Notes: Finite-length Protograph-based Spatially Coupled LDPC Codes

Abstract—The finite-length performance of multi-edge spatially coupled low-density parity-check (SC-LDPC) codes is analyzed over the binary erasure channel (BEC). Existing scaling laws are extended to arbitrary protograph base matrices including puncturing patterns and multiple edges between nodes. Compared to more involved rate-1/2 structures proposed to improve the threshold to minimum distance trade-off, constructions based on regular (4,8) protograph LDPC block codes work surprisingly well in the waterfall region. We design codes using scaling laws and estimate the block length an SC-LDPC code to match the performance of another given code. We also estimate the performance degradation if the chain length varies.

I. INTRODUCTION

state of the art: Spatially coupled Low-Density Parity-Check (SC-LDPC) codes have gained a lot of attention since it was shown that they achieve capacity over binary-input memoryless symmetric (BMS) channels under belief propagation (BP) decoding [1]. SC-LDPC codes are constructed by coupling $L$ LDPC codes with length $M$ each. For large $L$ and very large $M$, the BP threshold of this class of codes is very close to the maximum-a-posteriori (MAP) threshold of the underlying uncoupled LDPC code [1], [2].

For uncoupled LDPC code ensembles, it is well known that constructions based on protographs have practical advantages in comparison to regular and irregular constructions[3], [4].

The SC-LDPC code ensembles for which finite-length analyses exist were chosen to simplify the analysis rather than to construct strong codes [1], [5], [6]. There have been proposed several constructions of SC-LDPC codes which perform very well [7], [8]. The SC-LDPC ensembles are typically compared based on asymptotic arguments, e.g. threshold and minimum distance growth. However, the finite-length performance of these constructions is unknown since many of them are classes of multi-edge type LDPC codes which are hard to analyze. Proofs for achieving capacity over BMS channels [1] as well as finite-length performance analyses [5], [6] have been proposed only for random SC-LDPC codes.

contributions/what we achieved: We show that finite-length protograph-based codes provide better error rates than random constructions in both the waterfall and the error floor regions since their structure significantly increases the stability and the robustness of the decoding process. We also give the performance of higher structured code ensembles in the waterfall region where it turns out that more structure does not help in this region and match the performance of various code ensembles. We also scale the error rate with the coupling factor $L$.

structure of the paper/what we write: To show this, we extend the finite-length analysis based on scaling laws any cites? to protograph-based ensembles with puncturing and multiple edges. The finite-length performance of LDPC codes is analyzed over the binary erasure channel (BEC) where scaling laws between the finite-length performance and the LDPC code parameters can be analytically derived [9]. For the BEC, we consider a the peeling decoder (PD) with equivalent performance than BP decoding [3]. After removing all known variable nodes (VNs) from the Tanner graphs, PD iteratively resolves check nodes which yields a sequence of graphs whose statistics define the asymptotic as well as the finite-length properties of the code [10].

To obtain an estimate of the PD error probability, we proceed similar to [9]. We compute the average evolution of deg-1 check nodes for protograph-based ensembles under PD and develop a method for all constructions to estimate the variance around the expected evolution. The scaling law is obtained with a parameter correction in the SC-LDPC scaling law from [5], [6].

We then discuss the performance for regular (3,6) and (4,8) protograph-based ensembles as well as for the more structured (q,a, L)[7] and TAR4JAprotograph-based ensembles [8]. We also match the performance of code constructions by focussing on the stability in the steady-state regime and propose a method to estimate the performance degradation with increasing chain length.

We emphasize the MATLAB software publicly available on our website that calculates finite-length scaling parameters for user-specified SC-LDPC codes using their protograph base matrix.

II. CONSTRUCTIONS AND THRESHOLDS

Consider random constructions as in [11] and constructions based on protographs as in [12]. We will first give the random construction before we introduce the constructions based on protographs. We define the standard $(l, r)_P$ construction [13], a spatially coupled accumulate-repeat-accumulate (SC-ARA) construction [7] and a spatially coupled accumulate-repeat-by-4-jagged-accumulate (SC-TAR4A) construction [8].

A. Random $(l, r, L)$ Construction

At each of $L$ positions, an uncoupled $(l, r)$ regular LDPC code with $M$ variable nodes and $\frac{l}{r}$ check nodes is positioned. The variable nodes at each code position are then connected to one randomly chosen check node at each position $u - l, \ldots, u + l - 1$ so that we have $l$ connections per variable node. This way, we obtain $M$ variable nodes placed at positions $1, \ldots, L$ and $\frac{l}{r}M$ check nodes at positions $1, \ldots, L + (l - 1)$. Note that using this construction, it is guaranteed that each variable node is connected to a node of the respective blocks. However, it is not guaranteed that each check node within a
block has \(r\) connections. For \(L \to \infty\), the design rate tends to 
\[r = 1 - \frac{1}{r}\] which is the uncoupled code rate.

**B. Protograph-based Constructions**

Small Tanner graphs [14] called protographs are used as a blueprint for a larger structure. The protograph is copied several times. Connections between the same type of sockets are then randomly permuted between these copies to obtain larger girths since small girths often harm iterative decoding. This operation is also known as 'copy-and-permute'. The protograph can be used for analysis since it is a compact representation of structure of the resulting code.

**C. Standard \((l, r, L)_P\) Construction**

\(L\) protographs of a regular \((l, r)\) code are coupled to create an SC-LDPC protograph. Therefore, each variable node at position \(u\) is connected to the check nodes at the positions 
\[u, u + 1, \ldots, u + l - 1\] as shown in Fig. 1 for \((l, r, L)_P = (3, 6, 3)\). A code of the ensemble is obtained by copying and permuting \(M/k\) copies of the coupled protograph, where \(k = \frac{q}{l}\), \(k \in \mathbb{N}\) is the number of variable nodes per regular \((l, r)\) protograph. We calculate the code rate from the protograph as 
\[r_{(l, r, L)_P} = 1 - \frac{(L + l - 1)}{kL}\] This construction ensures regularity on the check node side where the construction in Section II-A allows more randomness.

![Fig. 1. A coupled \((l, r, L)_P = (3, 6, 3)_P\) protograph from small protographs.](image)

**D. \((q, a, L)\) SC-ARA Construction**

These codes as described in [7] base on the spatial coupling of accumulate-repeat-accumulate (ARA) codes [15]. ARA codes have a low error floor due to a linear growth of the minimum Hamming distance with the size of the code. Low-complexity encoding is also possible for spatially coupled constructions [7]. Let \(q\) be the degree of the message variable nodes and \(a\) the number of edges of the check node at a certain position connected to message variable nodes. While the message variable node of position \(u\) is connected to the check nodes at positions \(u, \ldots, u + a - 1\), the accumulator variable node at position \(u\) has two edges to the check node at the same position to keep the RA code accumulator structure as described in [7]. An example of a \((q, a, L) = (3, 3, 5)\) SC-ARA code is given in Fig. 2 where the accumulator nodes are marked grey. There are also added \(q - 1\) accumulator variable nodes at the termination to avoid deg-1 check nodes. Codes of the ensemble are obtained by copy-and-permute. The code rate is 
\[r_{(q, a, L)} = \frac{aL}{aL + qL + q - 1}\] We consider \((q, a, L) = (4, 4, L)\).

![Fig. 2. A coupled \((q, a, L) = (3, 3, 5)_P\) SC-ARA protograph.](image)

**E. SC-TAR4JA Construction**

The construction as described in [8] couples the accumulate-repeat-accumulate structure given in [16]. To construct C-LDPC codes from a base matrix \(B\) of a block code, certain edge types of the construction are connected to the following block code during the edge spreading procedure. In total, the number and the type of edges stays unchanged. Let \(R\) be the rate of the uncoupled protograph and \(m_s\) be the coupling memory. We obtain a code rate of 
\[r_{SC-TAR4JA} = 1 - \left(1 - \frac{1 + m_s}{L}\right) (1 - R)\] The used protograph construction is depicted in Fig. 3 for \(L = 3\) with 
\[r_{SC-TAR4JA} = 1 - \left(1 - \frac{1 + 3}{3}\right) (1 - \frac{1}{3})\]

![Fig. 3. A coupled SC-TAR4JA protograph with \(L = 3\). Punctured variable nodes are white.](image)

The minimum distances of these ensembles are compared in [17] where it is also shown that the minimum distance growth rates of the SC-LDPC codes are higher than for their uncoupled counterparts. Table I shows the threshold of all constructions introduced above for \(L = 100\). The best thresholds are obtained for the \((4, 8)_P\) and the TAR4JA ensemble.

**III. Degree distribution, BEC transmission and Peeling Decoding**

**A. Notation**

Vectors and matrices are denoted with \(v = (d_1, d_2, \ldots, d_n)\) and \(M\), respectively. \(0_{i,j,k}\) denotes a vector where all entries are zero except the entries at positions \(i, j, k\) whereas \(1_{i,j,k}\) denotes a vector where all entries are ones. \(|v| = \sum_{i=1}^{n} v_i\) is the sum of all entries of \(v\).

**B. Peeling decoding and degree distribution**

Consider a transmission over the binary erasure channel (BEC) and a peeling decoder (PD) [10]. The PD algorithm removes the variable nodes and their edges associated with non-erased symbols and any disconnected check nodes from the graph. The remaining edges form the residual graph. The PD proceeds by removing a check node connected to a single
variable node (referred to as a deg-1 check node), the variable node itself and all its adjacent edges. In [10], it was shown that the sequence of graphs follows a typical path or expected evolution. The graph degree distribution (DD) at any time during the decoding process constitutes a sufficient statistic for analysis. We define a proper DD for SC-LDPC protograph-based ensembles and characterize the DD of the residual graph along the decoding process.

Protograph LDPC ensembles are a particular class of multi-edge type (MET) LDPC codes. To define the DD, we use a MET-like notation as introduced in [3]. For any edge in the LDPC protograph connecting a different pair of two nodes, we define a new edge type. As an example, the protograph of an uncoupled (2, 4) LDPC ensemble is plotted in Fig. 4, including a labeling for each edge. Observe there are two edges of type 1 and two of type 2.

Let $m$ be the total number of different edge types in the protograph. With labelled edges in the graph, we define the type of a given variable(check) node in the protograph. The type of a given variable node is defined by a vector $d = (d_1, \ldots, d_m)$, where $d_j \in \mathbb{N}_0$ represents the number of edges of type $j$ connected to this variable node. Similarly, the type of a given check node is given by a vector $c$. Let $V_d(\Phi_v)$ represent the number of variable (check) nodes of multi-edge type $d$ ($c$). We denote by $\Phi_v$ ($\Phi_c$) the set of multi-edge variable (check) node types of the graph. As for single-type LDPC ensembles, the whole DD is represented using the multinomials

$$V(x) = \sum_{d \in \Phi_v} V_d x^d$$

$$R(x) = \sum_{c \in \Phi_c} R_c x^c$$

where $x^d = \prod_{i=1}^{m} x_i^{d_i}$. The total number of variable nodes and check nodes is given by $V(1), R(1)$. For our example illustrated in Fig. 4, the DD for the (2, 4) protograph-based LDPC ensemble is

$$V(x) = N x_1^2 + N x_2^2$$

$$R(x) = N x_1^2 x_2^2.$$  

Since we defined an edge type for each edge in the protograph that connects a different pair of nodes, the same edge type cannot be associated to two different variable nodes in the protograph. Consequently, for any edge type $j$, only one variable type $d \in \Phi_v$ has $d_j > 0$. We denote this variable type by $d^∗$.

The PD is initialized by removing all correctly received variable nodes with probability $(1 - \epsilon)$ from the graph. This residual graph includes check nodes with types that are not included in $\Phi_v$ while the set of variable types $\Phi_v$ does not change. For example, some check types that can occur for a (2, 4) protograph-based LDPC code are shown in Fig. 4. We define $D(c)$ as the set of check types in the graph after PD initialization deriving from a check node of type $c$ stemming from $c$ minus immediately resolved edges. We denote the extended set of all these possible multi-edge types derived from $c$ as $\Phi_c = \Phi_c \cup c D(c)$.

According to our definitions, the expected DD after PD initialization can be expressed as

$$E[V(x, 0)] = \sum_{d \in \Phi_v} \epsilon V_d x^d$$

$$E[R(x, 0)] = \sum_{c \in \Phi_c} R_c \sum_{c' \in D(c)} \epsilon|c'| (1 - \epsilon)^{|c' - c|} x^{c'}.$$  

For our (2, 4) LDPC protograph, the expected DD is

$$E[V(x, 0)] = \epsilon N (x_1^2 + x_2^2),$$

$$E[R(x, t = 0)] = N (c^2(1 - \epsilon)x_1^2 + c^2(1 - \epsilon)^2 x_2^2 + (1 - \epsilon)^4 x_1^2 x_2^2 + 2c(1 - \epsilon)^2 x_1 x_2 + 2c(1 - \epsilon)^2 x_1^2 x_2 + 2c(1 - \epsilon)^3 x_2^2).$$
process. Denote the expected values of \( r_c(\tau) \) and \( v_d(\tau) \) by \( \hat{r}_c(\tau) \) and \( \hat{v}_d(\tau) \), respectively. As shown in [10], these expectations are computed as the solution to the following system of differential equations:

\[
\frac{\partial \hat{v}_d(\tau)}{\partial \tau} = \frac{E[\Delta v_d(\tau)]}{M} \hat{v}(x, \tau), \hat{r}(x, \tau),
\]

\[
\frac{\partial \hat{r}_c(\tau)}{\partial \tau} = \frac{E[\Delta r_c(\tau)]}{M} \hat{v}(x, \tau), \hat{r}(x, \tau),
\]

(10)

(11)

where

\[
\Delta v_d(\tau) = v_d(\tau + \frac{1}{M}) - v_d(\tau),
\]

\[
\Delta r_c(\tau) = r_c(\tau + \frac{1}{M}) - r_c(\tau).
\]

From (8), each PD iteration takes a normalized time \( \frac{1}{M} \). As an intuition, the derivative of the expected evolution of \( \hat{r}_c(\tau) \) and \( \hat{v}_d(\tau) \) at \( \tau \) is given by their expected change after the next PD iteration divided by the normalized time of such an iteration. Also, note that

\[
\frac{\Delta r_c(\tau)}{M} = \mathbb{E}_c(\tau + \frac{1}{M}) - R_c(\tau)
\]

and hence the derivative of \( \hat{r}_c \) w.r.t. \( \tau \) in (11) at this time is evaluated by computing how many check nodes of type \( c \) are created (or resolved) after one PD iteration if the DD at \( \tau \) is \( \hat{v}(x, \tau), \hat{r}(x, \tau) \). A similar result holds for (10). The solution to (10) and (11) is unique and with probability \( 1 - \mathcal{O}(e^{-\sqrt{M}}) \), any particular realization of the normalized DD in (8) deviates from its mean by a factor of less than \( M^{-1/6} \) for the initial conditions

\[
\hat{r}_c(0) = E[r_c(\ell = 0)] = E[R_c(\ell = 0)] / M
\]

\[
\hat{v}_d(0) = E[v_d(\ell = 0)] = E[V_d(\ell = 0)] / M
\]

computed from (4) and (5) as in [9]. The expectations in (10) and (11) are computed in Section IV.

The ensemble BP threshold is given by the maximum \( \epsilon \) for which the mean of the total fraction of deg-1 check nodes

\[
\hat{c}_1(\tau) = \sum_{i=1}^{m} \hat{r}(0_{\sim i}, \tau)
\]

is positive for any \( \tau \in [0, L] \), where \( \hat{r}(x, \tau) \) is the mean of \( r(x, \tau) \) in (9). Based on (17), \( \hat{c}_1(\tau) \) is the mean of the random process

\[
c_1(\tau) = \sum_{i=1}^{m} r(0_{\sim i}, \tau).
\]

The solution to the system of equations (10) and (11) is used to characterize the asymptotic behavior, i.e., computing the threshold of the ensemble. It also determines quantities to assess the finite-length performance of the code. Critical points are points for which \( \hat{c}_1(\tau) \) presents local minima. As shown in [9], [18], the average error probability for an ensemble of codes is dominated by the probability that the process \( c_1(\tau) \) survives, i.e., does not go to zero around the critical points. Therefore, it is relevant to characterize the critical points and the expected number of deg-1 check nodes in the graph at them to determine the SC-LDPC finite-length performance computed from (10) and (11).

### D. Covariance evolution

To assess the survival probability of \( c_1(\tau) \) at critical points, we must determine the variance around the average \( \hat{c}_1(\tau) \), since errors in this framework correspond to deviations from the expected behavior beyond a certain threshold. The variance evolution of each of the DD components, i.e. \( v_d(\tau) \) for all \( d \in \Phi_c \) and \( r_c(\tau) \) for all \( c \in \Phi_c \) is computed by solving an extended set of differential equations, referred to as covariance evolution for LDPC ensembles [9]. The covariance function between any pair of DD components \( r_c(\tau) \) and \( r_{c'}(\tau) \) for \( c, c' \in \Phi_c \) is given by \( \delta_{c,c'}(\tau) / M \), which is the solution to the following differential equation:

\[
\frac{\partial \delta_{c,c'}(\tau)}{\partial \tau} = \frac{\text{CoVar}[\Delta r_c(\tau), \Delta r_{c'}(\tau)]}{M - 2}
\]

\[
+ \sum_{u \in \Phi_c} \delta_{c,u}(\tau) \frac{\partial f(\Delta r_u(\tau))}{\partial v_u} \bigg|_{\hat{v}(x, \tau), \hat{r}(x, \tau)}
\]

\[
+ \sum_{d \in \Phi_c} \delta_{c,d}(\tau) \frac{\partial f(\Delta r_d(\tau))}{\partial v_d} \bigg|_{\hat{v}(x, \tau), \hat{r}(x, \tau)}
\]

(19)

\[
\text{where } f(\Delta r_c(\tau)) \text{ is given by}
\]

\[
E[\Delta r_c(\tau)] \hat{v}(x, \tau), \hat{r}(x, \tau)
\]

\[
(M - 1)
\]

(20)

Note that the differential equation for \( \delta_{c,c'}(\tau) \) in (19) is coupled with similar differential equations for \( \delta_{c,u}(\tau) \) and \( \delta_{c,d}(\tau) \) for all \( u \in \Phi_c \) and \( d \in \Phi_c \). Also, all these equations are coupled with the differential equations in (10) and (11). By [9], the solution of the covariance evolution equations is unique. Given the initial conditions in (15) and (16) and

\[
\delta_{c,c'}(0) = \text{CoVar}[r_c(\ell = 0), r_{c'}(\ell = 0)]
\]

(21)

we obtain that \( \delta_{c,c'}(\tau) / M - \text{CoVar}[r_c(\tau), r_{c'}(\tau)] \) is in \( \mathcal{O}(M^{-1/2}) \). With \( M \to \infty \), the following properties hold:

I) \( r_c(\tau) \) is Gaussian distributed with mean \( \hat{r}_c(\tau) \) and variance \( \delta_{c,c'}(\tau) / M \).

II) For any pair \( (c, c') \in \Phi_c \times \Phi_c \), \( r_c(\tau) \) and \( r_{c'}(\tau) \) are jointly Gaussian distributed with cross covariance \( \delta_{c,c'}(\tau) / M \).

III) \( \delta_{c,c'}(\tau) / M \to 0 \) for any pair \( (c, c') \in \Phi_c \times \Phi_c \).

Assume we solve the covariance evolution system of equations (19) and denote by \( \text{Var}[c_1(\tau)] \) the variance of the process...
\( c_1(\tau) \) that represents the total fraction of deg-1 check nodes in the residual graph. Given the solution to \( \delta_{c,c'}(\tau) \) for any pair \((c,c')\) in (18), it is given by

\[
\text{Var}[c_1(\tau)] = \frac{\delta_1(\tau)}{M} + O(M^{-1/2}),
\]

(22)

where

\[
\delta_1(\tau) = \sum_{i=1}^{m} \sum_{j=1}^{m} \delta_{0_i,0_j},
\]

(23)

By the properties I) and II) above, \( c_1(\tau) \) tends in \( M \) to be Gaussian distributed. Putting altogether, we must solve two steps to compute both \( \hat{c}_1(\tau) \), the mean of \( c_1(\tau) \), and \( \delta_1(\tau) \), its variance. First, we compute the expectations inside the differential equations in (10), (11) and (19) for an arbitrary DD at time \( \tau \):

\[
E[\Delta_{\tau}|(v(\tau),r(\tau))]/M^{-1},
\]

(24)

\[
E[\Delta_{\tau}|(v(\tau),r(\tau))]/M^{-1},
\]

(25)

\[
\text{CoVar}[\Delta_{\tau},\Delta_{\tau}|(v(\tau),r(\tau))]/M^{-2},
\]

(26)

\[
\text{CoVar}[\Delta_{\tau},\Delta_{\tau}|(v(\tau),r(\tau))]/M^{-2},
\]

(27)

for \( c,c' \in \Phi_c \) and \( d \in \Phi_v \). Second, we use these solutions to calculate (10) and (11) by numerical integration.

This is done in Section IV. Howerver, solving the covariance evolution system of equations in (19) is prohibitively complex for large SC-LDPC codes due to the number of coupled differential equations to handle since computing moments of the form \( \text{CoVar}[\Delta_{\tau},\Delta_{\tau}|(v(\tau),r(\tau))]/M^{-2} \) already has high combinatorial complexity. In Section IV-C, we estimate \( \delta_1(\tau) \) accurately using the solution of the system in (10) and (11).

IV. Calculation of the Expected Graph Evolution

Assume the graph DD \((v(\tau),r(\tau))\) at a particular \( \tau \) is known. In this section, we want compute the expectations in (24) and (25) for \( c \in \Phi_c \) and \( d \in \Phi_v \).

A. Calculation of the Mean of \( c_1(\tau) \)

Recall that, by (14), (24) and (25) represent the expected number of variable (check) nodes of type \( d \) \((c)\) that are created in a single PD step. Using PD, only deg-1 check node can be directly resolved per iteration. When deg-1 check node is directly resolved, it is removed from the graph with the connected variable node and all attached edges to this variable node. When these edges are removed, a certain number of check nodes in the graph loose one connection so that the type of each one of these check nodes is modified, e.g. from \( c_1 \in \Phi_c \) to \( c_2 \in \Phi_c \). The graph looses a check node of type \( c_1 \) and wins a check node of multi-edge type \( c_2 \), i.e. the check node of multi-edge type \( c_1 \) is indirectly removed and the check node of type \( c_2 \) is created. No direct creation is possible, except for check nodes of type 0. Deg-1 check nodes can be removed from the graph in both a direct way or in an indirect way.
3) Creation of new check node types: because we got a comment in the review last time: i could replace 'remove' with 'resolve' For every edge indirectly removed from the graph, we create an additional check node of a different multi-edge type, i.e. if a check node of multi-edge type $c + O_k$ looses a type $k$ edge, the graph looses a check node of multi-edge type $c$ and gains a check node of multi-edge type $c$.

We compute the expected number $E_c^+ (\tau)$ of check nodes of type $c$ that are created after one PD update. If we ignore the probability of double edges in the SC-LDPC code graph, which is true in the limit $N \to \infty$, a check node that is indirectly removed from the graph only looses one edge. Therefore, a check node of type $c$ can only be created if a check node of type $c + O_1, c + O_2, \ldots, c + O_m$ is indirectly removed from the graph. Using (32), we obtain

$$E_c^+ (\tau) = \sum_{j=1}^{m} P_{O_{c,j}}^+ (\tau) \left( \sum_{k=1}^{m} E_{c+O_k,j}^- (\tau) \right).$$

4) Expected evolution in a single PD update: We now compute the expected evolution of the number of check nodes of multi-edge type $c$ in the graph in a single PD update if the DD at time $\tau$ is $(v(x, \tau), r(x, \tau))$. For any check type $c \in \mathcal{C}$ that does not correspond to a deg-1 check node, we have

$$E[\Delta c_e (\tau)] (v(x, \tau), r(x, \tau)) = E_c^+ (\tau) - E_c^- (\tau), \quad (33)$$

Finally, for $O_{c,j}$ for $j = 1, \ldots, m$, we have to also take into account that such a check node can be directly removed:

$$E[\Delta O_{c,j}^- (\tau)] (v(x, \tau), r(x, \tau)) = E_{O_{c,j},k,j}^+ (\tau) - E_{O_{c,j},k,j}^- (\tau) - P_{O_{c,j}}^- (\tau. \quad (34)$$

Unlike the SC-LDPC ensemble analyzed in [6], the expectations in (33) and (34) do not depend on the DD on the variable node side $v(x, \tau)$.

B. Expected Mean Evolution for Different SC-LDPC Ensembles

As discussed in Section III-D, $c_1(\tau)$ tends in $M$ to be Gaussian distributed with mean $\hat{c}_1(\tau)$ which can be computed by solving the set of differential equations in (10). The time-derivative of $\hat{c}_1(\tau)$ can be directly computed by summing (33) over all deg-1 check types:

$$\frac{\partial \hat{c}_1 (\tau)}{\partial \tau} = \sum_{j=1}^{m} E_{O_{c,j},k,j}^+ (\tau) - E_{O_{c,j},k,j}^- (\tau) - P_{O_{c,j}}^- (\tau \quad (35)$$

By numerical integration of (35) (Euler’s method with a sufficiently small step), the solution to $\hat{c}_1(\tau)$ can be evaluated for each $\epsilon$. Fig. 6 (a) shows the computed solution for the $(l, r, L)_P$ ensemble with $l = 3, r = 6$ and $L = 50$ for varying $\epsilon$. It includes actual decoding trajectories for $\epsilon = 0.45$ to show realizations of the decoding process. We also show the solution to $\hat{c}_1(\tau)$ for the $(l, r, L)_P$ same ensemble with double chain length, $L = 100$. In Fig. 6 (b) we show the same for the TAR4JA ensemble with $L = 50$.

The solutions obtained for $\hat{c}_1(\tau)$ are qualitatively similar to those obtained for non-protograph SC-LDPC ensembles in [6]. The evolution of $c_1(\tau)$ is divided in three phases:

1) The curves present an initial phase of decay, which corresponds to the PD eliminating the fraction of deg-1 check nodes created along the code after initialization. During this phase, deg-1 check nodes are removed roughly uniformly along the length of the chain. If $\epsilon > \epsilon^*$ for the uncoupled protograph LDPC block ensemble, the initial phase phase ends when all positions but those in the boundaries have run out of deg-1 check nodes. Denote this time by $\tau^*$.

2) The second phases starts at $\tau^*$ and corresponds to the “decoding” wave which moves at constant speed throughout the graph [19], [20]. Note that in this second phase, there is no single critical time point at which the decoder is most likely to stop, but the expected number of deg-1 check nodes is essentially constant throughout this critical phase. In this critical phase, we observe a constant mean evolution and thus $\frac{\partial \hat{c}_1 (\tau)}{\partial \tau} \approx 0$. Equation (35) indicates that at each PD step, we create as many deg-1 check nodes than we are deleting from the graph.

3) A third phase takes over when the two decoding waves started at the two ends of the chain are meeting in the middle of the chain.

Since the expected fraction of deg-1 check nodes in the residual graph at both the first and third phase is significantly larger than in the critical phase, it is unlikely that the decoder gets stuck in either the first or the last phase. We concentrate on the intermediate steady state phase and express the error probability as a function of the properties of the $c_1(\tau)$ random process during this phase. The length of the critical phase is upper bounded by $\epsilon L - \tau^*$. Denote the constant value of $\hat{c}_1(\tau)$ during the critical phase by $\hat{c}_1(*)$. Fig. 6 (a) and (b) show that $\hat{c}_1(*)$ does not depend on the chain length $L$. Therefore, the longer the chain length $L$ is, the longer decoder remains in the critical phase. As shown in [6], the statistical properties of $c_1(\tau)$ (mean, variance and correlation along the time) in this phase do not depend on $L$.

In order to relate the average block error probability with $\epsilon$, an approximate scaling of the mean and the variance of the process $c_1(\tau)$ in (18) as a function of $\epsilon$ is convenient. Following [9], we resort on a first order Taylor expansion of both quantities around the ensemble threshold $\epsilon^*$ for $\hat{c}_1(*)$:

$$\hat{c}_1(*)|_\epsilon \approx \hat{c}_1(*)|_{\epsilon^*} + \gamma (\epsilon^* - \epsilon) + \mathcal{O}((\epsilon^* - \epsilon)^2). \quad (36)$$

Since $c_1(*)|_{\epsilon^*} = 0$ by definition, $\gamma$ can be estimated from the numerical solution of $c_1(*)$ for a given $\epsilon < \epsilon^*$ by

$$\gamma \approx \frac{\hat{c}_1(*)|_\epsilon - \hat{c}_1(*)|_{\epsilon^*}}{(\epsilon^* - \epsilon)} = \frac{\hat{c}_1(*)|_\epsilon}{(\epsilon^* - \epsilon)}. \quad (37)$$

In Fig. 7, we plot (37) for the two ensembles considered above. For each ensemble, observe that all curves converge to the same constant, which indicates that ignoring the quadratic term (and all the rest of high order terms) in (36) is legitimate. During the critical phase, we assume

$$\hat{c}_1(*) \approx \gamma (\epsilon^* - \epsilon). \quad (38)$$
Fig. 6. Calculated trajectories of the number of deg-1 check nodes $c_1(\tau)$ during the decoding for codes of the ensemble $(3, 6, 50)$ (a) and for codes of the TAR4JA ensemble with $L = 50$ (b) for varying $\epsilon$.

The computed $\gamma$ for different protograph-based SC-LDPC ensembles are summarized in Table I. Its impact in the finite-length performance and the scaling in (38) is discussed in Section V. The higher $\gamma$ is, the more deg-1 check nodes exist during the critical phase and this reflects in a better finite-length performance. However, we have to take into account the variance of the $c_1(\tau)$ process.

C. An Estimation of the Second Order Moments of $c_1(\tau)$

As discussed in Section III-D, solving the system of differential equations that determines the variance of the $c_1(\tau)$ process in (23) is a complex problem due to the large number of coupled equations. An equivalent system was solved for a random SC-LDPC ensemble in [6], where it was observed that the variance of the process representing the fraction of deg-1 check nodes in the residual graph is almost constant during the critical phase. It was also shown that the dependency of the variance on $\epsilon$ is small during the critical phase further away from the ensemble threshold. By simulation, we observe the same behavior of $\text{Var}[c_1(\tau)]$ for protograph-based SC-LDPC codes. Fig. 8 shows $\text{Var}[c_1(\tau)]$ computed from 5000 simulations using the $(4, 8, 100)$ ensemble for three lifting factors $N = 500$, $N = 1000$ and $N = 2000$ bits. The solid lines refer to $\epsilon = 0.45$ and the dashed lines to $\epsilon = 0.46$. Observe that $\text{Var}[c_1(\tau)]$ is approximately constant during the critical phase seems not differ between both $\epsilon$ values. As predicted by equation (23), $\text{Var}[c_1(\tau)]$ decays linearly with $N$.

Denote $\delta_1(\tau)$ in (23) during the critical phase by $\delta_1(*)$ for which the following lemma holds.

**Lemma:**

$$\lim_{M \to \infty} \delta_1(*) = \text{Var}[\Delta_1|\hat{\theta}(x, \tau), \hat{r}(x, \tau)].$$

(39)

where $\Delta_1$ is the variation in the number of deg-1 check nodes between two PD consecutive iterations during the critical phase.

**Proof:**

Table I lists $\delta_1(*)$ for different SC-LDPC ensembles at $\epsilon = 0$.
\[ \gamma(\tau) = f(|c|) = \begin{cases} 0, & |c| > 2 \\ 1, & |c| = 2 \\ -1, & |c| = 1 \end{cases} \]

Recall that the \( i \)-th edge of the variable node (of type \( k_i \))
is connected to a type $c$ check node with probability

$$
\frac{c_{k_i} r_e(\tau)}{\sum_{e \in \mathcal{F}_c} c_{k_i} r_e(\tau)}.
$$

We now compute the p.m.f. of $\Delta_1(j, i)$, i.e., $P(\Delta_1(j, i) = a)$, $a \in \{-1, 0, +1\}$. Denote the type associated to the $i$-th edge of the variable node removed with $k_i \in \{1, \ldots, m\}$. Taking into account all possible check types $c$ and using the results derived in Section IV, the probability $P(\Delta_1(j, i) = a)$ can be computed as

$$
P(\Delta_1(j, i) = a) = \sum_{c \in \mathcal{F}_c} \sum_{\tau \in \Theta} \frac{c_{k_i} r_e(\tau)}{\sum_{e \in \mathcal{F}_c} c_{k_i} r_e(\tau)}
$$

By inserting (47) into (44), we compute the p.m.f. on the variation of the total number of deg-1 check nodes in the graph conditioned to the case $T_r = j$. Combining (46) and (44), we obtain (41) by averaging over all possible $T_r$:

$$
p_{\Delta_0}(u) = \sum_{j=1}^{m} p(\Delta_0_{j}, \tau) c_1(\tau) P(\Delta_1(j) = u),
$$

for $u \in \{-l, \ldots, l\}$.

1) Temporal Covariance: As discussed in Section IV-B, the error probability of the coupled ensemble is not determined by the behavior of the decoder at a particular critical point in time, but rather a critical phase during a period of time with length $\Theta(L)$ as depicted in Fig. 6. The error probability in the coupled case is given by the cumulative probability that the $c_1(\tau)$ process hits zero at some point during the decoding process. Recall we assume that the process is in the steady state phase essentially during the whole decoding process and we neglect the initial and the final phase.

To obtain the error probability during the steady state phase, we compute the covariance of the $r_1(\tau)$ process over time:

$$
\phi_1(\zeta, \tau) = \mathbb{E}[c_1(\zeta)c_1(\tau)] - \hat{c}_1(\tau) \hat{c}_1(\tau),
$$

where $\tau$ and $\zeta$ are two distinct time instances. The analytical computation is similar to the covariance evolution described in Section III-D. However, the number of coupled differential equations to solve is squared since we have to consider a minimum of two consecutive PD iterations. This approach is complex and computationally challenging.

As described in [6] for a random $(l, r, L)$ SC-LDPC ensemble, $\phi_1(\zeta, \tau)$ (49) is a function of $|\tau - \zeta|$ and the decay of the correlation is exponential:

$$
\phi_1(\zeta, \tau) \approx \frac{\delta_1}{M} e^{-|\theta| |\tau - \zeta|},
$$

where $\theta$ depends on the uncoupled LDPC ensemble and the coupling pattern of the SC-LDPC code. By simulation results, the same conjecture is verified for protograph-based SC-LDPC codes. In Fig. 10, we plot an estimate of $M\phi_1(\zeta, \tau)$ for several ensembles with $L = 100$ and $M = 4000$ bits per position at $\epsilon = 0.45$ computed by simulating 500 transmitted codewords.

![Fig. 10. Simulation-based estimation of $M\phi_1(\zeta, \tau)$ for several ensembles with $L = 100$ and $M = 4000$ bits per position at $\epsilon = 0.45$ computed by simulating 500 transmitted codewords.](image)

V. THE SCALING LAW FOR SPATIALLY COUPLED LDPC CODES

Based on the graph covariance evolution as a function of the code length at critical points where the expected evolution of the fraction of deg-1 check nodes presents a local minimum, scaling laws (SLs) predict the finite-length performance of code ensembles in the waterfall region [9].

As shown in [5], [6], $c_1(\tau)$ during the steady-state phase can be modeled by an Ornstein-Uhlenbeck (OU) process [21]. Based on this behavior, the zero-crossing probability of the estimate the value of $\theta$, only a few hundred transmitted codewords are required.

In Table I, we list $\theta$ for various SC-LDPC ensembles. As shown in [6], this parameter is related to the coupling pattern. If we increase the spatial memory of the code, $\theta$ decreases. Observe that $\theta$ for the $(q, a, L)$ ensemble is very high due to the strong coupling.
process $c_1(\tau)$ during the steady-state phase is estimated as

$$P^* \approx 1 - \exp\left(\frac{(\epsilon L - \tau^*)}{\mu_0(M, \epsilon, l, r)}\right)$$  \hspace{1cm} (51)

where $(\epsilon L - \tau^*)$ is the duration of the steady-state phase. $\mu_0$ is the average survival time of the $c_1(\tau)$ process:

$$\mu_0(l, r, M, \epsilon) \approx \sqrt{2\pi} \theta \int_0^{\sqrt{\theta} \lambda} \Phi(z) e^{\frac{-z^2}{2}} dz$$  \hspace{1cm} (52)

where $\Phi(z)$ is the c.d.f. of the Gaussian $\mathcal{N}(0,1)$, $\alpha = \delta_1(\sigma, \epsilon)\gamma^{-1}$ and $\delta_1(\star)$ is proportional to the variance of $c_1(\tau)$. While both $\theta$ and $\delta_1(\star)$ have the same value for the $(l, r, L)$ and $(l, r, L)_P$ ensembles. The higher $\gamma$ obtained for the protograp case yields an exponential increase of $\mu_0$ in (52) and, consequently, a drastic reduction in the error performance estimate in (51). If $M$ is sufficiently large so that (51) is small, a Taylor expansion around zero yields

$$P^* \approx \frac{(\epsilon L - \tau^*)}{\mu_0(M, \epsilon, l, r)}$$  \hspace{1cm} (53)

which suggests that the performance scales linearly with $L$ in the low-error rate regime.

VI. PERFORMANCE COMPARISON

A. Regular $(l, r, L)_P$ based Ensembles

The performance gain in comparison to the more random $(l, r, L)$ ensembles was already discussed in [13]. We quickly review the results and extend them to (4, 8) ensembles.

1) Mean Evolution of Deg-1 Check Nodes: Fig. 11 shows the analytical calculation and the simulation of the mean of the deg-1 check nodes $\bar{c}_1(\tau)$ during the decoding process. We analyze trajectories for the ensembles $(3, 6, 100)_P$, $(3, 6, 100)$, $(4, 8, 100)_P$ and $(4, 8, 100)$ for a BEC with $\epsilon = 0.45$ and verify the results with simulations of actual codes lifted by $M = 2000$. The BP thresholds are given in Table I. The means of the protograp-based constructions $(3, 6, 100)_P$ and $(4, 8, 100)_P$ are very close, as well as the means of the $(3, 6, 100)$ and the $(4, 8, 100)$ random constructions. Note that the steady-state phase of the $(4, 8)$ constructions are significantly longer. All calculations are accurate predictions of the experimental results.

2) Variance Evolution of Deg-1 Check Nodes: Fig. 12 shows the simulation of the variance of the aforementioned constructions. For all ensembles, $\text{Var} \left[ c_1(\tau) \right]$ is approximately constant in the steady-state phase. This constant value is denoted with $\delta_1(\star)$. The $(4, 8)$ constructions have a higher variance. In general, the variances of constructions with the same degree are more similar than constructions of the same type. The estimated values for the variances in the flat regime are also accurate.

3) Finite-length Scaling Parameters: Using the obtained parameters, we predict the finite-length performance as plotted in Fig. 13. The prediction of the slope is accurate while there is an offset, i.e. the code performs worse than the prediction. This is due to the neglect of finite-length effects in the structure of the code itself, e.g. smaller girths, which influences the performance as well, but is not captured.

Fig. 11. Calculated trajectories of $\gamma = \frac{\epsilon L}{\mu_0(M, \epsilon, l, r)}$ during the decoding for codes of the ensembles $(3, 6, 100)_P$, $(3, 6, 100)$, $(4, 8, 100)_P$ and $(4, 8, 100)$ for a BEC with $\epsilon = 0.45$.

Fig. 12. Simulated trajectories of the variance of deg-1 check nodes $\delta_1(\tau)$ during the decoding for codes of the ensembles $(3, 6, 100)_P$, $(3, 6, 100)$, $(4, 8, 100)_P$ and $(4, 8, 100)$ for $\epsilon = 0.45$.

B. Ensembles With Stronger Structure

We review the more structured $(q, a, L)$ and TAR4JA SC-LDPC ensembles and compare them to the $(4, 8)_P$ ensemble.

1) Mean Evolution of Deg-1 Check Nodes: Fig. 14 shows the mean evolution of the $(q, a, L)$ and TAR4JA SC-LDPC ensemble in comparison to the $(4, 8)_P$ ensemble for $L = 100$ and a BEC with $\epsilon = 0.45$. The mean of the TAR4JA ensemble is lower compared to the $(4, 8)_P$ ensemble but behaves quite similar. The mean of the $(q, a, L)$ ensemble enters the steady-state significantly later and is also higher. The experimental results are predicted accurately.

2) Variance and Covariance Evolution of Deg-1 Check Nodes: Since the computation of the variance is complex, we only calculate the values for a single point of the steady-state regime instead a plot of the complete evolution. The results are given in Table I. The variance of both the TAR4JA and the $(q, a, L)$ ensemble is higher than the variance for the $(4, 8)_P$ ensemble. The covariance of the $(q, a, L)$ ensemble is similar to the covariance of the $(4, 8)_P$ ensemble. Interestingly, the covariance of the TAR4JA ensemble is lower which is depicted in Fig. 10 (b).

3) Finite-length Scaling Parameters: Using the obtained parameters, we calculate the finite-length predictions for the $(q, a, L)$ and the TAR4JA ensemble. The predictions in Fig. 15 capture the slope of the actual performance well while
showing the same offset as the predictions in Fig. 13. The constructions are outperformed in the waterfall region by the \((4,8)_p\) ensemble. However, the finite-length analysis does not capture the performance in the error-floor region these codes were designed for.

### VII. MATCHING THE PERFORMANCE USING \((4,8)_p\) CODES

We have seen that in the waterfall region, \((4,8)_p\) codes perform significantly better than their \((3,6)_p\) counterparts. Alternatively, we can use smaller block sizes for these codes to obtain the same performance than a \((3,6)_p\) code. Thus, we are interested how to obtain a smaller lifting factor \(M\) with larger node degrees. From Table I, we see that the \((3,6)_p\) and the \((4,8)_p\) ensembles have a similar \(\gamma\) and \(\theta\).

(51) is dominated by \(M\) and \(\alpha = \gamma / \sqrt{\delta_1}\). We propose to approximate it by matching the upper-limit of the integral in (52) with

\[
M_{(3,6)} \approx \frac{\alpha_{(3,6)} \Delta_{(4,8)}}{\alpha_{(4,8)} \Delta_{(3,6)}} M_{(4,8)}.
\]  (54)

\[
M_{(3,6)} \approx \frac{\alpha_{(3,6)} (\epsilon - \epsilon_{(3,6)})}{\alpha_{(4,8)} (\epsilon - \epsilon_{(4,8)})} M_{(4,8)}.
\]  (55)

For every block size \(M\) and \(\epsilon\), we obtain a different \(M_{(4,8)}\) to match the performance of a single \((3,6)_p\) code.

### TABLE I

FINITE-LENGTH SCALING PARAMETERS FOR VARIOUS CODE ENSEMBLES WITH \(L = 100\) AND \(\epsilon = 0.45\).

<table>
<thead>
<tr>
<th>Code</th>
<th>(\epsilon^*)</th>
<th>(\tau^*)</th>
<th>(\gamma)</th>
<th>(\delta^*_1)</th>
<th>(\theta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((3,6,100))</td>
<td>0.4482</td>
<td>52</td>
<td>4.30</td>
<td>0.63</td>
<td>0.28</td>
</tr>
<tr>
<td>((4,8,100))</td>
<td>0.4947</td>
<td>74</td>
<td>4.28</td>
<td>0.86</td>
<td>0.28</td>
</tr>
<tr>
<td>((3,6,100)_p)</td>
<td>0.4955</td>
<td>49</td>
<td>5.34</td>
<td>0.73</td>
<td>0.23</td>
</tr>
<tr>
<td>((4,8,100)_p)</td>
<td>0.4968</td>
<td>73</td>
<td>5.26</td>
<td>1.02</td>
<td>0.23</td>
</tr>
<tr>
<td>((q,a,L))</td>
<td>0.4866</td>
<td>30</td>
<td>6.86</td>
<td>1.4</td>
<td>0.17</td>
</tr>
<tr>
<td>TAR4JA</td>
<td>0.4995</td>
<td>90</td>
<td>3.75</td>
<td>0.95</td>
<td>0.30</td>
</tr>
</tbody>
</table>

Fig. 13. Finite length scaling predictions and actual word error rate (WER) of \((3,6)_p\) and \((4,8)_p\) codes with \(L = 100\) and various lifting factors.

Fig. 14. Calculated trajectories of \(\gamma = \frac{\alpha_{(4,8)}}{\sqrt{\delta_1}}\) during the decoding for codes of the ensembles \((3,6,100)_p\), \((3,6,100)\), \((4,8,100)_p\) and \((4,8,100)\) for a BEC with \(\epsilon = 0.45\).

Fig. 15. Finite length scaling predictions and actual word error rate (WER) of \((q,a,L)\) and TAR4JA codes with \(L = 100\) and various lifting factors.

Fig. 16 shows the ratio between block sizes of the \((4,8)_p\) code to other codes with the same performance. While the performance of the \((3,6)_p\) ensemble is better below the threshold of the uncoupled code which is reflected in a ratio \(> 1\), the block size of this ensemble is larger in the area of interest close to the threshold when it is coupled. A similar effect can be observed for the \((q,a,L)\) ensemble. Observe the inverse effect for the TAR4JA ensemble since the \(\epsilon^*\) of the \((4,8)_p\) code is below \(\epsilon^*\) of the TAR4JA ensemble.

In Fig. 17, we calculate the matched \((4,8)_p\) ensembles for every \(\epsilon\) and verify it by simulation. For these two ensembles, \(54\) is a good estimate and the predicted and the simulated curves of the matched \((4,8)_p\) codes are close to their \((3,6)_p\) counterparts.

Fig. 18 shows the performance of \((4,8)_p\) ensembles matched to the \((q,a,L)\) and the TAR4JA ensembles. While the theoretic curves are matched accurately, the simulated performance of the codes differs stronger. This holds especially for the TAR4JA code and its matched \((4,8)_p\) code ensemble. There, it also comes to play that the \((4,8)_p\) code has a lower
threshold than the TAR4JA code it is matched to.

VIII. EXPECTED PERFORMANCE DEGRADATION FOR LONGER CHAIN LENGTHS

In VII, the lifting factor $M$ was predicted using (51) to match the performance of another code. We now examine the influence of the chain length $L$ on the performance. (53) suggests that in the low-error rate regime, the performance scales linearly with $L$.

In Fig. 19, we plot the error rates for both the SC-ARA ensemble with $q = a = 4$ and the SC-TAR4JA ensembles for $L = 100, L = 150$ and $L = 200$. Observe that in the low-error rate regime, using shorter chain lengths as prediction is accurate.

IX. CONCLUSION & OUTLOOK

The finite-length analysis was extended to protographs with higher edge degrees. Various SC-LDPC codes were analyzed and compared. The regular $(4, 8)_{p}$ ensemble is a promising construction in the waterfall region. We also developed a method to match the performances of various code ensembles.

REFERENCES


