Breaking the Computational Bottleneck: Probabilistic Optimization of High-Memory Spatially-Coupled Codes

Siyi Yang, *Member, IEEE*, Ahmed Hareedy, *Member, IEEE*, Robert Calderbank, *Life Fellow, IEEE*, and Lara Dolecek, *Senior Member, IEEE*

***Abstract*— Spatially-coupled (SC) codes, known for their threshold saturation phenomenon and low-latency windowed decoding algorithms, are ideal for streaming applications and data storage systems. SC codes are constructed by partitioning an underlying block code, followed by rearranging and concatenating the partitioned components in a convolutional manner. The number of partitioned components determines the memory of SC codes. In this paper, we investigate the relation between the performance of SC codes and the density distribution of partitioning matrices. While adopting higher memories results in improved SC code performance, obtaining finite-length, high-performance SC codes with high memory is known to be computationally challenging. We break this computational bottleneck by developing a novel probabilistic framework that obtains (locally) optimal density distributions via gradient descent. Starting from random partitioning matrices abiding by the obtained distribution, we perform low-complexity optimization algorithms that minimize the number of detrimental objects to construct high-memory, high-performance quasi-cyclic SC codes. We apply our framework to various objects of interest, from the simplest short cycles, to more sophisticated objects such as concatenated cycles aiming at finer-grained optimization. Simulation results show that codes obtained through our proposed method notably outperform state-of-the-art SC codes with the same constraint length and optimized SC codes with uniform partitioning. The performance gain is shown to be universal over a variety of channels, from canonical channels such as additive white Gaussian noise and binary symmetric channels, to practical channels underlying flash memory and magnetic recording systems.**

***Index Terms*— LDPC codes, spatially-coupled codes, absorbing sets, edge distribution, gradient descent, near-optimal partitioning, data storage, flash memories, magnetic recording, communications.**

Manuscript received 18 September 2021; revised 17 July 2022; accepted 5 September 2022. Date of publication 16 September 2022; date of current version 20 January 2023. This work was supported in part by the University of California, Los Angeles (UCLA) Dissertation Year Fellowship, in part by the Air Force Office of Scientific Research (AFOSR) under Grant FA 9550-20-1-0266, and in part by the National Science Foundation (NSF) under Grant CCF-FET 2008728 and Grant CCF 2106213. An earlier version of this paper was presented at the 2021 IEEE International Symposium on Information Theory (ISIT) [DOI: 10.1109/ISIT45174.2021.9517931]. *(Corresponding author: Siyi Yang.)*

Siyi Yang was with the University of California at Los Angeles, Los Angeles, CA 90095 USA. She is now with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: siyi.yang@duke.edu).

Ahmed Hareedy is with the Department of Electrical and Electronics Engineering, Middle East Technical University (METU), 06800 Ankara, Turkey (e-mail: ahareedy@metu.edu.tr).

Robert Calderbank is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: robert.calderbank@duke.edu).

Lara Dolecek is with the Department of Electrical and Computer Engineering, University of California at Los Angeles, Los Angeles, CA 90095 USA (e-mail: dolecek@ee.ucla.edu).

Communicated by D. Mitchell, Associate Editor for Coding Theory.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3207321.

Digital Object Identifier 10.1109/TIT.2022.3207321

I. Introduction

SPATIALLY-COUPLED (SC) codes, also known as low-density parity-check (LDPC) codes with convolutional structures, are an ideal choice for streaming applications and data storage devices thanks to their threshold saturation phenomenon [2], [3], [4], [5], [6] and amenability to low-latency windowed decoding [7]. SC codes are constructed by partitioning the parity-check matrix of an underlying block code, followed by rearranging the component matrices in a *convolutional* manner. In particular, component matrices are vertically concatenated into a *replica*, and then multiple replicas are horizontally placed together, resulting in a *coupled* code. The number of component matrices minus one is referred to as the *memory* of the SC code [8], [9], [10], [11].

It is known that the performance of an SC code improves as its memory increases. This is a byproduct of improved node expansion and additional degrees of freedom that can be utilized to decrease the number of short cycles and detrimental objects [9], [10], [12], [13], [14]. Mitchell and Rosnes demonstrated that SC codes designed using non-contiguous partitioning outperform codes designed using contiguous partitioning in [15], which was also demonstrated in [16]. A plethora of existing works [9], [10], [13], [17], [18], [19], [20] focus on minimizing the number of short cycles in the graph of the SC code. Although the optimization problem of designing SC codes with memory less than 4 has been efficiently solved [9], [10], there is still an absence of efficient algorithms that construct good enough SC codes with high memories systematically. Esfahanizadeh *et al.* [9] proposed a combinatorial framework to develop optimal quasi-cyclic (QC) SC codes, comprising so-called optimal overlap (OO) to search for the optimal partitioning matrices, and lifting optimization (CPO) to optimize the lifting parameters, which was extended by Hareedy *et al.* [10]. However, this method is hard to execute in practice for high-memory codes due to the increasing computational complexity. Efficient algorithms designing SC

codes with high memories have been investigated in [13], [17], [18], and [20]. Naseri. *et al.* [13] and Battaglioni *et al.* [17] presented lower bounds for the constraint lengths that enable SC codes free of small cycles. Beemer *et al.* [20] considered both partitioning (edge spreading in the paper) and lifting in tail-biting SC codes, which lead to QC-SC codes and allow the enumeration and elimination of small cycles efficiently, starting from small base matrices. Mo *et al.* [18] proposed to perform optimization over partitioning and lifting in a two-stage fashion, yielding a greater flexibility in designing QC-SC codes with a large girth. Codes designed in the aforementioned literature are typically based on enumeration of detrimental objects of interest followed by algorithmic optimization that eliminates these objects, e.g., exhaustive search and integer programming in [17] that removes short cycles, and a limited exhaustive search with iterative backtracking in [20] that enumerate $(3, 3)$-absorbing sets in regular VN degree 3 codes. A more advanced technique based on the parent/child relationship between trapping sets (TSs) that efficiently enumerates TSs by expanding short cycles and successively removes the most harmful ones was devised in [13]. Exclusive focus on algorithmic optimization, i.e., lack of theoretical guidance, in the design of SC codes in existing works can lead to search spaces that are away from the optimal solution; several of the codes produced by these works can be outperformed by optimally designed QC-SC codes with lower memories under the same constraint length [18]. We therefore establish a theoretically supported step that systematically targets an optimal search space for the algorithmic elimination step coming afterwards, universally reducing the expected number of different objects of interest and leading to better performance.

Inspired by the excellent performance and the low computational complexity offered by approaches comprising theoretical analysis guiding heuristic methods, such as degree distribution optimization followed by progressive edge-growth (PEG) algorithms in designing irregular codes, we propose a two-step hybrid optimization framework that has these advantages. The framework first specifies a search subspace that is theoretically proved to be locally optimal, followed by a semi-greedy algorithm within this targeted search space. By analogy with threshold optimization approaches that search for LDPC ensembles with the optimal degree distribution, our first step is to obtain an SC ensemble with the optimal edge distribution (i.e., density distribution of component matrices). The associated metric is the expected number of targeted detrimental objects in the protograph of the code. Having reached a locally optimal edge distribution through gradient descent, we then apply a semi-greedy algorithm to search for a locally optimal partitioning matrix that satisfies this edge distribution. Our probabilistic framework is referred to as **gradient-descent distributor, algorithmic optimizer (GRADE-AO)**.

Preliminary version of this work was presented in [1], where we focused only on the minimization of the number of short cycles. In this work, we develop a broadly applicable framework that handles more sophisticated objects. While cycles are detrimental in codes with low variable node (VN) degrees, such as regular codes with VN degree 2 or 3, objects that dominate the error profiles in higher-degree codes and irregular codes are typically more advanced. In particular, we focus on the concatenation of two short cycles in this paper. These concatenated cycles are common subgraphs of the detrimental objects, which are absorbing sets (ASs) [12], that govern the performance of LDPC codes with VN degree ≥ 3 in error floor region. These detrimental objects are also the major source of undesirable dependencies that undermine the performance in the waterfall region. While focusing on cycles for simplicity, which is the case for the majority of existing works, unnecessary degrees of freedom could be exhausted on isolated cycles that are much less problematic. Prior work aiming at eliminating objects instead of cycles in binary codes, which is important for a variety of applications including storage systems, includes [14] by Naseri and Banihashemi along with [21] by Hareedy *et al.*. While [14] adopts a greedy algorithm to remove the TSs and [21] focuses on a different class of codes, the multi-dimensional LDPC codes, our framework offers additional mathematical guidance and leads to systematic design of high-performance SC codes that do not floor over the storage channels. Hareedy *et al.* [22], [23] proposed the so-called weight consistency matrix (WCM) framework to search for edge-weight assignments that minimize the number of ASs in non-binary (NB) LDPC codes with a given underlying topology, and demonstrated performance gains in data storage systems. Simulation results show that our framework leads to codes with excellent performance in flash memory and magnetic recording systems. The proposed GRADE-AO framework not only opens a door to fine-grained optimization over detrimental objects in SC codes, but also can be applied in optimizing the unweighted graphs of non-binary (NB) SC codes, which can lead to excellent NB-SC codes when combined with the WCM framework. Because of the improved threshold and waterfall performance, GRADE-AO has potential to produce SC codes for communication systems as well.

In this paper, we propose a probabilistic framework that efficiently searches for near-optimal SC codes with high memories. In Section II, we introduce preliminaries of SC codes and the performance-related metrics. In Section III, we develop the theoretical basis of GRADE, which derives an edge distribution that determines a locally optimal SC ensemble. In Section IV, we introduce the theoretical details of how GRADE is generalized to more sophisticated objects. The distribution obtained through GRADE leads to effective initialization and specifies the search space of the semi-greedy algorithm adopted in AO afterwards. In Section V, we introduce two examples of GRADE-AO that result in near-optimal SC codes: the so-called **gradient-descent (GD) codes** and **topologically-coupled (TC) codes**. In summary, we generalize GRADE to a rich class of relevant objects and present examples of GRADE-AO that focus on concatenated cycles. Our proposed framework is supported in Section VI by simulation results of seven groups of codes, with the best code in each obtained from GRADE-AO. Finally, we make concluding remarks and introduce possible future work in Section VII.

II. Preliminaries

In this section, we recall the typical construction of SC codes with quasi-cyclic (QC) structure. Any QC code with a

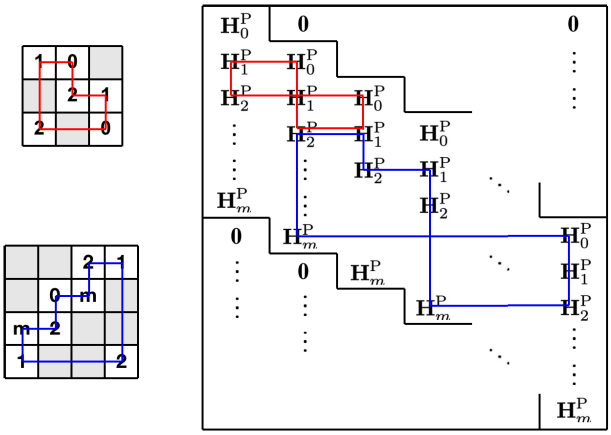


Fig. 1. Cycle candidates in the protograph (right panel) and their corresponding cycle candidates in the partitioning matrices (left panel).

parity-check matrix $\mathbf{H}$ is obtained by replacing each nonzero (zero) entry of some binary matrix $\mathbf{H}^{\mathrm{P}}$ with a circulant (zero) matrix of size z, $z \in \mathbb{N}$. The matrix $\mathbf{H}^{\mathrm{P}}$ and z are referred to as the protograph and the circulant size of the code, respectively. In particular, the protograph $\mathbf{H}^{\mathrm{P}}_{\mathrm{SC}}$ of an SC code has a convolutional structure composed of L replicas, as presented in Fig. 1. Each replica is obtained by stacking the disjoint component matrices $\{\mathbf{H}^{\mathrm{P}}_i\}_{i=0}^{m}$, where m is the memory and $\mathbf{\Pi} = \mathbf{H}^{\mathrm{P}}_0 + \mathbf{H}^{\mathrm{P}}_1 + \cdots + \mathbf{H}^{\mathrm{P}}_m$ is the protograph of the underlying block code.

In this paper, we constrain $\mathbf{\Pi}$ to be an all-one matrix of size $\gamma \times \kappa$, $\gamma, \kappa \in \mathbb{N}$. An SC code is then uniquely represented by its partitioning matrix $\mathbf{P}$ and lifting matrix $\mathbf{L}$, where $\mathbf{P}$ and $\mathbf{L}$ are all $\gamma \times \kappa$ matrices. The matrix $\mathbf{P}$ has $(\mathbf{P})_{i,j} = a$ if $(\mathbf{H}^{\mathrm{P}}_a)_{i,j} = 1$. The matrix $\mathbf{L}$ is determined by replacing each circulant matrix by its associated exponent. Here, this exponent represents the power to which the matrix $\boldsymbol{\sigma}$ defined by $(\boldsymbol{\sigma})_{i,i+1} = 1$ is raised, where $(\boldsymbol{\sigma})_{z,z+1} = (\boldsymbol{\sigma})_{z,1}$. The number of replicas is referred to as the *coupling length* and denoted by L. Under this notation, the parameters of an SC code are described by $(\gamma, \kappa, m, z, L)$.

The performance of finite-length LDPC codes is strongly affected by the number of detrimental objects that are subgraphs with certain structures in the Tanner graphs of those codes. Two major classes of detrimental objects are trapping sets and absorbing sets. Since enumerating and minimizing the number of detrimental objects is complicated, existing work typically focuses on common substructures of these objects: the short cycles [9], [10], [17]. A cycle-$2g$ candidate in $\mathbf{H}^{\mathrm{P}}_{\mathrm{SC}}$ ($\mathbf{\Pi}$) is a path of traversing a structure to generate cycles of length $2g$ after lifting (partitioning) [10]. In an SC code, each cycle in the Tanner graph corresponds to a cycle candidate in the protograph $\mathbf{H}^{\mathrm{P}}_{\mathrm{SC}}$, and each cycle candidate in $\mathbf{H}^{\mathrm{P}}_{\mathrm{SC}}$ corresponds to a cycle candidate C in the base matrix $\mathbf{\Pi}$. Lemma 1 specifies a necessary and sufficient condition for a cycle candidate in $\mathbf{\Pi}$ to become a cycle candidate in the protograph and then a cycle in the final Tanner graph.

Lemma 1: Let C be a cycle-$2g$ candidate in the base matrix, where $g \in \mathbb{N}$, $g \geq 2$. Denote C by $(j_1, i_1, j_2, i_2, \ldots, j_g, i_g)$, where (i_k, j_k), (i_k, j_{k+1}), $1 \leq k \leq g$, $j_{g+1} = j_1$, are nodes of C in $\mathbf{\Pi}$, $\mathbf{P}$, and $\mathbf{L}$. Then C becomes a cycle candidate in the protograph if and only if the following condition follows [17]:

$$\sum\nolimits_{k=1}^{g} \mathbf{P}(i_k, j_k) = \sum\nolimits_{k=1}^{g} \mathbf{P}(i_k, j_{k+1}). \tag{1}$$

This cycle candidate becomes a cycle in the Tanner graph if and only if [24]:

$$\sum\nolimits_{k=1}^{g} \mathbf{L}(i_k, j_k) \equiv \sum\nolimits_{k=1}^{g} \mathbf{L}(i_k, j_{k+1}) \mod z. \tag{2}$$

As shown in Fig. 1, a cycle-6 candidate and a cycle-8 candidate in the partitioning matrix with assignments satisfying condition (1), and their corresponding cycle candidates in the protograph are marked by red and blue, respectively. An optimization of a QC-SC code is typically divided into two major steps: optimizing $\mathbf{P}$ to minimize the number of cycle candidates in the protograph, and optimizing $\mathbf{L}$ to further reduce that number in the Tanner graph given the optimized $\mathbf{P}$ [9], [10]. The latter goal has been achieved in [9] and [10], using an algorithmic method called lifting optimization (CPO), while the former goal is yet to be achieved for large m. We note that the step separation highlighted above notably reduces the overall optimization complexity.

In the remainder of this paper, we first focus on QC-SC codes for the additive white Gaussian noise (AWGN) channel, where the most detrimental objects are the low weight absorbing sets (ASs) [9]. The ASs are defined in Definition 1.

Definition 1 (Absorbing Sets): Consider a subgraph induced by a subset $\mathcal{V}$ of VNs in the Tanner graph of a code. The set $\mathcal{V}$ is said to be an (a, b) **absorbing set (AS)** over GF(q) if the size of $\mathcal{V}$ is a, the number of unsatisfied neighboring CNs of $\mathcal{V}$ is b, and each VN in $\mathcal{V}$ is connected to strictly more satisfied than unsatisfied neighboring CNs, for some set of VN values in GF(q) $\setminus \{0\}$, assuming that the values of all other VNs in the Tanner graph are zeros.

An (a, b) **elementary AS** $\mathcal{V}$ over GF(q) is an (a, b) AS with the additional property that all the satisfied (resp., unsatisfied (if any)) neighboring CNs of $\mathcal{V}$ have degree 2 (resp., degree 1); otherwise the AS is referred to as an (a, b) **non-elementary AS**.

Consider a subgraph induced by a subset $\mathcal{V}$ of VNs in the Tanner graph of a binary code. The set $\mathcal{V}$ is said to be an (a, b) **binary AS** if the size of $\mathcal{V}$ is a, the number of odd-degree neighboring CNs of $\mathcal{V}$ is b, and each VN in $\mathcal{V}$ is connected to strictly more even-degree than odd-degree neighboring CNs.

Observe that the unlabeled configuration (all edge weights set to 1) underlying an (a, b) non-binary elementary AS is itself an (a, b) binary elementary AS.

Consequently, a simplified optimization focuses on cycle candidates of lengths 4, 6, and 8 [9], [10]. Existing literature, e.g., [9], that searches for the optimal $\mathbf{P}$ for an SC code with $m \leq 2$ typically assumes balanced (uniform) edge distribution among component matrices for reducing the complexity. However, in the remaining sections, we show that the edge distribution for optimal SC codes with large m is not uniform, and we propose the GRADE-AO framework that explores a locally optimal solution. With the success in cycle optimization, we step forward to a finer-grained optimization over more advanced objects, which can be applied in higher degree codes and irregular codes. While GRADE can be

generalized for arbitrary objects, we present constructions obtained though GRADE-AO focusing on concatenated cycles. Simulation results show that our proposed codes have excellent performance on practical channel models derived from flash memories and magnetic recording (MR), in both the waterfall and error floor regions.

III. A Probabilistic Optimization Framework

In this section, we present a probabilistic framework that searches for a locally optimal edge distribution for the partitioning matrices of SC codes with given memories through the gradient-descent algorithm.

Definition 2: Let $\gamma, \kappa, m, m_t \in \mathbb{N}$ and $\mathbf{a} = (a_0, a_1, \ldots, a_{m_t})$, where $0 = a_0 < a_1 < \cdots < a_{m_t} = m$. A (γ, κ) SC code with memory m is said to have **coupling pattern** $\mathbf{a}$ if and only if $\mathbf{H}_i^{\mathrm{P}} \neq \mathbf{0}^{\gamma \times \kappa}$, for all $i \in \{a_0, a_1, \ldots, a_{m_t}\}$, and $\mathbf{H}_i^{\mathrm{P}} = \mathbf{0}^{\gamma \times \kappa}$, otherwise. The value m_t is called the **pseudo-memory** of the SC code.

A. Probabilistic Metric

In this subsection, we define metrics relating the edge distribution to the expected number of cycle candidates in the protograph in Theorem 1 and Theorem 2. While Schmalen *et al.* have shown in [25] that nonuniform coupling (nonuniform edge distribution in our paper) yields an improved threshold, our work differs in two areas: 1) Explicit optimal coupling graphs were exhaustively searched and were restricted to small memories in [25], whereas our method produces near-optimal SC protographs for arbitrary memories. 2) Work [25] focused on the asymptotic analysis for the threshold region, while our framework is dedicated to the finite-length construction and has additional demonstrable gains in the error floor region.

Definition 3: Let $m, m_t \in \mathbb{N}$ and $\mathbf{a} = (a_0, a_1, \ldots, a_{m_t})$, where $0 = a_0 < a_1 < \cdots < a_{m_t} = m$. Let $\mathbf{p} = (p_0, p_1 \ldots, p_{m_t})$, where $0 < p_i \leq 1$, $p_0 + p_1 + \cdots + p_{m_t} = 1$: each p_i specifies the probability of a '1' in $\mathbf{\Pi}$ going to the component matrix $\mathbf{H}_{a_i}^{\mathrm{P}}$, thus $\mathbf{p}$ is referred to as **edge distribution** under random partition later on. Then, the following $f(X; \mathbf{a}, \mathbf{p})$, which is abbreviated to $f(X)$ when the context is clear, is called the **coupling polynomial** of an SC code with coupling pattern $\mathbf{a}$, associated with probability distribution $\mathbf{p}$:

$$f(X; \mathbf{a}, \mathbf{p}) \triangleq \sum\nolimits_{0 \leq i \leq m_t} p_i X^{a_i}. \tag{3}$$

Theorem 1: Let $[\cdot]_i$ denote the coefficient of X^i of a polynomial. Denote by $P_6(\mathbf{a}, \mathbf{p})$ the probability of a cycle-6 candidate in the base matrix becoming a cycle-6 candidate in the protograph under random partitioning with edge distribution $\mathbf{p}$. Then,

$$P_6(\mathbf{a}, \mathbf{p}) = \left[f^3(X) f^3(X^{-1}) \right]_0. \tag{4}$$

Proof: According to Lemma 1, suppose the cycle-6 candidate in the base matrix is represented by $C(j_1, i_1, j_2, i_2, j_3, i_3)$. Then,

$$\begin{aligned}
P_6(\mathbf{a}, \mathbf{p}) &= \mathbb{P}\left[\sum\nolimits_{k=1}^{3} \mathbf{P}(i_k, j_k) = \sum\nolimits_{k=1}^{3} \mathbf{P}(i_k, j_{k+1}) \right] \\
&= \sum_{\sum_{k=1}^{3} x_k = \sum_{k=1}^{3} y_k} \prod_{k=1}^{3} \mathbb{P}\left[\mathbf{P}(i_k, j_k) = x_k, \mathbf{P}(i_k, j_{k+1}) = y_k \right] \\
&= \sum_{\sum_{k=1}^{3} x_k = \sum_{k=1}^{3} y_k} p_{x_1} p_{x_2} p_{x_3} p_{y_1} p_{y_2} p_{y_3} \\
&= \left[\sum_{x_k, y_k \in \mathrm{vals}(\mathbf{a})} p_{x_1} p_{x_2} p_{x_3} p_{y_1} p_{y_2} p_{y_3} X^{x_1 + x_2 + x_3 - y_1 - y_2 - y_3} \right]_0 \\
&= \left[f^3(X) f^3(X^{-1}) \right]_0,
\end{aligned}$$

where $\mathrm{vals}(\mathbf{a})$ is the set $\{a_0, a_1, \ldots, a_{m_t}\}$. Thus, the theorem is proved. ∎

A similar approach of adopting generating functions to assist deriving the probability of breaking a cycle-6 was also presented the [26, Theorem 3.4.1], for the special case of a uniform edge distribution.

Example 1: Consider SC codes with full memories and uniform partition, i.e., $\mathbf{a} = (0, 1, \ldots, m)$ and $\mathbf{p} = \frac{1}{m+1}\mathbf{1}_{m+1}$. When $m = 2$, $P_6(\mathbf{a}, \mathbf{p}) = 0.1934$; when $m = 4$, $P_6(\mathbf{a}, \mathbf{p}) = 0.1121$.

Example 2: First, consider SC codes with $m = m_t = 2$. Let $\mathbf{a}_1 = (0, 1, 2)$ and $\mathbf{p}_1 = (2/5, 1/5, 2/5)$. According to Theorem 1, $f(X) = (2 + X + 2X^2)/5$, $f^3(X) f^3(X^{-1}) = 0.0041(X^6 + X^{-6}) + 0.0123(X^5 + X^{-5}) + 0.0399(X^4 + X^{-4}) + 0.0717(X^3 + X^{-3}) + 0.1267(X^2 + X^{-2}) + 0.1544(X + X^{-1}) + 0.1818$. Therefore, $P_6(\mathbf{a}_1, \mathbf{p}_1) = 0.1818$. Second, consider SC codes with $m = m_t = 4$. Let $\mathbf{a}_2 = (0, 1, 2, 3, 4)$ and $\mathbf{p}_2 = (0.31, 0.13, 0.12, 0.13, 0.31)$. According to Theorem 1, $P_6(\mathbf{a}_2, \mathbf{p}_2) = 0.0986$.

After we have derived the metric for cycle-6 candidates in the protograph, we now turn to the case of cycle-8 candidates. As shown in Fig. 2, cycle candidates in the base matrix that result in cycle-8 candidates in the protograph can be categorized into 6 different structures, labeled $S_1, \ldots, S_6$. Different cases are differentiated by the number of rows and columns (without order) the structures span in the partitioning matrix [10]. Specifically, $S_1, \ldots, S_6$ denote the structures that span submatrices of size 2×2, 2×3 or 3×2, 3×3, 2×4 or 4×2, 3×4 or 4×3, and 4×4, respectively. Any structure that belongs to S_2, S_4, S_5 has multiple cycle-8 candidates, and these distinct candidates are marked by blue in Fig. 2.

Lemma 2: Let $P_{8;i}(\mathbf{a}, \mathbf{p})$, $1 \leq i \leq 6$, denote the probability of a cycle-8 candidate of structure S_i in the base matrix becoming a cycle-8 candidate in the protograph, under random partition with edge distribution $\mathbf{p}$. Then,

$$\begin{aligned}
P_{8;1}(\mathbf{a}, \mathbf{p}) &= \left[f^2(X) f^2(X^{-1}) \right]_0, \\
P_{8;2}(\mathbf{a}, \mathbf{p}) &= \left[f(X^2) f(X^{-2}) f^2(X) f^2(X^{-1}) \right]_0, \\
P_{8;3}(\mathbf{a}, \mathbf{p}) &= \left[f(X^2) f^2(X) f^4(X^{-1}) \right]_0, \text{ and} \\
P_{8;4}(\mathbf{a}, \mathbf{p}) &= P_{8;5}(\mathbf{a}, \mathbf{p}) = P_{8;6}(\mathbf{a}, \mathbf{p}) = \left[f^4(X) f^4(X^{-1}) \right]_0.
\end{aligned}$$

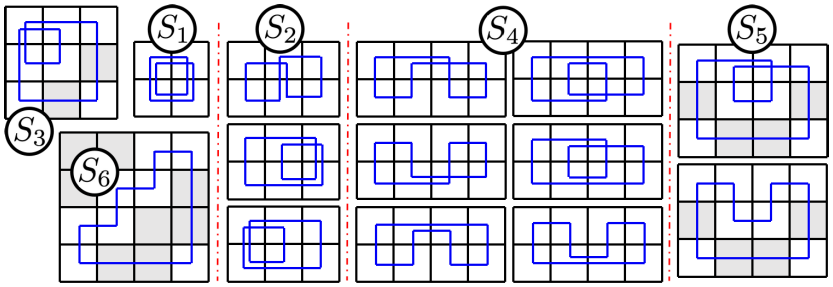


Fig. 2. Structures and cycle candidates for cycle-8. Note that structures S_i, $i \in \{2, 4, 5\}$, also include the transpose of each cycle candidate shown in the figure, which are omitted for simplicity. Permutations of rows and/or columns give equivalent arrangements.

Proof: For structures where the nodes of the cycle-8 candidates are pairwise different, namely, S_4, S_5, S_6, the result can be derived by following the logic in the proof of Theorem 1.

For S_1, suppose the indices of the rows and columns are i_1, i_2, and j_1, j_2, respectively. Then, the cycle condition in Lemma 1 is $\mathbf{P}(i_1, j_1) + \mathbf{P}(i_2, j_2) = \mathbf{P}(i_1, j_2) + \mathbf{P}(i_2, j_1)$.

For S_2, suppose the indices of the rows and columns are i_1, i_2, and j_1, j_2, j_3, respectively. Then, the cycle condition in Lemma 1 is $2\mathbf{P}(i_1, j_1) - 2\mathbf{P}(i_2, j_1) + \mathbf{P}(i_2, j_2) + \mathbf{P}(i_2, j_3) - \mathbf{P}(i_1, j_2) - \mathbf{P}(i_1, j_3) = 0$.

For S_3, suppose the indices of the rows and columns are i_1, i_2, i_3, and j_1, j_2, j_3, respectively. Then, the cycle condition in Lemma 1 is $2\mathbf{P}(i_1, j_1) + \mathbf{P}(i_2, j_2) + \mathbf{P}(i_3, j_3) - \mathbf{P}(i_1, j_2) - \mathbf{P}(i_2, j_1) - \mathbf{P}(i_1, j_3) - \mathbf{P}(i_3, j_1) = 0$.

Following the logic in the proof of Theorem 1, the case for S_1, S_2, S_3 can be proved. ∎

Theorem 2: Denote $N_8(\mathbf{a}, \mathbf{p})$ as the expectation of the number of cycle-8 candidates in the protograph. Then,

$$\begin{aligned} N_8(\mathbf{a}, \mathbf{p}) =& w_1 \left[f^2(X) f^2(X^{-1})\right]_0 \\ &+ w_2 \left[f(X^2) f(X^{-2}) f^2(X) f^2(X^{-1})\right]_0 \\ &+ w_3 \left[f(X^2) f^2(X) f^4(X^{-1})\right]_0 \\ &+ w_4 \left[f^4(X) f^4(X^{-1})\right]_0, \end{aligned} \tag{5}$$

where $w_1 = \binom{\gamma}{2}\binom{\kappa}{2}$, $w_2 = 3\binom{\gamma}{2}\binom{\kappa}{3} + 3\binom{\gamma}{3}\binom{\kappa}{2}$, $w_3 = 18\binom{\gamma}{3}\binom{\kappa}{3}$, $w_4 = 6\binom{\gamma}{2}\binom{\kappa}{4} + 6\binom{\gamma}{4}\binom{\kappa}{2} + 36\binom{\gamma}{3}\binom{\kappa}{4} + 36\binom{\gamma}{4}\binom{\kappa}{3} + 24\binom{\gamma}{4}\binom{\kappa}{4}$.

Proof: Provided the results in Lemma 2, we just need to prove that the numbers of cycle candidates of structures $S_1, S_2, \ldots, S_6$ in a $\gamma \times \kappa$ base matrix are $\binom{\gamma}{2}\binom{\kappa}{2}$, $3\binom{\gamma}{2}\binom{\kappa}{3} + 3\binom{\gamma}{3}\binom{\kappa}{2}$, $18\binom{\gamma}{3}\binom{\kappa}{3}$, $6\binom{\gamma}{2}\binom{\kappa}{4} + 6\binom{\gamma}{4}\binom{\kappa}{2}$, $36\binom{\gamma}{3}\binom{\kappa}{4} + 36\binom{\gamma}{4}\binom{\kappa}{3}$, and $24\binom{\gamma}{4}\binom{\kappa}{4}$, respectively.

Take $i = 5$ as an example. We first count the number of cycle candidates of structure S_5 in any 3×4 (4×3) matrix. We first choose the only row (column) that contains four edges of S_5, which has 3 ways in a 3×4 (4×3) matrix. Then, we choose the edge combination of the other two rows (columns), which has $\binom{4}{2}$ ways. There are 2 different ways to connect the four edges in the remaining row. Therefore, the number is $3 \cdot \binom{4}{2} \cdot 2 = 36$. The total number of 3×4 or 4×3 matrices in a $\gamma \times \kappa$ base matrix is $\binom{\gamma}{3}\binom{\kappa}{4} + \binom{\gamma}{4}\binom{\kappa}{3}$. Therefore, the total number of cycle candidates of structure S_5 is $36\binom{\gamma}{3}\binom{\kappa}{4} + 36\binom{\gamma}{4}\binom{\kappa}{3}$. By a similar logic, we can prove the result for the remaining structures. ∎

Remark 1: Note that each cycle candidate satisfying the cycle condition in the partitioning matrix can result in multiple cycle candidates in the protograph; the multiplicative factor is determined by its width, i.e., the number of replicas each resultant cycle candidate spans in the protograph. We ignore the number of replicas a cycle candidate spans in $\mathbf{H}_{\text{SC}}^{\text{P}}$. This is because $L \gg m$ typically holds (for small rate loss), which means that the multiplicative factor of each cycle-6 and each cycles-8 are in the ranges $[L - m, L]$ and $[L - 2m, L]$, respectively. Therefore, the deviation of the number of cycles derived while ignoring the multiplicative factor from the accurate number is upper bounded by a fraction of $O\left(\frac{m}{L}\right)$, which approaches zero when $L \gg m$, and its effect is negligible on the constructions obtained from our optimization framework. We address this number in the CPO stage.

B. Gradient-Descent Distributor

By contrasting Examples 1 and 2 it is clear that for a given coupling pattern, an optimal edge distribution is not necessarily reached by a uniform partition. In this subsection, we develop an algorithm that obtains a locally optimal distribution by gradient descent.

Lemma 3: Given $m_t \in \mathbb{N}$ and $\mathbf{a} = (a_0, a_1, \ldots, a_{m_t})$, a necessary condition for $P_6(\mathbf{a}, \mathbf{p})$ to reach its minimum value is that the following equation holds for some $c_0 \in \mathbb{R}$:

$$\left[f^3(X) f^2(X^{-1})\right]_{a_i} = c_0, \ \forall i, 0 \le i \le m_t. \tag{6}$$

Proof: Our objective is to minimize $P_6(\mathbf{a}, \mathbf{p})$, under the constraint $p_0 + p_1 + \cdots + p_{m_t} = 1$. Therefore, we consider the Lagrangian $L_6(\mathbf{a}, \mathbf{p}) = P_6(\mathbf{a}, \mathbf{p}) + c(1 - p_0 - p_1 - \cdots - p_{m_t})$ and compute its gradient.

$$\begin{aligned} &\nabla_{\mathbf{p}} L_6(\mathbf{a}, \mathbf{p}) \\ =& \nabla_{\mathbf{p}} \left(P_6(\mathbf{a}, \mathbf{p}) + c(1 - p_0 - p_1 - \cdots - p_{m_t})\right) \\ =& \nabla_{\mathbf{p}} \left[f^3(X) f^3(X^{-1})\right]_0 - c\mathbf{1}_{m_t+1} \\ =& \left[\nabla_{\mathbf{p}} \left(f^3(X) f^3(X^{-1})\right)\right]_0 - c\mathbf{1}_{m_t+1} \\ =& 3\left[f^2(X) f^2(X^{-1}) f(X) \nabla_{\mathbf{p}} f(X^{-1})\right]_0 \\ &+ 3\left[f^2(X) f^2(X^{-1}) f(X^{-1}) \nabla_{\mathbf{p}} f(X)\right]_0 - c\mathbf{1}_{m_t+1} \\ =& 6\left[f^3(X) f^2(X^{-1}) \left(X^{-a_0}, X^{-a_1}, \ldots, X^{-a_{m_t}}\right)\right]_0 - c\mathbf{1}_{m_t+1}. \end{aligned} \tag{7}$$

When $P_6(\mathbf{a}, \mathbf{p})$ reaches its minimum, $\nabla_{\mathbf{p}} \left[L_6(\mathbf{a}, \mathbf{p})\right] = \mathbf{0}_{m_t+1}$, which is equivalent to (6) by defining $c_0 = c/6$. ∎

Lemma 4: Given $\gamma, \kappa, m_t \in \mathbb{N}$ and $\mathbf{a} = (a_0, a_1, \ldots, a_{m_t})$, a necessary condition for $N_8(\mathbf{a}, \mathbf{p})$ to reach its minimum value

is that the following equation holds for some $c_0 \in \mathbb{R}$:

$$\begin{aligned}&\left[4f^2(X)f(X^{-1})\right])_{a_i} + \bar{w}_2\left[2f(X^2)f^2(X)f^2(X^{-1})\right]_{2a_i}\\ &+\bar{w}_2\left[4f(X^2)f(X^{-2})f^2(X)f(X^{-1})\right]_{a_i}\\ &+\bar{w}_3\left[f^2(X)f^4(X^{-1})\right]_{-2a_i}+\bar{w}_3\left[2f(X^2)f(X)f^4(X^{-1})\right]_{-a_i}\\ &+\bar{w}_3\left[4f(X^2)f^2(X)f^3(X^{-1})\right]_{a_i}\\ &+\bar{w}_4\left[8f^4(X)f^3(X^{-1})\right]_{a_i} = c_0,\ \forall i, 0 \le i \le m_t,\end{aligned}$$

where $\bar{w}_2 = \gamma + \kappa - 4$, $\bar{w}_3 = 2(\gamma-2)(\kappa-2)$, and $\bar{w}_4 = \frac{1}{2}\left[(\gamma-2)(\gamma-3)+(\kappa-2)(\kappa-3)\right]+(\gamma-2)(\kappa-2)(\gamma+\kappa-6)+\frac{1}{6}(\gamma-2)(\gamma-3)(\kappa-2)(\kappa-3)$.

Proof: Consider the gradient of $L_8(\mathbf{a},\mathbf{p}) = N_8(\mathbf{a},\mathbf{p}) + c(1-p_0-p_1-\cdots-p_{m_t})$.

$$\begin{aligned}&\nabla_{\mathbf{p}} L_8(\mathbf{a},\mathbf{p}) = \nabla_{\mathbf{p}}\left(N_8(\mathbf{a},\mathbf{p})+c(1-p_0-p_1-\cdots-p_{m_t})\right)\\ &= w_1\left[\nabla_{\mathbf{p}}\left(f^2(X)f^2(X^{-1})\right)\right]_0\\ &\quad + w_2\left[\nabla_{\mathbf{p}}\left(f(X^2)f(X^{-2})f^2(X)f^2(X^{-1})\right)\right]_0\\ &\quad + w_3\left[\nabla_{\mathbf{p}}\left(f(X^2)f^2(X)f^4(X^{-1})\right)\right]_0\\ &\quad + w_4\left[\nabla_{\mathbf{p}}\left(f^4(X)f^4(X^{-1})\right)\right]_0 - c\mathbf{1}_{m_t+1}\\ &= w_1\{\left[4f^2(X)f(X^{-1})(X^{-a_0},X^{-a_1},\ldots,X^{-a_{m_t}})\right])_0\\ &\quad + \bar{w}_2\left[2f(X^2)f^2(X)f^2(X^{-1})(X^{-2a_0},\ldots,X^{-2a_{m_t}})\right]_0\\ &\quad + \bar{w}_2\left[4f(X^2)f(X^{-2})f^2(X)f(X^{-1})(X^{-a_0},\ldots,X^{-a_{m_t}})\right]_0\\ &\quad + \bar{w}_3\left[f^2(X)f^4(X^{-1})(X^{2a_0},X^{2a_1},\ldots,X^{2a_{m_t}})\right]_0\\ &\quad + \bar{w}_3\left[2f(X^2)f(X)f^4(X^{-1})(X^{a_0},X^{a_1},\ldots,X^{a_{m_t}})\right]_0\\ &\quad + \bar{w}_3\left[4f(X^2)f^2(X)f^3(X^{-1})(X^{-a_0},X^{-a_1},\ldots,X^{-a_{m_t}})\right]_0\\ &\quad + \bar{w}_4\left[8f^4(X)f^3(X^{-1})(X^{-a_0},X^{-a_1},\ldots,X^{-a_{m_t}})\right]_0\}\\ &\quad - c\mathbf{1}_{m_t+1}. \qquad (8)\end{aligned}$$

When $P_8(\mathbf{a},\mathbf{p})$ reaches its minimum, $\nabla_{\mathbf{p}}\left[L_8(\mathbf{a},\mathbf{p})\right] = \mathbf{0}_{m_t+1}$, which is equivalent to (8) by defining $c_0 = c/w_1$. ■

Based on Lemma 3 and Lemma 4, we adopt the gradient-descent algorithm to obtain a locally optimal edge distribution for SC codes with coupling pattern $\mathbf{a}$, starting from the uniform distribution inside $\mathbf{P}$ as presented in Algorithm 1. Note that $\mathrm{conv}(\cdot)$ and $\mathrm{flip}(\cdot)$ refer to convolution and reverse of vectors, respectively. The weight 1 of cycles-8 is a normalized weight, while we let w to be the weight of cycles-6. Say if cycles-8 and cycles-6 are of weights w_8 and w_6, respectively, then it is equivalent to the case where their weights are 1 and $w = w_6/w_8$, respectively. Typically, cycles of shorter lengths are more problematic than cycles of larger lengths, thus we always choose $w > 1$ in the optimization algorithm.

The total number of cycle-6 candidates in the base matrix is $6\binom{\gamma}{3}\binom{\kappa}{3}$, thus the expected number of cycle-6 candidates in the protograph is $N_6(\mathbf{a},\mathbf{p}) = w_1\left(6\binom{\gamma}{3}\binom{\kappa}{3}P_6(\mathbf{a},\mathbf{p})\right)$. Our objective is to minimize the weighted sum $wN_6(\mathbf{a},\mathbf{p}) + N_8(\mathbf{a},\mathbf{p}) = w_1\left(6\binom{\gamma}{3}\binom{\kappa}{3}\frac{w}{w_1}P_6(\mathbf{a},\mathbf{p}) + \frac{1}{w_1}N_8(\mathbf{a},\mathbf{p})\right) = w_1\left(\frac{2w}{3}(\gamma-2)(\kappa-2)P_6(\mathbf{a},\mathbf{p}) + \frac{1}{w_1}N_8(\mathbf{a},\mathbf{p})\right)$. Therefore, we let the following equation be the objective function of the gradient-descent algorithm:

$$\begin{aligned}&\frac{2w}{3}(\gamma-2)(\kappa-2)P_6(\mathbf{a},\mathbf{p}) + \frac{1}{w_1}N_8(\mathbf{a},\mathbf{p})\\ &= \frac{2w}{3}(\gamma-2)(\kappa-2)\left[f^3(X)f^3(X^{-1})\right]_0 + \left[f^2(X)f^2(X^{-1})\right]_0\\ &\quad + \bar{w}_2\left[f(X^2)f(X^{-2})f^2(X)f^2(X^{-1})\right]_0\\ &\quad + \bar{w}_3\left[f(X^2)f^2(X)f^4(X^{-1})\right]_0 + \bar{w}_4\left[f^4(X)f^4(X^{-1})\right]_0. \qquad (9)\end{aligned}$$

Algorithm 1 Gradient-Descent Distributor (GRADE) for Cycle Optimization

Inputs and Parameters:

$\gamma, \kappa, m_t, m, \mathbf{a}$: parameters of the SC code;

w: weight of each cycle-6 candidate assuming that of a cycle-8 is 1;

ϵ, α: accuracy and step size of gradient descent;

Outputs and Intermediate Variables:

$\mathbf{p}$: a locally optimal edge distribution over $\mathrm{vals}(\mathbf{a})$

v_{prev}, v_{cur}: the value of the objective function in (9) at the previous and the current iterations;

$\mathbf{g}$: the gradient of the objective function;

1: $\bar{w}_1 \leftarrow \frac{2w}{3}(\gamma-2)(\kappa-2)$, obtain $\{\bar{w}_i\}_{i=2}^4$ in Lemma 4;

2: $v_{prev} = 1$; $v_{cur} = 1$;

3: $\mathbf{p} \leftarrow \frac{1}{m_t+1}\mathbf{1}_{m_t+1}$, $\mathbf{g} \leftarrow \mathbf{0}_{m_t+1}$, $\mathbf{f}, \bar{\mathbf{f}} \leftarrow \mathbf{0}_{m+1}$, $\mathbf{f}_2, \bar{\mathbf{f}}_2 \leftarrow \mathbf{0}_{2m+1}$;

4: $\mathbf{f}\left[a_0,\ldots,a_{m_t}\right] \leftarrow \mathbf{p}$, $\bar{\mathbf{f}} \leftarrow \mathrm{flip}(\mathbf{f})$; //$\mathbf{f}$, $\mathrm{flip}(\mathbf{f})$: the coefficients of $f(X)$, $f(X^{-1})$

5: $\mathbf{f}_2\left[1,3,\ldots,2m+1\right] \leftarrow \mathbf{f}$, $\bar{\mathbf{f}}_2 \leftarrow \mathrm{flip}(\mathbf{f}_2)$; //$\mathbf{f}_2$, $\mathrm{flip}(\mathbf{f}_2)$: the coefficients of $f(X^2)$, $f(X^{-2})$

//The coefficients of the product of polynomials can be computed as the convolution of the coefficients of all the polynomials. In MATLAB, a convolution is performed by using the "conv($\cdot$)" command

//We classify each term in the characteristic polynomials and their gradients by a degree pair (d_+, d_-), where d_+, d_- refer to the degree of the positive component (product of f's and f_2's) and the negative component (product of $\bar{f}$'s and $\bar{f}_2$'s), respectively

//Compute and update the value of the objective function. The sum of all terms with d_+ being $3m$, $2m$, and $4m$ in the objective function is represented by $\mathbf{q}_1$, $\mathbf{q}_2$, and $\mathbf{q}_3$, respectively.

6: $\mathbf{q}_1 \leftarrow \bar{w}_1\mathrm{conv}(\mathbf{f},\mathbf{f},\mathbf{f},\bar{\mathbf{f}},\bar{\mathbf{f}},\bar{\mathbf{f}})$, $\mathbf{q}_2 \leftarrow \mathrm{conv}(\mathbf{f},\mathbf{f},\bar{\mathbf{f}},\bar{\mathbf{f}})$;

7: $\mathbf{q}_3 \leftarrow \bar{w}_2\mathrm{conv}(\mathbf{f}_2,\bar{\mathbf{f}}_2,\mathbf{f},\mathbf{f},\bar{\mathbf{f}},\bar{\mathbf{f}}) + \bar{w}_3\mathrm{conv}(\mathbf{f}_2,\mathbf{f},\mathbf{f},\bar{\mathbf{f}},\bar{\mathbf{f}},\bar{\mathbf{f}},\bar{\mathbf{f}}) + \bar{w}_4\mathrm{conv}(\mathbf{f},\mathbf{f},\mathbf{f},\mathbf{f},\bar{\mathbf{f}},\bar{\mathbf{f}},\bar{\mathbf{f}},\bar{\mathbf{f}})$;

8: $v_{prev} = v_{cur}$, $v_{cur} = \mathbf{q}_1\left[3m\right] + \mathbf{q}_2\left[2m\right] + \mathbf{q}_3\left[4m\right]$;

//Compute and update the gradient of the objective function. The sum of all terms of degree pairs $(3m,-2m)$, $(2m,-m)$, $(4m,-3m)$, and $(4m,-2m)$ in the gradient is represented by $\mathbf{g}_1$, $\mathbf{g}_2$, $\mathbf{g}_3$, and $\mathbf{g}_4$, respectively.

9: $\mathbf{g}_1 \leftarrow 6\bar{w}_1\mathrm{conv}(\mathbf{f},\mathbf{f},\mathbf{f},\bar{\mathbf{f}},\bar{\mathbf{f}})$, $\mathbf{g}_2 \leftarrow 4\mathrm{conv}(\mathbf{f},\mathbf{f},\bar{\mathbf{f}})$;

10: $\mathbf{g}_3 \leftarrow 4\bar{w}_2\mathrm{conv}(\mathbf{f}_2,\bar{\mathbf{f}}_2,\mathbf{f},\mathbf{f},\bar{\mathbf{f}}) + 2\bar{w}_3\mathrm{conv}(\bar{\mathbf{f}}_2,\mathbf{f},\mathbf{f},\mathbf{f},\mathbf{f},\bar{\mathbf{f}}) + 4\bar{w}_3\mathrm{conv}(\mathbf{f}_2,\mathbf{f},\mathbf{f},\bar{\mathbf{f}},\bar{\mathbf{f}},\bar{\mathbf{f}}) + 8\bar{w}_4\mathrm{conv}(\mathbf{f},\mathbf{f},\mathbf{f},\mathbf{f},\bar{\mathbf{f}},\bar{\mathbf{f}},\bar{\mathbf{f}},\bar{\mathbf{f}})$;

11: $\mathbf{g}_4 \leftarrow 2\bar{w}_2\mathrm{conv}(\mathbf{f}_2,\mathbf{f},\mathbf{f},\bar{\mathbf{f}},\bar{\mathbf{f}}) + \bar{w}_3\mathrm{conv}(\mathbf{f},\mathbf{f},\mathbf{f},\mathbf{f},\bar{\mathbf{f}},\bar{\mathbf{f}})$;

12: $\mathbf{g} \leftarrow \mathbf{g}_1\left[2m+\mathbf{a}\right] + \mathbf{g}_2\left[m+\mathbf{a}\right] + \mathbf{g}_3\left[3m+\mathbf{a}\right] + \mathbf{g}_4\left[2m+2\mathbf{a}\right]$, $\mathbf{g} \leftarrow \mathbf{g} - \mathrm{mean}(\mathbf{g})$;

13: **if** $|v_{prev} - v_{cur}| > \epsilon$ **then**

14: $\quad \mathbf{p} \leftarrow \mathbf{p} - \alpha\frac{\mathbf{g}}{||\mathbf{g}||}$;

15: $\quad$ **goto** step 4;

16: **return** $\mathbf{p}$;

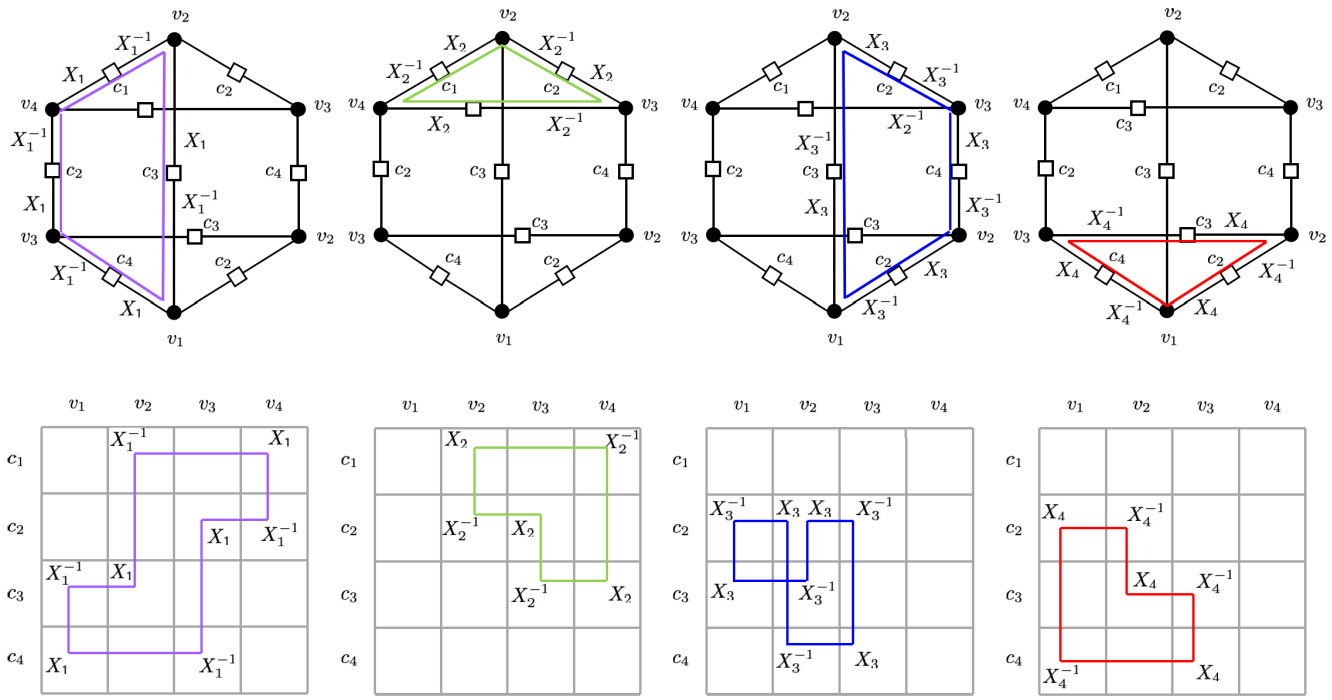


Fig. 3. The cycle basis of a typical $(6, 0)$ $((6, 6))$-AS in SC codes with $\gamma = 3$ ($\gamma = 4$) and their corresponding cycle candidates while pulled back to the base matrix. The cycle basis has 4 fundamental cycles as shown in the top 4 panels; each cycle decides a cycle candidate in the base matrix and an independent variable in the characteristic polynomial, as shown in the bottom panels.

IV. Generalization of GRADE

We have explained the basic idea of GRADE in optimizing the edge distribution of SC code ensembles with respect to the expected number of cycles. Cycles have been studied extensively in related literature (see e.g., [27], [28], [29]) due to their simplicity and presence in problematic objects. However, cycles alone do not always account for typical decoding failures; for example, isolated cycles are not as harmful as concentrated cycles in codes with VN degree 4 since single cycles on their own do not lead to decoding failures (as captured by e.g., ASs [10]), rather concatenated cycles do. An excessive focus on the removal of isolated cycles can lead to remarkably fewer degrees of freedom for the removal of dominant problematic objects. In this section, we therefore extend the theory of GRADE to arbitrary subgraphs.

A. Probabilistic Metric

In this subsection, we generalize the results presented in Section III-A to obtain closed-form representations of the expected number of objects with arbitrary topologies. The key idea is that the dependency among nodes within each object can be fully described by a minimal set of fundamental cycles (or basic cycles), which is referred to as the **cycle basis** of the object (see Definition 4) [21], [30], [31].

Definition 4: (**Cycle Basis**) A **cycle basis** of an object is a minimum-cardinality set of cycles using disjunctive unions[1] of which, each cycle in the object can be obtained; we call the cycles in this set **fundamental cycles**.[2]

In the remainder of this paper, we define a **prototype** of an object, for simplicity, as an assignment of the indices of its variable nodes (VNs) and its check nodes (CNs) in the base matrix. Prototype is a natural extension of *pattern*, i.e., a prototype of an object is exactly a pattern (discussed in [10]) when the object is a cycle. According to [21], we call a prototype **active** if all the fundamental cycles satisfy the cycle condition simultaneously, which means the detrimental object will be created in the protograph after partitioning the base matrix. The probability of a prototype becoming active under a random partition is proved to be represented by the constant term of a multi-variate polynomial, where each variable is associated with a cycle in the cycle basis: we refer to this polynomial as the **characteristic polynomial** of the object associated with fixed prototype. The overall characteristic polynomial of the object without specifying the prototype is then obtained as an average over the characteristic polynomials associated with all possible prototypes.

We start with a motivating example.

Example 3: Take the object (AS) with the node assignment shown in the top panel of Fig. 3 as an example.[3] Consider the cycle basis consisting of the 4 cycles highlighted in Fig. 3 and

[1]Disjunctive union of cycles refers to the subgraph consisting of all the edges contained in an odd number of cycles along with their neighboring vertices (VNs and CNs).

[2]While it is always possible to find a cycle basis for an object, some non-elementary configurations could possibly require different processing in order to enumerate the number of and specify the fundamental cycles, as explained in [21, Section VI].

[3]Note that we only keep nodes with intrinsic connections, i.e., we ignored the degree 1 CNs as they are not involved in any cycles and thus do not affect the probability of a prototype becoming active.

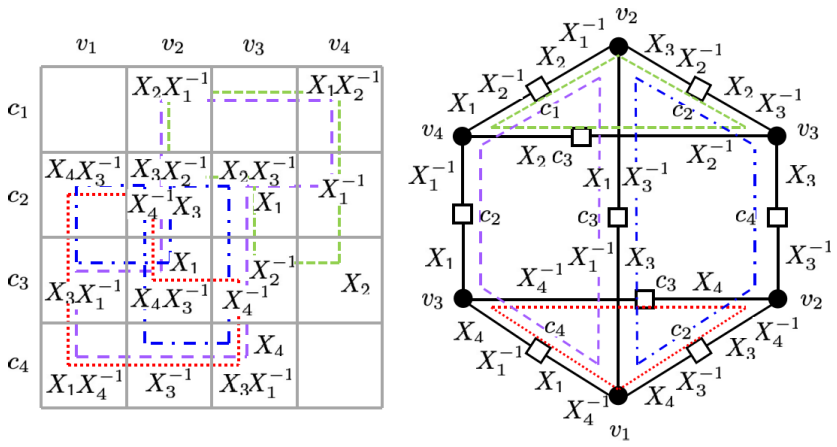


Fig. 4. The matrix representation of the characteristic polynomial of the AS in Fig. 3. The monomial in each entry corresponds to a factor in the characteristic polynomial in (10).

their associated variables X_i, $1 \leq i \leq 4$. We refer to the cycle associated with X_i, $1 \leq i \leq 4$, as cycle i. In the bottom panel of Fig. 3, the labels X_i and X_i^{-1} are placed alternately on the cycle candidate corresponding to cycle i in the base matrix.

We next briefly and intuitively explain how the characteristic polynomial of the object is specified as follows:

$$\begin{aligned} h(\mathbf{X}) =& f(X_2^{-1}X_3^2X_4^{-1})f(X_1X_2X_3^{-1})f(X_1X_3^{-1}X_4) \\ & f(X_1^{-1}X_3X_4)f(X_1X_2^{-1})f(X_1^{-1}X_2)f^2(X_1^{-1}X_3) \\ & f(X_1X_4^{-1})f(X_2^{-1}X_4^{-1})f(X_3^{-1}X_4)f(X_1^{-1}) \\ & f(X_2)f(X_3^{-1}), \end{aligned} \tag{10}$$

where $f(\cdot)$ is the coupling polynomial of a cycle as specified in Definition 3.

As shown in Fig. 4, we place the labels on all the cycle candidates (see Fig. 3) altogether in the base matrix. Then, each entry of the matrix becomes associated with the product of all the labels contained in it. Take the entry at the intersection of row c_3 and column v_2 as an example. This entry is labeled with X_1, X_3^{-1}, X_4 on the cycle candidates for cycles 1, 3, and 4, respectively. Therefore, the entry is associated with $X_1X_3^{-1}X_4$. Each product is the monomial corresponding to the matrix entry. The characteristic polynomial in (10) is exactly the product of all the factors obtained by replacing the variable in the coupling polynomial by the monomials corresponding to each entry.

In a way similar to the process described in the proof of Theorem 1, expanding the right-hand side (RHS) of (10) results in terms of the form $q_iX_1^{k_1}X_2^{k_2}X_3^{k_3}X_4^{k_4}$ for each, where q_i is the probability of a unique assignment to edges, i.e., matrix entries, on the prototype of the AS such that the alternating sum of entries on cycle i associated with X_i in the cycle basis is k_i, $1 \leq i \leq 4$. Therefore, the constant term is exactly the sum of the probabilities of all possible assignments (of the partitioning matrix) such that the cycle candidates of all the fundamental cycles satisfy their cycle conditions (all k_i's are zeros). In other words, the constant term is exactly the probability of the prototype becoming active in the Tanner graph.

In Example 3, we have briefly introduced the idea of how we define the characteristic polynomial of an object associated with a fixed prototype. However, as shown in the case of cycle-8 candidates, an object is typically associated with multiple prototypes (referred to as cycle candidates when the object is a cycle). We next present an efficient method to obtain the expected number of all possible prototypes corresponding to an object.

The major idea is described as follows. Each prototype of an object leads to an equivalence relation on the CNs and VNs of the object, in which nodes with identical indices are regarded as being equivalent. The set consisting of all the prototypes describing the same equivalence relation is referred to as a **prototype class**. The characteristic polynomials of the prototypes belonging to the same prototype class are identical, and the cardinality of each prototype class is determined by their associated equivalence relation. Therefore, the key steps to obtain the characteristic polynomial of an object are: 1) to enumerate all the possible prototype classes of (or non-isomorphic equivalence relations on) a given object, and then 2) to obtain their associated characteristic polynomials and cardinalities.

For example, consider the prototype class described by the graph in the left panel of Fig. 5. Throughout this paper, we use $[n]$ to represent the set $\{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}$. Any assignment of $c_1, c_2, c_3, c_4 \in [\gamma]$ and $v_1, v_2, v_3, v_4 \in [\kappa]$ such that c_1, c_2, c_3, c_4 are mutually different, v_1, v_2, v_3, v_4 are also mutually different belongs to a unique prototype in the prototype class described by the graph. Note that the uniqueness follows from the fact that the **automorphism group** of the prototype class has only the identity element, thus the cardinality of this prototype class is $4!\binom{\gamma}{4}4!\binom{\kappa}{4}$. The automorphism group of a prototype is defined later in Definition 6; however, the aforementioned cardinality intuitively implies that each row/column permutation results in a unique prototype. We then move on to obtain the characteristic polynomial directly through the prototype class. The equivalence relation on CNs and VNs induces an equivalence relation on edges, in which edges with VNs and CNs all from the same equivalence class are referred to as being equivalent. As shown in Fig. 5, edges from the same equivalence class are highlighted by identical markers.

Note that each equivalence class on edges corresponds to a unique entry in the base matrix. Recall that each entry is associated with the product of all labels contained in it,

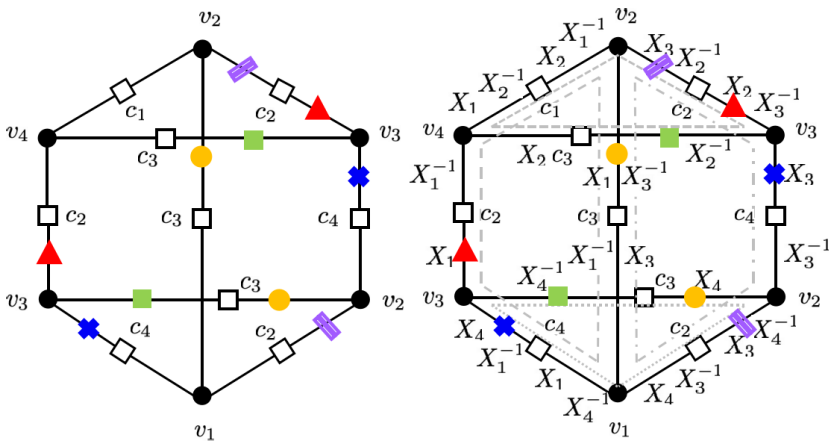


Fig. 5. The graph representation of the characteristic polynomial of the AS in Fig. 3. Edges in the same equivalence class are labeled with identical marks. The product of monomials at all the edges belonging to each equivalence class corresponds to a factor in the characteristic polynomial in (10).

which corresponds to a separate factor in the characteristic polynomial. In a similar way, if we represent each equivalence class by the product of all labels on all edges contained in it, the resultant product will be exactly the monomial associated with the entry corresponding to this equivalence class. Therefore, each factor of the characteristic polynomial in (10) is associated with an equivalence class on edges. For example, in Fig. 5, the two edges highlighted by red triangles belong to the same equivalence class and are labeled with X_1 and $X_2X_3^{-1}$, respectively; and they altogether correspond to the factor $f(X_1X_2X_3^{-1})$ in the characteristic polynomial.

In the remaining text, we represent the equivalence relation by $\sim$.

Definition 5 (Prototype Class): Let $\gamma, \kappa \in \mathbb{N}$. Consider an object represented by the bipartite graph $G(V, C, E)$, where V and C denote the set of VNs and CNs, respectively. The set E is the set of all edges, where each edge is represented by $e_{i,j}$, for some $i \in V$, $j \in C$, connecting nodes i and j. Let $\mathcal{V}, \mathcal{C}$ represent equivalence classes on V and C, respectively. A **prototype** is an assignment $P = (f, g)$, where $f : V \to [\kappa]$, $g : C \to [\gamma]$ such that:

1) For any $c \in C$ and $v_1, v_2 \in V$ such that $e_{v_1,c}, e_{v_2,c} \in E$, $f(v_1) \neq f(v_2)$;
2) For any $v \in V$ and $c_1, c_2 \in C$ such that $e_{v,c_1}, e_{v,c_2} \in E$, $g(c_1) \neq g(c_2)$.

The set consisting of all the prototypes $P = (f, g)$ that satisfy the following conditions is referred to as the **prototype class** associated with $(\mathcal{V}, \mathcal{C})$, denoted by $\mathcal{P}(\mathcal{V}, \mathcal{C})$:

1) For any $v_1, v_2 \in V$, $f(v_1) = f(v_2)$ iff. $v_1 \sim v_2$ in $\mathcal{V}$ (same column in the matrix);
2) For any $c_1, c_2 \in C$, $g(c_1) = g(c_2)$ iff. $c_1 \sim c_2$ in $\mathcal{C}$ (same row in the matrix).

Denote the equivalence class induced by the relation: $e_{v_1,c_1} \sim e_{v_2,c_2}$ iff. $v_1 \sim v_2$ and $c_1 \sim c_2$, for all $v_1, v_2 \in V$ and $c_1, c_2 \in C$, by $\mathcal{E}(\mathcal{V}, \mathcal{C})$, which is referred to as the equivalence class induced by $\mathcal{V}$ and $\mathcal{C}$.

Lemma 5: (**Characteristic Polynomial of Prototypes**) Consider the bipartite graph $G(V, C, E)$ of an object and the prototype class $\mathcal{P}(\mathcal{V}, \mathcal{C})$. Suppose $\mathcal{E}(\mathcal{V}, \mathcal{C})$ is the equivalence class on edges induced by $\mathcal{V}$ and $\mathcal{C}$.

Let S denote the cycle basis of G. Define $\delta : E \times S \to \{-1, 0, 1\}$ as follows: for any $s \in S$, $s = (v_1, c_1, v_2, c_2, \ldots, v_g, c_g)$, and $e \in E$, $\delta_{e,s} = 1$ if $e = e_{v_i,c_i}$ for some $i \in [g]$, $\delta_{e,s} = -1$ if $e = e_{v_{i+1},c_i}$ for some $i \in [g]$, otherwise $\delta_{e,s} = 0$.

Define $h(\mathbf{X}; G|\mathcal{V}, \mathcal{C})$ as a polynomial of G associated with $\mathcal{P}(\mathcal{V}, \mathcal{C})$ and given by:

$$h(\mathbf{X}; G|\mathcal{V}, \mathcal{C}) = \prod_{\bar{e} \in \mathcal{E}(\mathcal{V},\mathcal{C})} f\left(\prod_{e \in \bar{e}} \prod_{s \in S} X_s^{\delta_{e,s}}\right). \quad (11)$$

Then, the constant term of $h(\mathbf{X}; G|\mathcal{V}, \mathcal{C})$ is the probability that a prototype belonging to the class $\mathcal{P}(\mathcal{V}, \mathcal{C})$ is active in the Tanner graph after partitioning.

Proof: For simplicity, we write $\mathcal{E}$ instead of $\mathcal{E}(\mathcal{V}, \mathcal{C})$ in the proof. Any assignment on the set of edges E can be represented by $\mathbf{x} \in (x_1, x_2, \ldots, x_{|E|}) \in \text{vals}(\mathbf{a})^{|E|}$ ($\text{vals}(\mathbf{a})$ is defined in Theorem 1 as the set $\{a_0, a_1, \ldots, a_{m_t}\}$), where x_e denotes the assignment on edge e for any $e \in E$. Consider that all the edges belonging to the same equivalence class in $\mathcal{E}$ correspond to the same entry in the base matrix (and the partitioning matrix); these edges need to be assigned with an identical number in the partitioning matrix. Therefore, the assignment on the set of edges is essentially an assignment on the equivalence classes. Let $\mathbf{i} \in \{0, 1, \ldots, m_t\}^{|\mathcal{E}|}$ denote an assignment on the equivalence classes $\mathcal{E}$, where each element of $\mathbf{i}$ is represented by $i_{\bar{e}}$ for some $\bar{e} \in \mathcal{E}$; all the edges in the equivalence class $\bar{e}$ are assigned with $a_{i_{\bar{e}}}$ in the partitioning matrix, for any $\bar{e} \in \mathcal{E}$.

We know that

$$\begin{aligned} &h(\mathbf{X}; G|\mathcal{V}, \mathcal{C}) \\ &= \prod_{\bar{e} \in \mathcal{E}} f\left(\prod_{e \in \bar{e}} \prod_{s \in S} X_s^{\delta_{e,s}}\right) \\ &= \sum_{\mathbf{i} \in \{0,1,\ldots,m_t\}^{|\mathcal{E}|}} \prod_{\bar{e} \in \mathcal{E}} \left[p_{i_{\bar{e}}} \prod_{e \in \bar{e}} \prod_{s \in S} X_s^{\delta_{e,s} a_{i_{\bar{e}}}}\right] \\ &= \sum_{\mathbf{i} \in \{0,1,\ldots,m_t\}^{|\mathcal{E}|}} \left(\prod_{\bar{e} \in \mathcal{E}} p_{i_{\bar{e}}}\right) \prod_{\bar{e} \in \mathcal{E}} \prod_{e \in \bar{e}} \prod_{s \in S} X_s^{\delta_{e,s} a_{i_{\bar{e}}}} \end{aligned}$$

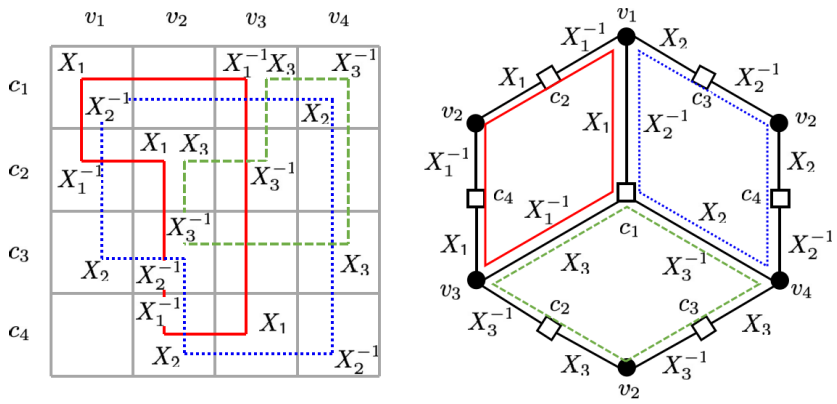


Fig. 6. A matrix representation (left panel) of the characteristic polynomial of the AS (right panel) in Remark 2.

$$= \sum_{\mathbf{i}\in\{0,1,\ldots,m_t\}^{|\mathcal{E}|}} \left(\prod_{\bar{e}\in\mathcal{E}} p_{i_{\bar{e}}}\right) \prod_{s\in S} X_s^{\sum_{\bar{e}\in\mathcal{E}} a_{i_{\bar{e}}} \sum_{e\in\bar{e}} \delta_{e,s}}$$
$$= \sum_{\mathbf{i}\in\{0,1,\ldots,m_t\}^{|\mathcal{E}|}} \left(\prod_{\bar{e}\in\mathcal{E}} p_{i_{\bar{e}}}\right) \prod_{s\in S} X_s^{l_s(\mathbf{i})}, \tag{12}$$

where $l_s(\mathbf{i}) = \sum_{\bar{e}\in\mathcal{E}} a_{i_{\bar{e}}} \sum_{e\in\bar{e}} \delta_{e,s} = \sum_{e\in s, e\in\bar{e}} \delta_{e,s} a_{i_{\bar{e}}}$ is exactly the alternating sum of the assignment $\mathbf{i}$ on cycle s.

Denote by $\mathcal{Z}(G)$ the set of assignments of the prototype in the partitioning matrix such that this prototype becomes active in the Tanner graph of the code after partitioning. Then,

$$\begin{aligned}
&[h(\mathbf{X}; G|\mathcal{V},\mathcal{C})]_0 \\
=& \sum_{\mathbf{i}\in\{0,1,\ldots,m_t\}^{|\mathcal{E}|}: l_s(\mathbf{i})=0, \forall s\in S} \prod_{\bar{e}\in\mathcal{E}} p_{i_{\bar{e}}} \\
=& \sum_{\mathbf{i}\in\{0,1,\ldots,m_t\}^{|\mathcal{E}|}: l_s(\mathbf{i})=0, \forall s\in S} \mathbb{P}\left[x_e = a_{i_{\bar{e}}}, \forall \bar{e}\in\mathcal{E}, e\in\bar{e}\right] \\
=& \mathbb{P}\left[\mathcal{Z}(G)\right],
\end{aligned} \tag{13}$$

which indicates that the constant term of $h(\mathbf{X}; G|\mathcal{V},\mathcal{C})$ is the probability we are seeking. ■

In fact, the coefficients of other terms of $h(\mathbf{X}; G|\mathcal{V},\mathcal{C})$ also specify the probabilities of partitioning assignments other than $\mathcal{Z}(G)$. Consequently, $h(\mathbf{X}; G|\mathcal{V},\mathcal{C})$ in (11) represents the characteristic polynomial of G associated with $\mathcal{P}(\mathcal{V},\mathcal{C})$.

While elementary objects (absorbing sets in particular) dominate the error floor of binary LDPC codes and NB-LDPC codes over the AWGN channel, non-elementary objects are observed to notably contribute to the error floor of NB-LDPC codes over non-canonical channels, e.g., practical magnetic recording and Flash channels [22], [23], [32].

Remark 2: [Non-Elementary Objects] Note that Lemma 5 extends beyond elementary objects. Fig. 6 shows a prototype of a non-elementary object. This prototype can still be described by a set of 3 elementary cycles, as shown in Fig. 6 via colors. According to Lemma 5, the characteristic polynomial of the prototype is:

$$\begin{aligned}
&h(\mathbf{X}; G|\mathcal{V},\mathcal{C}) \\
=& f(X_1X_2^{-1}) f(X_2X_3^{-1}) f(X_3X_1^{-1}) f(X_1X_3) f(X_1^{-1}X_2) \\
& f(X_2^{-1}X_3^{-1}) f(X_1) f(X_1^{-1}) f(X_2) f(X_2^{-1}) f(X_3) f(X_3^{-1}).
\end{aligned} \tag{14}$$

In combination with the WCM framework proposed in [23] that optimizes the edge weights of NB-LDPC codes on fixed unweighted graphs, our method can open a door to systematically optimizing NB-SC codes with high memories, which have potential to be adopted in storage systems among other applications.

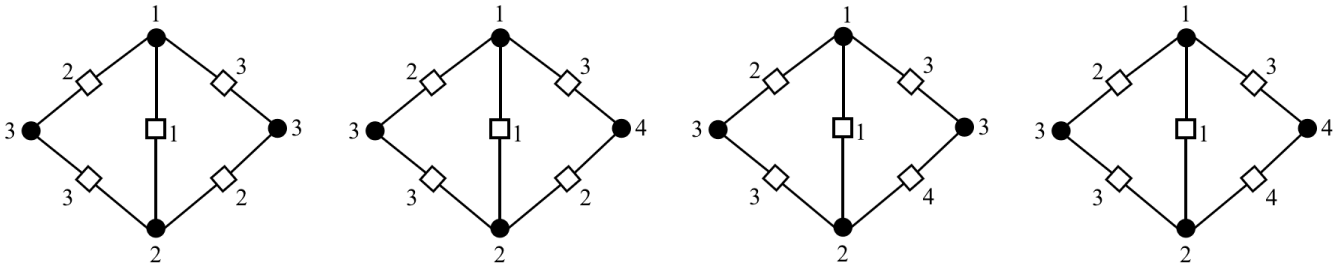
After obtaining the characteristic polynomial of any object associated with a fixed prototype class, we proceed to obtain the expectation of the number of active prototypes over all prototype classes. The essential step here is to calculate the cardinality of each prototype class. A natural property here is that each prototype from a specific class corresponds to assigning non-repeated elements with order from $[\kappa]$ and $[\gamma]$ to the equivalence classes in $\mathcal{V}$ and $\mathcal{C}$, respectively. However, specific permutations of values assigned to the nodes can lead to some other assignments that are isomorphic to each other because of the intrinsic symmetry of the prototypes. For example, in Fig. 7(a), the assignment that exchanges values v_1 and v_2 while keeping values on remaining VNs as they are is equivalent to the original assignment. We call this exchange operation an **automorphism** over G under $\mathcal{P}(\mathcal{V},\mathcal{C})$ and denote it by (v_1v_2). The automorphisms over G under each prototype class form a group, which is defined in Definition 6.

Definition 6 (Automorphism Group of an Object Under a Prototype Class): For any object represented by a bipartite graph $G(V,C,E)$, let $\mathcal{P}(\mathcal{V},\mathcal{C})$ be a prototype class of G. An **automorphism** over G under $\mathcal{P}(\mathcal{V},\mathcal{C})$ is a pair of bijections (π_V,π_C) written as $\pi_V\pi_C$, where $\pi_V : V \to V$ and $\pi_C : C \to C$ are bijections such that

1) $\forall v\in V$, $c\in C$, $e_{v,c}\in E$ iff. $e_{\pi_V(v),\pi_C(c)}\in E$;
2) $\forall v_1, v_2\in V$, $v_1\sim v_2$ iff. $\pi_V(v_1)\sim\pi_V(v_2)$;
3) $\forall c_1, c_2\in C$, $c_1\sim c_2$ iff. $\pi_C(c_1)\sim\pi_C(c_2)$.

The set containing all automorphisms over G under $\mathcal{P}(\mathcal{V},\mathcal{C})$ is closed under permutation compositions, and is referred to as the **automorphism group** of G under $\mathcal{P}(\mathcal{V},\mathcal{C})$.

Remark 3: From Definition 6, we know that any automorphism over G under $\mathcal{P}(\mathcal{V},\mathcal{C})$ is a graph isomorphism that preserves the equivalence relation specified by $(\mathcal{V},\mathcal{C})$, i.e., $\{\pi_V(\bar{v}), \forall\bar{v}\in\mathcal{V}\} = \mathcal{V}$, $\{\pi_C(\bar{c}), \forall\bar{c}\in\mathcal{C}\} = \mathcal{C}$. Therefore,



(a) Prototype class 1. (b) Prototype class 2. (c) Prototype class 3. (d) Prototype class 4.

Fig. 7. The 4 prototype classes of 2 concatenated cycles-6 and their corresponding topologies.

each automorphism can be simply represented as a pair of permutations over $\mathcal{V}$ and $\mathcal{C}$.

Lemma 6: Given the bipartite graph $G(V, C, E)$ of an object, let $\mathcal{B}(G)$ denote the set consisting of all prototype classes of G. Define the characteristic polynomial $h(\mathbf{X}; G)$ of object G as follows:

$$h(\mathbf{X};G)=\sum_{\mathcal{P}(\mathcal{V},\mathcal{C})\in\mathcal{B}(G)}\frac{|\mathcal{V}|!|\mathcal{C}|!}{|\mathrm{Aut}(G|\mathcal{V},\mathcal{C})|}\binom{\kappa}{|\mathcal{V}|}\binom{\gamma}{|\mathcal{C}|}h(\mathbf{X};G|\mathcal{V},\mathcal{C}), \tag{15}$$

where $\mathrm{Aut}(G|\mathcal{V},\mathcal{C})$ denotes the automorphism group of the bipartite graph G under prototype class $\mathcal{P}(\mathcal{V},\mathcal{C})$. Then, $[h(\mathbf{X};G)]_0$ is exactly the expected number of active prototypes of object G.

Proof: There are $|\mathcal{V}|!|\mathcal{C}|!\binom{\kappa}{|\mathcal{V}|}\binom{\gamma}{|\mathcal{C}|}$ assignments on indices of nodes in G in the base matrix that satisfy the equivalence relation specified by $(\mathcal{V},\mathcal{C})$. Among these assignments, each one has been counted exactly $|\mathrm{Aut}(G|\mathcal{V},\mathcal{C})|$ times. Therefore, the cardinality of the prototype class $\mathcal{P}(\mathcal{V},\mathcal{C})$ is exactly $\frac{|\mathcal{V}|!|\mathcal{C}|!}{|\mathrm{Aut}(G|\mathcal{V},\mathcal{C})|}\binom{\kappa}{|\mathcal{V}|}\binom{\gamma}{|\mathcal{C}|}$.

For any prototype P of G, define a Bernoulli random variable X_P, where $\mathbb{P}[X_P=1]=\mathbb{P}[P \text{ is active}]$, $\mathbb{P}[X_P=0]=\mathbb{P}[P \text{ is not active}]$. Let $X=\sum_P X_P$ denote the summation of X_P over all possible prototypes of G. Then,

$$\begin{aligned}\mathbb{E}[X]&=\sum_P\mathbb{E}[X_P]=\sum_{\mathcal{P}(\mathcal{V},\mathcal{C})\in\mathcal{B}(G)}\sum_{P\in\mathcal{P}(\mathcal{V},\mathcal{C})}\mathbb{E}[X_P]\\&=\sum_{\mathcal{P}(\mathcal{V},\mathcal{C})\in\mathcal{B}(G)}\sum_{P\in\mathcal{P}(\mathcal{V},\mathcal{C})}\mathbb{P}[X_P=1]\\&=\sum_{\mathcal{P}(\mathcal{V},\mathcal{C})\in\mathcal{B}(G)}\sum_{P\in\mathcal{P}(\mathcal{V},\mathcal{C})}[h(\mathbf{X};G|\mathcal{V},\mathcal{C})]_0\\&=\sum_{\mathcal{P}(\mathcal{V},\mathcal{C})\in\mathcal{B}(G)}\frac{|\mathcal{V}|!|\mathcal{C}|!}{|\mathrm{Aut}(G|\mathcal{V},\mathcal{C})|}\binom{\kappa}{|\mathcal{V}|}\binom{\gamma}{|\mathcal{C}|}[h(\mathbf{X};G|\mathcal{V},\mathcal{C})]_0\\&=[h(\mathbf{X};G)]_0\,.\end{aligned} \tag{16}$$

Thus, the lemma is proved. ∎

Example 4: Suppose $\gamma\in\{3,4\}$. Take the graph G of two concatenated cycles-6 as an example; there are 4 different possible prototype classes $(\mathcal{V}_i,\mathcal{C}_i)$, $1\le i\le 4$, as shown in Fig. 7. The automorphism groups corresponding to the 4 prototype classes, denote by $\mathrm{Aut}(G|\mathcal{V}_i,\mathcal{C}_i)$, $1\le i\le 4$, respectively, are:

$$\begin{aligned}&\mathrm{Aut}(G|\mathcal{V}_1,\mathcal{C}_1)=\{e,(v_1v_2),(c_2c_3),(v_1v_2)(c_1c_3)\},\\&\mathrm{Aut}(G|\mathcal{V}_2,\mathcal{C}_2)=\{e,(v_1v_2)(c_2c_3),(v_3v_4)(c_2c_3),(v_1v_2)(v_3v_4)\},\\&\mathrm{Aut}(G|\mathcal{V}_3,\mathcal{C}_3)=\{e,(v_1v_2)(c_2c_4)\},\\&\mathrm{Aut}(G|\mathcal{V}_4,\mathcal{C}_4)=\{e,(v_1v_2)(v_3v_4)(c_2c_4)\},\end{aligned} \tag{17}$$

where the element e in these groups is the identity element (no permutations). Therefore, the cardinality of the automorphism groups corresponding to the 4 prototype classes are $|\mathrm{Aut}(G|\mathcal{V}_1,\mathcal{C}_1)|=4$, $|\mathrm{Aut}(G|\mathcal{V}_2,\mathcal{C}_2)|=4$, $|\mathrm{Aut}(G|\mathcal{V}_3,\mathcal{C}_3)|=2$, and $|\mathrm{Aut}(G|\mathcal{V}_4,\mathcal{C}_4)|=2$, respectively. Moreover, the characteristic polynomials for G corresponding to each prototype class are:

$$\begin{aligned}h(\mathbf{X};G|\mathcal{V}_1,\mathcal{C}_1)=&f(X_1X_2)f(X_1^{-1}X_2^{-1})f(X_1X_2^{-1})\\&f(X_1^{-1}X_2)f(X_1)f(X_1^{-1})f(X_2)f(X_2^{-1}),\\h(\mathbf{X};G|\mathcal{V}_3,\mathcal{C}_3)=&f(X_1X_2)f(X_1^{-1}X_2^{-1})\\&f(X_1^{-1}X_2)f^2(X_1)f(X_1^{-1})f(X_2)f^2(X_2^{-1}),\\h(\mathbf{X};G|\mathcal{V}_2,\mathcal{C}_2)=&h(\mathbf{X};G|\mathcal{V}_4,\mathcal{C}_4)=f(X_1X_2)f(X_1^{-1}X_2^{-1})\\&f^2(X_1)f^2(X_1^{-1})f^2(X_2)f^2(X_2^{-1}).\end{aligned} \tag{18}$$

According to Lemma 6, when $\gamma=3$, the characteristic polynomial $h(\mathbf{X};G)$ is derived in (19), shown at the bottom of the next page.

When $\gamma=4$, the characteristic polynomial $h(\mathbf{X};G)$ is derived in (20), shown at the bottom of the next page.

Remark 4: Observe that in Example 4, the number of assignments such that all edges are distinct dominates among all the cases, especially when κ is large enough; we refer to such dominant assignments as the **typical assignments**. Therefore, it is normally sufficiently accurate to optimize over the characteristic polynomial corresponding to the typical assignments only. Specifically, for 2 concatenated cycles of length $2i$ and $2j$, suppose the number of edges in common is $2k$. Then, the characteristic polynomial can be well approximated by $\tilde{h}(\mathbf{X};G)=Cf^k(X_1X_2)f^k(X_1^{-1}X_2^{-1})f^{i-k}(X_1)f^{i-k}(X_1^{-1})f^{j-k}(X_2)f^{j-k}(X_2^{-1})$ for some constant $C\in\mathbb{N}$.

B. Gradient-Descent Distributor

Theorem 3: Given the bipartite graph $G(V, C, E)$ of an object and the prototype class $\mathcal{P}(\mathcal{V},\mathcal{C})$. Suppose $\mathcal{E}(\mathcal{V},\mathcal{C})$ is the equivalence class induced by $\mathcal{V}$ and $\mathcal{C}$. Let $(\mathbf{v})_t$ be the t-th

entry of the vector $\mathbf{v}$. Following the notation in Lemma 5, the gradient of $[h(\mathbf{X};G|\mathcal{V},\mathcal{C})]_0$ with respect to $\mathbf{p}$ is given by:

$$\begin{aligned}&\left(\nabla_{\mathbf{p}}\left[h(\mathbf{X};G|\mathcal{V},\mathcal{C})\right]_0\right)_t\\&=\left[\sum_{\bar{e}\in\mathcal{E}}\prod_{s\in S}X_s^{a_t\sum_{e\in\bar{e}}\delta_{e,s}}\prod_{\bar{e}'\in\mathcal{E}\backslash\{\bar{e}\}}f\left(\prod_{e\in\bar{e}'}\prod_{s\in S}X_s^{\delta_{e,s}}\right)\right]_0.\end{aligned}\tag{21}$$

Proof: We obtain the gradient with respect to $\mathbf{p}$ as follows:

$$\begin{aligned}&\left(\nabla_{\mathbf{p}}h(\mathbf{X};G|\mathcal{V},\mathcal{C})\right)_t\\&=\sum_{\bar{e}\in\mathcal{E}}\left(\nabla_{\mathbf{p}}f\left(\prod_{e\in\bar{e}}\prod_{s\in S}X_s^{\delta_{e,s}}\right)\right)_t\prod_{\bar{e}'\in\mathcal{E}\backslash\{\bar{e}\}}f\left(\prod_{e\in\bar{e}'}\prod_{s\in S}X_s^{\delta_{e,s}}\right)\\&=\sum_{\bar{e}\in\mathcal{E}}\left(\nabla_{\mathbf{p}}f\left(\prod_{s\in S}\prod_{e\in\bar{e}}X_s^{\delta_{e,s}}\right)\right)_t\prod_{\bar{e}'\in\mathcal{E}\backslash\{\bar{e}\}}f\left(\prod_{e\in\bar{e}'}\prod_{s\in S}X_s^{\delta_{e,s}}\right)\\&=\sum_{\bar{e}\in\mathcal{E}}\left(\nabla_{\mathbf{p}}f\left(\prod_{s\in S}X_s^{\sum_{e\in\bar{e}}\delta_{e,s}}\right)\right)_t\prod_{\bar{e}'\in\mathcal{E}\backslash\{\bar{e}\}}f\left(\prod_{e\in\bar{e}'}\prod_{s\in S}X_s^{\delta_{e,s}}\right)\\&=\sum_{\bar{e}\in\mathcal{E}}\prod_{s\in S}X_s^{a_t\sum_{e\in\bar{e}}\delta_{e,s}}\prod_{\bar{e}'\in\mathcal{E}\backslash\{\bar{e}\}}f\left(\prod_{e\in\bar{e}'}\prod_{s\in S}X_s^{\delta_{e,s}}\right).\end{aligned}\tag{22}$$

Therefore,

$$\begin{aligned}&\left(\nabla_{\mathbf{p}}\left[h(\mathbf{X};G|\mathcal{V},\mathcal{C})\right]_0\right)_t=\left[\left(\nabla_{\mathbf{p}}h(\mathbf{X};G|\mathcal{V},\mathcal{C})\right)_t\right]_0\\&=\left[\sum_{\bar{e}\in\mathcal{E}}\prod_{s\in S}X_s^{a_t\sum_{e\in\bar{e}}\delta_{e,s}}\prod_{\bar{e}'\in\mathcal{E}\backslash\{\bar{e}\}}f\left(\prod_{e\in\bar{e}'}\prod_{s\in S}X_s^{\delta_{e,s}}\right)\right]_0.\end{aligned}\tag{23}$$

■

Provided the explicit expression of $h(\mathbf{X};G)$ and its gradient, one can easily apply the gradient-descent algorithm to obtain an edge distribution that locally minimizes the expected number of active prototypes of the object specified by G. In Example 5 and Example 6, we apply GRADE to two concatenated cycles-8 as shown in Fig. 8, under any prototype class $\mathcal{P}(\mathcal{V},\mathcal{C})$ such that $\mathcal{V}$ and $\mathcal{C}$ induce no equivalent edges in E. Denote by $P_{8-8}(\mathbf{a},\mathbf{p}|\mathcal{V},\mathcal{C})$ the probability that a prototype of this object becomes active after partitioning in an SC ensemble with coupling pattern $\mathbf{a}$ and edge distribution $\mathbf{p}$.

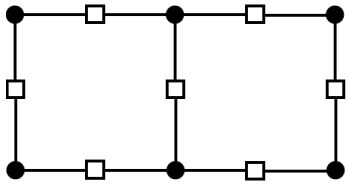


Fig. 8. The targeted object consisting of two concatenated cycle-8 in Example 5 and Example 6.

Example 5: Consider the following three cases of SC ensembles with $m=6$:

1) Full-memory codes with uniform edge distribution: $m_t=m=6$, $\mathbf{a}=(0,1,\ldots,6)$ and $\mathbf{p}=\frac{1}{7}\mathbf{1}_7$. Then, $P_{8-8}(\mathbf{a},\mathbf{p}|\mathcal{V},\mathcal{C})=0.0049$;
2) Full-memory codes with distribution obtained from GRADE: $m_t=m=6$, $\mathbf{a}=(0,1,\ldots,6)$ and $\mathbf{p}=(0.2991,0.0899,0.0749,0.0733,0.0749,0.0896,0.2984)$. Then, $P_{8-8}(\mathbf{a},\mathbf{p}|\mathcal{V},\mathcal{C})=0.0032$;
3) Non-full-memory codes with distribution obtained from GRADE: $m_t=3$, $\mathbf{a}=(0,1,4,6)$ and $\mathbf{p}=(0.2604,0.2063,0.2219,0.3114)$. Then, $P_{8-8}(\mathbf{a},\mathbf{p}|\mathcal{V},\mathcal{C})=0.0035$.

Example 6: Consider the following three cases of SC ensembles with $m=9$:

1) Full-memory codes with uniform edge distribution: $m_t=m=9$, $\mathbf{a}=(0,1,\ldots,9)$ and $\mathbf{p}=\frac{1}{10}\mathbf{1}_{10}$. Then, $P_{8-8}(\mathbf{a},\mathbf{p}|\mathcal{V},\mathcal{C})=0.0024$;

$$\begin{aligned}h(\mathbf{X};G)=&\frac{\kappa!\gamma!}{4(\kappa-3)!(\gamma-3)!}f(X_1X_2)f(X_1^{-1}X_2^{-1})f(X_1X_2^{-1})f(X_1^{-1}X_2)f(X_1)f(X_1^{-1})f(X_2)f(X_2^{-1})\\&+\frac{\kappa!\gamma!}{4(\kappa-4)!(\gamma-3)!}f(X_1X_2)f(X_1^{-1}X_2^{-1})f^2(X_1)f^2(X_1^{-1})f^2(X_2)f^2(X_2^{-1})\\=&\frac{3\kappa!}{2(\kappa-4)!}\Bigg(f(X_1X_2)f(X_1^{-1}X_2^{-1})f^2(X_1)f^2(X_1^{-1})f^2(X_2)f^2(X_2^{-1})\\&+\frac{1}{\kappa-3}f(X_1X_2)f(X_1^{-1}X_2^{-1})f(X_1X_2^{-1})f(X_1^{-1}X_2)f(X_1)f(X_1^{-1})f(X_2)f(X_2^{-1})\Bigg).\end{aligned}\tag{19}$$

$$\begin{aligned}h(\mathbf{X};G)=&\frac{\kappa!\gamma!}{4(\kappa-3)!(\gamma-3)!}f(X_1X_2)f(X_1^{-1}X_2^{-1})f(X_1X_2^{-1})f(X_1^{-1}X_2)f(X_1)f(X_1^{-1})f(X_2)f(X_2^{-1})\\&+\frac{\kappa!\gamma!}{2(\kappa-3)!(\gamma-4)!}f(X_1X_2)f(X_1^{-1}X_2^{-1})f(X_1^{-1}X_2)f^2(X_1)f(X_1^{-1})f(X_2)f^2(X_2^{-1})\\&+\left(\frac{\kappa!\gamma!}{4(\kappa-4)!(\gamma-3)!}+\frac{\kappa!\gamma!}{2(\kappa-4)!(\gamma-4)!}\right)f(X_1X_2)f(X_1^{-1}X_2^{-1})f^2(X_1)f^2(X_1^{-1})f^2(X_2)f^2(X_2^{-1})\\=&\frac{18\kappa!}{(\kappa-4)!}\Bigg(f(X_1X_2)f(X_1^{-1}X_2^{-1})f^2(X_1)f^2(X_1^{-1})f^2(X_2)f^2(X_2^{-1})\\&+\frac{2}{3(\kappa-3)}f(X_1X_2)f(X_1^{-1}X_2^{-1})f(X_1^{-1}X_2)f^2(X_1)f(X_1^{-1})f(X_2)f^2(X_2^{-1})\\&+\frac{1}{3(\kappa-3)}f(X_1X_2)f(X_1^{-1}X_2^{-1})f(X_1X_2^{-1})f(X_1^{-1}X_2)f(X_1)f(X_1^{-1})f(X_2)f(X_2^{-1})\Bigg).\end{aligned}\tag{20}$$

2) Full-memory codes with distribution obtained from GRADE: $m_t = m = 9$, $\mathbf{a} = (0, 1, \ldots, 9)$ and $\mathbf{p} = (0.2648, 0.0803, 0.0509, 0.0526, 0.0519, 0.0519, 0.0525, 0.0508, 0.0801, 0.2644)$. Then, $P_{8-8}(\mathbf{a}, \mathbf{p}|\mathcal{V}, \mathcal{C}) = 0.0015$;
3) Non-full-memory codes with distribution obtained from GRADE: $m_t = 4$, $\mathbf{a} = (0, 1, 4, 7, 9)$ and $\mathbf{p} = (0.2479, 0.1799, 0.1262, 0.1645, 0.2814)$. Then, $P_{8-8}(\mathbf{a}, \mathbf{p}|\mathcal{V}, \mathcal{C}) = 0.0016$.

Remark 5: In contrast to what we have shown regarding applying GRADE to single cycles, the gains obtained from applying GRADE to concatenated cycles are much more evident. As shown in Example 5 and Example 6, for $m = 6$ and $m = 9$, the local minima obtained for full memory codes are quite close to the gains obtained for codes with coupling patterns $(0, 1, 4, 6)$ and $(0, 1, 4, 7, 9)$, respectively (referred to as **topologically-coupled (TC) codes** later on).

Note that some probabilities can approach 0 directly in the GRADE algorithm without a brute-force search on all possible coupling patterns if the objective function is modified by adding a regularization term with some finely tuned $\beta \in \mathbb{R}$, $\beta > 0$, $n \in \mathbb{N}^*$ as follows:

$$h_0(\mathbf{X}; G) = [h(\mathbf{X}; G)]_0 + \beta \sum_{i=0}^{m} (1 + \tanh(np_i)). \tag{24}$$

The corresponding gradient changes as follows:

$$\nabla_{\mathbf{p}} h_0(\mathbf{X}; G) = \nabla_{\mathbf{p}} [h(\mathbf{X}; G)]_0 + n\beta \left(\operatorname{sech}^2(np_0), \operatorname{sech}^2(np_1), \ldots, \operatorname{sech}^2(np_m)\right). \tag{25}$$

The term $\beta \sum_{i=0}^{m} (1 + \tanh(np_i))$ is a differentiable approximation of the truncated gradient proposed in [33], and is adopted as a regularization term to help constrain the number of entries that are bounded away from zero.

We show next in Section V and Section VI that TC codes have close performance to GD codes with full-memories, where both are obtained from applying GRADE-AO followed by CPO to concatenated cycles.

Remark 6: Note that although we focus on non-tail-biting SC codes throughout this paper, this condition is by no means necessary. To extend our method to tail-biting codes, one just needs to change the cycle condition in (1) from "$\sum_{k=1}^{g} \mathbf{P}(i_k, j_k) = \sum_{k=1}^{g} \mathbf{P}(i_k, j_{k+1})$" to "$\sum_{k=1}^{g} \mathbf{P}(i_k, j_k) \equiv \sum_{k=1}^{g} \mathbf{P}(i_k, j_{k+1}) \mod L$", as presented in [20]. If $L > m + 1$, which is the typical case, the resultant optimal distribution is still very likely to be nonuniform because of the asymmetry among components indexed by $\{0, 1, \ldots, m\}$. This fact is important since while constructing non-tail-biting codes with large κ and m, a large L is typically desired due to the notable rate loss resulting from a small L. However, in certain practical applications, codes are typically of moderate length, which implies that a moderate L is desirable. In such situations where κ and m are large, tail-biting codes do not suffer the same rate loss, and they can offer high error floor performance despite limiting the gain obtained from threshold saturation.

V. Algorithmic Optimization

We have developed the theory and the algorithm to obtain edge distributions that locally minimize the number of short cycles in Section III-B and generalized the results from cycles to more sophisticated objects in Section IV-B. In this section, we investigate algorithmic optimizers (AO) that search for excellent partitioning matrices under the guidance of GRADE. In particular, the edge distribution $\mathbf{p}_{opt}$ obtained through GRADE confines the search space to only contain matrices that have edge distributions near $\mathbf{p}_{opt}$.

We discuss both heuristic AOs based on semi-greedy algorithms and globally-optimal AOs based on variations of the OO technique proposed in [9] and [10]. The heuristic AOs require low computational complexity and are applicable to a rich set of objects and any code parameters, but are only locally optimal. The OO-based AOs obtain the globally-optimal solutions, but currently only work on short cycles in SC codes with small pseudo-memories; we refer to these codes as **topologically-coupled (TC) codes**. The reason behind the nomenclature "TC codes" is the topological degrees of freedom they offer the code designer via the selection of the non-zero component matrices.

A. Heuristic AO

In this subsection, we consider AOs that are based on heuristic methods. In this case, our proposed GRADE algorithm obtains an edge distribution to guide the AO. Starting from a random partitioning matrix $\mathbf{P}$ with the derived distribution, one can perform a semi-greedy algorithm that searches for partitioning matrix near the initial $\mathbf{P}$ that locally minimizes the number of targeted objects. Constraining the search space to contain $\mathbf{P}$'s that have distributions within small L_1 and L_∞ distances from that of the original $\mathbf{P}$, and adopting the CPO next, significantly reduces the computational complexity to find a strong high-memory code. GRADE-guided heuristic AO has advantages in two aspects: 1) low complexity by reduced search space, and 2) higher probability of arriving at superior solution by providing a good enough initialization to AO that can avoid undesirable local minima.

1) Cycle-Based Optimization: Based on the GRADE specified in Algorithm 1, we first present in Algorithm 2 a corresponding AO that focuses on minimizing the weighted sum of the number of cycles-6 and cycles-8. We refer to codes obtained from GRADE-AO as **gradient-descent (GD) codes**. By replacing the initial distribution $\mathbf{p}$ with the uniform distribution, we obtain the so-called **uniform (UNF) codes**. In the special cases where the SC codes are not of full memory, i.e., the pseudo-memory is not identical with the memory, we refer to the codes obtained from GRADE-AO as **topologically-coupled (TC) codes**. We show in Section VI by simulation that the distribution obtained by GRADE results in constructions that are better than those adopting uniform distribution and in existing literature.

2) Finer-Grained Optimization: We next develop a finer-grained optimizer in Algorithm 3, in which the targeted objects are two concatenated cycles where each of them is a cycle-6 or a cycle-8, as discussed in the examples of Section IV-B. The critical part of the algorithm is enumerating

Algorithm 2 Cycle-Based GRADE-A Optimizer (AO)

Inputs and Parameters:

$\gamma, \kappa, m, m_t, \mathbf{a}$: parameters of an SC ensemble;
$\mathbf{p}$: edge distribution obtained from Algorithm 1;
w: weight of each cycle-6 assuming that of a cycle-8 is 1;
d_1, d_2: parameters indicating the size of the search space;

Outputs and Intermediate Variables:

$\mathbf{P}$: a locally optimal partitioning matrix;

Obtain the lists $\mathcal{L}_6(i,j)$, $\mathcal{L}_8(i,j)$ of cycles-6 candidates and cycle-8 candidates in the base matrix that contain node (i,j), $1 \le i \le \gamma$, $1 \le j \le \kappa$;
Obtain $\mathbf{u} = \arg\min_{\mathbf{x} \in \{0,1,\ldots,\gamma\kappa\}^{m+1}, ||\mathbf{x}||_1 = \gamma\kappa} ||\frac{1}{\gamma\kappa}\mathbf{x} - \mathbf{p}||_2$;
for $i \in \{0, 1, \ldots, m\}$ **do**
  Place $\mathbf{u}[i+1]$ i's into $\mathbf{P}$ randomly;
$\mathbf{d} \leftarrow \mathbf{0}_{m+1}$;
noptimal $\leftarrow$ False;
for $i \in \{1, 2, \ldots, \gamma\}$, $j \in \{1, 2, \ldots, \kappa\}$ **do**
  $n_6 \leftarrow |\mathcal{L}_6(i,j)|$, $n_8 \leftarrow |\mathcal{L}_8(i,j)|$, $n \leftarrow wn_6 + n_8$;
  for $v \in \{0, 1, \ldots, m\}$ **do**
    $\mathbf{d}' = \mathbf{d}$, $\mathbf{d}'[v+1] \leftarrow \mathbf{d}'[v+1] + 1$, $p \leftarrow \mathbf{P}(i,j)$;
    if $||\mathbf{d}'||_1 \le d_1$ and $||\mathbf{d}'||_\infty \le d_2$ **then**
      $\mathbf{P}(i,j) \leftarrow v$;
      $t_6 \leftarrow |\mathcal{L}_6(i,j)|$, $t_8 \leftarrow |\mathcal{L}_8(i,j)|$, $t \leftarrow wt_6 + t_8$;
      if $t < n$ **then**
        noptimal $\leftarrow$ True, $n \leftarrow t$, $\mathbf{d} \leftarrow \mathbf{d}'$, $\mathbf{P}(i,j) \leftarrow v$;
if noptimal **then**
  goto step 6;
return $\mathbf{P}$;

all the objects of interest efficiently. Since we focus on *concatenated* cycles of length 6 or 8, the key idea is to characterize concatenated cycles by the positions of the two degree 3 VNs and the three paths connecting them. Here, we only consider objects that are elementary ASs. We call a path $\mathcal{P} = v_1$-c_1-v_2-$\cdots$-c_l-v_{l+1} a type-l path connecting (c_1, v_1) and (c_l, v_{l+1}) and denote it by $L(\mathcal{P}) = l$. Paths of type-1, type-2, and type-3 are shown in Fig. 9. Each concatenation of two cycles can be referred to as an i-j-k object, where i, j, k are the types of the three paths connecting the degree 3 nodes in this object. Our targeted objects, where each of the two cycles is either a cycle-6 or a cycle-8, can only be 2-1-2, 2-1-3, 2-2-2, or 3-1-3 objects. Steps 1-5 in Algorithm 3 are aimed at listing the paths of type 1–3, and all the possible combinations of indices of the beginning and the ending CNs on each path.

Algorithm 3 Fine-Grained GRADE-A Optimizer (AO)

Inputs and Parameters:

$\gamma, \kappa, m, m_t, \mathbf{a}$: parameters of an SC code with full memory;
$\mathbf{p}$: edge distribution obtained from Algorithm 1;
$\mathbf{w} = (w_1, w_2, w_3, w_4)$: the weights of 2-1-2, 2-1-3, 2-2-2, 3-1-3 objects, respectively.
d_1, d_2: parameters indicating the size of the search space;

Outputs and Intermediate Variables:

$\mathbf{P}$: a locally optimal partitioning matrix;

Obtain the lists $\mathcal{L}_1(i,j_1,j_2)$, $\mathcal{L}_2(i_1,i_2,j_1,j_2)$, and $\mathcal{L}_3(i_3,i_4,j_1,j_2)$, $1 \le j_1 < j_2 \le \kappa$, $1 \le i, i_1, i_2, i_3, i_4 \le \gamma$, $i_1 \ne i_2$, where the lists are specified as follows:
a) $\mathcal{L}_1(i,j_1,j_2)$: all type-1 paths connecting (i,j_1) and (i,j_2) in the base matrix;
b) $\mathcal{L}_2(i_1,i_2,j_1,j_2)$: all type-2 paths connecting (i_1,j_1) and (i_2,j_2) in the base matrix;
c) $\mathcal{L}_3(i_3,i_4,j_1,j_2)$: all type-3 paths connecting (i_3,j_1) and (i_4,j_2) in the base matrix;
Obtain the sets $\mathcal{I}_{212}$, $\mathcal{I}_{213}$, $\mathcal{I}_{222}$, and $\mathcal{I}_{313}$ as follows:
a) $\mathcal{I}_{212} \leftarrow \{(i,i_1,i_2,i_3,i_4) : 1 \le i, i_1, i_2, i_3, i_4 \le \gamma, i_1 < i_3, i_1 \ne i_2, i_3 \ne i_4\}$;
b) $\mathcal{I}_{213} \leftarrow \{(i,i_1,i_2,i_3,i_4) : 1 \le i, i_1, i_2, i_3, i_4 \le \gamma, i_1 \ne i_2\}$;
c) $\mathcal{I}_{222} \leftarrow \{(i_1,i_2,i_3,i_4,i_5,i_6) : 1 \le i_1, i_2, i_3, i_4, i_5, i_6 \le \gamma, i_1 < i_3 < i_5, i_1 \ne i_2, i_3 \ne i_4, i_5 \ne i_6\}$;
d) $\mathcal{I}_{313} \leftarrow \{(i,i_1,i_2,i_3,i_4) : 1 \le i, i_1, i_2, i_3, i_4 \le \gamma, i_1 < i_3\}$;
Obtain $\mathbf{u} = \arg\min_{\mathbf{x} \in \{0,1,2,\ldots,\gamma\kappa\}^{m+1}, ||\mathbf{x}||_1 = \gamma\kappa} ||\frac{1}{\gamma\kappa}\mathbf{x} - \mathbf{p}||_2$;
for $i \in \{0, 1, \ldots, m\}$ **do**
  Place $\mathbf{u}[i+1]$ i's into $\mathbf{P}$ randomly;
$\mathbf{d} \leftarrow \mathbf{0}_{m+1}$;
$n \leftarrow M$; //M is some very large constant
noptimal $\leftarrow$ False;
for $i \in \{1, 2, \ldots, \gamma\}$, $j \in \{1, 2, \ldots, \kappa\}$ **do**
  for $v \in \{0, 1, \ldots, m\}$ **do**
    $\mathbf{d}' = \mathbf{d}$, $\mathbf{d}'[v+1] \leftarrow \mathbf{d}'[v+1] + 1$, $p \leftarrow \mathbf{P}(i,j)$;
    if $||\mathbf{d}'||_1 \le d_1$ and $||\mathbf{d}'||_\infty \le d_2$ **then**
      $\mathbf{P}(i,j) \leftarrow v$;
  //Compute the number $v_{t,l}$ of Type-t paths with alternating sum l in matrix $\mathcal{P}$, where $1 \le t \le 3$
  for $1 \le j_1 < j_2 \le \kappa$, $1 \le i_0, i_1, i_2, i_3, i_4 \le \gamma$, $i_1 \ne i_2$ **do**
    $v_{1,l}(i_0,j_1,j_2) \leftarrow |\{\mathcal{P} | S(\mathcal{P};\mathbf{P}) = l, \mathcal{P} \in \mathcal{L}_1(i_0,j_1,j_2)\}|$, $-m \le l \le m$;
    $v_{2,l}(i_1,i_2,j_1,j_2) \leftarrow |\{\mathcal{P} | S(\mathcal{P};\mathbf{P}) = l, \mathcal{P} \in \mathcal{L}_2(i_1,i_2,j_1,j_2)\}|$, $-2m \le l \le 2m$;
    $v_{3,l}(i_3,i_4,j_1,j_2) \leftarrow |\{\mathcal{P} | S(\mathcal{P};\mathbf{P}) = l, \mathcal{P} \in \mathcal{L}_3(i_3,i_4,j_1,j_2)\}|$, $-m \le l \le m$;
  //Compute the number t_{ijk} of prototypes that result in i-j-k objects in the protograph
  $t_{212} \leftarrow \sum_{1 \le j_1 < j_2 \le \kappa} \sum_{(i_0,i_1,i_2,i_3,i_4) \in \mathcal{I}_{212}} \sum_{-m \le l \le m} v_{1,l}(i_0,j_1,j_2) v_{2,l}(i_1,i_2,j_1,j_2) v_{2,l}(i_3,i_4,j_1,j_2)$;
  $t_{213} \leftarrow \sum_{1 \le j_1 < j_2 \le \kappa} \sum_{(i_0,i_1,i_2,i_3,i_4) \in \mathcal{I}_{213}} \sum_{-m \le l \le m} v_{1,l}(i_0,j_1,j_2) v_{2,l}(i_1,i_2,j_1,j_2) v_{3,l}(i_3,i_4,j_1,j_2)$;
  $t_{222} \leftarrow \sum_{1 \le j_1 < j_2 \le \kappa} \sum_{(i_1,i_2,i_3,i_4,i_5,i_6) \in \mathcal{I}_{222}} \sum_{-2m \le l \le 2m} v_{2,l}(i_1,i_2,j_1,j_2) v_{2,l}(i_3,i_4,j_1,j_2) v_{2,l}(i_5,i_6,j_1,j_2)$;
  $t_{313} \leftarrow \sum_{1 \le j_1 < j_2 \le \kappa} \sum_{(i_0,i_1,i_2,i_3,i_4) \in \mathcal{I}_{313}} \sum_{-m \le l \le m} v_{1,l}(i_0,j_1,j_2) v_{3,l}(i_1,i_2,j_1,j_2) v_{3,l}(i_3,i_4,j_1,j_2)$;
  $t \leftarrow w_1 t_{212} + w_2 t_{213} + w_3 t_{222} + w_4 t_{313}$;
  if $t < n$ **then**
    noptimal $\leftarrow$ True, $n \leftarrow t$;
    $\mathbf{d} \leftarrow \mathbf{d}'$, $\mathbf{P}(i,j) \leftarrow v$;
if noptimal **then**
  goto step 8;
return $\mathbf{P}$;

Based on the aforementioned notations, a prototype of an i-j-k object consists of three paths of type-i,j,k, where all paths connect two entries with column indices j_1, j_2, $1 \le j_1 < j_2 \le \kappa$. Denote by $\mathcal{I}_{ijk}$ the set containing all possible combinations of row indices of the endpoints of paths in i-j-k

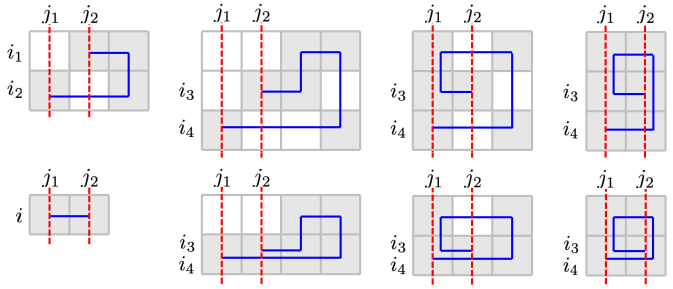


Fig. 9. Type 1-3 paths mentioned in Algorithm 3: $\mathcal{L}_1(i, j_1, j_2)$ (bottom left panel), $\mathcal{L}_2(i_1, i_2, j_1, j_2)$ (top left panel), and $\mathcal{L}_3(i_3, i_4, j_1, j_2)$ (the rightmost 6 panels), $1 \leq j_1 < j_2 \leq \kappa$, $1 \leq i, i_1, i_2, i_3, i_4 \leq \gamma$, $i_1 \neq i_2$. A type-k ($1 \leq k \leq 3$) path is a length $2i$ path in the base matrix that traverses between entries from two different columns, which depicts all possible assignments on the non-overlapping part of each cycle in the prototype of a concatenated-cycle pair connecting cycles of lengths 6 or 8.

object. The only constraint is that endpoints belonging to the same column are pairwise different. Take $j = 2$ as an example, only 2-2-2 objects are under our discussion. The set $\mathcal{I}_{222}$ has parameters $(i_1, i_2, i_3, i_4, i_5, i_6)$, where entries (i_{2t-1}, j_1), (i_{2t}, j_2), $1 \leq t \leq 3$, are the endpoints of the three paths comprising the object. Fig. 9 indicates that $i_{2t-1} \neq i_{2t}$, $1 \leq t \leq 3$. Therefore, $\mathcal{I}_{222} = \{(i_1, i_2, i_3, i_4, i_5, i_6) : 1 \leq i_1, i_2, i_3, i_4, i_5, i_6 \leq \gamma, i_1 < i_3 < i_5, i_1 \neq i_2, i_3 \neq i_4, i_5 \neq i_6\}$. Other cases are specified in Step 2 in Algorithm 3.[4]

The other major step in Algorithm 3 is to count the total number t_{ijk} of prototypes that result in i-j-k objects in the protograph. Given a partitioning matrix $\mathbf{P}$, denote by $S(\mathcal{P}; \mathbf{P})$ the alternating sum of the entries along the path $\mathcal{P}$. Then, we are able to count the number of Type-t paths, $1 \leq t \leq 3$, with alternating sum l, which is denoted by $v_{t,l}$ in Steps 14-17 in Algorithm 3. Therefore, t_{ijk} is the sum of $v_{i,l} v_{j,l} v_{k,l}$ over all possible combinations of l, row indices from $\mathcal{I}_{ijk}$, and column indices $1 \leq j_1 < j_2 \leq \kappa$, as shown in Steps 18-21 in Algorithm 3.

Remark 7: Note that in the finer-grained optimization, the condition of a concatenated-cycle pair in the protograph becoming a pair of concatenated cycles in the Tanner graph after lifting is that the two cycle candidates contained in this prototype all satisfy the cycle condition on lifting parameters specified in Lemma 1. Therefore, after applying AO to minimize the number of concatenated-cycle pairs, instead of using the original CPO designed for cycle optimization in [9], we adopt a modified version that is tailored for optimization over the number of pairs of concatenated cycles accordingly.

In a way similar to what we have done with approaches eliminating cycles, we define GD codes, UNF codes, and TC codes here. Moreover, as shown in Section IV-B, the expected number of concatenated-cycle pairs in GD codes (with full memories) is close to that of GD-TC codes with carefully chosen pseudo-memories and coupling patterns.[5] In particular, memory 6 GD ensembles can be approximated by GD-TC ensembles with pseudo-memory 3 and coupling pattern $(0, 1, 4, 6)$; memory 9 GD ensembles can be approximated by GD-TC ensembles with pseudo-memory 4 and coupling pattern $(0, 1, 4, 7, 9)$. This leads to the conjecture that the performance of GD-TC codes and the performance of GD codes are quite close, which is somewhat surprising provided that they differ a lot in edge distribution, and the impact of concatenated cycles on waterfall performance is not strictly characterized. In Section VI-B, Monte-Carlo simulations support our conjecture, which enables more possibilities in TC codes. For example, TC codes can be globally-optimized given that their pseudo-memories are low; details will be discussed later on in Section V-B.

Remark 8: Despite that both Algorithm 2 and Algorithm 3 assume all-one base matrices in this paper, while extending to generalized base matrices, the only part that may increase the complexity is the enumeration of the cycles and concatenated cycles in each algorithm.

The complexity is primarily controlled by the degrees of the nodes rather than the sizes of the base matrices. In particular, the major strategy of enumerating concatenated cycles is based on enumerating the type-i paths, $1 \leq i \leq 3$, which requires complexity $O(\gamma \kappa d_v^{i-1} d_c^i)$, where d_v and d_c represent the average VN degree and CN degree, respectively. Moreover, the individual cycle enumeration can also be performed utilizing the idea of paths adopted in the concatenated-cycles enumeration. Each cycle-6 candidate is composed of a type-1 path and a type-2 path. Each cycle-8 of structure S_1 in Fig. 2 is composed of two type-1 paths, while each of the others is composed of two type-2 paths. Therefore, the complexity of enumerating the cycles and the concatenated-cycles are $O(\gamma \kappa d_v d_c^2)$ and $O(\gamma \kappa d_v^2 d_c^3)$, respectively.

As for the part that calculates the total number of concatenated cycles, the number of multiplications performed at each iteration is shown to be dominated by twice the number of type-3 paths in the base matrix as presented in (26), shown at the bottom of the next page.

Therefore, the number of operations at each iteration is also bounded by $O(\gamma \kappa d_v^2 d_c^3)$. While we are not able to estimate the number of iterations at which the algorithm stops (despite that, experimentally it is typically small), we can stop the algorithm when the number of iterations reaches K, for some big enough $K \in \mathbb{N}$.[6] Under this modification, the complexity of Algorithm 3 is $O(Km\gamma \kappa d_v^2 d_c^3)$. A similar result can be applied to estimate the complexity of Algorithm 2 as $O(Km\gamma \kappa d_v d_c^2)$.

Compared with the complexity of the OO technique described in [10] that is governed by discrete optimization on $((m+1)^\gamma - 1)$ independent parameters, which also grows with κ, the complexity of the GRADE-AO method grows much slower with the size of the base matrix and the memory of the SC code. The GRADE-AO algorithm has been adopted

[4]Note that here we ignored the case where $j_1 = j_2$ for 2-2-2 objects, which does not happen when $j = 1$, and it leads to 3 pairwise concatenated cycle-4 candidates. Objects resulting from these prototypes will be removed in the CPO stage by removing cycles-4 when the circulant size is odd.

[5]We refer to the prototype of a pair of concatenated cycles as a concatenated-cycle pair.

[6]Note that specifying a closed form relation between m and the number of iterations is hard: despite that increasing m can enlarge the search space in the AO step and thus can increase the number of iterations, it also allows the algorithm to converge faster thanks to the additional degrees of freedom and thus can reduce the number of iterations. Having said that, the number of iterations depends not only on m but also on the initialization of the algorithm. Since the GRADE step provides a locally-optimal initialization, it leads to fast convergence and allows us to assume that the number of iterations is sufficiently small.

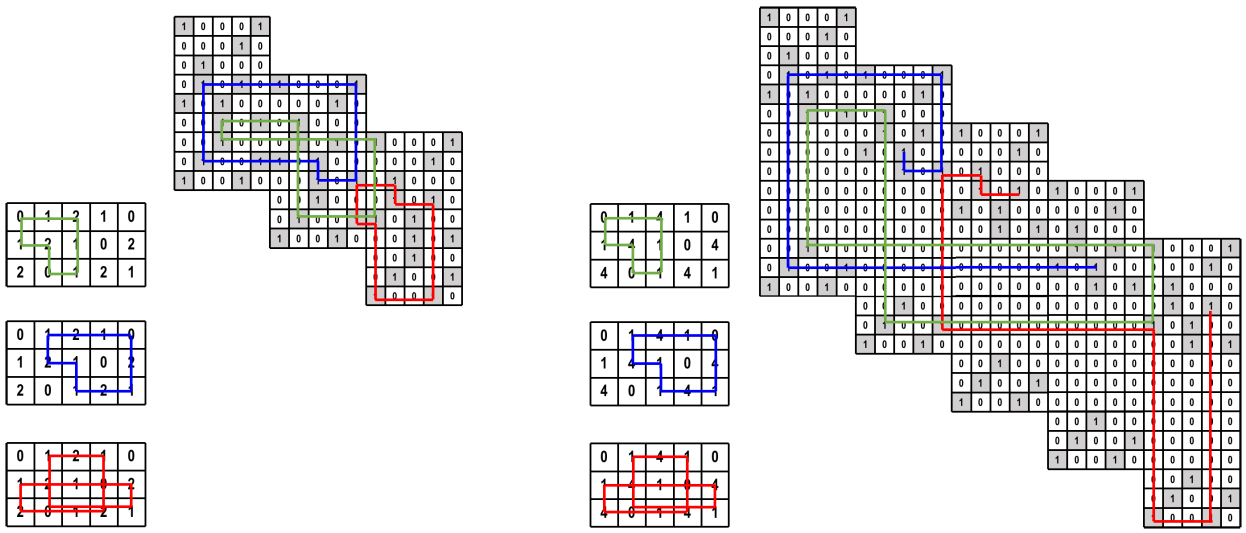


Fig. 10. The first 5 replicas of the protograph of a TC code with coupling pattern $(0, 1, 4)$ (the right panel), and the first 3 replicas of the protograph of its corresponding SC code with memomry 2 (the left panel). Three cycle candidates (colored by green, blue, and red, respectively) in the base matrix and their corresponding paths in the two protographs are marked out.

to construct SC-IRA codes, i.e., SC codes with base matrices being irregular-repeat-accumulate (IRA) codes in the thesis of Yang in [34].

B. Globally-Optimal AO

In this subsection, we explore globally-optimal constructions of TC codes with small pseudo-memories. The motivation behind this task is to construct an SC code with memory m under the same computational complexity needed to construct a full memory m_t code, where $m_t < m$. Given m_t and m, we first find the optimal $\mathbf{a}$, in terms of the minimum number of prototypes of interest, with length $m_t + 1$ in a brute-force manner. Taking $m = 4$ and $m_t = 2$ as an example, the optimal coupling pattern with respect to the number of cycles is $\mathbf{a} = (0, 1, 4)$ and the corresponding optimal distribution is almost uniform. Moreover, we already know from Section IV-B that regarding the optimal coupling pattern with respect to the number of concatenated-cycle pairs, $\mathbf{a} = (0, 1, 4, 6)$ and $\mathbf{a} = (0, 1, 4, 7, 9)$ are not only the optimal coupling patterns for $(m, m_t) = (6, 3)$ and $(m, m_t) = (9, 4)$, respectively, but also approximate the optimal full memory GD ensembles quite closely in terms of performance.

Given the optimal coupling pattern $\mathbf{a}$, we then obtain an optimal partitioning matrix by the OO method proposed in [9] and [10]. We extend the OO method for memory m_0 SC codes to any TC code with pseudo-memory $m_t = m_0$, which does not increase the complexity of the approach. Note that despite the current OO works only on cycles, future steps can be taken towards the extension of OO into concatenated cycles, which has potential to lead to TC codes with excellent performance that is even better than the GD codes with full memories, provided that it is much harder to obtain globally-optimal solutions for GD codes with full memories.

Optimal TC codes with pseudo-memory m_t have strictly fewer cycle candidates in their protographs than optimal SC codes with full memory $m = m_t$. Take $m = 4$ and $m_t = 2$ as an example. Suppose the optimal SC code has the partition $\mathbf{\Pi} = \mathbf{H}_0^{\mathrm{P}} + \mathbf{H}_1^{\mathrm{P}} + \mathbf{H}_2^{\mathrm{P}}$. Consider the TC code with partition $\mathbf{\Pi} = \mathbf{H}_0^{\mathrm{P}} + \mathbf{H}_1^{\mathrm{P}} + \mathbf{H}_4^{\mathrm{P}}$ such that $\mathbf{H}_2^{\mathrm{P}} = \mathbf{H}_4^{\mathrm{P}}$. Then, any cycle-6 candidate resulting from a cycle candidate in the base matrix assigned with 0-1-0-1-2-0, 1-2-1-2-2-0, or 0-1-2-1-x-x, $x \in \{0, 1, 2\}$, in $\mathbf{P}$ no longer has a counterpart in the TC code, since by replacing 2's with 4's, assignments 0-1-0-1-4-0, 1-4-1-4-4-0, and 0-1-4-1-x-x, $x \in \{0, 1, 4\}$, no longer satisfy the cycle condition in Lemma 1. Moreover, there exists a bijection between the remaining candidates in the SC code and all candidates in the TC code through the replacement of 2's with 4's.

Fig. 10 presents part of the protograph of a TC code with coupling pattern $(0, 1, 4)$ and that of its corresponding SC code with full memory 2. The cycle-6 candidate colored by blue is assigned with 0-1-2-1-1-1 in the SC code, which satisfies the cycle condition cycle candidates are generated in the protograph. However, the assignment becomes 0-1-4-1-1-1 in the TC code, which no longer satisfies the cycle condition and no cycle candidates are generated in the protograph. The cycle-6 candidate colored by green corresponds to one that results in cycle-6 candidates in both the SC and the TC codes shown in the figure. We also marked out a cycle-8 candidate (colored by red) that only leads to cycle candidates in the SC code.

$$
\begin{aligned}
&\sum\nolimits_{1\le j_1<j_2\le\kappa}\sum\nolimits_{(i_0,i_1,i_2,i_3,i_4)\in\mathcal{I}_{313}}\sum\nolimits_{-m\le l\le m} 2\mathbb{1}\left(v_{1,l}(i_0,j_1,j_2)>0\right)\mathbb{1}\left(v_{3,l}(i_1,i_2,j_1,j_2)>0\right)\mathbb{1}\left(v_{3,l}(i_3,i_4,j_1,j_2)>0\right)\\
\le&\sum\nolimits_{1\le j_1<j_2\le\kappa}\sum\nolimits_{(i_0,i_1,i_2,i_3,i_4)\in\mathcal{I}_{313}}\sum\nolimits_{-m\le l\le m} 2\mathbb{1}\left(v_{3,l}(i_1,i_2,j_1,j_2)>0\right)\\
\le&\sum\nolimits_{1\le j_1<j_2\le\kappa}\sum\nolimits_{(i_0,i_1,i_2,i_3,i_4)\in\mathcal{I}_{313}}\sum\nolimits_{-m\le l\le m} 2v_{3,l}(i_1,i_2,j_1,j_2). \qquad (26)
\end{aligned}
$$

TABLE I
STATISTICS OF THE NUMBER OF CYCLES

(γ, κ)	Code	Rate	Codeword Length	Constraint Length	Cycles-6	Cycles-8
$(3, 7)$	GD	-	-	-	0	0
	UNF	-	-	-	0	6,292
$(3, 17)$	GD	0.8076	11900	70	0	397,880
	UNF	0.8076	11900	70	0	559,902
	Battaglioni *et al.* [17]	0.8059	11900	71	0	451,337
$(4, 29)$	GD	-	-	-	0	528,090
	UNF	-	-	-	0	1,087,268
$(4, 17)$	TC	0.7459	14450	85	15,436	-
	SC (matched constraint length)	0.7490	14280	84	19,180	-
	SC (matched circulant size)	0.7553	14450	51	74,579	-

According to the aforementioned discussion, TC codes are better (have less cycles) than SC codes with the same circulant size and $m = m_t$. In Section VI, we present simulation results of such codes and show that they can also outperform SC codes with the same constraint length (larger circulant size) and $m = m_t$.

VI. Simulation Results

In this section, we show the frame error rate/uncorrectable bit error rate (FER/UBER) curves of seven groups of SC codes designed by the GRADE-AO methods presented in Section V. We demonstrate that codes constructed by the GRADE-AO methods offer significant performance gains compared with codes with uniform edge distributions and codes constructed through purely algorithmic methods. Note that all the codes constructed in this paper have no cycles-4.

A. Optimization Over Cycles

In this subsection, we simulate codes constructed based on optimizing the number of cycles using GRADE-AO specified in Section III over the AWGN channel. Out of these three plots, Fig. 11 and Fig. 12 compare GD codes with UNF codes designed as in Section V-A. Fig. 13 compares a TC code designed as in Section V-B with optimal SC codes constructed through the OO-CPO method proposed in [9]. The GD/UNF codes have parameters $(\gamma, \kappa, m, z, L) = (3, 7, 5, 13, 100)$, $(3, 17, 9, 7, 100)$, and $(4, 29, 19, 29, 20)$. We also compare our $(3, 17)$ codes with the code presented by Battaglioni *et al.* in [17]. For fair comparison, we choose the partitioning matrix $\mathbf{P}_{C_2}$ in [17], which has memory $m = 70$ and $z = 1$ (similar constraint length to our codes). Therefore, we choose $L = 700$ to let the code have a rate and overall length that are close to those of our proposed codes. In particular, both codes have length 11900. The code in [17] has constraint length $(70 + 1) \times 1 = 71$ and rate $1 - (1 + 70/700) \times (3/17) \simeq 0.8059$, while our codes have constraint length $(9+1) \times 7 = 70$ and rate $1 - (1 + 9/100) \times (3/17) \simeq 0.8076$.

The TC code has parameters $(\gamma, \kappa, m_t, z, L) = (4, 17, 2, 17, 50)$ with the coupling pattern $\mathbf{a} = (0, 1, 4)$. For a fair comparison, we have selected two SC codes: one with a similar constraint length $(m + 1)z$ and the other with an identical circulant power z. To ensure that the SC codes and the TC code have close rates and codelengths, the two SC codes have parameters $(\gamma, \kappa, m, z, L) = (4, 17, 2, 28, 30)$ and $(4, 17, 2, 17, 50)$, respectively. The TC code has codeword length $17 \times 17 \times 50 = 14450$, constraint length $(4+1) \times 17 = 85$, and rate $1-(1+4/50) \times (4/17) \simeq 0.7459$. The SC code with matched constraint length has codeword length $17 \times 28 \times 30 = 14280$, constraint length $(2+1) \times 28 = 84$, and rate $1-(1+2/30) \times (4/17) \simeq 0.7490$; the SC code with matched circulant size has codeword length $17 \times 17 \times 50 = 14450$, constraint length $(2+1) \times 17 = 51$, and rate $1 - (1 + 2/50) \times (4/17) \simeq 0.7553$. The statistics regarding the number of cycles of each code are presented in Table I.

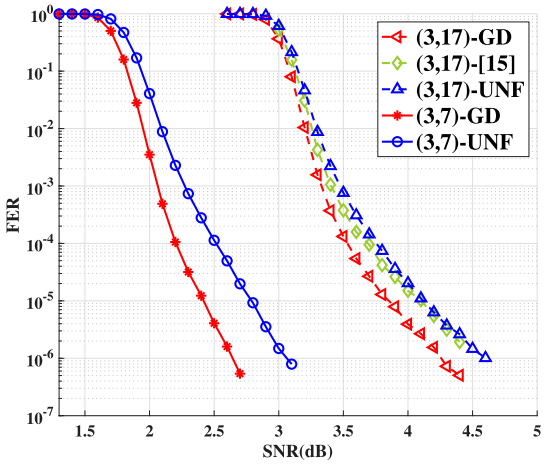


Fig. 11. FER curves of GD/UNF codes with $\gamma = 3$ over the AWGN channel. The leftmost two FER curves correspond to GD/UNF codes with parameters $(\gamma, \kappa, m, z, L) = (3, 7, 5, 13, 100)$. The rightmost curves colored with red and blue correspond to GD/UNF codes with parameters $(\gamma, \kappa, m, z, L) = (3, 17, 9, 7, 100)$. The green curve corresponds to the code with parameters $(\gamma, \kappa, m, z, L) = (3, 17, 70, 1, 700)$, the partitioning matrix of which is $\mathbf{P}_{C_2}$ from [17, Equation (46)].

Fig. 11 shows FER curves of our GD/UNF comparisons with $(\gamma, \kappa) = (3, 7)$ and $(3, 17)$. The partitioning matrices and the lifting matrices of the codes are specified in Appendix A and Appendix B. When $\gamma = 3$, cycles-6 are easily removed by the CPO. Therefore, we perform joint optimization on the number of cycles-6 and cycles-8 candidates by assigning different weights to cycle candidates in Algorithm 2. We observe a performance gain for the GD code with respect to the UNF code in both the waterfall region and the error floor region. The FER curve of the code [17] is shown to lay between FER curves of the GD and UNF codes in the waterfall region, and close to the UNF codes in the error floor region. Moreover, the number of cycles-8 in the $(3, 17)$ GD code is reduced by 29% and 12% compared with the UNF code and the code constructed by Battaglioni *et al.* in [17], respectively. In addition, the $(3, 17)$ GD code has no weight-6 absorbing

TABLE II
STATISTICS OF THE NUMBER OF CONCATENATED CYCLES

(γ, κ)	Code	Cycles-6	2-1-2 Objects	2-2-2 Objects	2-1-3 Objects	3-1-3 Objects
$(4, 24)$ (NLM)	GD	4,794	1,751	807,534	1,162,035	125,869,717
	TC	4,352	4,301	816,816	1,125,400	126,903,436
	UNF	11,713	7,293	1,308,762	2,615,297	208,425,933
$(4, 24)$ (BSC)	GD	4,794	1,802	822,120	1,212,100	126,514,918
	TC	4,403	4,097	807,534	1,139,000	126,434,525
	UNF	11,713	7,293	1,308,762	2,615,297	208,425,933
$(4, 20)$	GD	2,171	338	178,334	238,160	19,638,372
	TC	2,665	1,404	178,828	262,509	20,571,499
	UNF	2,444	2,860	287,014	540,865	34,362,393

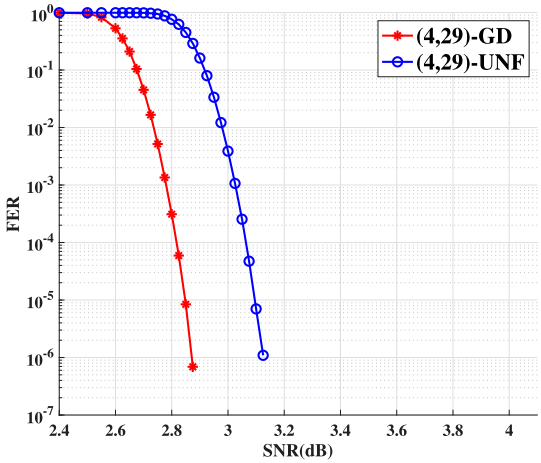


Fig. 12. FER curves of GD/UNF codes with $(\gamma, \kappa, m, z, L) = (4, 29, 19, 29, 20)$ over the AWGN channel.

sets (ASs) and 133 weight-7 ASs, whereas the UNF code has 6 weight-6 ASs and 361 weight-7 ASs. As for the $(3, 7)$ GD code, all cycles-6 and cycles-8 are removed, while the $(3, 7)$ UNF code still has 6292 cycles-8. Thus, the gain of the GD code compared with the UNF code exceeds the gain observed in the $(3, 17)$ codes.

Fig. 12 shows FER curves of the GD/UNF comparison with $(\gamma, \kappa) = (4, 29)$. The partitioning matrices and the lifting matrices of the codes are specified in Appendix C. Cycles-6 in the GD code and the UNF code are both removed, and the number of cycles-8 in the GD code demonstrates a 51.4% reduction from the count observed in the UNF code. It is worth mentioning that both codes have no ASs of weights up to 8, which is reflected in their FER curves via the sharp waterfall regions and the non-existing error floor regions. The FER of the GD/UNF codes decreases with a rate exceeding 12 orders of magnitude per 0.5 dB signal-to-noise ratio (SNR) increase. Moreover, the GD code has a significant gain of about 0.25 dB over the UNF code.

Fig. 13 shows the FER curves of the TC/SC codes with $(\gamma, \kappa) = (4, 17)$. The partitioning matrices and the lifting matrices of the codes are specified in Appendix D. The number of cycles-6 in the $(4, 17)$ TC code demonstrates a 79% and a 20% reduction from the counts observed in the SC codes with a matched constraint length and a matched circulant size, respectively. Moreover, the TC code has no weight-6 nor weight-8 ASs. It is shown that the TC code outperforms the optimal SC code with a matched constraint length, and that the gain is of greater magnitude when compared with the SC code of identical circulant size.

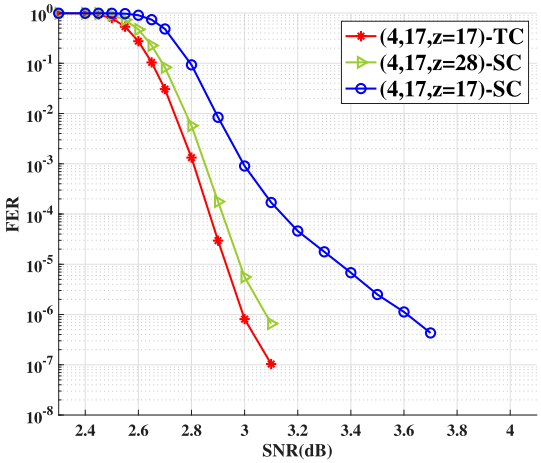


Fig. 13. FER curves of TC/SC codes with $(\gamma, \kappa) = (4, 17)$ over the AWGN channel. The red curve corresponds to a TC code with parameters $(\gamma, \kappa, m, z, L) = (4, 17, 4, 17, 50)$ and $\mathbf{a} = (0, 1, 4)$. The green and blue curves correspond to SC codes with a similar constraint length ($(\gamma, \kappa, m, z, L) = (4, 17, 2, 28, 30)$) and an identical circulant size ($(\gamma, \kappa, m, z, L) = (4, 17, 2, 17, 50)$), respectively.

Remark 9: Note that although TC codes have higher memories and thus larger constraint lengths than SC codes of matched circulant sizes, they possess the same number of nonzero component matrices, and thus the same degrees of freedom in construction. This fact makes TC codes even more promising if we can devise for them windowed decoding algorithms with window sizes that are comparable to the corresponding SC codes of matched circulant sizes.

B. Optimization Over Concatenated Cycles

In this subsection, we simulate codes constructed based on minimizing the number of concatenated-cycle pairs using the GRADE-AO specified in Section IV and Section V-A. We compare GD/UNF codes with $m = 6$ in addition to TC codes with $m_t = 3$ and $\mathbf{a} = (0, 1, 4, 6)$ on the binary symmetric channel (BSC), Flash channel, and magnetic recording channel. There are two groups of GD/TC/UNF codes that have parameters $(\gamma, \kappa, m, z, L) = (4, 24, 6, 17, 40)$ and $(4, 20, 6, 13, 20)$, respectively. The statistics regarding the number of cycles of each code are presented in Table II.

Fig. 14 shows UBER curves of the GD/TC/UNF codes with $(\gamma, \kappa) = (4, 24)$ on a Flash channel.[7] The Flash channel used in this section is a practical, asymmetric Flash channel, which is the normal-Laplace mixture (NLM) Flash channel [37].

[7]Note that although we only provide simulation results on some typical channels for brevity in this paper, our approach is generally applicable to many other channels, such as the ones underlying three-dimensional cross point (3D XPoint) [35] and two-dimensional magnetic recording (TDMR) [36] systems.

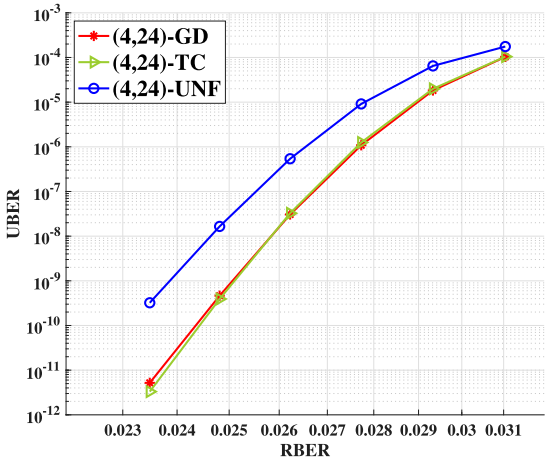


Fig. 14. UBER curves of GD/TC/UNF codes with $(\gamma, \kappa, m, z, L) = (4, 24, 6, 17, 40)$ over the NLM channel, where the TC code has the coupling pattern $\mathbf{a} = (0, 1, 4, 6)$ and the edge distribution obtained in Example 5. The codes are non-binary codes on GF(4).

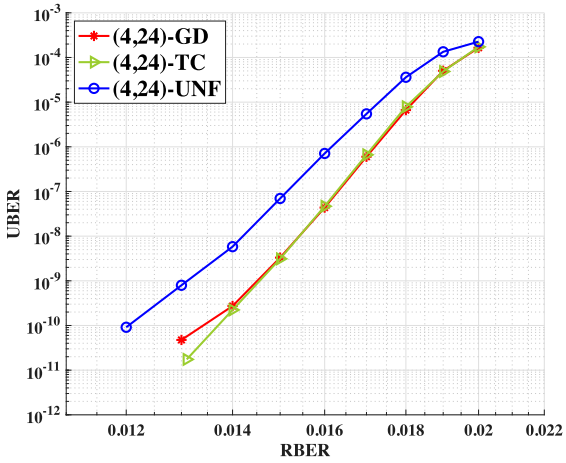


Fig. 15. UBER curves of GD/TC/UNF codes with $(\gamma, \kappa, m, z, L) = (4, 24, 6, 17, 40)$ over the BSC. The partitioning matrices of these codes are identical with partitioning matrices of GD/TC/UNF codes simulated over the NLM channel in Fig. 14.

In the NLM channel, the threshold voltage distribution of sub-20nm multi-level cell (MLC) Flash memories is carefully modeled. The four levels are modeled as different NLM distributions, incorporating several sources of error due to wear-out effects, e.g., programming/erasing problems, thereby resulting in significant asymmetry. Furthermore, the authors of [37] provided accurate fitting results of their model for program/erase (P/E) cycles up to 10 times the manufacturer's endurance specification (up to 30,000 P/E cycles). We implemented the NLM channel based on the parameters described in [37]. Here, we use 3 threshold voltage reads, and the sector size is 512 bytes. For decoding, we use a finite-precision (FP) fast Fourier transform based q-ary sum-product algorithm (FFT-QSPA) LDPC decoder [38]. The decoder performs a maximum of 50 iterations, and it stops if a codeword is reached sooner.

The partitioning matrices and the lifting matrices of the codes are specified in Appendix F. The non-binary edge weights are set as in [39] and [40]. The codes can be further optimized by applying the more advanced WCM framework presented in [22] and [23]. The first row of Table II shows the statistics of unlabeled cycles and concatenated cycles in each code. The number of objects in GD/TC codes are reduced by around 40% compared with the UNF code. No error floors are observed in any one of the UBER curves. The UBER of the GD/TC codes decreases with a rate exceeding 14 orders of magnitude per 0.01 RBER decrease. Moreover, the GD/TC codes have a significant gain of about 2 orders of magnitude over the UNF code at RBER 0.0235. It is worth mentioning that the UBER curves of the GD/TC codes nearly overlap, which is in accordance with the closeness of the statistics of objects in them.

While NB codes are adopted in the simulations over the NLM channel, independent coding are more widely applied in practical Flash solutions in order to preserve high access speed. Therefore, we next present in Fig. 15 the UBER curves of the GD/TC/UNF codes with $(\gamma, \kappa) = (4, 24)$ over the BSC, as a simplified model of single-level cell (SLC) channel with 1 threshold voltage read.

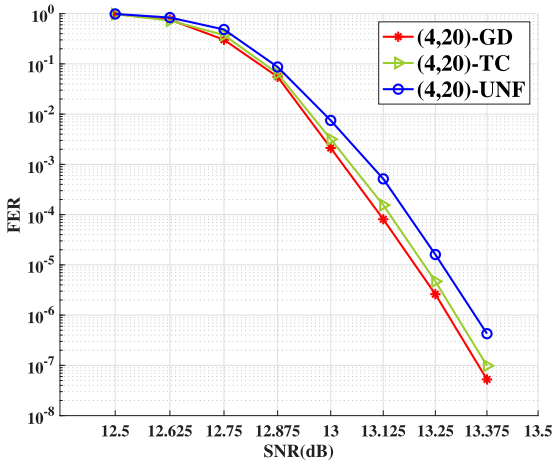


Fig. 16. FER curves of GD/TC/UNF codes with $(\gamma, \kappa, m, z, L) = (4, 20, 6, 13, 20)$ over the MR channel.

The partitioning matrices and the lifting matrices of the codes are specified in Appendix E. While partitioning matrices of the $(4, 24)$ NB codes constructed for the NLM channels are adopted as they are here, we have modified the lifting parameters slightly in order to remove all unlabeled ASs with weights less than or equal to 7 in the GD/TC codes: this is achieved by changing one entry in each lifting matrix. According to Table II, the number of objects in the GD/TC codes has not changed dramatically, and they still demonstrate a 40% reduction compared with the count observed in the UNF code. As shown in Fig. 15, the UBER curves of GD/TC codes are still close in the early waterfall region like they are over the NLM channel simulations; however, they start to deviate at RBER less than 0.014. The TC code has no observed error floor in its performance curve as expected, and it has a 2 orders of magnitude gain over the UNF code at RBER 0.013. The GD code curve surprisingly floors despite that there are no ASs with weight less than 8 in it: error profile analysis shows that the error floor at RBER 0.013 results from only 2 different large weight errors (one of weight 78 and another of weight 168) instead of structured small weight errors.

2	0	5	5	0	0	4
5	5	0	0	1	1	0
0	2	4	5	5	5	0

3	12	3	2	0	4	9
5	2	4	6	8	10	12
0	8	3	11	6	1	9

(a) GD code.

1	3	0	3	2	5	5
4	0	2	0	5	3	1
0	4	2	1	4	2	1

2	5	6	4	0	3	0
9	2	8	11	5	10	12
0	8	3	11	1	1	9

(b) UNF code.

Fig. 17. Partitioning matrices (left) and lifting matrices (right) of GD/UNF codes with $(\gamma, \kappa, m, z, L) = (3, 7, 5, 13, 100)$.

7	8	0	9	9	9	7	0	0	2	9	9	0	3	3	7	3
9	5	9	1	0	0	9	6	4	7	2	2	4	0	0	9	8
1	1	8	0	8	8	0	9	9	6	0	0	9	9	9	0	0

0	6	0	3	0	6	0	6	6	6	4	2	0	0	6	0	0
3	2	2	6	2	3	4	0	0	0	6	1	2	5	0	1	4
5	1	0	3	6	6	6	0	1	2	3	4	5	6	0	1	2

(a) GD code.

9	2	9	4	8	8	4	6	0	3	9	3	1	2	5	8	3
3	9	8	1	0	8	4	0	5	7	7	0	4	9	5	0	3
5	6	2	9	1	0	7	2	7	1	2	7	6	0	8	1	6

0	3	2	2	2	0	3	0	2	1	0	1	2	0	0	0	0
4	2	4	6	1	1	2	1	0	6	1	1	3	3	0	1	4
4	1	2	1	4	5	6	0	1	2	3	4	5	6	0	1	2

(b) UNF code.

Fig. 18. Partitioning matrices (top) and lifting matrices (bottom) of GD/UNF codes with $(\gamma, \kappa, m, z, L) = (3, 17, 9, 7, 100)$.

Remark 10: While the reason why the large weight errors observed in the error profile of the GD code are detrimental over the BSC simulations remains unexplored and is left for future investigation, the TC code is observed to be robust against these errors, which is specifically intriguing. Moreover, the fact that the waterfall performance of GD/TC codes is remarkably superior to that of UNF codes calls for an asymptotic analysis that takes edge distribution into consideration. The significant gain achieved by TC codes substantiates the potential of TC codes in Flash memories.

Fig. 16 shows FER curves of the GD/TC/UNF codes with $(\gamma, \kappa) = (4, 20)$ over the MR channel. The MR system adopts the partial-response (PR) channel presented in [32] and sequence detection. This PR channel incorporates the MR channel effects: inter-symbol interference (intrinsic memory), jitter, and electronic noise. The normalized channel density [32], [41] is 1.4, and the PR equalization target is $(8, 14, 2)$. The filtering units are followed by a Bahl-Cocke-Jelinek-Raviv (BCJR) detector [42], which is based on pattern-dependent noise prediction (PDNP) [43], and again an FP FFT-QSPA ($q = 2$) LDPC decoder [24]. The number of global (detect-decoder) iterations is 10, and the number of local (decoder only) iterations is 20. Unless a codeword is reached, the decoder performs its prescribed number of local iterations for each global iteration. More details can be found in [32].

The partitioning matrices and the lifting matrices of the codes are specified in Appendix G. The number of targeted objects (concatenated-cycle pairs) observed in the GD code demonstrates an approximate 40% reduction from the count observed in the UNF code. The FER of the GD/TC codes decreases with a rate that is approximately 13 orders of magnitude per 1 dB SNR increase. Moreover, the GD code has a significant gain of about one order of magnitude over the UNF code at SNR 13.375 dB. These results substantiate the remarkable impact of the GRADE-AO method in constructing SC codes with superior performance for storage devices, with potential usage in further applications including wireless communication systems.

VII. Conclusion

Discrete optimization of the constructions of spatially-coupled (SC) codes with high memories is known to be computationally expensive. Heuristic algorithms are efficient, but their lack of theoretical guidance makes it difficult to guarantee satisfactory performance. In this paper, we proposed the so-called GRADE-AO method, a probabilistic framework that efficiently searches for locally optimal QC-SC codes with arbitrary memories. We obtained a locally optimal edge distribution that minimizes the expected number of the most detrimental objects via gradient descent. Starting from a random partitioning matrix with the derived edge distribution, we then applied a semi-greedy algorithm to find a locally optimal partitioning matrix near it. While the application of GRADE-AO in optimizing the number of short cycles has shown noticeable gains, we focused in this paper on minimizing the number of more detrimental objects, the concatenated cycles. This finer-grained optimization avoids unnecessary attention on individual cycles which are typically not problematic on their own, especially in codes with high VN degrees and irregular codes. Simulation results show that our proposed constructions have a significant performance gain over state-of-the-art codes; this gain is shown to be universal in both waterfall and error floor regions, as well as on channels underlying various practical systems. Future work includes extending the framework to other classes of underlying block codes.

Appendix A
Partitioning Matrices and Lifting Matrices for $(3, 7)$ Codes on AWGN Channel

See Fig. 17.

Appendix B
Partitioning Matrices and Lifting Matrices for $(3, 17)$ Codes on AWGN Channel

See Fig. 18.

Appendix C
Partitioning Matrices and Lifting Matrices for $(4, 29)$ Codes on AWGN Channel

See Figs. 19 and 20.

Appendix D
Partitioning Matrices and Lifting Matrices for $(4, 17)$ Codes on AWGN Channel

See Fig. 21.

0	0	0	19	17	17	1	11	13	5	18	10	19	13	1	6	19	8	19	0	19	0	0	0	2	17	6	19	4
19	18	18	2	0	19	19	5	3	19	9	2	9	9	3	17	6	0	2	16	12	13	8	18	16	0	17	10	0
1	14	3	16	7	1	4	19	5	0	0	16	0	0	7	19	10	19	16	18	3	18	15	3	19	8	19	1	15
16	0	14	1	11	2	15	2	19	16	18	0	19	19	19	0	0	5	1	0	9	4	19	14	7	12	0	19	1

7	1	18	21	5	4	17	0	6	16	26	8	13	7	5	6	9	2	0	0	0	0	0	5	0	4	19	0	0
3	15	4	1	5	3	12	19	10	21	19	3	4	19	28	1	3	5	12	18	11	10	15	17	19	21	18	25	27
0	8	16	24	3	11	20	20	6	14	22	1	9	2	25	21	12	7	28	6	15	23	19	10	0	1	4	13	21
0	18	7	25	1	3	21	10	28	17	6	24	13	2	20	9	27	16	5	23	12	1	19	8	26	15	4	22	11

Fig. 19. Partitioning matrix (top) and lifting matrix (bottom) for GD code with $(\gamma, \kappa, m, z, L) = (4, 29, 19, 29, 20)$.

0	17	3	13	1	14	4	12	4	15	10	2	17	2	18	11	17	15	11	3	13	12	13	6	2	5	14	13	14
8	0	19	19	18	8	5	18	13	6	11	3	2	4	11	3	9	15	16	7	7	12	19	16	4	9	0	13	3
14	6	12	10	12	1	17	9	7	5	16	19	1	15	5	19	6	5	15	7	0	2	3	10	15	9	6	7	11
17	14	0	2	9	18	12	1	8	11	4	4	7	10	1	8	14	8	0	16	17	16	1	0	10	18	18	10	8

12	1	1	7	14	27	4	26	25	2	0	6	15	7	24	1	1	6	17	5	13	19	2	0	11	0	0	0	1
5	2	4	22	8	5	23	1	4	18	28	1	19	17	22	6	3	3	14	9	11	13	15	3	0	2	3	25	27
23	8	2	24	3	7	1	27	6	14	21	12	9	17	5	4	12	20	28	7	7	13	2	25	18	26	5	13	21
28	18	7	4	14	3	21	10	28	17	6	24	13	2	9	8	1	26	5	23	12	1	19	8	26	15	4	22	11

Fig. 20. Partitioning matrix (top) and lifting matrix (bottom) for UNF code with $(\gamma, \kappa, m, z, L) = (4, 29, 19, 29, 20)$.

4	4	4	4	4	0	0	0	1	1	4	1	1	1	0	0	0
1	1	0	0	0	1	1	1	0	0	0	4	4	4	4	4	1
0	0	1	1	1	4	4	4	4	4	1	0	0	0	1	1	4
1	1	1	1	1	0	0	0	0	0	0	4	4	4	4	4	4

16	7	5	15	0	2	9	2	1	3	2	7	0	12	7	13	13
0	6	4	0	10	10	12	14	9	1	11	2	7	1	16	4	13
1	8	16	7	15	6	14	5	13	3	5	3	9	11	7	0	11
0	1	2	3	4	5	6	7	8	9	1	11	9	5	4	15	16

(a) TC code with $(z, L) = (4, 17, 2, 17, 50)$ and $\mathbf{a} = (0, 1, 4)$.

2	2	2	2	2	0	0	0	1	1	2	1	1	1	0	0	0
1	1	0	0	0	1	1	1	0	0	0	2	2	2	2	2	2
0	0	1	1	1	2	2	2	2	2	1	0	0	0	1	1	1
1	1	1	1	1	0	0	0	0	0	0	2	2	2	2	2	2

5	5	12	1	7	16	10	0	0	0	14	15	1	10	7	0	10
0	2	15	0	2	9	12	14	16	1	3	5	7	9	11	13	15
1	8	16	7	15	6	16	5	13	4	12	3	12	2	10	1	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

21	27	3	13	1	5	27	16	21	4	0	15	26	2	11	8	0
15	11	1	20	22	11	21	20	3	0	22	22	24	26	0	2	4
25	13	22	24	9	12	15	0	18	16	24	21	12	20	0	8	18
0	18	8	26	16	17	24	14	1	22	12	2	11	10	0	18	8

(b) SC codes with $(z, L) = (17, 50)$ (middle) and $(z, L) = (28, 30)$ (bottom).

Fig. 21. Partitioning matrices (top) and lifting matrices (bottom) for TC/SC codes with $(\gamma, \kappa, m_t) = (4, 17, 2)$.

0	6	2	0	6	1	0	1	6	6	6	6	0	0	0	4	5	4	5	0	0	5	0	1
0	0	6	2	6	6	6	0	1	1	5	6	4	6	0	0	1	6	6	0	6	1	4	0
3	3	6	6	0	6	1	5	3	0	0	1	2	6	6	6	2	0	0	6	4	6	0	6
6	5	1	5	2	0	3	5	0	3	0	0	6	0	6	3	6	2	0	6	0	0	6	4

11	15	1	7	8	11	14	1	5	6	16	9	12	0	5	13	1	0	3	5	15	0	0	1
9	2	8	6	8	10	10	3	5	8	5	5	7	12	9	14	15	0	2	4	6	8	10	12
0	2	6	7	15	12	4	5	13	4	7	8	11	2	1	1	8	0	8	16	9	15	11	5
0	1	2	3	3	5	6	7	14	8	14	11	12	13	14	15	16	0	1	2	3	4	11	6

Fig. 22. Partitioning matrix (top) and lifting matrix (bottom) for GD Code with $(\gamma, \kappa, m, z, L) = (4, 24, 6, 17, 40)$.

0	1	4	6	6	1	0	1	4	6	4	6	6	6	0	6	1	1	1	0	4	0	1	0
1	6	0	6	4	6	6	0	1	6	1	6	0	0	6	0	0	6	6	4	0	1	0	4
6	6	6	0	0	0	1	4	0	0	6	1	1	4	6	4	6	4	0	4	6	6	4	0
4	0	1	1	1	6	4	6	6	0	0	1	4	4	0	1	6	0	4	6	4	4	6	6

13	4	1	7	1	11	3	6	6	6	1	12	15	14	5	4	0	12	0	0	0	0	0	0
5	11	4	9	13	15	0	14	11	1	7	8	7	9	9	12	2	0	2	4	6	14	4	16
0	8	16	1	15	6	14	5	14	4	9	3	11	2	10	1	9	0	8	13	12	15	6	11
0	9	2	3	4	5	6	2	8	9	8	4	12	13	14	15	0	9	15	13	3	2	8	4

Fig. 23. Partitioning matrix (top) and lifting matrix (bottom) for TC code with $(\gamma, \kappa, m, z, L) = (4, 24, 6, 17, 40)$.

0	5	6	5	0	6	2	2	5	1	2	6	2	0	6	2	2	3	2	3	0	3	1	1
5	4	0	5	0	2	3	1	0	0	4	0	1	4	3	6	2	4	6	4	6	6	4	3
1	1	5	0	6	4	6	5	1	5	1	4	2	5	0	3	1	3	2	1	5	3	4	3
6	1	2	2	4	4	1	5	6	6	3	1	5	0	0	4	5	1	4	3	0	0	3	6

8	16	7	7	0	9	6	14	6	14	1	13	14	8	0	0	4	6	0	0	0	0	3	16
3	2	6	6	13	10	12	14	10	3	3	11	7	9	11	7	15	0	13	4	4	8	10	12
2	8	16	1	15	6	14	5	13	2	12	3	11	2	13	1	9	0	8	16	7	15	6	14
1	1	2	3	13	5	11	7	8	9	10	11	12	13	14	15	16	0	1	2	3	4	5	6

Fig. 24. Partitioning matrix (top) and lifting matrix (bottom) for UNF code with $(\gamma, \kappa, m, z, L) = (4, 24, 6, 17, 40)$.

0	6	2	0	6	1	0	1	6	6	6	6	0	0	0	4	5	4	5	0	0	5	0	1
0	0	6	2	6	6	6	0	1	1	5	6	4	6	0	0	1	6	6	0	6	1	4	0
3	3	6	6	0	6	1	5	3	0	0	1	2	6	6	6	2	0	0	6	4	6	0	6
6	5	1	5	2	0	3	5	0	3	0	0	6	0	6	3	6	2	0	6	0	0	6	4

16	15	1	7	8	11	14	1	5	6	16	9	12	0	5	13	1	0	3	5	15	0	0	1
9	2	8	6	8	10	10	3	5	8	5	5	7	12	9	14	15	0	2	4	6	8	10	12
0	2	6	7	15	12	4	5	13	4	7	8	11	2	1	1	8	0	8	16	9	15	11	5
0	1	2	3	3	5	6	7	14	8	14	11	12	13	14	15	16	0	1	2	3	4	11	6

Fig. 25. Partitioning matrix (top) and lifting matrix (bottom) for GD code with $(\gamma, \kappa, m, z, L) = (4, 24, 6, 17, 40)$.

0	1	4	6	6	1	0	1	4	6	4	6	6	6	0	6	1	1	1	0	4	0	1	0
1	6	0	6	4	6	6	0	1	6	1	6	0	0	6	0	0	6	6	4	0	1	0	4
6	6	6	0	0	0	1	4	0	0	6	1	1	4	6	4	6	4	0	4	6	6	4	0
4	0	1	1	1	6	4	6	6	0	0	1	4	4	0	1	6	0	4	6	4	4	6	6

13	4	1	7	1	11	3	6	6	6	1	12	15	14	5	4	0	12	0	0	0	0	0	0
5	11	4	9	13	15	0	14	11	1	7	8	7	9	9	12	2	0	2	4	6	14	4	16
0	8	16	1	15	6	14	5	14	4	9	3	11	2	10	1	9	0	8	13	12	15	6	11
0	5	2	3	4	5	6	2	8	9	8	4	12	13	14	15	0	9	15	13	3	2	8	4

Fig. 26. Partitioning matrix (top) and lifting matrix (bottom) for TC code with $(\gamma, \kappa, m, z, L) = (4, 24, 6, 17, 40)$.

0	5	6	5	0	6	2	2	5	1	2	6	2	0	6	2	2	3	2	3	0	3	1	1
5	4	0	5	0	2	3	1	0	0	4	0	1	4	3	6	2	4	6	4	6	6	4	3
1	1	5	0	6	4	6	5	1	5	1	4	2	5	0	3	1	3	2	1	5	3	4	3
6	1	2	2	4	4	1	5	6	6	3	1	5	0	0	4	5	1	4	3	0	0	3	6

8	16	7	7	0	9	6	14	6	14	1	13	14	8	0	0	4	6	0	0	0	0	3	16
3	2	6	6	13	10	12	14	10	3	3	11	7	9	11	7	15	0	13	4	4	8	10	12
2	8	16	1	15	6	14	5	13	2	12	3	11	2	13	1	9	0	8	16	7	15	6	14
1	1	2	3	13	5	11	7	8	9	10	11	12	13	14	15	16	0	1	2	3	4	5	6

Fig. 27. Partitioning matrix (top) and lifting matrix (bottom) for UNF code with $(\gamma, \kappa, m, z, L) = (4, 24, 6, 17, 40)$.

0	0	3	6	0	1	2	2	0	6	6	0	6	5	6	0	6	0	5	6
3	0	1	0	5	0	6	3	2	0	6	3	1	6	6	6	0	6	6	5
6	6	0	1	1	6	1	6	5	0	0	4	5	0	0	4	0	5	2	3
1	6	6	4	6	4	6	0	6	4	0	6	0	1	0	0	6	2	0	0

3	5	4	8	3	11	0	2	0	1	0	9	6	9	0	0	2	2	1	1
7	2	6	6	8	4	8	1	2	6	6	4	6	7	7	7	6	8	5	10
0	7	3	11	0	7	9	1	12	0	2	10	5	0	8	3	11	7	1	9
0	5	10	2	7	12	4	11	1	6	11	5	8	7	5	10	2	7	12	4

Fig. 28. Partitioning matrix (top) and lifting matrix (bottom) for GD code with $(\gamma, \kappa, m, z, L) = (4, 20, 6, 13, 20)$.

6	0	0	6	0	0	4	6	0	1	6	6	6	1	6	6	0	0	1	1
1	6	4	4	4	1	1	6	4	6	0	0	0	6	0	4	6	0	4	6
4	0	6	4	1	6	1	1	4	4	4	6	0	0	1	0	1	6	6	4
0	6	4	0	6	4	6	0	6	0	1	1	4	4	6	1	6	6	0	0

1	2	8	5	11	3	0	1	10	4	1	12	10	0	10	3	0	6	0	0
8	7	4	12	5	10	12	1	3	5	10	10	4	8	2	7	6	12	10	12
0	11	3	11	0	1	9	4	12	7	8	4	9	10	8	3	11	2	1	10
0	12	10	2	7	12	4	9	1	6	4	10	11	0	5	9	2	7	12	4

Fig. 29. Partitioning matrix (top) and lifting matrix (bottom) for TC code with $(\gamma, \kappa, m, z, L) = (4, 20, 6, 13, 20)$.

0	0	6	5	5	1	5	1	6	3	3	2	2	3	2	6	0	3	0	2
5	5	2	1	1	2	5	2	0	3	5	3	5	5	1	6	6	1	2	2
6	0	1	5	2	4	1	4	4	0	4	4	6	0	2	0	6	3	4	4
0	4	0	3	4	5	1	5	1	6	1	0	1	4	6	3	0	6	3	6

6	11	1	1	4	9	11	0	3	11	12	0	0	1	0	9	2	0	10	1
10	3	4	5	9	2	2	3	8	5	7	9	5	12	0	4	6	8	10	12
12	0	3	11	5	1	9	12	12	7	4	4	5	12	3	7	11	6	0	9
9	5	10	2	7	10	4	9	1	6	11	5	8	5	5	11	9	7	12	4

Fig. 30. Partitioning matrix (top) and lifting matrix (bottom) for UNF code with $(\gamma, \kappa, m, z, L) = (4, 20, 6, 13, 20)$.

APPENDIX E
PARTITIONING MATRICES AND LIFTING MATRICES FOR $(4, 24)$ CODES IN SIMULATIONS ON BSC

See Figs. 22–24.

APPENDIX F
PARTITIONING MATRICES AND LIFTING MATRICES FOR $(4, 24)$ CODES IN SIMULATIONS ON NLM CHANNEL

See Figs. 25–27.

APPENDIX G
PARTITIONING MATRICES AND LIFTING MATRICES FOR $(4, 20)$ CODES ON MR CHANNEL

See Figs. 28–30.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor Dr. David Mitchell and the anonymous reviewers for their careful reading of our manuscript and their insightful comments and suggestions, also would like to thank Shyam Venkatasubramanian for his assistance in carrying out part of the simulations in this research, and also would like to thank Christopher Cannella and Arnab Kar for useful discussions regarding probabilistic optimization on SC code constructions while being at Duke University.

REFERENCES

[1] S. Yang, A. Hareedy, S. Venkatasubramanian, R. Calderbank, and L. Dolecek, "GRADE-AO: Towards near-optimal spatially-coupled codes with high memories," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 587–592.

[2] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.

[3] S. Kumar, A. J. Young, N. Macris, and H. D. Pfister, "Threshold saturation for spatially coupled LDPC and LDGM codes on BMS channels," *IEEE Trans. Inf. Theory*, vol. 60, no. 12, pp. 7389–7415, Dec. 2014.

[4] P. M. Olmos and R. L. Urbanke, "A scaling law to predict the finite-length performance of spatially-coupled LDPC codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3164–3184, Jun. 2015.

[5] A. Hareedy, H. Esfahanizadeh, and L. Dolecek, "High performance non-binary spatially-coupled codes for flash memories," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2017, pp. 229–233.

[6] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.

[7] A. R. Iyengar, P. H. Siegel, R. L. Urbanke, and J. K. Wolf, "Windowed decoding of spatially coupled codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2277–2292, Apr. 2013.

[8] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, "Spatially coupled LDPC codes constructed from protographs," *IEEE Trans. Inf. Theory*, vol. 61, no. 9, pp. 4866–4889, Sep. 2015.

[9] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "Finite-length construction of high performance spatially-coupled codes via optimized partitioning and lifting," *IEEE Trans. Communications*, vol. 67, no. 1, pp. 3–16, Jan. 2018.

[10] A. Hareedy, R. Wu, and L. Dolecek, "A channel-aware combinatorial approach to design high performance spatially-coupled codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 8, pp. 4834–4852, Aug. 2020.

[11] A. E. Pusane, R. Smarandache, P. O. Vontobel, and D. J. Costello, "Deriving good LDPC convolutional codes from LDPC block codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 835–857, Feb. 2011.

[12] 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 Trans. Inf. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.

[13] S. Naseri and A. H. Banihashemi, "Spatially coupled LDPC codes with small constraint length and low error floor," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 254–258, Feb. 2020.

[14] S. Naseri and A. H. Banihashemi, "Construction of time invariant spatially coupled LDPC codes free of small trapping sets," *IEEE Trans. Commun.*, vol. 69, no. 6, pp. 3485–3501, Jun. 2021.

[15] 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 (ISIT)*, Jun. 2017, pp. 2940–2944.

[16] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "A novel combinatorial framework to construct spatially-coupled codes: Minimum overlap partitioning," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 1693–1697.

[17] M. Battaglioni, A. Tasdighi, G. Cancellieri, F. Chiaraluce, and M. Baldi, "Design and analysis of time-invariant SC-LDPC convolutional codes with small constraint length," *IEEE Trans. Commun.*, vol. 66, no. 3, pp. 918–931, Mar. 2018.

[18] S. Mo, L. Chen, D. J. Costello, D. G. M. Mitchell, R. Smarandache, and J. Qiu, "Designing protograph-based quasi-cyclic spatially coupled LDPC codes with large girth," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5326–5337, Jun. 2020.

[19] M. Battaglioni, F. Chiaraluce, M. Baldi, and D. G. M. Mitchell, "Efficient search and elimination of harmful objects for the optimization of QC-SC-LDPC codes," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[20] A. Beemer, S. Habib, C. A. Kelley, and J. Kliewer, "A generalized algebraic approach to optimizing SC-LDPC codes," in *Proc. 55th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2017, pp. 672–679.

[21] A. Hareedy, R. Kuditipudi, and R. Calderbank, "Minimizing the number of detrimental objects in multi-dimensional graph-based codes," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5299–5312, Sep. 2020.

[22] A. Hareedy, C. Lanka, and L. Dolecek, "A general non-binary LDPC code optimization framework suitable for dense flash memory and magnetic storage," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 9, pp. 2402–2415, Sep. 2016.

[23] A. Hareedy, C. Lanka, N. Guo, and L. Dolecek, "A combinatorial methodology for optimizing non-binary graph-based codes: Theoretical analysis and applications in data storage," *IEEE Trans. Inf. Theory*, vol. 65, no. 4, pp. 2128–2154, Apr. 2019.

[24] M. P. C. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.

[25] L. Schmalen, V. Aref, and F. Jardel, "Non-uniformly coupled LDPC codes: Better thresholds, smaller rate-loss, and less complexity," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 376–380.

[26] A. Beemer, "Design and analysis of graph-based codes using algebraic lifts and decoding networks," Ph.D. dissertation, Dept. Math., Univ. Nebraska-Lincoln, Lincoln, NE, USA, 2018.

[27] Y. Wang, Y. Wang, J. S. Yedidia, and S. C. Draper, "Construction of high-girth QC-LDPC codes," in *Proc. 5th Int. Symp. Turbo Codes Rel. Topics*, Sep. 2008, pp. 180–185.

[28] I. E. Bocharova, R. Johannesson, F. Hug, B. D. Kudryashov, and R. V. Satyukov, "Searching for voltage graph-based LDPC tailbiting codes with large girth," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2265–2279, Apr. 2012.

[29] A. Tasdighi, A. H. Banihashemi, and M. R. Sadeghi, "Efficient search of girth-optimal QC-LDPC codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1552–1564, Feb. 2016.

[30] J. Wang, L. Dolecek, and R. D. Wesel, "The cycle consistency matrix approach to absorbing sets in separable circulant-based LDPC codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2293–2314, Apr. 2013.

[31] B. Amiri, J. Kliewer, and L. Dolecek, "Analysis and enumeration of absorbing sets for non-binary graph-based codes," *IEEE Trans. Commun.*, vol. 62, no. 2, pp. 398–409, Feb. 2014.

[32] A. Hareedy, B. Amiri, R. Galbraith, and L. Dolecek, "Non-binary LDPC codes for magnetic recording channels: Error floor analysis and optimized code design," *IEEE Trans. Commun.*, vol. 64, no. 8, pp. 3194–3207, Aug. 2016.

[33] J. Langford, L. Li, and T. Zhang, "Sparse online learning via truncated gradient," *J. Mach. Learn. Res.*, vol. 10, no. 3, pp. 1–25, 2009.

[34] S. Yang, *Application-Driven Coding Techniques: From Cloud Storage to Quantum Communications*. Los Angeles, CA, USA: University of California, Los Angeles, CA, USA, 2021.

[35] F. T. Hady, A. Foong, B. Veal, and D. Williams, "Platform storage performance with 3D XPoint technology," *Proc. IEEE*, vol. 105, no. 9, pp. 1822–1833, Sep. 2017.

[36] R. Wood, M. Williams, A. Kavcic, and J. Miles, "The feasibility of magnetic recording at 10 Terabits per square inch on conventional media," *IEEE Trans. Magn.*, vol. 45, no. 2, pp. 917–923, Feb. 2009.

[37] T. Parnell, N. Papandreou, T. Mittelholzer, and H. Pozidis, "Modelling of the threshold voltage distributions of sub-20nm NAND flash memory," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 2351–2356.

[38] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.

[39] L. Dolecek, D. Divsalar, Y. Sun, and B. Amiri, "Non-binary protograph-based LDPC codes: Enumerators, analysis, and designs," *IEEE Trans. Inf. Theory*, vol. 60, no. 7, pp. 3913–3941, Jul. 2014.

[40] A. Bazarsky, N. Presman, and S. Litsyn, "Design of non-binary quasi-cyclic LDPC codes by ACE optimization," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Sep. 2013, pp. 1–5.
[41] S. G. Srinivasa, Y. Chen, and S. Dahandeh, "A communication-theoretic framework for 2-D MR channel modeling: Performance evaluation of coding and signal processing methods," *IEEE Trans. Magn.*, vol. 50, no. 3, pp. 6–12, Mar. 2014.
[42] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.
[43] J. Moon and J. Park, "Pattern-dependent noise prediction in signal-dependent noise," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 4, pp. 730–743, Apr. 2001.

Siyi Yang (Member, IEEE) received the B.S. degree in electrical engineering from Tsinghua University, Beijing, China, in 2016, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California at Los Angeles (UCLA), in 2018 and 2021, respectively. She is currently a Post-Doctoral Associate with the Electrical and Computer Engineering Department, Duke University. She is broadly interested in error correction codes that are tailored for modern storage devices, communication systems, and quantum computing. She worked at Intel Corporation, Non-Volatile Memory Solution (NSG) Group, as a System on Chip (SoC) Design Engineer, in Summer 2020. She won the (2020–2021) Dissertation Year Fellowship (DYF) at UCLA.

Ahmed Hareedy (Member, IEEE) received the bachelor's and M.S. degrees in electronics and communications engineering from Cairo University, Egypt, in 2006 and 2011, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California at Los Angeles (UCLA) in 2018. He is currently an Assistant Professor with the Electrical and Electronics Engineering Department, Middle East Technical University (METU), Turkey. He is interested in questions in coding/information theory that are fundamental to opportunities created by the current, unparalleled access to data and computing. He was a Post-Doctoral Associate with the Electrical and Computer Engineering Department, Duke University, between 2018 and 2021. He worked with Mentor Graphics Corporation (currently, Siemens EDA) between 2006 and 2014. He worked as an Error-Correction Coding Architect with Intel Corporation in Summer 2015 and Summer 2017.

He was a recipient of the Best Paper Award from the 2015 IEEE Global Communications Conference (GLOBECOM), Selected Areas in Communications, Data Storage Track. He won the (2016–2017) Electrical Engineering Henry Samueli Excellence in Teaching Award for teaching Probability and Statistics at UCLA, where he won the (2017–2018) Dissertation Year Fellowship (DYF). He won the (2018–2019) Distinguished Ph.D. Dissertation Award in Signals and Systems from the Electrical and Computer Engineering Department, UCLA. He was also a recipient of the Memorable Paper Award from the 2018 Non-Volatile Memories Workshop (NVMW) in the area of devices, coding, and information theory. He was also a recipient of the (2018–2019) Best Student Paper Award from the IEEE Data Storage Technical Committee (DSTC). He has been recently awarded the TÜBİTAK 2232-B International Fellowship for Early Stage Researchers in 2022.

Robert Calderbank (Life Fellow, IEEE) received the B.S. degree in mathematics from Warwick University, U.K., in 1975, the M.S. degree in mathematics from Oxford University, U.K., in 1976, and the Ph.D. degree in mathematics from the California Institute of Technology in 1980.

He is currently a Professor in electrical and computer engineering at Duke University, where he directs the Rhodes Information Initiative at Duke (iiD). Prior to joining Duke in 2010, he was a Professor in electrical engineering and mathematics at Princeton University, where he directed the Program in applied and computational mathematics. Prior to joining Princeton in 2004, he was the Vice President of Research at AT&T, responsible for directing the first industrial research laboratory in the world where the primary focus is data at scale. At the start of his career at the Bell Laboratories, innovations by him were incorporated in a progression of voiceband modem standards that moved communications practice close to the Shannon limit. Together with Peter Shor and colleagues at the AT&T Laboratories, he developed the mathematical framework for quantum error correction. He is also a co-inventor of space-time codes for wireless communication, where correlation of signals across different transmit antennas is the key to reliable transmission.

Dr. Calderbank served as the Editor-in-Chief of the IEEE TRANSACTIONS ON INFORMATION THEORY from 1995 to 1998 and as an Associate Editor for Coding Techniques from 1986 to 1989. He was a member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and from 2006 to 2008. He was honored by the IEEE Information Theory Prize Paper Award in 1995 for his work on the $\mathbb{Z}_4$ linearity of Kerdock and Preparata Codes (joint with A. R. Hammons Jr., P. V. Kumar, N. J. A. Sloane, and P. Sole), and again in 1999 for the invention of space-time codes (joint with V. Tarokh and N. Seshadri). He has received the 2006 IEEE Donald G. Fink Prize Paper Award, the IEEE Millennium Medal, the 2013 IEEE Richard W. Hamming Medal, and the 2015 Shannon Award. He was elected to the U.S. National Academy of Engineering in 2005.

Lara Dolecek (Senior Member, IEEE) received the B.S. (Hons.), M.S., and Ph.D. degrees in electrical engineering and computer sciences and the M.A. degree in statistics from the University of California at Berkeley. She is currently a Full Professor with the Electrical and Computer Engineering Department and Mathematics Department (courtesy), University of California at Los Angeles (UCLA). She received the 2007 David J. Sakrison Memorial Prize for the Most Outstanding Doctoral Research with the Department of Electrical Engineering and Computer Sciences, UC Berkeley. Prior to joining UCLA, she was a Post-Doctoral Researcher with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology. She received IBM Faculty Award (2014), Northrop Grumman Excellence in Teaching Award (2013), Intel Early Career Faculty Award (2013), University of California Faculty Development Award (2013), Okawa Research Grant (2013), NSF Career Award (2012), and Hellman Fellowship Award (2011). With her research group and collaborators, she received numerous best paper awards. Her research interests span coding and information theory, graphical models, statistical methods, and algorithms, with applications to emerging systems for data storage and computing. She currently serves as an Associate Editor for IEEE TRANSACTIONS ON INFORMATION THEORY and as the Secretary of the IEEE Information Theory Society. She is a (2021–2022) Distinguished Lecturer of the IEEE Information Theory Society. She has served as a consultant for a number of companies specializing in data communications and storage.