{H}(\beta)} L_{ex}^{\mathrm{H}}(\alpha,\beta) + L_{ch}^{\mathrm{PVN}}(\beta). \quad (3)$$

   b) Compute $L_{ex}^{\mathrm{PVN}}(\alpha,\beta)$ by subtracting $L_{ex}^{\mathrm{H}}(\alpha,\beta)$ from $L_{app}^{\mathrm{PVN}}(\beta)$, i.e.,

$$L_{ex}^{\mathrm{PVN}}(\alpha,\beta) = L_{app}^{\mathrm{PVN}}(\beta) - L_{ex}^{\mathrm{H}}(\alpha,\beta);\ \ \forall\alpha \in \mathcal{H}(\beta). \quad (4)$$

4) Decoding: Repeat Step 2) and Step 3) I times and make decisions based on the sign of $L_{app}^{\mathrm{PVN}}(\beta)$ ($\beta = 1,2,\ldots,N$).

B. Layered Decoding Algorithm

Recall in Section II that the base matrix $\boldsymbol{B}_{m\times n}$ is lifted twice using factors z_1 and z_2, respectively, to form $\boldsymbol{H}_{M\times N}$ which has a size of $M \times N$ ($= mz_1z_2 \times nz_1z_2$). Here we divide $\boldsymbol{H}_{M\times N}$ into mz_1 layers, where each layer is composed of $1 \times nz_1$ CPMs each of size $z_2 \times z_2$ (or equivalently a block-row of size $z_2 \times nz_1z_2$). In other words, each layer consists of z_2 HCNs, each connected to d independent PVNs. ($d = r+2$ is the row weight of $\boldsymbol{B}_{m\times n}$ and also that of $\boldsymbol{H}_{M\times N}$.)

To improve the convergence rate of the decoder, we propose a layered decoding algorithm. We use the same symbols as above. Moreover, we define k as the layer number ($k =$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2021.3057717, IEEE Communications Letters

1	2	3	4	5	6 : 8	9	10 : 15	16
$L_{ex}^{\mathrm{PVN}}(\alpha,\beta_1)$	$L_{ex}^{\mathrm{PVN}}(\alpha,\beta_2)$	$L_{ex}^{\mathrm{PVN}}(\alpha,\beta_3)$	$L_{ch}^{\mathrm{D1H}(\alpha)}(1)$	$L_{ex}^{\mathrm{PVN}}(\alpha,\beta_4)$	$\boldsymbol{L}_{ch}^{\mathrm{D1H}(\alpha)}(2:4)$	$L_{ex}^{\mathrm{PVN}}(\alpha,\beta_5)$	$\boldsymbol{L}_{ch}^{\mathrm{D1H}(\alpha)}(5:10)$	$L_{ex}^{\mathrm{PVN}}(\alpha,\beta_6)$

Fig. 6. Arrangement of the $2^r = 16$ inputs when they are fed to the FHT block in a symbol-MAP Hadamard decoder for the case $r = 4$. $\{\beta_1,\beta_2,\beta_3,\beta_4,\beta_5,\beta_6\} = \mathcal{P}(\alpha)$.

$1,2,\ldots,mz_1$) and $\mathcal{L}(k)$ as the set of HCNs in layer k. Our layered decoding algorithm is described as follows.

1) Initialization: Set $L_{app}^{\mathrm{PVN}}(\beta) = L_{ch}^{\mathrm{PVN}}(\beta)$, $\forall\beta = 1,2,\ldots,N$; and set $L_{ex}^{H}(\alpha,\beta) = 0$ $\forall\alpha = 1,2,\ldots,M$ and $\beta = 1,2,\ldots,N$.
2) Symbol-MAP Hadamard layered decoder: Set $k = 1$.
   a) For the α-th HCN in layer k ($\alpha \in \mathcal{L}(k)$), compute the following.
      i) For $\beta \in \mathcal{P}(\alpha)$, compute $L_{ex}^{\mathrm{PVN}}(\alpha,\beta)$ by subtracting $L_{ex}^{\mathrm{H}}(\alpha,\beta)$ from $L_{app}^{\mathrm{PVN}}(\beta)$, i.e.,

$$L_{ex}^{\mathrm{PVN}}(\alpha,\beta) = L_{app}^{\mathrm{PVN}}(\beta) - L_{ex}^{\mathrm{H}}(\alpha,\beta);\ \ \forall\beta\in\mathcal{P}(\alpha). \quad (5)$$

      ii) Compute $L_{app}^{\mathrm{H}}(\alpha,\beta)$ for the β-th PVN ($\beta \in \mathcal{P}(\alpha)$) using

$$\begin{aligned}\boldsymbol{L}_{app}^{\mathrm{H}}(\alpha) &= \{L_{app}^{\mathrm{H}}(\alpha,\beta) : \beta\in\mathcal{P}(\alpha)\}\\ &= \mathcal{T}\left[\{L_{ex}^{\mathrm{PVN}}(\alpha,\beta) : \beta\in\mathcal{P}(\alpha)\}, \boldsymbol{L}_{ch}^{\mathrm{D1H}(\alpha)}\right]. \quad (6)\end{aligned}$$

      iii) Update $L_{ex}^{\mathrm{H}}(\alpha,\beta)$ and $L_{app}^{\mathrm{PVN}}(\beta)$ using

$$L_{ex}^{\mathrm{H}}(\alpha,\beta) = L_{app}^{\mathrm{H}}(\alpha,\beta) - L_{ex}^{\mathrm{PVN}}(\alpha,\beta);\ \ \forall\beta\in\mathcal{P}(\alpha) \quad (7)$$

$$L_{app}^{\mathrm{PVN}}(\beta) = L_{app}^{\mathrm{H}}(\alpha,\beta);\ \ \forall\beta\in\mathcal{P}(\alpha). \quad (8)$$

   b) If k is smaller than the number of layers, i.e., $k < mz_1$, increment k by 1 and goto Step 2a).
3) Repeat Step 2) I times and make decisions based on the sign of $L_{app}^{\mathrm{PVN}}(\beta)$ ($\beta = 1,2,\ldots,N$).

Note that (8) is derived as follows. We consider the associated PVNs in layer k. Note that each of the associated PVNs is connected to one and only one HCN in layer k. We suppose the β-th PVN is connected to the α-th HCN in layer k. After this layer is processed, the updated APP for the β-th PVN is given by

$$\begin{aligned}&L_{app}^{\mathrm{PVN}}(\beta) = \textstyle\sum_{\alpha\in\mathcal{H}(\beta)} L_{ex}^{\mathrm{H}}(\alpha,\beta) + L_{ch}^{\mathrm{PVN}}(\beta)\\ &= L_{ex}^{\mathrm{H}}(\alpha,\beta) + \sum_{\substack{\alpha'\in\mathcal{H}(\beta)\\ \alpha'\notin\mathcal{L}(k)}} L_{ex}^{\mathrm{H}}(\alpha',\beta) + L_{ch}^{\mathrm{PVN}}(\beta)\\ &= L_{ex}^{\mathrm{H}}(\alpha,\beta) + L_{ex}^{\mathrm{PVN}}(\alpha,\beta) = L_{app}^{\mathrm{H}}(\alpha,\beta).\end{aligned} \quad (9)$$

C. Complexity Analysis

In this section, we compare the complexity of the proposed layered decoding algorithm and the standard decoding algorithm in terms of memory requirement and computational logic.

1) Memory requirement: Considering the layered decoding algorithm in Section III-B, memory storage (i.e., RAM) for the following sets of LLRs is required — $\{L_{ch}^{\mathrm{PVN}}(\beta)\}$, $\{L_{app}^{\mathrm{PVN}}(\beta)\}$, $\{L_{ex}^{\mathrm{H}}(\alpha,\beta)\}$ and $\{\boldsymbol{L}_{ch}^{\mathrm{D1H}(\alpha)}\}$. Moreover, $\{L_{app}^{\mathrm{H}}(\alpha,\beta)\}$ and $\{L_{ex}^{\mathrm{PVN}}(\alpha,\beta)\}$ are only intermediate variables generated during the computation process and thus need no storage. Note also that $\{L_{ch}^{\mathrm{PVN}}(\beta)\}$ is only required during the initialization process but not in the iterative process. Thus, it can be immediately released for storing the LLRs for the next codeword.

For the standard decoding algorithm in Section III-A, besides $\{L_{ch}^{\mathrm{PVN}}(\beta)\}$, $\{L_{app}^{\mathrm{PVN}}(\beta)\}$, $\{L_{ex}^{\mathrm{H}}(\alpha,\beta)\}$ and $\{\boldsymbol{L}_{ch}^{\mathrm{D1H}(\alpha)}\}$, $\{L_{ex}^{\mathrm{PVN}}(\alpha,\beta)\}$ needs to be stored after the computation in (4). On the other hand, we can observe that $L_{ex}^{\mathrm{H}}(\alpha,\beta)$, after being used to update $L_{ex}^{\mathrm{PVN}}(\alpha,\beta)$ in (4), is no longer needed. The memory location used to store $L_{ex}^{\mathrm{H}}(\alpha,\beta)$ can therefore be used to store $L_{ex}^{\mathrm{PVN}}(\alpha,\beta)$. Similarly, $L_{ex}^{\mathrm{PVN}}(\alpha,\beta)$ is no longer needed after computing (2), and its memory location can be used to store $L_{ex}^{\mathrm{H}}(\alpha,\beta)$ afterwards. In other words, $\{L_{ex}^{\mathrm{H}}(\alpha,\beta)\}$ and $\{L_{ex}^{\mathrm{PVN}}(\alpha,\beta)\}$ can share the same set of memory locations. But unlike in the layered decoding algorithm, $\{L_{ch}^{\mathrm{PVN}}(\beta)\}$ in the standard decoding algorithm is required throughout the iterative process (in (3)). Thus another set of memory is required to store $\{L_{ch}^{\mathrm{PVN}}(\beta)\}$ for the next codeword. Note that the number of $L_{ch}^{\mathrm{PVN}}(\beta)$ is equal to number of PVNs, i.e., $N = nz_1z_2$. For the $r = 4$ PLDPC-HC optimized in [6], N equals $11\times32\times512 = 180224$, which implies quite a large memory.

2) Computational logic: Both the layered decoding algorithm and the standard decoding algorithm involve FHT, DFHT and simple additions/subtractions. Moreover, the summation term in (3) of the standard decoding algorithm requires the addition of all $L_{ex}^{\mathrm{H}}(\alpha,\beta)$ terms corresponding to the same PVN. The number of terms equals the column weight and varies from column to column. In the example given in Fig. 2, the column weight ranges from 1 to 9. When 9 values are to be added together, more combinational logics (especially many PVNs are processed in parallel) are required and a slightly larger latency is needed.

In summary, the layered decoding algorithm requires less memory storage and computational logic compared with the standard decoding algorithm.

IV. Simulation Results

We simulate the $r = 4$ and $R = 0.0494$ PLDPC-HC optimized in [6] (whose base matrix and protograph are shown in Fig. 2). We transmit all-zero codewords using binary-phase-shift-keying modulation over an additive white Gaussian noise channel. To compare with the BER performance of the standard decoder in [6], we use the same lifting factors, i.e., $z_1 = 32$ and $z_2 = 512$, and the same code length, i.e., $l = 1,327,104$ (See Appendix C in [7] for details of the code structure after the lifting process). Fig. 7 plots the bit error rate (BER) results of the standard and layered decoders. We denote the maximum number of decoding iterations used by the layered decoder as I. When $I = 30,40,50,60,75,150$, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCOMM.2021.3057717, IEEE Communications Letters

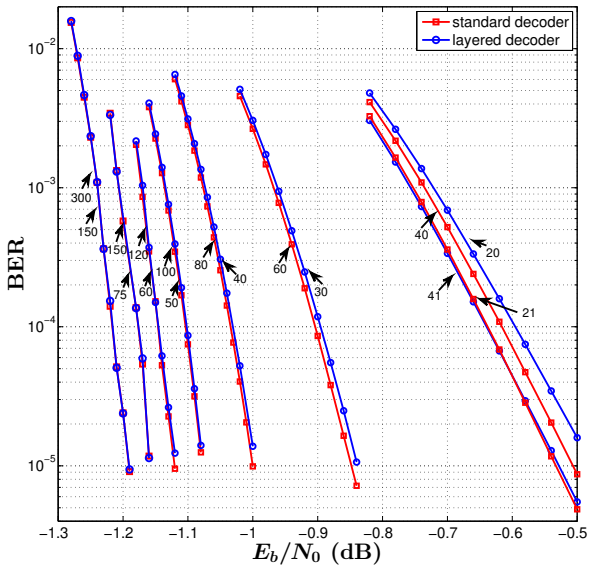

Fig. 7. Comparison of BER performance of the standard PLDPC-Hadamard decoder and the layered PLDPC-Hadamard decoder. The maximum number of decoding iterations ranges from 40 to 300 for the standard decoder; and ranges from 20 to 150 for the layered decoder. $r = 4$ and $R = 0.0494$.

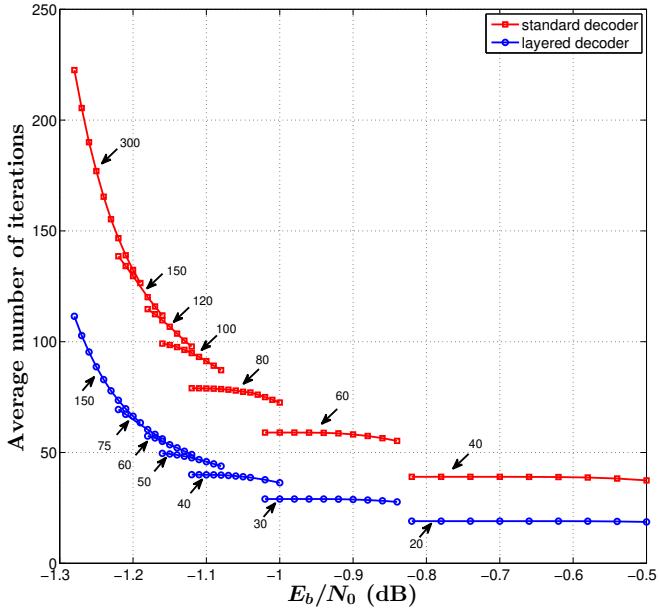

Fig. 8. Average number of iterations required for a standard PLDPC-Hadamard decoder and a layered PLDPC-Hadamard decoder to decode a codeword. The maximum numbers of iterations allowed are given next to the curves.

layered decoding algorithm using I iterations has almost the same error rate as the standard decoder using $2I$ iterations. When $I = 20$, there is a 0.03 dB difference between the layered decoding algorithm using $I = 20$ iterations and the standard decoder using $2I = 40$ iterations at a bit error rate of 2.0×10^{-5}. Note that in most scenarios, a 0.03 dB difference is considered as insignificant, but has been shown in our figure to be a relatively large gap due to the scale being used. We further find that when $I = 21$, the layered decoding algorithm outperforms the standard decoder using 40 iterations and has the same performance of the standard decoder using 41 iterations. Thus we can conclude that compared with the standard decoding algorithm, the layered decoding algorithm improves the convergence rate by about two times. Fig. 8 plots the corresponding average number of iterations required to decode a codeword. At a given E_b/N_0, the average number of iterations required by the layered decoder is about half of that required by the standard decoder.

V. Conclusion

This paper has proposed a layered decoding algorithm for the ultimate-Shannon-limit-approaching PLDPC-Hadamard code. Simulation results have verified that the layered decoding method can speed up the PLDPC-Hadamard decoder by about two times compared with the standard decoder. Though an even-order PLDPC-HC is illustrated, the proposed algorithm can be readily applied to odd-order PLDPC-HCs and other generalized LDPC codes by making appropriate modifications. The algorithm is also suitable for hardware decoder design. When multiple symbol-MAP Hadamard decoders are used, multiple HCNs in the same layer can be processed simultaneously so as to increase the overall throughput.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," *in Proc. IEEE ICC*, vol. 2, pp. 1064–1070, May 1993.

[2] R. G. Gallager, *Low-Density Parity-Check Codes*. PhD thesis, Cambridge, MA, USA, 1963.

[3] P. Li, W. K. Leung, and K. Y. Wu, "Low-Rate Turbo-Hadamard Codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 12, pp. 3213–3224, Dec. 2003.

[4] G. Yue, L. Ping, and X. Wang, "Low-rate generalized low-density parity-check codes with Hadamard constraints," in *Proc. IEEE ISIT*, pp. 1377–1381, 2005.

[5] G. Yue, L. Ping, and X. Wang, "Generalized low-density parity-check codes based on Hadamard constraints," *IEEE Trans. Inf. Theory*, vol. 53, no. 3, pp. 1058–1079, 2007.

[6] P. W. Zhang, F. C. M. Lau, and C.-W. Sham, "Protograph-based LDPC-Hadamard Codes," in *2020 IEEE WCNC*, pp. 1–6, 2020.

[7] P. W. Zhang, F. C. M. Lau, and C.-W. Sham, "Protograph-Based Low-Density Parity-Check Hadamard Codes," in *arXiv:2010.08285*, 2020, available at https://arxiv.org/abs/2010.08285.

[8] P. W. Zhang, F. C. M. Lau, and C.-W. Sham, "Protograph-Based Low-Density Parity-Check Hadamard Codes," *submitted to IEEE Trans. on Communications*.

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

[10] J. Ha, D. Klinc, J. Kwon, and S. W. Mclaughlin, "Layered BP decoding for rate-compatible punctured LDPC codes," *IEEE Commun. Lett.*, vol. 11, no. 5, pp. 440–442, 2007.

[11] G. Han and X. Liu, "An efficient dynamic schedule for layered belief-propagation decoding of LDPC codes," *IEEE Commun. Lett.*, vol. 13, no. 12, pp. 950–952, 2009.

[12] M. Ferrari, S. Bellini, and A. Tomasoni, "Safe early stopping for layered LDPC decoding," *IEEE Commun. Lett.*, vol. 19, no. 3, pp. 315–318, 2015.

[13] M. Gholami, M. Esmaeili, and M. Samadieh, "Quasi-cyclic low-density parity-check codes based on finite set systems". *IET commun.*, vol. 8, no. 10, pp.1837-1849, 2014.