A Unified, SNR-Aware SC-LDPC Code Design with Applications to Magnetic Recording

Homa Esfahanizadeh*[1], Eshed Ram*[2], Yuval Cassuto[2], and Lara Dolecek[3]
[1]EECS Department, Massachusetts Institute of Technology (MIT), Cambridge, MA 02139 USA
[2]Andrew and Erna Viterbi Department of Electrical and Computer Engineering, Technion, Haifa, Israel
[3]ECE Department, University of California, Los Angeles (UCLA), Los Angeles, CA 90095 USA
*Equal contributions

Spatially-Coupled (SC)-LDPC codes are known to have outstanding error-correction performance and low decoding latency, which make them an excellent choice for high-density magnetic recording technologies. Whereas previous works on LDPC and SC-LDPC codes mostly take either an asymptotic or a finite-length design approach, we propose a unified framework for jointly optimizing the codes' thresholds and cycle counts to address both regimes. We focus on circulant-based SC-LDPC code family as a representative, high-performance exemplar of structured SC-LDPC codes. The framework is based on efficient traversal and pruning of the code search space, building on the fact that the performance of a circulant-based SC-LDPC code depends on some characteristics of the code's partitioning matrix, which by itself is much smaller than the code's full parity-check matrix. We then propose an algorithm that traverses all non-equivalent partitioning matrices, and outputs a list of codes, each offering an attractive point on the trade-off between asymptotic and finite-length performance. Our simulations show that our framework results in SC-LDPC codes that outperform the state-of-the-art constructions, over both Additive White Gaussian Noise (AWGN) and Partial Response (PR) channel models, and that it offers the flexibility to choose low-SNR, high-SNR, or in-between SNR region considering system requirements e.g, that of the magnetic recording device.

***Index Terms*—channel coding, partial-response (PR) channel, signal-to-noise ratio (SNR), spatially-coupled LDPC codes.**

I. Introduction

MODERN DENSE magnetic recording (MR) devices require low-latency error-correction codes with outstanding error-correction capability. The spatially-coupled low-density parity-check (SC-LDPC) codes are a family of graph-based codes that have attracted a lot of attention thanks to their capacity-approaching performance and low-latency decoding [2], [3]. SC-LDPC codes are constructed by coupling a series of disjoint block LDPC codes into a single coupled chain. We use circulant-based (CB) LDPC codes as the underlying LDPC block codes due to their practical simplicity [4]. SC-LDPC codes are known to have many desirable properties, such as threshold saturation [5] and linear-growth of the size of minimal trapping sets in typical codes from an ensemble [6]. These properties, respectively, imply good bit-error rate (BER) performance in the *waterfall* and *error floor* regions, using the belief-propagation (BP) decoder. Recently, [7] introduced SC-LDPC codes with a sub-block locality feature, where in addition to the usual decoding, the codes can be decoded *locally* in small sub-blocks for fast read access.

Modern graph-based codes are typically studied from either the asymptotic or from the finite-length perspective, with (so far) little intellectual overlap between them. From the asymptotic perspective, density evolution (DE) techniques have been routinely used to study the decoding threshold of SC-LDPC codes, e.g., in the works [5], [8], among others. From the finite-length perspective, methodologies for the evaluation and optimization of the number of problematic combinatorial objects are studied in e.g., [9]–[12]. The asymptotic properties (e.g., the decoding threshold) of LDPC codes are the dominant performance determinants in the low-SNR region, and the finite-length properties (e.g., the number of short cycles) are the dominant ones in the error-floor (high-SNR) region [13], [14]. This is because in the low-SNR region, the performance is typically dominated by the properties of the code's tree ensemble, while in the high-SNR region the performance critically depends on the incidence of problematic combinatorial objects in the code's graph [15]–[20].

In this paper, we pursue a principled comprehensive approach that combines both asymptotic and finite-length design metrics. We evaluate each candidate code in terms of its exact threshold and cycle-counts, and extract a small set of attractive codes from a large pool of initial candidates. This approach reflects its potential the most when we can examine each possible code given its design parameters. However, the space of possible candidates with various design parameters may blow up quickly, thus requiring efficient traversal and pruning methods. One particularly effective such method that we introduce in this paper is to quickly eliminate multiple candidates that are equivalent in terms of their performance. The elimination of this multiplicity reduces the candidate list – and in turn curbs the complexity of code design.

To enable the aforementioned comprehensive design approach, in this paper we formalize the notion of *performance-equivalent* SC-LDPC codes. It is known that two linear codes are equivalent in terms of most performance figures if one's parity-check matrix can be obtained from the other's by a sequence of row and column permutations. We prove that for SC-LDPC codes, the same property holds for the code's *partitioning matrix*, meaning two SC-LDPC codes are performance-wise equivalent if their partitioning matrices are row and/or column permuted versions of each other. The space of partitioning matrices is much smaller than the one for the full (coupled+lifted) code. This motivates our exact identification of non-equivalent binary matrices with unit memory. This identification can be easily translated to an efficient traversal of all non-equivalent matrices, thus enabling an efficient SC-

For smoother reading and space limitations, extensive discussions and additional examples appear in the extended pre-print of this manuscript [1].

LDPC code design. Next, we detail a joint threshold+cycle code-design algorithm, which outputs a final list of candidates with the property that each candidate has: 1) it possesses the best threshold among all codes with equal or better cycle-counts, and 2) and it possesses the best cycle-count among all codes with equal or better thresholds.

Our codes are shown to outperform prior constructions based on cutting-vector [21] and optimal-overlap partitioning [9] over Additive White Gaussian Noise (AWGN) and Partial Response (PR) channels. The software and data used in the paper are available for public access at https://github.com/hesfahanizadeh/Unified_SC_LDPCL/.

II. Preliminaries

Throughout this paper, matrices, vectors, and scalars are represented by uppercase bold letters (e.g., $\mathbf{A}$), lowercase italic letters with an overline (e.g., $\overline{a}$), and lowercase italic letters (e.g., a), respectively. Sets and functions are represented by calligraphic italic letters (e.g., $\mathcal{A}$) and uppercase italic letters (e.g., $A(\cdot)$), respectively. The matrix transpose operation, the cardinality of a set, and the factorial function are denoted by $(\cdot)^T$, $|\cdot|$, and $(\cdot)!$, respectively. The notation $\mathbf{A}=[a_{i,j}]$ refers to a matrix $\mathbf{A}$ where $a_{i,j}$ is the element in row i and column j. We denote the all-one and all-zero matrices with size $m\times n$ as $\mathbf{1}_{m\times n}$ and $\mathbf{0}_{m\times n}$, respectively.

A. Construction of SC-LDPC Codes

An LDPC protograph is a small bipartite graph represented by a $\gamma\times\kappa$ bi-adjacency proto-matrix $\mathbf{B}=[b_{i,j}]$ (where γ and κ are positive integers and $\gamma<\kappa$), i.e., there is an edge between check node (CN) i and variable node (VN) j if and only if $b_{i,j}=1$. In general, $b_{i,j}>1$ (parallel edges) are allowed in the protograph. In this work, without loss of generality[1], we focus on $b_{i,j}\in\{0,1\}$. A sparse parity-check matrix $\mathbf{H}$ (or its corresponding representation as a Tanner graph) is generated from $\mathbf{B}$ by a *lifting* operation with a positive integer z that is called the circulant size. The rows (resp. columns) of $\mathbf{H}$ corresponding to row $i\in\{1,\ldots,\gamma\}$ (resp. column $j\in\{1,\ldots,\kappa\}$) of $\mathbf{B}$, are called row group i (resp. column group j).

In this paper, we use *circulant-based* (CB) lifting [4], where the circulants, each with size $z\times z$, are either all-zero or an identity matrix shifted by a certain number of units to the left, described by the *circulant power*. The powers of the circulants are represented by the *power matrix* $\mathbf{C}=[c_{i,j}]$ of size $\gamma\times\kappa$, such that the non-zero elements in row group i and column group j in $\mathbf{H}$ form a single-shift identity matrix raised to the power $c_{i,j}$. In our simulations, the power matrix $\mathbf{C}=[c_{i,j}]$ is defined as $c_{i,j}=\alpha\cdot i\cdot j$, for a constant positive integer α. This choice ensures that no length-4 cycle (cycle-4 for short) exists when the circulant size is a prime number [22]. Thus, this paper focuses on length-6 cycles (cycles-6 for short) as the most problematic cycles. Further optimization of circulant powers to enhance the cycle properties is not a contribution of this paper, and more information can be found in [9], [10].

[1]Parallel edges can be avoided by duplicating protograph nodes.

Let proto-matrix $\mathbf{B}$ be the parity-check matrix of a protograph block code. The matrix of an SC-LDPC protograph [2] with memory m and coupling length l is constructed from $\mathbf{B}$ by partitioning it into $m+1$ matrices $\mathbf{B}_0,\ldots,\mathbf{B}_m$ such that $\mathbf{B}=\sum_{k=0}^{m}\mathbf{B}_k$, and stacking l *replicas* of $[\mathbf{B}_0;\mathbf{B}_1;\ldots;\mathbf{B}_m]$ (where ';' represents vertical concatenation here) on the diagonal of the coupled proto-matrix $\mathbf{B}_{\text{SC}}$. For proto-matrix $\mathbf{B}$ of size $\gamma\times\kappa$, the resulting coupled proto-matrix $\mathbf{B}_{\text{SC}}$ has size $(l+m)\gamma\times l\kappa$. We represent this partitioning by a matrix $\mathbf{P}=[p_{i,j}]$, called *partitioning matrix*, where $p_{i,j}\in\{0,1,\ldots,m,\star\}$. If $p_{i,j}=\star$, then there is a zero in row i and column j of $\mathbf{B}$. Otherwise, the non-zero element is assigned to $\mathbf{B}_{p_{i,j}}$. Some examples of partitioning matrices for $m=1$ SC-LDPC codes are given in Appendix B. This description captures both regular and irregular SC constructions. In this work, we focus on SC codes with $m=1$, thus the partitioning matrix determines which (non-zero) elements are assigned to $\mathbf{B}_0$ and which ones are assigned to $\mathbf{B}_1$ (when referring to lifted graphs, we use $\mathbf{H}_0$ and $\mathbf{H}_1$).

B. Asymptotic Analysis: The EXIT Method

The EXtrinsic Information Transfer (EXIT) method [23] is a useful tool for analyzing and designing LDPC codes in the asymptotic regime over the AWGN channel with the channel parameter σ. Let $J\colon[0,\infty)\to[0,1)$ be a function that represents the mutual information between the channel input and a corresponding message passing in the Tanner graph. For a VN of degree d_v in the protograph, with incoming EXIT values $\{J_i\}_{i=1}^{d_v-1}$, the VN→CN EXIT value is

$$J_{out}^{(V)}(s_{ch},J_1,\ldots,J_{d_v-1})=J\left(\sqrt{\sum_{i=1}^{d_v-1}\left(J^{-1}(J_i)\right)^2+s_{ch}^2}\right), \quad (1)$$

where $s_{ch}^2=4/\sigma^2$. For a CN of degree d_c in the protograph with incoming EXIT values $\{J_j\}_{j=1}^{d_c-1}$, the CN→VN EXIT value is approximated by

$$J_{out}^{(C)}(J_1,\ldots,J_{d_c-1})=1-J_{out}^{(V)}(0,1-J_1,\ldots,1-J_{d_c-1}). \quad (2)$$

The functions $J_{out}^{(V)}$ and $J_{out}^{(C)}$ are monotonically non decreasing with respect to all their arguments. In simulations, we use approximations of $J(\cdot)$ and $J^{-1}(\cdot)$ [23]. By alternately applying (1) and (2) for every edge in a protograph and by varying σ, a threshold value σ^* can be found such that all EXIT values on VNs approach 1 as the number of iterations increases if and only if $\sigma<\sigma^*$ [24]. We mark the threshold of a protograph $\mathbf{B}$ by $\sigma^*(\mathbf{B})$.

C. Finite-Length Analysis: Cycles and Overlap Parameters

Short cycles have a negative impact on the performance of block-LDPC and SC-LDPC codes under BP decoding: 1) they affect the independence of the messages exchanged on the graph, 2) they enforce upper-bounds on the minimum distance, and 3) they form combinatorial objects in the Tanner graphs that fail the iterative decoder in different known ways [9], [25].

Consider a binary matrix $\mathbf{B}$. A degree-d overlap parameter $t_{\{i_1,\ldots,i_d\}}$ is the number of columns in which all rows of

$\mathbf{B}$ indexed by $\{i_1, \ldots, i_d\}$ have 1s. The overlap parameters contain all the information we need to find the number of cycles in the graph represented by the matrix. We are particularly interested in cycles-6 (i.e., cycles with 6 nodes), as they are the shortest cycles for practical LDPC codes (most practical high-rate LDPC codes, in particular the codes in this paper, are designed to have girth at least 6). Consider a binary matrix $\mathbf{B}$ with γ rows and κ columns. The number of cycles-6 in the graph of matrix $\mathbf{B}$ can be expressed in terms of the overlap parameters of matrix $\mathbf{B}$ as follows:

$$F(\mathbf{B}) = \sum_{\{i_1,i_2,i_3\} \subseteq \{1,\ldots,\gamma\}} A(t_{\{i_1,i_2,i_3\}}, t_{\{i_1,i_2\}}, t_{\{i_1,i_3\}}, t_{\{i_2,i_3\}}), \tag{3}$$

where A is given in [9]. The optimization problem for identifying the optimal overlap parameters, and consequently the optimal partitioning, for designing SC-LDPC protographs with the minimum number of cycles-6 is presented in [9]. The approach is called the optimal overlap (OO) partitioning, and is one of the baselines to which we compare our empirical results.

III. Reducing Search Space: Non-Equivalent Binary Matrices

In this section, we explore the space of all possible binary matrices of a given size $\gamma \times \kappa$. In the next section, these binary matrices will correspond to partitioning matrices defining the SC-LDPC codes, but in the meantime, it will be instructive to think about these matrices as parity-check matrices of some protograph-based code (i.e., proto-matrices). We introduce a combinatorial representation that allows to significantly reduce the search space size by capturing the equivalence among the codes and only keeping one candidate from each class of equivalent codes. Equivalent codes are codes whose proto-matrices can be obtained from one another by a sequence of row and column permutations. This equivalence definition is motivated by the fact that permutation of rows and columns in proto-matrices affect neither the asymptotic threshold nor the number of cycles in the protograph. In Section IV, we theoretically prove that equivalence under this definition for partitioning matrices implies performance-equivalent SC-LDPC codes. By introducing a new technique for representing the binary matrices, we only consider one code from each equivalence class, thereby significantly reducing the search complexity.

Definition 1 (Column and Row Permutation)**.** *A column permutation of matrix $\mathbf{A}$ with κ columns is denoted by a vector $\overline{\pi}_c = [\rho_1, \ldots, \rho_\kappa]$, that is a permutation of the elements in the vector $[1, \ldots, \kappa]$. When $\mathbf{A} \xrightarrow{\overline{\pi}_c} \mathbf{A}'$, $\mathbf{A}'$ is obtained from $\mathbf{A}$ such that the j-th column of $\mathbf{A}$ is the ρ_j-th column of $\mathbf{A}'$. Similarly, a row permutation of matrix $\mathbf{A}$ with γ rows is denoted by a vector $\overline{\pi}_r = [\nu_1, \ldots, \nu_\gamma]$, that is a permutation of the elements in the vector $[1, \ldots, \gamma]$. When $\mathbf{A} \xrightarrow{\overline{\pi}_r} \mathbf{A}'$, $\mathbf{A}'$ is obtained from $\mathbf{A}$ such that the i-th row of $\mathbf{A}$ is the ν_i-th row of $\mathbf{A}'$.*

Definition 2 (Equivalent Matrices)**.** *Two binary matrices $\mathbf{A}$ and $\mathbf{A}'$ are column-wise equivalent (resp., row-wise equivalent) if they can be obtained by column (resp., row) permutations of each other. Two binary matrices $\mathbf{A}$ and $\mathbf{A}'$ are equivalent if they can be derived from each other by a sequence of row and column permutations.*

Remark 1. *We use the notion of equivalence in this paper to highlight that neither decoding threshold nor the number of combinatorial objects, e.g., cycles, absorbing sets [21], trapping sets [26], etc., change with a sequence of row and column permutations of a binary matrix.*

In this section, we present a new efficient combinatorial approach for identifying the non-equivalent binary matrices given their size. Consider a binary matrix $\mathbf{B}$ with size $\gamma \times \kappa$. There are 2^γ distinct choices for each column of $\mathbf{B}$, i.e., $[0, 0, \ldots, 0]^T$, $[0, 0, \ldots, 1]^T$, $\ldots$, $[1, 1, \ldots, 1]^T$. The set of all binary matrices with size $\gamma \times \kappa$ is of cardinality $2^{\gamma\kappa}$. In what follows, we show that the search space is effectively much smaller due to the equivalence among matrices. Our goal is to identify the set of non-equivalent binary matrices with γ rows and κ columns, denoted by $\mathcal{K}_{\kappa,\gamma}$, and to find a closed-form expression for its cardinality. This reduction, as we numerically verify, combined with an algorithm for iterating over non-equivalent binary matrices, allows a significantly more efficient optimization of LDPC protographs in terms of short cycles and thresholds

Definition 3 (Column Type)**.** *The type of a column of a binary matrix is defined as the decimal representation of the binary vector with the top element being the most significant bit.*

Definition 4 (Column Distribution)**.** *We associate with matrix $\mathbf{B} \in \{0,1\}^{\gamma\times\kappa}$ a vector $\overline{n}(\mathbf{B}) = [n_0, n_1, \ldots, n_{2^\gamma-1}]$ such that for every $i \in \{0, 1, \ldots, 2^\gamma - 1\}$, n_i is the number of* columns *in $\mathbf{B}$ with type i. We call $\overline{n}(\mathbf{B})$ the* column distribution *of $\mathbf{B}$, where the term stems from the fact that for every matrix $\mathbf{B} \in \{0,1\}^{\gamma\times\kappa}$, the entries of $\overline{n}(\mathbf{B})$ sum to κ, i.e., $\sum_{i=0}^{2^\gamma-1} n_i = \kappa$.*

Example 1. *Consider matrix*

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

Then, $\overline{n}(\mathbf{B}) = [0, 1, 1, 0, 0, 0, 1, 2]$.

Since column permutations do not change the column distribution of a matrix, we identify the number of column-wise non-equivalent matrices by counting the number of distinct column distributions. We note that a family of column-wise non-equivalent matrices can include row-wise equivalent matrices. At the same time, by excluding column-wise equivalent multiplicities, some row-wise equivalent multiplicities will also be excluded. For example, consider the 2×2 binary matrices: for matrix $[1\ 0; 1\ 1]$, no column permutation will lead to the row-permuted version $[1\ 1; 1\ 0]$; however, for matrix $[1\ 0; 0\ 1]$, swapping the columns will yield $[0\ 1; 1\ 0]$, i.e., swapping rows. The relation between families of column-wise non-equivalent matrices, denoted with $\mathcal{S}_{\kappa,\gamma}$, and non-equivalent matrices, denoted with $\mathcal{K}_{\kappa,\gamma}$, is not trivial, and how to derive the family of non-equivalent matrices is one contribution of this paper.

A. Column-Wise non-equivalent Binary Matrices

In this part, we explore the family of column-wise non-equivalent binary matrices, their connection to the stars-and-bars problem in combinatorics [27]. We also derive a closed-form expression for the number of column-wise non-equivalent matrices and describe how to efficiently iterate over them to evaluate their properties, e.g., threshold, cycle-counts, etc.

Lemma 1. *Let γ and κ be two positive integers, and let $\mathcal{S}_{\kappa,\gamma}$ be the set of all distinct column distributions for a $\gamma \times \kappa$ binary matrix, i.e.,*

$$\mathcal{S}_{\kappa,\gamma} = \left\{ [n_0, n_1, \ldots, n_{2^\gamma-1}] : n_i \geq 0, \sum_{i=0}^{2^\gamma-1} n_i = \kappa \right\}.$$

Then,

$$|\mathcal{S}_{\kappa,\gamma}| = \binom{\kappa + 2^\gamma - 1}{\kappa} = \binom{\kappa + 2^\gamma - 1}{2^\gamma - 1}. \tag{4}$$

Proof. The proof follows by applying the elementary stars-and-bars method [27], where each *bin* represents a column type, and the number of *stars* in bin i is n_i. □

We remind that $\mathcal{S}_{\kappa,\gamma}$ is the set of column-wise non-equivalent binary matrices of size $\gamma \times \kappa$. A recursive algorithm that iterates over all column-wise non-equivalent binary matrices is given in the long version of this paper [1, Algorithm 1].

Example 2. *Let $\kappa = 11$ and $\gamma = 3$. Then, there are $\binom{18}{7} = 31,824$ column-wise non-equivalent binary matrices with γ rows and κ columns, which is $2.7{\cdot}10^5$ times smaller then the entire space $\{0,1\}^{\gamma\kappa}$.*

In the next subsections, we further reduce the search space by taking into account the row permutations. In Lemma 1, we identified the set of distinct column distributions. It is somewhat challenging to calculate how many of these distributions result in equivalent matrices and thus can still be obtained from each other by a sequence of row and column permutations. In what follows, we complete this derivation, and to keep the analysis in this paper tractable, we derive the closed-form expressions only for $\gamma = 3$. This case gives a practically-important class of regular SC-LDPC codes with column weight 3 (the simpler case of $\gamma = 2$ appears in [1, Theorem 1].).

B. non-equivalent Binary Matrices with $\gamma = 3$

In the following lemma, we first state necessary and sufficient conditions for two column distributions to correspond to a pair of equivalent matrices. Then, in a subsequent theorem, we show how to use this lemma to reduce the search space of column distributions such that it consists only of those corresponding to non-equivalent matrices, and we identify the cardinality of this reduced search space.

Lemma 2. *Two binary matrices with $\gamma = 3$ and with column distributions $[n_0, n_1, \ldots, n_7]$ and $[m_0, m_1, \ldots, m_7]$ are equivalent if and only if $n_0 = m_0$, $n_7 = m_7$, and*

$$\begin{aligned}
&(n_1 = m_1, n_2 = m_2, n_4 = m_4, n_6 = m_6, n_5 = m_5, n_3 = m_3), \text{ or}\\
&(n_1 = m_1, n_2 = m_4, n_4 = m_2, n_6 = m_6, n_5 = m_3, n_3 = m_5), \text{ or}\\
&(n_1 = m_2, n_2 = m_1, n_4 = m_4, n_6 = m_5, n_5 = m_6, n_3 = m_3), \text{ or}\\
&(n_1 = m_2, n_2 = m_4, n_4 = m_1, n_6 = m_5, n_5 = m_3, n_3 = m_6), \text{ or}\\
&(n_1 = m_4, n_2 = m_1, n_4 = m_2, n_6 = m_3, n_5 = m_6, n_3 = m_5), \text{ or}\\
&(n_1 = m_4, n_2 = m_2, n_4 = m_1, n_6 = m_3, n_5 = m_5, n_3 = m_6).
\end{aligned}$$

Proof. For $\gamma = 3$, there are eight different column types and $3! = 6$ possible row permutations. The proof follows by tracking the changes in column types when applying each of the possible row permutations (from top to bottom): 1) $\overline{\pi}_r = [1,2,3]$ (identity permutation), 2) $\overline{\pi}_r = [2,1,3]$ 3) $\overline{\pi}_r = [1,3,2]$ 4) $\overline{\pi}_r = [2,3,1]$ 5) $\overline{\pi}_r = [3,1,2]$ 6) $\overline{\pi}_r = [3,2,1]$. □

The following theorem gives a concise characterization of non-equivalent matrices, leading to a closed-form count. The proof is deferred to the Appendix A.

Theorem 1. *Let $\gamma = 3$ and κ be a positive integer, and let $\mathcal{K}_{\kappa,3}$ be the set of column distributions for all $3 \times \kappa$ non-equivalent binary matrices. Then,*

$$\begin{aligned}
\mathcal{K}_{\kappa,3} = \Bigg\{ &[n_0, n_1, \ldots, n_7] \colon n_i \geq 0, \sum_{i=0}^{7} n_i = \kappa,\\
&(n_1 < n_2 < n_4) \text{ or}\\
&(n_1 = n_2 < n_4 \text{ and } n_6 \leq n_5) \text{ or}\\
&(n_1 < n_2 = n_4 \text{ and } n_5 \leq n_3) \text{ or}\\
&(n_1 = n_2 = n_4 \text{ and } n_6 \leq n_5 \leq n_3) \Bigg\},
\end{aligned} \tag{5}$$

and $|\mathcal{K}_{\kappa,3}| = a_\kappa + b_\kappa + c_\kappa$, where

$$\begin{aligned}
a_\kappa &= \sum_{\substack{i,j \in \mathbb{N}:\\ 3(i+j) \leq \kappa}} (\kappa - 3(i+j) + 1),\\
b_\kappa &= \sum_{\substack{i,j \in \mathbb{N}:\\ 2(i+j) \leq \kappa}} \binom{\kappa - 2(i+j) + 3}{3} - a_\kappa,\\
c_\kappa &= \frac{1}{6}\left(\binom{\kappa+7}{7} - 3b_\kappa - a_\kappa \right).
\end{aligned} \tag{6}$$

Example 3. *Let $\kappa = 11$ and $\gamma = 3$. Then, there are $|\mathcal{K}_{11,3}| = 60 + 1{,}452 + 4{,}568 = 6{,}080$ non-equivalent binary matrices with γ rows and κ columns, which is $1.41{\cdot}10^6$ times smaller then the cardinality of the entire space $\{0,1\}^{\gamma\kappa}$.*

We remind that $\mathcal{S}_{\kappa,\gamma}$ represents the set of $\gamma \times \kappa$ binary matrices that are column-wise non-equivalent, and $\mathcal{K}_{\kappa,\gamma}$ represents the set of $\gamma \times \kappa$ binary matrices that are non-equivalent (both column-wise and row-wise). In our code-design algorithm, we first iterate over the set of all distinct column distributions described in Lemma 1, and then exclude the options that do not satisfy the conditions in (5).

Example 4. *Fig. 1 shows the cardinality of the search space as a function of κ for $\gamma = 3$, using three different search schemes. As we see for the exhaustive search scheme (red)*

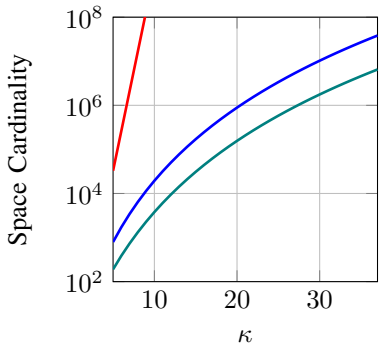

Fig. 1. The cardinality of the search space for binary matrices with $\gamma = 3$ rows: entire search space (red), reduced search space by exploiting the column-wise equivalence (blue), and reduced search space by exploiting both the row-wise equivalence and column-wise equivalence (green).

which performs on the full search space, the cardinality, i.e., $2^{\gamma\kappa}$, grows exponentially with κ and quickly goes beyond the practical feasibility. In contrast, our scheme (searching only non-equivalent matrices, green) exhibits much more graceful growth with κ. Also exploiting both pruning steps results in a significant gap (i.e., an order of magnitude) from the case when only the first step (pruning due to the column-wise equivalence, blue) is applied.

We finally note that evaluating each option in the search space, particularly identifying its decoding threshold, is computationally heavy and any reduction in the search space results in a reduction of the complexity of the design algorithm.

IV. An algorithm For Joint Finite-Length Asymptotic Design of SC Codes

In this section, we focus on partitioning matrices $\mathbf{P} \in \{0,1,\star\}^{\gamma\times\kappa}$ corresponding to SC-LDPC codes with memory 1, i.e., $\mathbf{B}_0+\mathbf{B}_1 = \mathbf{B}_{\gamma\times\kappa}$ (binary partitioning matrices without $\star$ symbols, as considered in Section II-A, correspond to all-one (regular) $\mathbf{B}$ matrices).

The derivations in this section can be generalized to regular/irregular $\mathbf{B}$ matrices with arbitrary memory $m \geq 1$, i.e., $\mathbf{P} \in \{\star, 0, \ldots, m\}^{\gamma\times\kappa}$. In the generalized case, the reduced search space in the previous section follows similarly by applying the stars-and-bars method with $(m+1)^\gamma$ bins, or $(m+2)^\gamma$ bins in case of irregular design, rather than 2^γ bins In what follows, given γ and κ, we produce a (short) list of partitioning matrices that offer a meaningful trade-off between threshold and cycle population. By meaningful we mean that no member of this list results in an SC protograph that is inferior to any other one (in the entire search space) in both the threshold and cycle-count properties.

A. Reduced Search Space of SC-LDPC Codes

Define $\mathbf{B}_0 = [u_{i,j}]$, $\mathbf{B}_1 = [v_{i,j}]$, and $\mathbf{P} = [p_{i,j}]$ where all with size $\gamma \times \kappa$, and $\mathbf{P}$ has elements in $\{0,1,\star\}$ that fully characterizes the construction of a regular/irregular SC code with memory 1 as follows: If $p_{i,j} = 0$, $u_{i,j} = 1$ and $v_{i,j} = 0$; If $p_{i,j} = 1$, $u_{i,j} = 0$ and $v_{i,j} = 1$; If $p_{i,j} = \star$, $u_{i,j} = v_{i,j} = 0$.

Lemma 3 derives the congruence between coupled proto-matrices and the partitioning matrices that are used to construct them. This congruence allows searching for a coupled SC-LDPC code over a reduced search space of small $(\gamma \times \kappa)$ non-equivalent partitioning matrices, instead of the large proto-matrices of the coupled code.

Lemma 3. *Any column/row permutation of the partitioning matrix $\mathbf{P}$ of an SC code results in an SC proto-matrix that is a column/row permuted version of the original SC proto-matrix $\mathbf{B}_{SC}$.*

Proof. By definition, any row and column permutations on $\mathbf{P}$ automatically applies to both $\mathbf{B}_0$ and $\mathbf{B}_1$. This means that when $\mathbf{P} \xrightarrow{\overline{\pi}_c,\overline{\pi}_r} \mathbf{P}'$, we have $\mathbf{B}_0 \xrightarrow{\overline{\pi}_c,\overline{\pi}_r} \mathbf{B}'_0$ and $\mathbf{B}_1 \xrightarrow{\overline{\pi}_c,\overline{\pi}_r} \mathbf{B}'_1$. Thus, the matrix $[\mathbf{B}'_1\ \mathbf{B}'_0]$ is a row-permuted version of $[\mathbf{B}_1\ \mathbf{B}_0]$ using $\overline{\pi}_r$, and the matrix $[\mathbf{B}'_0;\mathbf{B}'_1]$ is a column-permuted version of $[\mathbf{B}_0;\mathbf{B}_1]$ using $\overline{\pi}_c$. In view of the diagonal structure of $\mathbf{B}_{\text{SC}}$, we can infer that $\mathbf{B}'_{\text{SC}}$ which has $\mathbf{B}'_0$ and $\mathbf{B}'_1$ as component matrices is row and column permuted version of $\mathbf{B}_{\text{SC}}$, with column permutation $[\overline{\pi}_c, \overline{\pi}_c+\kappa, \ldots, \overline{\pi}_c+(l-1)\kappa]$ and row permutation $[\overline{\pi}_r, \overline{\pi}_r+\gamma, \ldots, \overline{\pi}_r+l\gamma]$, where the addition is performed element-wise. □

In our long version [1, Lemma 10], we show that the state-of-the-art approach of constructing SC-LDPC codes with overlap parameters [9] reduces the search space only up to column-wise equivalence, while leaving in the search space many row-wise equivalent matrices. Moreover, unlike the representation of column-wise non-equivalent matrices introduced in this paper, in the prior scheme, it is computationally difficult to iterate over all possible overlap parameters due to their dependencies.

B. An Algorithm For SC Code Design Offering A Design Trade-Off

We now present our code-design algorithm. The inputs of the algorithm are the code parameters γ, κ, z, l, and the output is a list of protographs such that no member in this list is inferior to any other protograph in the entire search space (we say that protograph $\mathcal{G}_1$ is inferior to protograph $\mathcal{G}_2$ if $\mathcal{G}_1$ has both lower threshold and larger number of cycles-6 than $\mathcal{G}_2$ in the corresponding lifted graph). This candidate list is often very short, and it is sorted such that its first member is the protograph with the best (lowest) cycle-count and the worst (lowest) threshold, and the last member has the best threshold and worst number of cycles.

Here, we focus on cycles-6, however, the approach presented in this subsection can be extended to longer cycles (and other problematic combinatorial objects) with some modifications. The methodology for counting the number of cycles and computing the decoding threshold considering SC-LDPC structure is described in detail in [1].

The algorithm consists of the following steps:

1) Generate a list of non-equivalent partitioning matrices of size $\gamma \times \kappa$ (which according to Lemma 3 corresponds to a list of non-equivalent SC proto-matrices).
2) Calculate the number of cycles-6 in the lifted coupled graphs corresponding to each partitioning matrix.

3) Sort the list in ascending order according to the number of cycles-6.
4) Iterate over the sorted list and for each partitioning matrix, generate the coupled protograph and calculate its decoding threshold.
5) Filter the list by removing inferior partitioning matrices using the following method:
   - Initialize the final list of candidates to be empty and set $\sigma^* = 0$ (σ^* records the highest found threshold).
   - Iterate, in order, over the sorted list. If a member has a higher threshold than σ^*, append the partitioning matrix to the final candidate-list and update σ^*.

Remark 2. *Although in this work we use CB lifting, one can easily use any other lifting method while keeping the general structure of the algorithm. For example, one can perform the cycle optimization of step 2 over the protograph (to obtain the minimum number of cycles-6 in the protograph) and then later use a lifting optimization program as in [9].*

The output of the above algorithm is a candidate list whose first member represents a choice that has the best cycle-count properties in the list, called cycle-driven (CD) choice, and the last member has the best threshold properties in the list, called threshold-driven (TD) choice.

V. Simulation Results

In our simulations, we consider parameters $\kappa = 11$, $z = 67$ $\gamma = 3$, $m = 1$, $l = 5$, and power matrix $C = [c_{i,j}]$ with $c_{i,j} = 6{\cdot}i{\cdot}j \pmod z$, which yields cycle-4 free graphs [28]. We investigate the performance of SC-LDPC codes constructed using four different design methods for $\mathbf{P} = [p_{i,j}]$ (the new introduced methods and existing methods).

- *Cutting-vector (CV) partitioning* [21]: This is partitioning via a cutting vector with size γ whose elements $0 < \zeta_1 < \ldots < \zeta_\gamma$ are natural numbers. Then, $p_{i,j} = 0$ if and only if $j < \zeta_i$. We consider the cutting vector $[4, 8, 11]$ for the simulations.
- *Optimal overlap (OO) partitioning* [9]: The OO partitioning results in the minimum number of cycles-6 in the protograph SC code.
- *Cycle-driven (CD) partitioning*: This is the partitioning within our reduced search space that results in the minimum number of cycles-6 in the *lifted* graph. This objective is set since short cycles in the *lifted* graph affect the performance in the high-SNR regime.
- *Threshold-driven (TD) partitioning*: This is the partitioning within our reduced search space that has the maximum threshold.

The partitioning matrices for all above constructions are given in Appendix B. We highlight that choosing a partitioning matrix that optimizes the cycle and threshold properties, *in the lifted graph*, have become a viable and practical option in our proposed constructions (CD and TD), thanks to the dramatic reduction in the search space. Since optimizing the circulant powers is not the focus of this paper, we used the same set of circulant powers for all four constructions described above. For Monte Carlo simulations, we observed at least 50 frame errors in every reported point. Our results include the BER performance over AWGN and PR channels, cycle-counts, threshold values.

TABLE I
Cycle and threshold properties of various design methods for SC-LDPC codes with $\kappa = 11$, $\gamma = 3$, $z = 67$, $m = 1$, $l = 5$. All codes have length 3685 bits.

Design	Actual rate	Cycle-6 count	Threshold σ^*
Cutting vector (CV) [21]	0.69	7,638	0.6779
Optimal overlap (OO) [9]	0.67	7,571	0.6901
Cycle-driven (CD)	0.67	3,551	0.6851
Threshold-driven (TD)	0.67	5,628	0.6909

We first record the populations of cycles-6 along with the threshold values for the four constructions of the SC-LDPC codes. The results are given in Table I, where it is shown that the CD method yields 54% reduction in the population of cycles-6 (in lifted graphs) compared to the CV method, while the OO method only improves this count by less than 1%. In terms of the asymptotic behavior, the TD method results in the highest threshold while also having fewer number of cycles compared to the CV and OO methods.

Fig. 2 compares the BER performance for these SC-LDPC codes over AWGN channel. The top sub-figure shows the BER performance in the low-SNR region and in particular the superiority of the TD partitioning with about half an order of magnitude compared to the CV partitioning at SNR$=2.5$ dB. The bottom sub-figure shows the BER performance in the high-SNR region and the superiority of the CD partitioning with about one order of magnitude compared to the CV partitioning at SNR$=5$ dB. Moreover, there is a crossover point at SNR $\simeq 3$ dB, where the BER performance curves of CD and TD methods intersect.

We also perform experiments over the partial response (PR) channel. We use a similar PR setting as in our previous work [29], which is briefly reviewed here: Our PR setting includes a magnetic recording channel model that incorporates inter-symbol interference in addition to transition jitter noise and electronic noise. The channel density is set to 1.4, and the equalization target used is $[8, 14, 2]$. The message is iteratively recovered via a min-sum LDPC decoding algorithm in addition to Bahl Cocke Jelinek Raviv (BCJR) detector based on pattern-dependent noise prediction. The internal iterations inside the LDPC decoder are called local iterations, while a global iteration is the one looping between the detector and the decoder. The decoder performs a specified number of local iterations (fewer if a codeword is reached) between any two successive global iterations. We use 20 global iterations and 200 local iterations for our simulations.

Fig. 3 shows the BER comparison between three SC-LDPC code constructions: cutting vector (CV), optimal overlap (OO), and cycle-driven (CD), over PR channel model with two different levels of transition jitter noise. Since our proposed threshold-driven (TD) construction optimizes the threshold for AWGN channel, we did not incorporate it in our BER evaluation over PR channel. We see that, while all having the same latency and rate, our CD construction enjoys about one order of magnitude performance improvement at SNR$=14$ dB

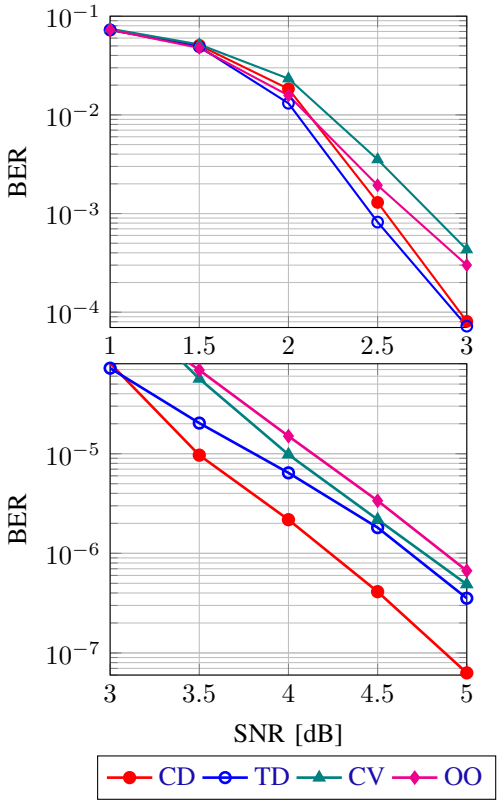

Fig. 2. BER performance over AWGN channel of various code constructions for SC-LDPC codes with parameters $\kappa = 11$, $z = 67$, $\gamma = 3$, $m = 1$, and $l = 5$. Top: low-SNR region; Bottom: high-SNR region.

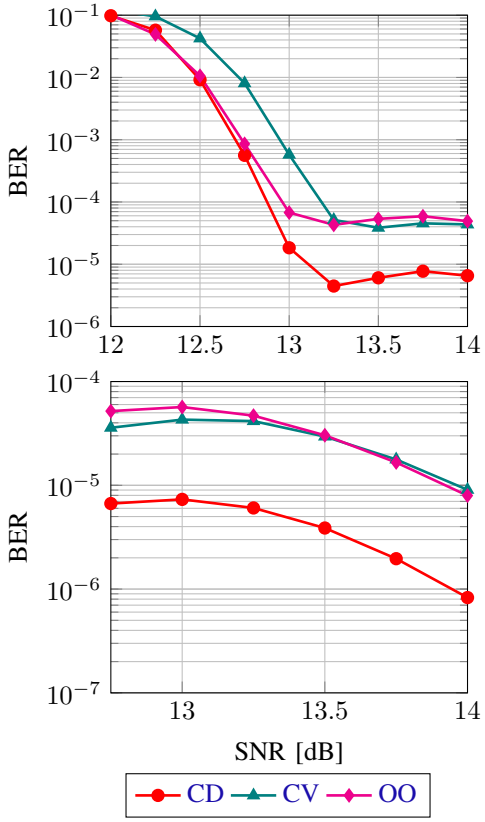

Fig. 3. BER performance over PR channel of various code constructions for SC-LDPC codes with parameters $\kappa = 11$, $z = 67$, $\gamma = 3$, $m = 1$, and $l = 5$. The transition jitter is stronger in the top panel (i.e., 70%) compared to the bottom panel (i.e., 50%)

thanks to the dramatically lower number of cycles-6 compared to the CV and OO constructions in the lifted graph. This observation is supported by the fact that short cycles are sub-graphs of the detrimental combinatorial objects over PR channels [29]. The deeper error floor that is observed in the lower panel is due to lower level of the jitter noise.

Fig. 3 highlights the advantage of the proposed optimization: improving the reliability of the storage device by an order of magnitude, without any additional cost in the encoding and decoding procedure. One can further improve the performance of the code by changing the code parameters, such as increasing the row weight κ, column weight γ, increasing the field size, among others.

VI. Conclusion

In this paper, we proposed a novel framework to reduce the search space of block LDPC and SC-LDPC codes via only keeping one member from a family of equivalent matrices that share identical finite-length and asymptotic metrics (cycles-6 and thresholds, respectively). Then, we proposed a design method that identifies all constructions that offer a trade-off between finite-length and asymptotic performances in this reduced search space. Our simulation results verify our theoretical derivations and show the improved outstanding performance and flexibility of the codes designed using our method over AWGN and PR channels. Beyond the promising results shown here, it is an interesting future-work direction to use the method to construct state-of-the-art codes for commercial magnetic-recording channels. In such pursuits, one can also incorporate additional constraints over the search space of non-equivalent matrices introduced in this paper, e.g., all columns must be at least of certain weight.

VII. Acknowledgements

Research supported in part by a grant from ASRC-IDEMA, and in part by a grant from the Israel Science Foundation. The authors would like to thank Lev Tauz and Debarnab Mitra for their assistance in carrying the PR experiments.

References

[1] H. Esfahanizadeh, E. Ram, Y. Cassuto, and L. Dolecek, "A unified spatially coupled code design: Threshold, cycles, and locality," http://arxiv.org/abs/2203.02052.

[2] A. J. Felstrom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2181–2191, Sep. 1999.

[3] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[4] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, "LDPC block and convolutional codes based on circulant matrices," *IEEE Transactions on Information Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.

[5] S. Kudekar, T. Richardson, and R. L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 7761–7813, Dec. 2013.

[6] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, "AWGN channel analysis of terminated LDPC convolutional codes," in *Proc. Information Theory and Applications Workshop (ITA)*, La Jolla, CA, Feb. 2011, pp. 1–5.
[7] E. Ram and Y. Cassuto, "Spatially coupled LDPC codes with sub-block locality," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2739–2757, 2021.
[8] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.
[9] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "Finite-length construction of high performance spatially-coupled codes via optimized partitioning and lifting," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 3–16, Jan. 2019.
[10] A. Hareedy, H. Esfahanizadeh, A. Tan, and L. Dolecek, "Spatially-coupled code design for partial-response channels: Optimal object-minimization approach," in *IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–7.
[11] D. G. M. Mitchell and E. Rosnes, "Edge spreading design of high rate array-based SC-LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2017, pp. 2940–2944.
[12] A. Beemer and C. A. Kelley, "Avoiding trapping sets in SC-LDPC codes under windowed decoding," in *Proc. IEEE Int. Symp. Inf. Theory and Its Applications*, Oct. 2016, pp. 206–210.
[13] B. K. Butler and P. H. Siegel, "Error floor approximation for LDPC codes in the awgn channel," *IEEE Transactions on Information Theory*, vol. 60, no. 12, pp. 7416–7441, 2014.
[14] E. Sharon and S. Litsyn, "Generating good finite length LDPC codes based on lifted graphs," *Proc. of the 44'th Allerton Conf. on Communication Control and Computing*, pp. 1–10, 09 2006.
[15] B. Vasić, S. K. Chilappagari, and D. V. Nguyen, *Failures and Error Floors of Iterative Decoders*. Elsevier Inc., june 2014, pp. 299–341.
[16] C. Di, D. Proietti, I. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1570–1579, 2002.
[17] A. Tomasoni, S. Bellini, and M. Ferrari, "Thresholds of absorbing sets in low-density parity-check codes," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3238–3249, 2017.
[18] C. Di, D. Proietti, I. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1570–1579, 2002.
[19] M. Karimi and A. H. Banihashemi, "An efficient algorithm for finding dominant trapping sets of irregular LDPC codes," in *IEEE International Symposium on Information Theory Proceedings*, 2011, pp. 1091–1095.
[20] T. Richardson, "Error floors of LDPC codes," in *41th Annual Allerton Conference on Communication, Control and Computing*, Oct. 2003, pp. 1426–1435.
[21] D. G. M. Mitchell, L. Dolecek, and D. J. Costello, "Absorbing set characterization of array-based spatially coupled LDPC codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Honolulu, HI, Jun. 2014, pp. 886–890.
[22] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 181–201, 2010.
[23] S. Ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 670–678, Apr. 2004.
[24] G. Liva and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," in *Proc. IEEE Global Telecommunications Conference (GLOBCOM)*, Washington, DC, Nov. 2007, pp. 3250–3254.
[25] R. Smarandache and P. O. Vontobel, "Quasi-cyclic LDPC codes: Influence of proto- and Tanner-graph structure on minimum hamming distance upper bounds," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 585–607, Feb. 2012.
[26] M. Karimi and A. H. Banihashemi, "On characterization of elementary trapping sets of variable-regular LDPC codes," *IEEE Transactions on Information Theory*, vol. 60, no. 9, pp. 5188–5203, 2014.
[27] R. P. Stanley, *Enumerative Combinatorics: Volume 1*, 2nd ed. USA: Cambridge University Press, 2011.
[28] J. L. Fan, "Array codes as low-density parity-check codes," in *Proc. International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Brest, France, Sep. 2000, pp. 543–546.
[29] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "Spatially coupled codes optimized for magnetic recording applications," *IEEE Transactions on Magnetics*, vol. 53, no. 2, pp. 1–11, 2017.

APPENDIX

A. Proof of Theorem 1

The first part is a direct consequence of Lemma 2 that ensures the distinct column distributions of equivalent matrices are only considered once by imposing appropriate constraints. For the second part, we find the number of distinct non-equivalent binary matrices with $\gamma = 3$ rows. In order to do so, we partition the column distributions in $\mathcal{S}_{\kappa,3}$ into three classes:

- Class-A column distributions $S^A_{\kappa,3}$: column distributions that are invariant to any row permutation. In other words, for a matrix with column distribution in Class-A, all (3! out of 3!) row permutations of the matrix results in the same column distribution.
- Class-B column distributions $S^B_{\kappa,3}$: column distributions that are invariant to permutation of one pair of rows. In other words, for a matrix with column distribution in Class-B, $3!/2! = 3$ row permutations exist that result in distinct column distributions in Class-B.
- Class-C column distributions $S^C_{\kappa,3}$: column distributions that are variant to any row permutation. In other words, for a matrix with column distribution in Class-A, all $3! = 6$ row permutations of the matrix result in distinct column distributions in Class-C.

Consequently,

$$|\mathcal{K}_{\kappa,3}| = |S^A_{\kappa,3}| + |S^B_{\kappa,3}|/3 + |S^C_{\kappa,3}|/6. \tag{7}$$

We first identify $S^A_{\kappa,3}$ as follows:

$$S^A_{\kappa,3} = \{[n_0, n_1, n_2, n_4, n_6, n_5, n_3, n_7] \in \mathcal{S}_{\kappa,3} : n_1 = n_2 = n_4, n_6 = n_5 = n_3\}.$$

Thus,

$$S^A_{\kappa,3} = \bigcup_{\substack{i,j\in\mathbb{N}:\\ 3(i+j)\le\kappa}} \{[n_0, n_1, n_2, n_4, n_6, n_5, n_3, n_7] \in \mathcal{S}_{\kappa,3} : n_1 = n_2 = n_4 = i, n_6 = n_5 = n_3 = j\},$$

where the union is disjoint. In view of Lemma 1,

$$\begin{aligned} \left|S^A_{\kappa,3}\right| &= \sum_{\substack{i,j\in\mathbb{N}:\\ 3(i+j)\le\kappa}} \left|\{[n_0, n_7] : n_0 + n_7 = \kappa - 3(i+j)\}\right| \\ &= \sum_{\substack{i,j\in\mathbb{N}:\\ 3(i+j)\le\kappa}} (\kappa - 3(i+j) + 1) = a_\kappa. \end{aligned} \tag{8}$$

Next, we identify $S^B_{\kappa,3}$. We define $\mathcal{T}^{1,2}_{\kappa,3}$ as the set of column distributions that are invariant to swapping the first and second rows (from the bottom):

$$\mathcal{T}^{1,2}_{\kappa,3} = \{[n_0, n_1, n_2, n_4, n_6, n_5, n_3, n_7] \in \mathcal{S}_{\kappa,3} : n_1 = n_2, n_6 = n_5\},$$

and similarly, $\mathcal{T}^{1,3}_{\kappa,3}$ and $\mathcal{T}^{2,3}_{\kappa,3}$ can be defined. Note that $S^A_{\kappa,3}$ is a subset of $\mathcal{T}^{1,2}_{\kappa,3}$, $\mathcal{T}^{1,3}_{\kappa,3}$, and $\mathcal{T}^{2,3}_{\kappa,3}$. Therefore,

$$S^B_{\kappa,3} = (\mathcal{T}^{1,2}_{\kappa,3} \setminus S^A_{\kappa,3}) \cup (\mathcal{T}^{1,3}_{\kappa,3} \setminus S^A_{\kappa,3}) \cup (\mathcal{T}^{2,3}_{\kappa,3} \setminus S^A_{\kappa,3}).$$

Because of the disjoint property and since $|\mathcal{T}_{\kappa,3}^{1,2} \setminus S_{\kappa,3}^{A}| = |\mathcal{T}_{\kappa,3}^{1,3} \setminus S_{\kappa,3}^{A}| = |\mathcal{T}_{\kappa,3}^{2,3} \setminus S_{\kappa,3}^{A}|$ (due to the symmetry), we have $|S_{\kappa,3}^{B}| = 3|\mathcal{T}_{\kappa,3}^{1,2} \setminus S_{\kappa,3}^{A}| = 3(|\mathcal{T}_{\kappa,3}^{1,2}| - a_\kappa)$. Besides

$$\mathcal{T}_{\kappa,3}^{1,2} = \bigcup_{\substack{i,j\in\mathbb{N}:\\ 2(i+j)\leq\kappa}} \{[n_0,n_1,n_2,n_3,n_4,n_5,n_4,n_7] \in \mathcal{S}_{\kappa,3} : n_1 = n_2 = i, n_5 = n_6 = j\},$$

where the union is disjoint. In view of Lemma 1,

$$\left|\mathcal{T}_{\kappa,3}^{1,2}\right| = \sum_{\substack{i,j\in\mathbb{N}:\\ 2(i+j)\leq\kappa}} |\{[n_0,n_3,n_4,n_7] \colon n_0+n_3+n_4+n_7 = \kappa - 2(i+j)\}| = \sum_{\substack{i,j\in\mathbb{N}:\\ 2(i+j)\leq\kappa}} \binom{\kappa-2(i+j)+3}{3}.$$

Thus, the number column distributions in Class-B is:

$$|\mathcal{S}_{\kappa,3}^{B}| = 3\left(\sum_{\substack{i,j\in\mathbb{N}:\\ 2(i+j)\leq\kappa}} \binom{\kappa-2(i+j)+3}{3} - A_\kappa\right) = 3b_\kappa. \tag{9}$$

Finally, we identify $\mathcal{S}_{\kappa,3}^{C}$, i.e., column distributions that are variant to any permutations. We remind that the total number of column distributions is $|\mathcal{S}_{\kappa,3}|$, a_k of them belong to Class-A, and $3b_k$ of them belong to Class-B. As a result,

$$|\mathcal{S}_{\kappa,3}^{C}| = \binom{\kappa+7}{7} - a_k - 3b_k = 6c_k. \tag{10}$$

Combining (7)-(10) completes the proof. □

B. SC-LDPC Code Details

In this subsection, we provide the partitioning matrices of the four SC-LDPC codes that were used in our simulation results in Section V:

- Cutting-vector (CV) partitioning:

$$\mathbf{P} = \begin{pmatrix} 0&0&0&0&1&1&1&1&1&1&1\\ 0&0&0&0&0&0&0&0&1&1&1\\ 0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix}$$

- Optimal overlap (OO) partitioning:

$$\mathbf{P} = \begin{pmatrix} 0&0&0&0&0&1&1&1&1&1&1\\ 1&1&1&1&1&0&0&0&0&0&0\\ 0&0&1&1&1&0&0&0&1&1&1 \end{pmatrix}$$

- Cycle-driven (CD) partitioning:

$$\mathbf{P} = \begin{pmatrix} 1&1&1&1&1&0&0&0&0&0&0\\ 1&1&0&0&0&1&1&0&0&0&0\\ 1&1&1&0&0&1&1&1&1&0&0 \end{pmatrix}$$

- Threshold-driven (TD) partitioning:

$$\mathbf{P} = \begin{pmatrix} 1&1&1&1&1&1&0&0&0&0&0\\ 1&1&0&0&0&0&1&1&1&1&0\\ 0&0&1&1&0&0&1&1&0&0&1 \end{pmatrix}$$