Improved Low-Density Parity Check Accumulate (LDPCA) Codes

Chao Yu and Gaurav Sharma, *Fellow, IEEE*

***Abstract*—We present improved constructions for Low-Density Parity-Check Accumulate (LDPCA) codes, which are rate-adaptive codes commonly used for distributed source coding (DSC) applications. Our proposed constructions mirror the traditional LDPCA approach; higher rate codes are obtained by splitting the check nodes in the decoding graph of lower rate codes, beginning with a lowest rate *mother code*. In a departure from the uniform splitting strategy adopted by prior LDPCA codes, however, the proposed constructions introduce non-uniform splitting of the check nodes at higher rates. Codes are designed by a global minimization of the average rate gap between the code operating rates and the corresponding theoretical lower bounds evaluated by density-evolution. In the process of formulating the design framework, the paper also contributes a formal definition of LDPCA codes.**

Performance improvements provided by the proposed non-uniform splitting strategy over the conventional uniform splitting approach used in prior work are substantiated via density evolution based analysis and DSC codec simulations. Optimized designs for our proposed constructions yield codes with a lower average rate gap than conventional designs and alleviate the trade-off between the performance at different rates inherent in conventional designs. A software implementation is provided for the codec developed.

***Index Terms*—LDPCA codes, LDPC codes, distributed source coding, code design.**

I. Introduction

EMERGING applications of battery-powered mobile devices and sensor networks have motivated the development of DSC techniques that exploit inter-dependency between sensor data at different nodes to reduce communication requirements, thereby improving energy-efficiency and operating times [1]–[4].

Although information theoretic results for DSC appeared nearly 40 years ago [5], [6], practical code constructions that achieve close to promised performance have only been developed in the past decade. Most constructions, and the discussion in this paper, restrict attention to the binary-input memoryless *side-informed* coding scenario: a block $\mathbf{x} = [x_1, x_2, \ldots x_L]^T$ of L independent bits available at one terminal, the *encoder*, needs to be communicated to a second terminal, the *decoder*, that has *a priori* information consisting of a corresponding block of side information $\mathbf{y} = [y_1, y_2, \ldots y_L]^T$, where[1] the elements of $\mathbf{Y}$ are independent and for each $1 \leq i \leq L$, Y_i is (potentially) correlated with X_i and independent of $\mathbf{X}_{\sim i} \stackrel{\text{def}}{=} [X_1, X_2, \ldots, X_{i-1}, X_{i+1}, X_{i+2}, \ldots X_L]^T$. The encoder generates a vector of j bits $\tilde{\mathbf{p}} = [p_1, p_2, \ldots p_j]^T$, which is (noiselessly) sent to the decoder and using which the decoder must recover $\mathbf{x}$. The objective is to minimize the *rate* $r = (j/L)$ required per encoded bit by exploiting, in the decoding process, the side information $\mathbf{y}$ that is available at the decoder but not at the encoder. Practical side-informed coding methods leverage channel coding techniques: $\mathbf{y}$ is interpreted as the noisy output of a *virtual channel* with input $\mathbf{x}$ and error correction decoding is used to recover $\mathbf{x}$ at the decoder. DSC constructions have been developed based on trellis codes [7], Turbo codes [8], and Low-Density Parity-Check (LDPC) codes [9]. Information theoretic results for side-informed coding imply that the conditional entropy per symbol $H(\mathbf{X}|\mathbf{Y})/L$ is the minimum required rate (on average). In the memoryless setting where the pairs of random variables $\{(X_i, Y_i)\}_{i=1}^{L}$ are drawn independently from the same joint distribution $p_{XY}(x, y)$, the minimum required rate becomes the conditional entropy $H(X|Y)$.

To simplify practical implementations and handle the varying correlation (between X and Y) encountered in DSC applications, *rate-adaptive* DSC techniques have been developed based on punctured Turbo codes [8], [10], [11], or using an LDPC-Accumulate (LDPCA) construction [12] that builds on LDPC codes. LDPCA codes, in particular, offer superior performance for side-informed source coding and have been adopted for several DSC applications such as distributed video coding [13]–[16] and image authentication [17].

Previously reported LDPCA codes follow the framework introduced in [12], [18]. Rate adaptivity is obtained by beginning with an LDPC code at the lowest rate from which higher rate codes are obtained by uniformly splitting a fraction of the check nodes in the decoding graph to define additional check nodes, which then define the additional bits to be communicated from the encoder to the decoder for the higher operating rate. Within this framework, optimized designs were developed in [19]. An examination of the performance of these previously reported codes (See results in Section IV), reveals

Manuscript received November 20, 2012; revised April 7, June 16, and August 1, 2013. The editor coordinating the review of this paper and approving it for publication was D. Declercq.

This work was supported in part by the National Science Foundation under grant number ECS-0428157.

C. Yu is with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627-0126 USA (e-mail: chyu@ece.rochester.edu).

G. Sharma is with the Department of Electrical and Computer Engineering, the Department of Biostatistics and Computational Biology, and the Department of Oncology, University of Rochester, NY 14627-0126 USA (e-mail: gaurav.sharma@rochester.edu).

Digital Object Identifier 10.1109/TCOMM.2013.13.120892

[1]We adopt the standard notational convention where upper case letters represent the random variables corresponding to their lower case counterparts, both being bold when these are vectors. Elements of a vector are represented by corresponding non-bold subscripted variables. The notation $\mathbf{H}$ for denoting parity check matrices (with various superscripts) is the exception to the notational convention.

a performance trade-off between different rate regions. The optimized designs in [19] exhibit good performance in the low through mid rate regions but perform relatively poorly at high rates. Other codes presented in [12] exhibit either similar performance, or, if they do not exhibit the poor performance at high rates, offer performance that is worse than the optimized designs over the mid and low rate regions.

In this paper, we revisit the LDPCA code construction and propose a method for alleviating the performance trade-off of LDPCA codes in different rate regions. *Specific contributions of the work include: a) introduction of a non-uniform partitioning in the process used to generate higher rate codes by splitting a fraction of the check nodes in the decoding graph of the lower rate code, b) extension of density evolution based performance analysis to the proposed construction methodology, c) introduction of a principled method for LDPCA code design based on optimization of the average rate gap across all operating rates, d) clear demonstration, through both density evolution based analysis and actual code simulations, of the trade-off between performance at different rates for codes developed with the prior uniform splitting approach and of the improvement offered by the proposed non-uniform splitting methodology. In addition, the paper also provides a formal definition of LDPCA codes and a codec implementation based on the optimized designs, which we hope will support further investigations in this area.*

The paper is organized as follows. Section II provides a formal description and definition of LDPCA codes. Section III describes the proposed construction and design framework. Results validating the benefits of the proposed designs and benchmarking performance against other alternatives are presented in Section IV. Section V summarizes conclusions. The appendix outlines key details of the differential evolution (DE) procedure used for optimizing code designs.

II. LDPCA Code Construction

The LDPCA encoder has three stages. The first stage is an invertible linear transformation $\mathbf{s} = \mathbf{H}\mathbf{x}$ of the source data sequence $\mathbf{x}$, where $\mathbf{H}$ is a $L \times L$ non-singular sparse binary matrix (over $GF(2)$). The second stage is a rate 1 accumulator that takes the length L sequence $\mathbf{s} \stackrel{\text{def}}{=} [s_1, s_2, \ldots s_L]^T$ and generates the length L sequence $\mathbf{c} = [c_1, c_2, \ldots c_L]^T$, where $c_i = \sum_{j=1}^{i} s_j$. The third stage permutes the sequence $\mathbf{c}$, using a permutation $\boldsymbol{\pi}$ of the indices $[1, 2, \ldots L]$, to obtain a sequence $\mathbf{p} = [p_1, p_2, \ldots p_L]^T$. The sequence $\mathbf{p}$ is then transmitted progressively to the decoder, in sequence, with the total number of transmitted bits determined by the decoder feedback. $\mathbf{s}$ and $\mathbf{c}$ are referred to as *syndrome* and *accumulated syndrome* sequences, respectively, or simply as syndromes when context eliminates ambiguity. A discrete set of N possible rates is enabled by the rate scalability, where the integer parameter N is a factor of the block length L so that $M = L/N$ is an integer. The encoder first sends M syndromes to the decoder and responds with M additional syndromes in response to each decoder request for additional bits sent when the decoder's attempt to recover $\mathbf{x}$ based on already received data fails. The process continues until decoding succeeds, which is usually verified using a check sum of the source message that is independently communicated to the decoder, resulting in a (negligible) overhead. The rate-adaptivity offers the discrete set of N rates $(1/N, 2/N, \cdots, (N-1)/N, 1)$. The permutation $\boldsymbol{\pi}$ that maps $\mathbf{c}$ to $\mathbf{p}$ (specifically, $p_{\pi_i} = c_i$) is designed to ensure that for any number of transmitted bits, the transmitted sequence represents a nearly uniform (and regular) sampling of the accumulated syndrome sequence $\mathbf{c}$. Algorithm 1 summarizes the computation of the permutation $\boldsymbol{\pi}$. The encoder is shown in Fig. 1 for a toy example with $L = 8$, $N = 4$, and $M = L/N = 2$, where we have $\boldsymbol{\pi} = [4, 8, 2, 6, 1, 5, 3, 7]$ from Algorithm 1.

Algorithm 1 Computation of the permutation $\boldsymbol{\pi}$ from $\mathbf{c}$ to $\mathbf{p}$ determining the LDPCA transmission order

Input: block-length L and rate scalability parameter N, where N is a factor of L.
Output: A permutation $\boldsymbol{\pi}$ of the integers from 1 through L.

```
Initialize: l^1 ← 1, u^1 ← N, h ← 1, t ← 1, π ← [N, 2N, ..., L − N, L]
while t ≥ h do
    if l^h ≠ u^h then
        l^{t+1} ← l^h, u^{t+1} ← l^h + ⌊(u^h − l^h)/2⌋
        l^{t+2} ← l^h + ⌊(u^h − l^h)/2⌋ + 1, u^{t+2} ← u^h
        π ← [π, u^{t+1} + [0, N, 2N, ..., (L − N)]]
        t ← t + 2
    end if
    h ← h + 1
end while
```

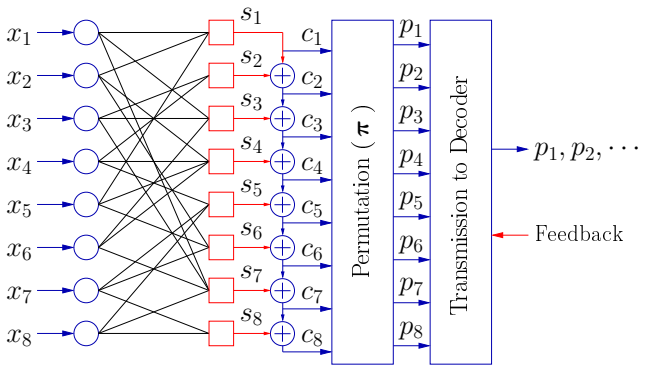


Fig. 1: An example LDPCA encoder with block-length $L = 8$.

At rate 1, the decoder recovers the message by inverting the permutation, accumulation, and the linear transformation ($\mathbf{H}$) steps performed at the encoder. For rates below 1, the decoding is posed as an error correction decoding problem for recovery of $\mathbf{x}$ from the noisy version $\mathbf{y}$ available at the decoder as side-information and the encoded data $\mathbf{p}^{(k)}$ received from the encoder. Specifically, let $\mathbf{p}^{(k)} = [p_1, p_2, \ldots p_{kM}]^T$ denote the sequence of bits received from the encoder (thus far), where k denotes the number of requests received by the encoder (starting with $k = 1$ for the first transmission). $\mathbf{p}^{(k)}$ is the leading subsequence of kM bits from $\mathbf{p}$. The decoder first (partly) undoes the permutation and accumulation operations performed by the encoder. Let $\tilde{\boldsymbol{\pi}}^k = [\tilde{\pi}_1^{(k)}, \tilde{\pi}_2^{(k)}, \cdots, \tilde{\pi}_{kM}^{(k)}]$ denote the first kM entries in the permutation vector $\boldsymbol{\pi}$, *sorted in ascending-order*, i.e., $1 \leq \tilde{\pi}_1^{(k)} < \tilde{\pi}_2^{(k)} \cdots \tilde{\pi}_{kM}^{(k)} \leq L$. Using

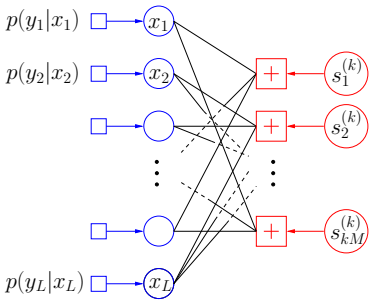
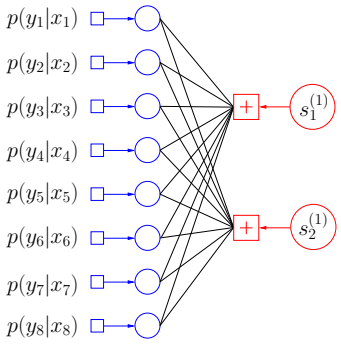


Fig. 2: Decoding graph $\mathcal{G}(\mathbf{H}^{(k)})$ for the effective LDPC code at rate (k/N).

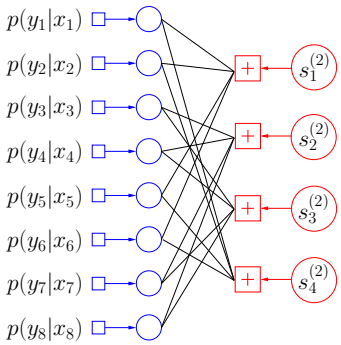
$\tilde{\boldsymbol{\pi}}^{(k)}$ and the received $\mathbf{p}^{(k)}$, we obtain a subsequence $\mathbf{c}^{(k)} = [c_{\tilde{\pi}_1^{(k)}}, c_{\tilde{\pi}_2^{(k)}} \ldots c_{\tilde{\pi}_{kM}^{(k)}}]^T$ of the sequence $\mathbf{c}$. $\mathbf{c}^{(k)}$, in turn, can be used to obtain $s_l^{(k)} \stackrel{\text{def}}{=} \sum_{j=\tilde{\pi}_{l-1}^{(k)}}^{\tilde{\pi}_l^{(k)}} s_j = (c_{\tilde{\pi}_l^{(k)}} - c_{\tilde{\pi}_{l-1}^{(k)}})$ for $l = 1, 2, \ldots kM$, where we introduce $\tilde{\pi}_0^{(k)} = 0$ and $s_0 = 0$ to simplify notation. Now, letting $\mathbf{s}^{(k)} = [s_1^{(k)}, s_2^{(k)}, \ldots s_{kM}^{(k)}]^T$ we see that $\mathbf{s}^{(k)} = \mathbf{H}^{(k)}\mathbf{x}$ where $\mathbf{H}^{(k)}$ is a $(kM)\times L$ binary matrix whose l^{th} row is the sum of the rows $\tilde{\pi}_{l-1}^{(k)}$ through $\tilde{\pi}_l^{(k)}$ of $\mathbf{H}$. The sparsity of $\mathbf{H}$ ensures that $\mathbf{H}^{(k)}$ is also sparse as long as successive indices of subsequence $(\tilde{\pi}_0^{(k)}, \tilde{\pi}_1^{(k)}, \ldots \tilde{\pi}_{kM}^{(k)})$ are not too far apart; the design of the permutation $\boldsymbol{\pi}$ ensures this constraint is met.

The matrix $\mathbf{H}^{(k)}$ is interpreted as the parity check matrix for an $(L, L-kM)$ low density parity check (LDPC) [20], [21] code, for which, the vector $\mathbf{x}$ lies in the coset uniquely identified by the syndrome $\mathbf{s}^{(k)} = \mathbf{H}^{(k)}\mathbf{x}$. Using $\mathbf{s}^{(k)}$ the decoder computes a symbol-by-symbol maximum (approximate) *a posteriori* probability decoding $\hat{\mathbf{x}}$ for $\mathbf{x}$ via belief propagation on a modified decoding graph for the code illustrated in Fig. 2. The graph, has two types of nodes: L *variable nodes* corresponding to the L bits in $\mathbf{x}$ (depicted by circles), and kM *check nodes* corresponding to the kM bits in the syndrome $\mathbf{s}^{(k)}$ (depicted by squares). Edges in the graph are defined by the $kM \times L$ parity check matrix $\mathbf{H}^{(k)}$ for the LDPC code, an edge connects the i^{th} check node and the j^{th} variable node if and only if $H_{ij}^{(k)} = 1$. We denote the decoding graph for the LDPC code with parity check matrix $\mathbf{H}^{(k)}$ by $\mathcal{G}(\mathbf{H}^{(k)})$.

For our example code, the decoder starts with a low rate of $1/4$ and first receives $\mathbf{p}^{(1)} = [p_1, p_2]$, which after undoing the permutation and summation provides $\mathbf{c}^{(1)} = [c_4, c_8]$ and $\mathbf{s}^{(1)}$, respectively. The decoder attempts decoding on the graph of Fig. 3(a) that describes $\mathbf{s}^{(1)} = \mathbf{H}^{(1)}\mathbf{x}$. If decoding fails at rate $1/4$, the decoder requests additional bits and the encoder sends $[p_3, p_4]$. Combining these with the previously received information and again undoing the permutation and summation, the decoder obtains $\mathbf{p}^{(2)} = [p_1, p_2, p_3, p_4]$, $\mathbf{c}^{(2)} = [c_2, c_4, c_6, c_8]$, and $\mathbf{s}^{(2)} = \mathbf{H}^{(2)}\mathbf{x} = [s_1^{(2)}, s_2^{(2)}, s_3^{(2)}, s_4^{(2)}]$ and attempts decoding using the corresponding graph of Fig. 3(b). If the decoding fails at rate $1/2$ and also at the next rate of $3/4$, the encoder sends all remaining bits in $\mathbf{p}$ to the decoder, which then (cumulatively) has $\mathbf{p}^{(4)} = [p_1, p_2, \ldots, p_8] = \mathbf{p}$ from which it recovers $\mathbf{s} = \mathbf{s}^{(4)} = [s_1^{(4)}, s_2^{(4)}, \ldots, s_8^{(4)}]$ and then the data as $\mathbf{x} = \mathbf{H}^{-1}\mathbf{s}$.



(a) rate $1/4$.



(b) rate $1/2$.

Fig. 3: Decoding graph for our example LDPCA code at rate: (a) $1/4$, where $s_1^{(1)} = s_1+s_2+s_3+s_4$, $s_2^{(1)} = s_5+s_6+s_7+s_8$, and (b) $1/2$, where $s_1^{(2)} = s_1 + s_2$, $s_2^{(2)} = s_3 + s_4$, $s_3^{(2)} = s_5 + s_6$, and $s_4^{(2)} = s_7 + s_8$.

III. Proposed LDPCA Construction: Code Analysis and Design

The LDPCA code is defined by the matrix $\mathbf{H}$ along with the permutation $\boldsymbol{\pi}$ and its performance can be characterized by analyzing the series of resulting LDPC codes at each of the N rates. For symmetric virtual channels, the standard LDPC analysis methodology of density evolution [22], [23] applies, based on which we establish an objective function for designing LDPCA codes.

Our designs consider scenarios where the side information correlation $p_{Y|X}$ is modeled either as a binary symmetric (BSC) or as a binary-input additive white Gaussian noise (BIAWGN) channel [23]. To unify notation and description, we use a common scalar *channel degradation parameter* q to denote either the probability of bit error for the BSC setting or the standard deviation of the noise for the BIAWGN setting. The minimum required rate corresponding to the channel degradation parameter q is then designated by $H(X|Y;q)$. The code performance at the rate (k/N) is quantified by estimating, via density evolution, the maximum channel parameter q_k^* for which successful decoding is expected and evaluating the rate *gap* $g_k \stackrel{\text{def}}{=} (k/N) - H(X|Y;q_k^*)$, where a smaller rate gap is clearly desirable. The value q_k^* is referred to as the *threshold* of the LDPC code with parity check matrix $\mathbf{H}^{(k)}$ and in the asymptotic regime of large block-lengths depends only on the statistical distribution of edges in the decoding graph $\mathcal{G}(\mathbf{H}^{(k)})$. Specifically, in $\mathcal{G}(\mathbf{H}^{(k)})$, let $\lambda_i^{(k)}$ and $\rho_i^{(k)}$ denote, the fraction

of edges (out of the total edges in the graph) that emanate from degree-i variable and check nodes, respectively, where the degree of a node is the number of edges connected to the node. The *edge-wise* degree distribution for $\mathcal{G}(\mathbf{H}^{(k)})$ can then be summarized via the pair of polynomials $(\lambda^{(k)}(x), \rho^{(k)}(x))$, where $\lambda^{(k)}(x) = \sum_{i\geq 2} \lambda_i^{(k)} x^{i-1}$, $\rho^{(k)}(x) = \sum_{i\geq 2} \rho_i^{(k)} x^{i-1}$ and $\sum_{i\geq 2} \lambda_i^{(k)} = \sum_{i\geq 2} \rho_i^{(k)} = 1$. For large enough block-length, the decoding performance of LDPCA code at rate (k/N) closely matches the average performance of an ensemble of codes that share the same statistical distribution of edges $(\lambda^{(k)}(x), \rho^{(k)}(x))$ and density evolution [22] allows quantification of this average performance via estimation of the threshold q_k^*.

We use the *average gap*

$$g_A \stackrel{\text{def}}{=} \frac{1}{N}\sum_{k=1}^{N} g_k \tag{1}$$

across the operating rates of the code as a single numerical figure of merit quantifying the performance of the LDPCA code, and as a cost function for code design. This specific choice allows for consistent comparison with previously reported designs in [12], [19]. Note, however, that the design framework we propose can also readily handle alternative performance metrics, such as the worst-case gap.

A. Proposed LDPCA Code Construction

The complete space of $L \times L$ sparse binary matrices $\mathbf{H}$ is prohibitively large, practical designs explore a smaller region of this space that is large enough to include good codes and small enough to facilitate design. We restrict our attention to code constructions that enable generation of $\{\mathbf{H}^{(i)}\}_{i=2}^{N}$ from the lowest rate mother code $\mathbf{H}^{(1)}$. Each row in $\mathbf{H}^{(1)}$ arises as the sum of a selection of rows from $\mathbf{H}$, when there is no overlap between the non-zero entries in the selected rows, the total number of edges in the decoding graph $\mathcal{G}(\mathbf{H})$ is preserved in the decoding graphs $\left\{\mathcal{G}(\mathbf{H}^{(k)})\right\}_{k=1}^{N-1}$. It follows that under this *non-overlapping constraint*, the decoding graph $\mathcal{G}(\mathbf{H}^{(k)})$ can be viewed as arising from a splitting of M check nodes in $\mathcal{G}(\mathbf{H}^{(k-1)})$ into two check nodes each in $\mathcal{G}(\mathbf{H}^{(k)})$. Equivalently, $\mathbf{H}^{(k)}$ can be generated from $\mathbf{H}^{(k-1)}$ by identifying, based on the scheduling permutation $\boldsymbol{\pi}$, M rows in $\mathbf{H}^{(k-1)}$ designated for splitting and splitting each of these rows in $\mathbf{H}^{(k-1)}$ into a pair of rows in $\mathbf{H}^{(k)}$, where, in the splitting process, the nonzero entries in a row in $\mathbf{H}^{(k-1)}$ are partitioned into the nonzero entries in the pair of rows generated in $\mathbf{H}^{(k)}$. The code is therefore constructed by choosing the parity check matrix $\mathbf{H}^{(1)}$ for the mother code and progressively generating $\mathbf{H}^{(k)}$ from $\mathbf{H}^{(k-1)}$ for $2 \leq k \leq N$ by splitting the rows as per the scheduling order defined by the permutation $\boldsymbol{\pi}$. Flexibility in partitioning of the nonzero entries in the process of splitting a row in $\mathbf{H}^{(k-1)}$ into a pair of rows in $\mathbf{H}^{(k)}$, *allows design choices in the degree distributions at the different stages, which in turn influences the performance at the corresponding rates*. We further simplify the code design by designing the mother code $\mathbf{H}^{(1)}$ with a *concentrated* check-node degree distribution, where check node degrees differ by at most one – a constraint under which performance extremely close to capacity has been demonstrated with LDPC codes [24]. Preserving check-node concentration across all rates (k/N) $(1 \leq k \leq N)$ motivates *uniform splitting* wherein when splitting a selected row $\boldsymbol{\phi}^T$ in $\mathbf{H}^{(k)}$ into two rows $\boldsymbol{\epsilon}^T$ and $\boldsymbol{\delta}^T$ in $\mathbf{H}^{(k+1)}$, non-zero entries are partitioned equally (off by one, if necessary) and randomly between $\boldsymbol{\epsilon}^T$ and $\boldsymbol{\delta}^T$. However, as we subsequently demonstrate in Section IV, LDPCA codes constructed with uniform splitting exhibit an inherent performance trade-off between different rate regions. We alleviate this trade-off by using non-uniform splitting.

After considering different non-uniform splitting options, we adopt a strategy that is relatively simple yet offers good performance. The strategy is motivated in part by observed statistics of degree-distributions of non-adaptive LDPC codes. Specifically, we note that good concentrated LDPC codes designed for individual DSC rates, have variable and check node degrees that are: (a) relatively large at low rates, with average variable node degrees close to 6 at rates close to 0 and (b) small for high rates, taking on values of 2 or 3 when the rate approaches 1, which are the smallest degrees that can be meaningfully used for belief-propagation. Because the total number of edges remains constant in the LDPCA graph, the non-uniform splitting must maintain the same average degree as the uniform splitting. Non-uniform splitting, however, allows introduction of degree 2 and 3 nodes at the higher rates, offering an advantage, as we subsequently see. Because previously designed LDPCA codes with (almost) concentrated check-node degrees can offer good performance at mid/low rates, uniform splitting is utilized for rates (k/N) lower than a chosen upper bound (k_u/N), where $2 \leq k_u < N$ is an integer design parameter. At higher rates (k/N), with $k \geq k_u$, each split of a row in $\mathbf{H}^{(k)}$ to generate two rows in $\mathbf{H}^{(k+1)}$ is performed non-uniformly to create, in the resulting pair, one low degree row with degree 2 or 3. A parameter η in $[0, 1]$ controls the relative fractions of these degree 2 and 3 nodes. Specifically, of the total M rows in $\mathbf{H}^{(k)}$ to be split into pairs during the process of generating $\mathbf{H}^{(k+1)}$, fractions η and $(1-\eta)$ are forced during the splitting process to have rows of degree 2 and 3, respectively, as one of the rows generated by the split, where η is another parameter in the code design. That is, of the M rows $\mathbf{H}^{(k)}$ designated for splitting by the transmission order $\boldsymbol{\pi}$, ηM are randomly selected and each selected row $\boldsymbol{\phi}^T$ is split into two rows $\boldsymbol{\epsilon}^T$ and $\boldsymbol{\delta}^T$ in $\mathbf{H}^{(k+1)}$, with $\boldsymbol{\epsilon}^T$ having degree 2 and $\boldsymbol{\delta}^T$ having degree $(w_{\mathcal{H}}(\boldsymbol{\phi}) - 2)$ where $w_{\mathcal{H}}(\mathbf{x})$ denotes the Hamming weight of the binary vector $\mathbf{x}$. Similarly, for the remaining $(1-\eta)M$ rows designated for splitting, each row $\boldsymbol{\phi}^T$ is split into two rows $\boldsymbol{\epsilon}^T$ and $\boldsymbol{\delta}^T$ with degrees 3 and $(w_{\mathcal{H}}(\boldsymbol{\phi}) - 3)$, respectively. For the splits into degree 2 (3) nodes, 2 (3) of the nonzero entries of $\boldsymbol{\phi}$ are randomly selected for allocation to $\boldsymbol{\epsilon}^T$. Algorithm 2 summarizes the procedure for generating $\{\mathbf{H}_k\}_{k=2}^{N}$ using $\mathbf{H}^{(1)}$, $\boldsymbol{\pi}$, k_u and η and the splitting process just outlined.

For our example code in Section II, each degree 8 check node for the rate $1/4$ code graph in Fig. 3(a) is split into two degree 4 nodes to obtain the rate $1/2$ code of Fig. 3(b). An alternative rate $1/2$ code shown in Fig. 4 is obtained by non-uniform splitting, where each degree-8 node in Fig. 3(a) is

Algorithm 2 Construct LDPCA code from the lowest rate mother code using splitting

Input: $\mathbf{H}^{(1)}, \boldsymbol{\pi}, k_u, \eta$

$\mathbf{H}^{(1)}$: lowest rate mother code, $\boldsymbol{\pi}$: permutation vector determined by Alg. 1.

k_u and η: non-uniform splitting parameters

Output: The matrix $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{H}^{(N)}$ defining the LDPCA code (along with $\boldsymbol{\pi}$)

1: **repeat**
2: **for** $k \leftarrow 2$ to N **do**
Construct the $(kM) \times L$ matrix $\mathbf{H}^{(k)}$ by splitting rows of $\mathbf{H}^{(k-1)}$
Denote m^{th} row of $\mathbf{H}^{(k)}$ ($\mathbf{H}^{(k-1)}$) by $\mathbf{H}^{(k)}_m$ ($\mathbf{H}^{(k-1)}_m$)
3: $d \leftarrow l$, where $\tilde{\pi}^{(k)}_l = \pi_k$
$\tilde{\boldsymbol{\pi}}^{(k)}$ is the vector of the first k elements of $\boldsymbol{\pi}$ sorted in ascending order
$\mathbf{H}^{(k-1)}$ is vertically divided into M sub-matrices, from each, split the d^{th} row into two rows
4: **for** $i \leftarrow 1$ to M **do**
Split $\mathbf{H}^{(k-1)}_{(k-1)(i-1)+d}$ into two rows $\mathbf{H}^{(k)}_{k(i-1)+d}$ and $\mathbf{H}^{(k)}_{k(i-1)+d+1}$ (see details in text)
Copy other rows from $\mathbf{H}^{(k-1)}$ into $\mathbf{H}^{(k)}$
5: $\mathbf{H}^{(k)}_{k(i-1)+j} \leftarrow \mathbf{H}^{(k-1)}_{(k-1)(i-1)+j}$ for $1 \le j < d$
6: $\mathbf{H}^{(k)}_{k(i-1)+j} \leftarrow \mathbf{H}^{(k-1)}_{(k-1)(i-1)+j-1}$ for $(d+1) < j \le k$
7: **end for**
8: $k \leftarrow k+1$
9: **end for**
10: **until** $\mathbf{H}^{(N)}$ is non-singular

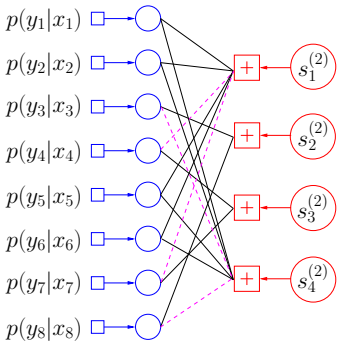


Fig. 4: A decoding graph at rate $1/2$ obtained via a non-uniform splitting from Fig. 3(a), in contrast with the graph of Fig. 3(b) obtained by uniform splitting.

split into two check nodes with degree 2 and 6, respectively.

B. Degree Distributions for Proposed LDPCA codes

The splitting process used to obtain $\mathbf{H}^{(k)}$ from $\mathbf{H}^{(1)}$ in Algorithm 2, can be mirrored in analysis to infer degree distributions $(\lambda^{(k)}(x), \rho^{(k)}(x))$ for the effective LDPC code at rate k/N from the degree distributions $(\lambda^{(k-1)}(x), \rho^{(k-1)}(x))$ for the effective LDPC code at rate $(k-1)/N$. Applying this process recursively, the code parameters $(\lambda^{(1)}(x), \rho^{(1)}(x), \eta, k_u)$, yield all the degree distributions $\{(\lambda^{(k)}(x), \rho^{(k)}(x))\}_{k=2}^{N}$.

To proceed with the analysis, we introduce the *node-wise* degree distributions for the decoding graph $\mathcal{G}(\mathbf{H}^{(k)})$ summarized by the pair of polynomials $(\Lambda^{(k)}(x), \Gamma^{(k)}(x))$, where $\Lambda^{(k)}(x) = \sum_{i\ge 2} \Lambda_i^{(k)} x^i$, $\Gamma^{(k)}(x) = \sum_{i\ge 2} \Gamma_i^{(k)} x^i$, and $\Lambda_i^{(k)}$ ($\Gamma_i^{(k)}$) indicates the fraction of variable (check) nodes that have degree-i in $\mathcal{G}(\mathbf{H}^{(k)})$. One can readily convert between the node-wise and edge-wise degree distributions [23, pp. 79].

Because variable node degrees remain unchanged in the splitting process used to generate $\mathbf{H}^{(k)}$ from $\mathbf{H}^{(k-1)}$, we readily see that $\Lambda^{(k)}(x) = \Lambda^{(k-1)}(x)$. To obtain $\Gamma^{(k)}(x)$ from $\Gamma^{(k-1)}(x)$, note that $\mathbf{H}^{(k)}$ is generated from $\mathbf{H}^{(k-1)}$ by splitting M rows, equivalently, a fraction $1/(k-1)$ of the total rows. To obtain $\Gamma^{(k)}(x)$, we first rewrite $\Gamma^{(k-1)}(x)$ as

$$\Gamma^{(k-1)}(x) = \Omega^{(k-1)}(x) + \Theta^{(k-1)}(x), \tag{2}$$

where $\Theta^{(k-1)}(x) = \sum_{i\ge 2} \Theta_i^{(k-1)} x^i$ and $\Omega^{(k-1)}(x) = \sum_{i\ge 2} \Omega_i^{(k-1)} x^i$. $\Omega^{(k-1)}(x)$ describes the part of the degree distribution corresponding to the check nodes selected for splitting and $\Theta^{(k-1)}(x)$ the degree distribution corresponding to the remaining check nodes. In the splitting process, because we use concentrated mother codes and use $k_u > (N/2)$, i.e., we split non-uniformly only for rates larger than $1/2$, the rows selected for splitting have degrees greater than or equal to the maximum degree for the rows not selected. Therefore, $\Omega^{(k-1)}(x)$ can be readily obtained by accumulating a fraction $1/(k-1)$ of the edges represented in $\Gamma^{(k-1)}(x)$ proceeding in order from the highest degree edges toward lower degree edges. In this process, we are assured that the minimum polynomial exponent in $\Omega^{(k-1)}(x)$ is no smaller than the maximum polynomial exponent in $\Theta^{(k-1)}(x)$, and $\Omega^{(k-1)}(1) = \sum_{i\ge 2} \Omega_i^{(k-1)} = 1/(k-1)$.

Now we can write

$$\Gamma^{(k)}(x) = \frac{k-1}{k}\left(\Theta^{(k-1)}(x) + \tilde{\Omega}^{(k-1)}(x)\right), \tag{3}$$

where $\tilde{\Omega}^{(k-1)}(x)$ describes the contribution, to the degree distribution $\Gamma^{(k)}(x)$, of the $2M$ nodes generated by splitting the M nodes described by $\Omega^{(k-1)}(x)$ and the normalizing factor $(k-1)/k$ accounts for the fact that each split node generates two nodes and ensures $\Gamma^{(k)}(1) = \sum_{i\ge 2} \Gamma_i^{(k)} = 1$. For $k < k_u$, we use uniform splitting and $\tilde{\Omega}^{(k-1)}(x)$ can be written as [18]:

$$\tilde{\Omega}^{(k-1)}(x) = 2\Omega_e^{(k-1)}(x^{\frac{1}{2}}) + x^{\frac{1}{2}}\Omega_o^{(k-1)}(x^{\frac{1}{2}}) + x^{-\frac{1}{2}}\Omega_o^{(k-1)}(x^{\frac{1}{2}}), \tag{4}$$

where $\Omega_e^{(k-1)}(x)$ and $\Omega_o^{(k-1)}(x)$, respectively, represents the polynomial terms within $\Omega^{(k-1)}(x)$ with even and odd degrees, and $\Omega^{(k-1)}(x) = \Omega_e^{(k-1)}(x) + \Omega_o^{(k-1)}(x)$. The check nodes described by $\Omega_e^{(k-1)}(x)$ and $\Omega_o^{(k-1)}(x)$, respectively, are described by $2\Omega_e^{(k-1)}(x^{\frac{1}{2}})$ and $(x^{\frac{1}{2}}\Omega_o^{(k-1)}(x^{\frac{1}{2}}) + x^{-\frac{1}{2}}\Omega_o^{(k-1)}(x^{\frac{1}{2}}))$ after splitting.

For $k \ge k_u$, non-uniform splitting is used, and the new

nodes are described by

$$\tilde{\Omega}^{(k-1)}(x) = \eta\left(\frac{\Omega^{(k-1)}(x)}{x^2} + \frac{1}{k-1}x^2\right) + (1-\eta)\left(\frac{\Omega^{(k-1)}(x)}{x^3} + \frac{1}{k-1}x^3\right). \quad (5)$$

The first term in (5) can be seen by recalling that $M\eta$ check nodes are split into $2M\eta$ new check nodes, $M\eta$ of which are degree 2 nodes and described by $(\eta/(k-1))x^2$, and the remaining $M\eta$ nodes are described by $\eta\Omega^{(k-1)}(x)/x^2$. The second term in (5) can be similarly interpreted for the splits generating degree 3 nodes.

The overall parameterization for the LDPCA code can be further compacted by using the fact that the mother code is concentrated and has rate $r = 1/N$ to obtain the check-node degree distribution $\rho^{(1)}(x) = (1-\rho)x^{j-1} + \rho x^j$, where [23] $\bar{j} = 1/\left(r\int_0^1 \lambda^{(1)}(x)dx\right)$, $j = \lfloor \bar{j} \rfloor$, and $\rho = (\bar{j} - j)(j+1)/\bar{j}$. Putting together the steps developed in this subsection, the degree distributions $\{(\lambda^{(k)}(x), \rho^{(k)}(x))\}_{k=2}^{N}$ can be obtained from $(\lambda^{(1)}(x), k_u, \eta)$.

We note that our analysis in this subsection builds upon and extends the analysis presented by Varodayan in [18], which addressed only the uniform splitting used in prior code designs.

C. Code Design for LDPCA Codes

For an LDPCA code described by the parameters $(\lambda^{(1)}(x), k_u, \eta)$, the analysis procedure of the preceding section yields the degree distributions $\{(\lambda^{(k)}(x), \rho^{(k)}(x))\}_{k=2}^{N}$, which used with density evolution [22] provide the rate gaps $\{g_k\}_{k=1}^{N-1}$ and in turn and the average gap g_A. We formulate the LDPCA code design problem as a search for code parameters minimizing the average gap, i.e.,

$$(\lambda^{(1)*}(x), k_u^*, \eta^*) = \underset{(\lambda^{(1)}(x), k_u, \eta)}{\operatorname{argmin}} \; g_A. \quad (6)$$

A global optimization strategy is necessary because the optimization problem in (6) is non-convex with an objective function that must be numerically evaluated. We perform our search by first generating candidates for (k_u, η) over the permissible ranges for these parameters using combination of gridding and random generation. For each candidate choice of the parameters (k_u, η), to search for a degree distribution $\lambda^{(1)}(x)$ that minimizes the average gap g_A, we employ the differential evolution (DE) [25] global optimization technique, which has proven to be useful in designing LDPC codes [24], [26]. Problem specific considerations for DE are summarized in the appendix.

After a good set of parameter values $(\lambda^{(1)*}(x), k_u^*, \eta^*)$ are determined, first the parity check matrix $\mathbf{H}^{(1)}$ is generated for the mother code using the progressive edge growth [27] algorithm which is a greedy approach to generate LDPC parity check matrices that avoids undesirable short length cycles in the decoding graph. Then, using the permutation $\boldsymbol{\pi}$ from Algorithm 1 and the parameters (k_u^*, η^*) in Algorithm 2, we obtain the matrix $\mathbf{H}$ that defines the LDPCA code (along with $\boldsymbol{\pi}$).

IV. Results

To illustrate the benefit of the proposed LDPCA code constructions, we benchmark their performance against alternative constructions using density evolution analysis of the designed degree distributions and Monte Carlo simulations of the actual codec. We consider the BSC and BIAWGN virtual channels. The latter channel model is commonly employed for practical distributed source coding applications where the side-information $\mathbf{y}$ at the receiver is continuous-valued, though the input $\mathbf{x}$ is binary (See, for example [28], [29]). For the BSC correlation channel, the channel degradation parameter is the cross-over probability $q = p_{Y|X}(0|1) = p_{Y|X}(1|0)$, and $H(X|Y;q) = -q\log_2 q - (1-q)\log_2(1-q)$. The BIAWGN channel is represented as $Y = (2X-1) + Z$, where $Z \sim \mathcal{N}(0, q^2)$ is the additive white Gaussian noise and q is the standard deviation of Z. Then, $H(X|Y;q) = 1 - C(q)$, where $C(q) = -\int \phi_q(x)\log_2 \phi_q(x)dx - \frac{1}{2}\log_2(2\pi e q^2)$ is the capacity for the BIAWGN channel, with $\phi_q(x) = \frac{1}{\sqrt{8\pi q^2}}\left(e^{-\frac{(x+1)^2}{2q^2}} + e^{-\frac{(x-1)^2}{2q^2}}\right)$.

A. Parameter Selections for Code Design

To facilitate comparisons with previously reported results in [12], [19], we selected the rate scalability parameter $N = 66$. We first generate a set of candidate values for k_u in the range $40 \le k_u \le 65$ and η in the interval $0 \le \eta \le 1$; the latter by combining a coarse grid corresponding to $\eta = 0.25, 0.50, 0.75$ with a set of randomly generated candidates. Next, random values are generated for the average variable-node degree $\bar{\lambda}$ (see Appendix for definition) in the range $3 \le \bar{\lambda} \le 6$. This range has been found to be adequate in prior work on LDPC code design [18], [26], [28]. For the DE procedure, a population size of $N_c = 48$ and a differential mixing parameter $F = 1/2$ are used. Initial candidates are generated with maximum polynomial exponent $D_{\max} = 33$ and $D_n = 6$ non-zero terms. Iterations terminate when the average gap falls below a threshold of $T = 0.02$ or after a maximum iteration count of 50. To accelerate the DE step, instead of the average gap, we use the sum of gaps at the subset of rates (k/N) for $k \in \{5, 10, 20, 30, 40, 45, 50, 55, 58, 62\}$.

From the parameters $(\lambda^{(1)*}(x), k_u^*, \eta^*)$ obtained via the optimization process, we design an actual LDPCA code matrix $\mathbf{H}$ using a value of $M = 249$, resulting in an overall code block-length of $L = NM = 16434$. These values are also chosen for compatibility with prior designs [12], [19] against which we benchmark the code's performance. To highlight the impact of the proposed nonuniform splitting, we include in our benchmarking, designs and codes constrained to the conventional uniform splitting but obtained with our methodology by setting $k_u = (N+1)$. Additional parameters for these alternative designs are introduced subsequently as required.

B. Designs for the BSC Channel

Using the proposed method, for the BSC channel, we obtain an optimized set of LDPCA code parameters given by $\lambda^{(1)*}(x) = 0.1166x + 0.221x^2 + 0.2732x^5 + 0.2232x^{24} + 0.1222x^{31} + 0.0439^{32}$, $\eta^* = 0.5$ and $k_u^* = 49$. This parameter

set is designated NU to indicate that it is obtained with the proposed nonuniform splitting strategy.

We benchmark the code against four other LDPCA code designs. The first of these is the code in [19], which represents the best reported previous design, and has a mother code degree distribution $\lambda^{(1)}(x) = 0.071112x + 0.238143x^2 + 0.182737x^3 + 0.073795x^9 + 0.079317x^{14} + 0.354896x^{32}$. We label this code as U$[0.15, 0.75]$ since the code uses uniform splitting and explicitly optimizes for two rates 0.15 and 0.75. Following the methodology in [19], i.e., minimizing the gap at two code rates, we also use our proposed design procedure to design two alternative LDPCA codes that conform to the conventional uniform splitting: U$[0.1, 0.7]$ which minimizes the sum of the gap at the two rates $r = 0.1, 0.7$ and has $\lambda^{(1)}(x) = 0.0987x + 0.2322x^2 + 0.1816x^4 + 0.0478x^8 + 0.0688x^{14} + 0.3711x^{32}$, and U$[0.1, 0.9]$ which minimizes the sum of the gap at the two rates $r = 0.1, 0.9$, with $\lambda^1(x) = 0.1471x + 0.4642x^2 + 0.1102x^8 + 0.022x^{25} + 0.0575x^{28} + 0.199x^{32}$. To specifically highlight the benefit of the proposed nonuniform splitting, we also obtain a series of degree distributions by starting with the mother code degree distribution in [19] and using the proposed nonuniform splitting procedure (instead of the conventional uniform splitting used in [19]); NU$[0.15, 0.75]$ represents this design. In our density evolution performance comparisons, we also include the predicted performance for non-adaptive LDPC code designs (designated LDPC-NA) obtained with the method described in [26].

LDPCA codes generated using the optimized design parameters were also experimentally evaluated using Monte-Carlo simulations and compared against existing LDPCA codes. The simulations were conducted with the channel degradation parameter q chosen to sample the conditional entropy $H(X|Y;q)$ in uniform steps of size 0.05 over the range from 0 to 1. For each choice of q, $N_s = 200$ pairs[2] $(\mathbf{x}, \mathbf{y})$ of data and side-information vectors, each matching the code block-length $L = 16434$, were generated where the x_i's were iid with $p(x_i = 1) = p(x_i = 0) = (1/2)$ and the corresponding side information was obtained as $\mathbf{y} = \mathbf{x} + \mathbf{z}$ where $\mathbf{z}$ was chosen as an iid binary vector with $p(z_i = 1) = q$. For each generated data vector $\mathbf{x}$, the LDPCA codec was simulated and the number of M-bit blocks sent from the encoder to the decoder for successful recovery[3] was recorded. Aggregating the data recorded over the simulations, the average rate $\bar{r}$ over the $N_s = 200$ simulations was computed. The corresponding gap $\bar{r} - H(X|Y;q)$ to the lower-bound was used to assess the effectiveness of code. For codes corresponding to the designs already presented, we re-use the corresponding designations NU and U$[0.15, 0.75]$, where the first is based on the proposed non-uniform splitting design and the second is the previously best reported code from [19]. In addition, we include three codes U2-4, U3 and U2-21 from [12] obtained via uniform splitting of the mother codes with node-wise degree distributions $\Lambda^1(x) = 0.3x^2 + 0.4x^3 + 0.3x^4$, $\Lambda^1(x) = x^3$, and $\Lambda^1(x) = 0.316x^2 + 0.415x^3 + 0.128x^7 + 0.069x^8 + 0.02x^{19} + 0.052x^{21}$, respectively.

For the proposed design, and the alternative designs, Table I lists the average gap g_A and Fig. 5 plots the rate gaps at the different operational rates, where results are included for both the density evolution analysis and for the actual codec. Several observations can be made from these results. First, the values in Table I show that the proposed NU design offers a *significant improvement over the best reported previous design* U$[0.15, 0.75]$. Compared with U$[0.15, 0.75]$, NU reduces the average rate gap by 35%. The plots in Figs. 5(a) and 5(b) reveal that NU maintains a low rate gap at all operating rates, which offers a significant improvement over U$[0.15, 0.75]$ at high rates ($r \geq 0.7$), while matching the performance of U$[0.15, 0.75]$ at lower rates. The two additional designs U$[0.1, 0.7]$ and U$[0.1, 0.9]$ illustrate that *the performance trade-off between the different rates appears intrinsic to the uniform splitting design*: for U$[0.1, 0.7]$ performance deteriorates rapidly at rates $r > 0.7$ and whereas U$[0.1, 0.9]$ offers more uniform performance across rates, the performance is markedly poorer than NU across the entire rate region. Finally the results for the NU$[0.15, 0.75]$ design obtained using the proposed splitting methodology but using the mother code degree distribution corresponding to U$[0.15, 0.75]$ (from [19]) also offer good performance, which though worse than the optimized NU design is better than the performance obtained with any of the designs obtained with uniform splitting. The performance of the codecs U2-4, U3 and U2-21 is markedly worse than the proposed NU codec design; though these codes do not exhibit an exaggerated decline in performance at high rates, their performance across the entire rate region is poorer. Overall, *the proposed* NU *designs incorporating non-uniform splitting of the check nodes in the process of developing higher rate codes from lower rate codes offer a significant improvement over codes constructed using the previously reported methodology using uniform splitting alone*.

C. Designs for the BIAWGN Channel

For the BIAWGN channel, the optimized code parameters obtained by using the proposed design procedure are: $\lambda^{(1)*}(x) = 0.1079x + 0.2898x^2 + 0.2174x^9 + 0.0448x^{12} + 0.0058x^{15} + 0.3342x^{32}$, $\eta^* = 0.765$ and $k_u^* = 48$. Using this design, an LDPCA code parity check matrix $\mathbf{H}$ was also obtained. Following a procedure similar to the one described in Section IV-B for the BSC setting, the performance of the design was analyzed by density evolution and of the code by simulations. For these evaluations, the channel degradation parameter q, which now represents the noise standard deviation, varied over the range corresponding to signal to noise ratios from 6.98 dB to -11.44 dB. For benchmarking purposes, similar evaluations were also performed for two designs and codes (each) obtained via uniform splitting, the U$[0.15, 0.75]$ code from [19] and U$[0.1, 0.7]$, designed for the BIAWGN channel to minimize the sum of gaps at two rates 0.1 and 0.7, having the mother code degree distribution $\lambda^{(1)}(x) = 0.1135x + 0.3361x^2 + 0.1947x^8 + 0.0979x^{13} + 0.0629x^{25} + 0.1948x^{32}$. Table II and Figure 6 summarize the results obtained for these codes, combining results from both

[2]Because of the relatively large block-length L, 200 simulations suffice. The estimated standard deviation of the average gap over the $N_S = 200$ simulations is only 0.00091 bits.

[3]A 32 bit cyclic redundancy check (CRC) is used to identify successful decoding. The resulting 0.002 bit overhead, although negligible, is included in our computed rate $\bar{r}$.

TABLE I. Average gap g_A for the different code designs: (a) predicted values from density evolution, (b) from Monte Carlo simulations using actual codec.

(a) Density evolution

Code	NU	NU$[0.15, 0.75]$	U$[0.15, 0.75]$	U$[0.1, 0.7]$	U$[0.1, 0.9]$	LDPC-NA
g_A	0.0398	0.0425	0.0615	0.0574	0.0638	0.0144

(b) Actual codec

Code	NU	NU$[0.15, 0.75]$	U$[0.15, 0.75]$	U2-4	U3	U2-21
g_A	0.0483	0.0536	0.0683	0.0816	0.0936	0.0696

TABLE II. Average gap g_A for the different code designs under BIAWGN channel. (DE): predicted values from density evolution; (MC): from Monte Carlo simulations using actual codec.

Code	NU	U$[0.1, 0.7]$	U$[0.15, 0.75]$
g_A (DE)	0.0272	0.0463	0.0583
g_A (MC)	0.0405	0.0561	0.0666

density evolution and simulations in a single table/graph for succinct presentation.

The observed trends in the results are similar to the BSC channel setting. Compared with U$[0.1, 0.7]$ and U$[0.15, 0.75]$, the proposed NU code offers improved performance at high rates while maintaining comparable performance in low and mid rate regions demonstrating clearly the advantage of the proposed nonuniform splitting strategy. Also, U$[0.1, 0.7]$, which is explicitly designed for BIAWGN, outperforms U$[0.15, 0.75]$ which is designed for BSC channel.

V. Conclusion

An improved construction of LDPC-Accumulate (LDPCA) codes for rate-adaptive distributed source coding is proposed. Analysis and simulation results demonstrate that the proposed construction alleviates the trade-off in the performance between different rates inherent in previous constructions. LDPCA codes designed using the proposed constructions and design methodology outperform prior designs and, in particular, offer a significant improvement in the performance at high rates without compromising performance at low rates. A software implementation of the codec is provided[4].

VI. Acknowledgment

We thank the Center for Integrated Research Computing, University of Rochester, for making available computation time required for the code design optimizations and simulations. This work was supported in part by the National Science Foundation under grant number ECS-0428157. We thank the anonymous reviewers and the associate editor for their careful reading and detailed comments that have significantly improved the presentation in this paper.

[4]The software codec is available at http://www.ece.rochester.edu/projects/siplab/networks.html.

Appendix

Differential evolution is an iterative procedure. The l^{th} iteration, or generation, has a pool of alternative candidate variable-node degree distributions $\{\tau_i^l(x)\}_{i=1}^{N_c}$, where N_c represents the number of candidates, which is constant through the iterations. To obtain the population for the $(l+1)^{\text{th}}$ generation, using the "DE/best/2/bin" variant of DE cited in [25] for its beneficial behavior, we generate a set of N_c candidate mutants,

$$\tilde{\tau}_i^{l+1}(x) = \tau_{\text{best}}^l(x) + F\left(\tau_{i_1}^l(x) + \tau_{i_2}^l(x) - \tau_{i_3}^l(x) - \tau_{i_4}^l(x)\right), \tag{7}$$

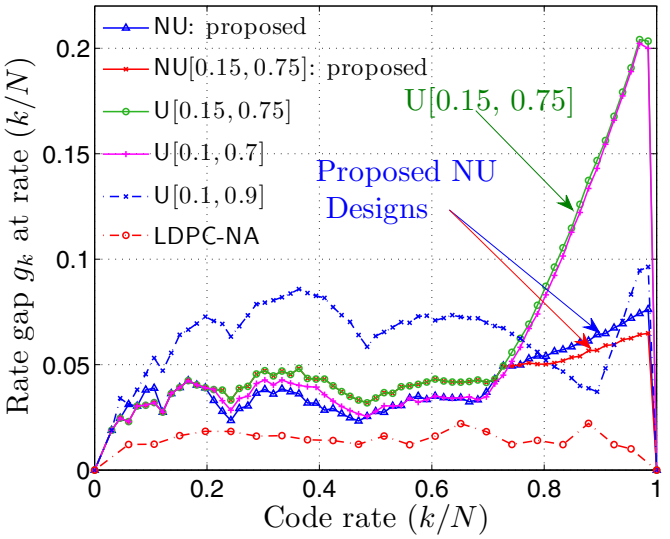
where $\tau_{i_1}^l(x), \tau_{i_2}^l(x), \tau_{i_3}^l(x)$ and $\tau_{i_4}^l(x)$ are four distinct random selections from $\{\tau_i^l(x)\}_{i=1}^{N_c}$, $\tau_{\text{best}}^l(x)$ denotes the distribution among $\{\tau_i^l(x)\}_{i=1}^{N_c}$ that minimizes the average gap g_A, and $F > 0$ is the non-negative differential mixing parameter.

By selecting between each candidate and its mutant the one that offers the smaller average gap, the $(l+1)^{\text{th}}$ generation is then obtained as

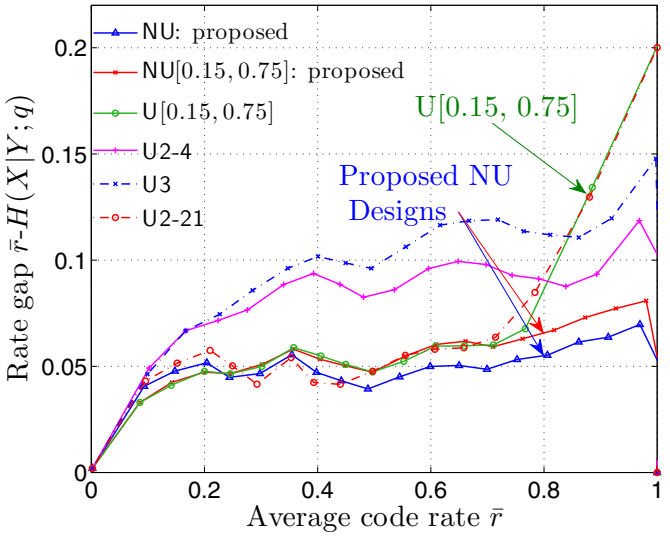
$$\tau_i^{l+1}(x) = \begin{cases} \tilde{\tau}_i^l(x), & \text{if } g_A(\tilde{\tau}_i^l(x), k_u, \eta) < g_A(\tau_i^l(x), k_u, \eta) \\ \tau_i^l(x), & \text{otherwise} \end{cases}. \tag{8}$$

for $i = 1, 2, \ldots N_c$.

Constraints and modifications are introduced for DE process to ensure stability and a concentrated check-node degree distribution for the mother code. To maintain concentration over a fixed set of adjacent degrees during the DE iterations, we structure the search for the mother code variable-degree distribution $\lambda^{(1)*}(x)$ as a series of DE searches, where each search is constrained to a fixed value for the average variable-node degree, which can be shown [26] to be $\bar{\lambda} = 1/\left(\int_0^1 \lambda^{(1)}(x)dx\right)$. The discussion in Section III-B (second paragraph from the end) indicates that fixing the variable-node degree $\bar{\lambda}$ also results in a fixed value for the concentrated check-node distribution $\rho^{(1)}(x)$, both of which also remain unchanged in the mutation process in (7), ensuring that concentration is maintained for $\rho^{(1)}(x)$. To preserve this fixed concentrated check-node degree distribution during the search, our implementation also eliminates the cross-over step in the original formulation of DE [25]. Also, candidate mutations in (7) are screened to ensure all coefficients are nonnegative and the LDPC stability constraint [26] is met. Initial candidate degree distributions $\{\tau_i^1(x)\}_{i=1}^{N_c}$ are randomly generated matching the average variable-node degree $\bar{\lambda}$ with each $\tau_i^1(x)$ constrained to a maximum polynomial exponent $D_{\max}$ and at most D_n non-zero terms. $D_{\max}$ and D_n are additional parameters defining the search process.



(a) Density evolution



(b) Actual codec

Fig. 5: Performance of the LDPCA code designs for the BSC channel evaluated via: (a) Density evolution based analysis and (b) Monte-Carlo simulations of actual codec with block-length $L = 16434$. Fig. 5(a) plots the gaps g_k to the theoretical lower bound at rate (k/N). Fig. 5(b) plots the gap between the average operational code rate $\bar{r}$ and the lower bound $H(X|Y;q)$ for a simulation with channel degradation parameter q. See text for identifying the code labels for additional details.

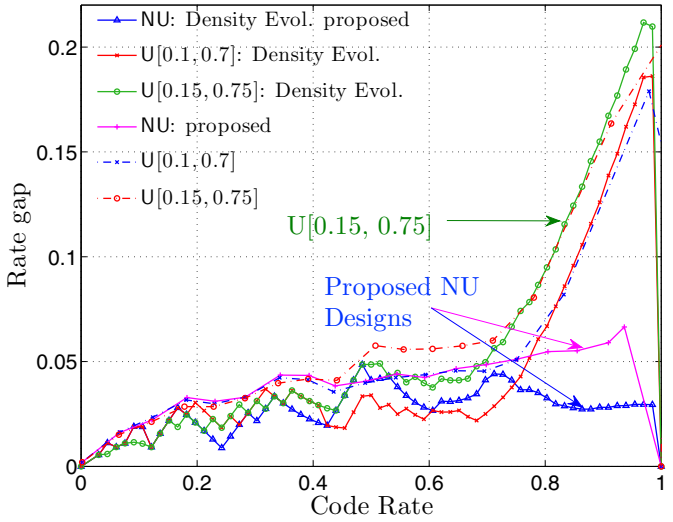


Fig. 6: Performance gap between the code rates and the lower bound $H(X|Y;q)$ for codes designed for the BIAWGN channel. Results from both density evolution (Density Evol.) and Monte-Carlo simulations are included in the same plot. See text and captions of Fig. 5 for designations of the codes included in the benchmarking.

REFERENCES

[1] Z. Xiong, A. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Process. Mag.*, vol. 21, no. 5, pp. 80–94, Sept. 2004.

[2] N. Wernersson and M. Skoglund, "Nonlinear coding and estimation for correlated data in wireless sensor networks," *IEEE Trans. Commun.*, vol. 57, no. 10, pp. 2932–2939, Oct. 2009.

[3] "Special issue: distributed signal processing in sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, Jul. 2006.

[4] J. Garcia-Frias and Y. Zhao, "Near-Shannon/Slepian-Wolf performance for unknown correlated sources over AWGN channels," *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 555–559, Apr. 2005.

[5] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inf. Theory*, vol. 19, no. 4, pp. 471–480, Jul. 1973.

[6] A. D. Wyner, "On source coding with side information at the decoder," *IEEE Trans. Inf. Theory*, vol. 21, no. 3, pp. 294–300, May 1975.

[7] S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction," *IEEE Trans. Inf. Theory*, vol. 49, no. 3, pp. 626–643, Mar. 2003.

[8] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proc. 2002 Data Comp. Conf.*, pp. 252–261.

[9] A. Liveris, Z. Xiong, and C. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 10, pp. 440–442, Oct. 2002.

[10] Y. Zhao and J. Garcia-Frias, "Joint estimation and compression of correlated nonbinary sources using punctured turbo codes," *IEEE Trans. Commun.*, vol. 53, no. 3, pp. 385–390, Mar. 2005.

[11] J. Hagenauer, J. Barros, and A. Schaefer, "Lossless turbo source coding with decremental redundancy," in *Proc. 2004 International ITG Conference on Source and Channel Coding*, vol. 181, pp. 333–339.

[12] D. Varodayan, A. Aaron, and B. Girod, "Rate-adaptive codes for distributed source coding," *Signal Process.*, vol. 86, no. 11, pp. 3123–3130, 2006.

[13] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret, "The DISCOVER codec: architecture, techniques and evaluation," in *Proc. 2007 Picture Coding Symposium*, vol. 17, no. 9, pp. 1103–1120, Nov. 2007.

[14] X. Zhu, A. Aaron, and B. Girod, "Distributed compression for large camera arrays," in *Proc. 2003 IEEE Workshop on Statistical Signal Processing*, pp. 30–33.

[15] F. Pereira, L. Torres, C. Guillemot, T. Ebrahimi, R. Leonardi, and S. Klomp, "Distributed video coding: selecting the most promising application scenarios," *Signal Process.: Image Commun.*, vol. 23, no. 5, pp. 339–352, Jun. 2008.

[16] P. L. Dragotti and M. Gastpar, *Distributed Source Coding: Theory, Algorithms and Applications*. Academic Press, 2009.

[17] Y.-C. Lin, D. Varodayan, and B. Girod, "Image authentication based on distributed source coding," in *Proc. 2007 IEEE Intl. Conf. Image Proc.*, vol. 3, pp. III–5–III–8.

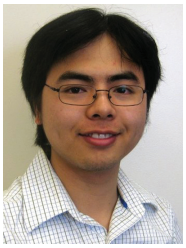
[18] D. Varodayan, "Adaptive distributed source coding," Ph.D. dissertation, Stanford University, Mar. 2010.

[19] F. Cen, "Design of degree distributions for LDPCA codes," *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 525–527, Jul. 2009.

[20] R. G. Gallager, *Low Density Parity Check Codes*. MIT Press, 1963.

[21] D. J. MacKay, *Information Theory, Inference, and Learning*

Algorithms. Cambridge University Press, 2003. Available: http://www.inference.phy.cam.ac.uk/mackay/itila/.
[22] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.
[23] ——, *Modern Coding Theory*. Cambridge University Press, 2008.
[24] S. Y. Chung, G. D. Forney Jr., T. J. Richardson, and R. L. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.
[25] R. M. Storn and K. V. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
[26] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.
[27] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.
[28] Y. Yang, S. Cheng, Z. Xiong, and W. Zhao, "Wyner-Ziv coding based on TCQ and LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 2, pp. 376–387, Feb. 2009.
[29] C. Yu and G. Sharma, "Distributed estimation and coding: a sequential framework based on a side-informed decomposition," *IEEE Trans. Signal Process.*, vol. 59, no. 2, pp. 759–773, Feb. 2011.



Chao Yu received his B.S. degree in electronic engineering from Tsinghua University, China, in 2004. He also received his M.S. and Ph.D. degree in electrical and computer engineering from University of Rochester, Rochester NY, in 2005 and 2013 respectively. His research interests lie in the area of signal and image/video processing, and specifically in distributed signal processing, coding, image/video compression and processing. Mr. Yu is a recipient of the best paper awards at the SPIE Visual Communications and Image Processing (VCIP) conference, San Jose, CA, 2009.

Gaurav Sharma is an associate professor at the University of Rochester in the Department of Electrical and Computer Engineering, in the Department of Biostatistics and Computational Biology, and in the Department of Oncology. From 2008–2010, he served as the Director for the Center for Emerging and Innovative Sciences (CEIS), a New York state funded center for promoting joint university-industry research and technology development, which is housed at the University of Rochester. He received the BE degree in electronics and communication engineering from Indian Institute of Technology Roorkee (formerly Univ. of Roorkee), India in 1990; the ME degree in electrical communication engineering from the Indian Institute of Science, Bangalore, India in 1992; and the MS degree in applied mathematics and PhD degree in electrical and computer engineering from North Carolina State University, Raleigh in 1995 and 1996, respectively. From Aug. 1996 through Aug. 2003, he was with Xerox Research and Technology, in Webster, NY, initially as a member of research staff and subsequently at the position of principal scientist.

Dr. Sharma's research interests include distributed signal processing, image processing, media security, and bioinformatics. He is the editor of the *Color Imaging Handbook*, published by CRC press in 2003. He is a fellow of the IEEE, of SPIE, and of the Society of Imaging Science and Technology (IS&T) and a member of Sigma Xi, Phi Kappa Phi, Pi Mu Epsilon, and the signal processing and communications societies of the IEEE. He served as a Technical Program Chair for the 2012 IEEE International Conference on Image Processing (ICIP), as the Symposium Chair for the 2012 SPIE/IS&T Electronic Imaging symposium, as the 2010-2011 Chair IEEE Signal Processing Society's Image Video and Multi-dimensional Signal Processing (IVMSP) technical committee, the 2007 chair for the Rochester section of the IEEE and the 2003 chair for the Rochester chapter of the IEEE Signal Processing Society. He is member of the IEEE Signal Processing Society's Information Forensics and Security (IFS) technical committee and an advisory member of the IEEE Standing committee on Industry DSP. He is the Editor-in-Chief for the *Journal of Electronic Imaging* and in the past has served as an associate editor for the *Journal of Electronic Imaging*, IEEE TRANSACTIONS ON IMAGE PROCESSING and IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY.