

Design of Irregular SC-LDPC Codes With Non-Uniform Degree Distributions by Linear Programing

Heeyoul Kwak, Jong-Seon No, *Fellow, IEEE*, Hosung Park, *Member, IEEE*

Abstract

In this paper, we propose a new design method of irregular spatially-coupled low-density parity-check (SC-LDPC) codes with non-uniform degree distributions by linear programming (LP). In general, irregular SC-LDPC codes with non-uniform degree distributions is difficult to design with low complexity because their density evolution equations are multi-dimensional. To solve the problem, the proposed method is based on two main ideas: a local design of the degree distributions for each pair of positions and pre-computation of the input/output message relationship. These ideas make it possible to design the degree distributions of irregular SC-LDPC codes by solving low complexity LP problems which are used when optimizing uncoupled low-density parity-check (LDPC) codes over the binary erasure channel. We also find a proper objective function for the proposed design methodology to improve the performance of SC-LDPC codes. It is shown that the irregular SC-LDPC codes obtained by the proposed method are superior to regular SC-LDPC codes in terms of both asymptotic and finite-length performances.

Index Terms

Density evolution, linear programming (LP) problem, low-density parity-check (LDPC) codes, non-uniform degree distributions, spatially-coupled low-density parity-check (SC-LDPC) codes.

H. Kwak and J.-S. No are with the Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul 08826, Korea (e-mail: ghy1228@ccl.snu.ac.kr, jsno@snu.ac.kr).

H. Park is with the School of Electronics and Computer Engineering, Chonnam National University, Gwangju 61186, Korea (e-mail: hpark1@jnu.ac.kr).

I. INTRODUCTION

Spatially-coupled low-density parity-check (SC-LDPC) codes have attracted much attention due to their ability to universally achieve channel capacity over general binary memoryless symmetric (BMS) channels under iterative belief propagation (BP) decoding [1]. By coupling L disjoint component low-density parity-check (LDPC) codes arranged at L positions, SC-LDPC codes are constructed and their decoding performances under BP decoding approaches the maximum a posteriori (MAP) decoding performance of uncoupled LDPC codes. This phenomenon is termed the *threshold saturation effect*, which is empirically observed in [2] and analytically proved for the binary erasure channel (BEC) [3] and general BMS channels [1], [4].

Most studies of SC-LDPC codes focus on spatially coupled codes composed of regular LDPC codes. However, irregular LDPC codes can be coupled to construct irregular SC-LDPC codes, where the threshold saturation effect also occurs [1], [5], [6]. Thus, there have been researches to construct SC-LDPC codes from irregular LDPC codes which outperform regular LDPC codes. The first approach to construct irregular SC-LDPC codes is coupling protograph-based codes [7] such as repeat-accumulate (RA) [8], accumulate-repeat-jagged-accumulate (ARJA) [9], and MacKay-Neal (MN) [10] codes. Spatially coupled ARJA codes [11], [12] and spatially coupled MN codes [13], [14] show better asymptotic performance than regular SC-LDPC codes with bounded degrees. However, it is later known that the finite-length performance of spatially coupled ARJA codes is rather worse than that of other spatially coupled codes due to their inferior scaling behavior [15]. On the contrary, spatially coupled repeat accumulate (SC-RA) codes [16] show the best finite-length performance compared to other spatially coupled codes [15].

Similar to protograph-based spatially coupled codes, randomly constructed irregular SC-LDPC codes can be constructed by coupling irregular LDPC codes with variable and check node degree distributions $\lambda(x)$ and $\rho(x)$. However, it is difficult to globally optimize the degree distributions $\lambda(x)$ and $\rho(x)$ of irregular SC-LDPC codes with low-complexity because their density evolution (DE) equations [3] are multi-dimensional for the number of positions L . Thus, many studies assume a nearly regular degree distribution with two distinct degrees because the exhaustive searches are possible for this optimization. For example, SC-LDPC ensembles with two distinct degrees on check nodes are designed to improve the BP threshold over wide range of code

rate [17]. In [18], they define the convergence speed of SC-LDPC codes and show that slightly imposing irregularity on the variable node degree distribution $\lambda(x)$ can improve the convergence speed. Also, it has been reported that this improvement in the convergence speed also results in an improvement of the finite-length performance [19]. Recently, an optimization method for irregular SC-LDPC codes without any constraints on the degree distributions is proposed [20]. Further, an SC-LDPC ensemble with so called *non-uniform degree distributions* is designed, where the degree distributions can differ for each position. By optimizing the degree distributions using a genetic algorithm, it is shown that the optimized SC-LDPC codes exhibit improved decoding performance compared to regular SC-LDPC codes.

In this paper, motivated by the ideas in [20], we propose a systematic method with low-complexity to design irregular SC-LDPC codes with non-uniform degree distributions. To reduce the complexity of the design, the proposed method iteratively solves linear programming (LP) problems, which are widely used when optimizing uncoupled irregular LDPC codes over the BEC [21], [22]. However, it is hard to directly apply the LP problem to the design of the degree distributions of SC-LDPC codes because the DE equations are multi-dimensional. Thus, we introduce two important methods for designing the degree distributions. First, instead of designing the degree distributions at once, local designs are iteratively performed, where the variable node degree distributions of each pair of positions are obtained one at a time while keeping the degree distributions of the other positions constant. Second, the input/output message relationship between the variable nodes at the target positions and the remaining graph is computed to design the variable node degree distributions of the target positions. These two methods allow us to obtain a *one-dimensional DE equation*, which enables the designing of the degree distributions simply, such as by solving LP problems.

The one-dimensional DE equation is used in [23] to optimize the degree distribution of additional variable nodes to mitigate the rate-loss of SC-LDPC codes, where the objective function for optimization is to maximize the design rate. However, we observe that the objective function of maximizing the design rate is not proper when designing the degree distributions of irregular SC-LDPC codes. Thus, we use another objective function which minimizes the number of required iterations for successful decoding [22], [24]. The proposed design method finds the degree distributions of a pair of target positions one at a time to minimize the number of required iterations in the one-dimensional DE equation. Although the proposed method aims to minimize

the number of required iterations in the one-dimensional DE equation, it is observed that the overall number of required iterations in the multi-dimensional DE equations is also properly decreased by the proposed design method. In addition, the BP threshold is improved by the proposed design method if all of the degree distributions are locally designed, which is confirmed graphically from an analysis of the expected graph evolution [25]. Numerical results show that irregular SC-LDPC codes with the degree distributions obtained by the proposed method exhibit superior BP threshold and finite-length performance compared to regular SC-LDPC codes. It is also shown that an additional degree of freedom on the check node degrees can improve the performance further. Finally, we introduce several methods which can be used to improve the performance by adopting the multi-edge type (MET) structure [21] and applying the proposed method to SC-RA codes.

The remainder of the paper is organized as follows. Section II introduces the construction methods of SC-LDPC ensembles and the tools for analyzing the ensembles. In Section III, the methods to design the degree distributions of SC-LDPC ensembles are proposed. In Section IV, the finite-length performance of the SC-LDPC codes obtained by the proposed methods is shown and several methods for improving the performance further are presented. Finally, the conclusion is given in Section V.

II. SC-LDPC ENSEMBLES

A. Construction of SC-LDPC Ensembles

Let l and r denote the variable and check node degrees of regular LDPC ensembles, respectively. The SC-LDPC ensemble considered in this paper is irregular SC-LDPC ensembles, where the degree distributions of the variable nodes at each position can be irregular while the check node degrees at each position are regular but can differ at each position. The SC-LDPC ensemble consists of M variable nodes located at each of L positions with M_c check nodes located at each of $L + w - 1$ positions, where $M_c \triangleq M \frac{l}{r} \in \mathbb{N}$. The variable nodes at position u follow the edge perspective degree distribution $\lambda_u(x) = \sum_d \lambda_{u,d} x^{d-1}$ [21] while the check nodes at position v have degree $r_v \in \mathbb{N}$. The edge connectivity between the nodes in the graph is represented by a $(L + w - 1) \times L$ connectivity matrix \mathbf{T} , where the entry $T_{v,u}$ is the number of edges between the check nodes at position v and the variable nodes at position u . For example, the regular SC-

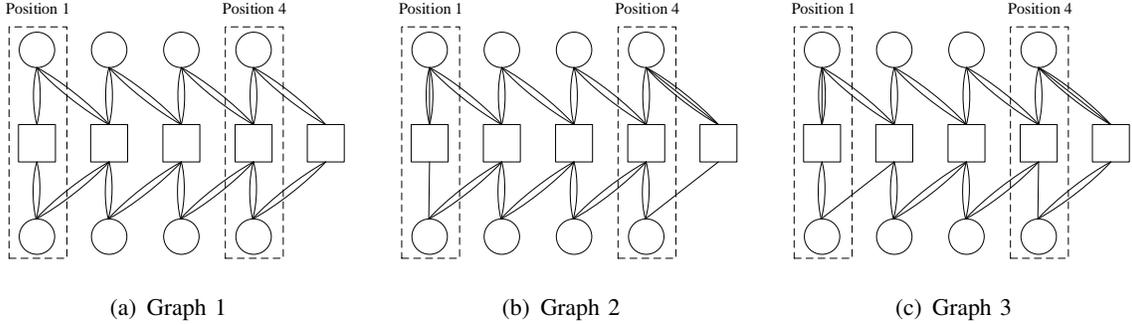


Fig. 1. Examples of Tanner graphs of SC-LDPC codes for $L = 4$, $w = 2$, and $M = 2$.

LDPC ensemble [3] constructed by coupling regular LDPC ensembles has code parameters of $\lambda_u(x) = x^{l-1}$ for $1 \leq u \leq L$, $r_v = r$ for $1 \leq v \leq L+w-1$, and $T_{v,u} = \frac{Ml}{w}$ for $u \leq v \leq u+w-1$.

In Fig. 1, Tanner graphs of three different SC-LDPC codes with $L = 4$, $w = 2$, and $M = 2$ are shown as examples and their code parameters are represented in Table I. Note that there are multiple edges in the Tanner graphs to represent various degree distributions with a small number of nodes. First, Graph 1 represents a regular SC-LDPC code, where the degrees of the variable nodes are equal to 4. In contrast, the degree distributions of the variable nodes at positions 1 and 4 in Graph 2 are irregular. Graph 2 is called a SC-LDPC code with non-uniform degree distributions in that the degree distributions of the variable nodes at each position differ from each other. The objective of this paper is to find good degree distributions of the variable nodes at each position of the SC-LDPC ensemble. In addition, Graph 2 becomes Graph 3 after permitting a different number of connected edges between the positions, giving more freedom in the design of the degree distributions. Note that we assume a symmetric structure, that is, $\lambda_u(x) = \lambda_{L+1-u}(x)$, $r_v = r_{L+w-v}$, $T_{v,u} = T_{L+w-v, L+1-u}$ for all the codes in this paper.

For the given code parameters $\lambda_u(x)$, r_v , and \mathbf{T} , the detailed construction method of the SC-LDPC ensemble is described as follows. Let $[m, n] \triangleq \{m, m+1, \dots, n\}$. First, place M variable nodes with degree distribution $\lambda_u(x)$ at position u for $u \in [1, L]$. Then, there are $M \frac{1}{\int_0^1 \lambda_u(x) dx}$ variable node sockets at position u , where $\frac{1}{\int_0^1 \lambda_u(x) dx}$ is the average variable node degree of the variable nodes at position u . To connect all of the sockets, the number of edges connected to the variable nodes at position u should be equal to the number of sockets, that is, $M \frac{1}{\int_0^1 \lambda_u(x) dx} = \sum_i T_{i,u}$. Let π_u be a random permutation on $[1, \sum_i T_{i,u}]$ for position u . Divide π_u into w disjoint subsets denoted by π_u^1, \dots, π_u^w such that the size of π_u^t becomes $T_{u+t-1,u}$ for

TABLE I
PARAMETERS OF THE TANNER GRAPHS IN FIG. 1

	$\lambda_i(x)$	\mathbf{T}
Graph 1	$\lambda_1(x) = x^3$ $\lambda_2(x) = x^3$ $\lambda_3(x) = x^3$ $\lambda_4(x) = x^3$	$\begin{bmatrix} 4 \\ 4 & 4 \\ & 4 & 4 \\ & & 4 & 4 \\ & & & 4 \end{bmatrix}$
Graph 2	$\lambda_1(x) = \frac{3}{8}x^2 + \frac{5}{8}x^4$ $\lambda_2(x) = x^2$ $\lambda_3(x) = x^2$ $\lambda_4(x) = \frac{3}{8}x^2 + \frac{5}{8}x^4$	$\begin{bmatrix} 4 \\ 4 & 4 \\ & 4 & 4 \\ & & 4 & 4 \\ & & & 4 \end{bmatrix}$
Graph 3	$\lambda_1(x) = \frac{3}{8}x^2 + \frac{5}{8}x^4$ $\lambda_2(x) = x^2$ $\lambda_3(x) = x^2$ $\lambda_4(x) = \frac{3}{8}x^2 + \frac{5}{8}x^4$	$\begin{bmatrix} 5 \\ 3 & 4 \\ & 4 & 4 \\ & & 4 & 3 \\ & & & 5 \end{bmatrix}$

$t \in [1, w]$. Similarly, place M_c check nodes of degree r_v at position v for $v \in [1, L + w - 1]$ in the graph. Accordingly, there are $M_c r_v$ check node sockets at position v but some sockets cannot be filled if $\sum_j T_{v,j} < M_c r_v$. Divide the randomly selected $\sum_j T_{v,j}$ check node sockets among the $M_c r_v$ check node sockets at position v into w groups randomly such that the size of group t becomes $T_{v,v-t+1}$ for $t \in [1, w]$, where $T_{v,u} = 0$ for $u < 0$ or $u > L$. Then, the j th check node socket in group t at position $u + t - 1$ is connected to the $\pi_u^t(j)$ th variable node socket at position u , where $\pi_u^t(j)$ denotes the j th element of π_u^t .

The total numbers of variable nodes V and check nodes C placed in the graph are $V = LM$ and $C = (L + w - 1)M_c$, respectively. However, some check nodes at position v for $v \in [1, w]$ or $v \in [L + 1, L + w - 1]$ cannot be connected to any variable nodes in the graph because the number $\sum_j T_{v,j}$ of edges connected to check nodes is lower than the number of sockets $M_c r_v$. To be specific, a check node socket at position v for $v \in [1, w]$ or $v \in [L + 1, L + w - 1]$ cannot be connected to any variable node in the graph with probability $1 - \sum_j T_{v,j} / (M_c r_v)$. Then, the expected number of check nodes \bar{C} with at least one connection to the variable nodes in the

graph is given as

$$\bar{C} = \left(L + w - 1 - 2 \sum_{v=1}^{w-1} \left(1 - \frac{\sum_j T_{v,j}}{M_c r_v} \right)^{r_v} \right) M_c,$$

which is less than $C = (L + w - 1)M_c$. Therefore, the design rate of the SC-LDPC ensemble is expressed as

$$\begin{aligned} R_{\text{SC}} &= 1 - \frac{\bar{C}}{V} \\ &= \left(1 - \frac{l}{r} \right) - \frac{l}{r} \frac{w-1}{L} + \frac{l}{r} \frac{2 \sum_{v=1}^{w-1} \left(1 - \frac{\sum_j T_{v,j}}{M_c r_v} \right)^{r_v}}{L}. \end{aligned} \quad (1)$$

Generally, because the last term of (1) is much smaller than the other terms, we ignore the last term when calculating the design rate for convenience.

B. DE Equations of SC-LDPC Ensembles

In this paper, the channel is assumed to be the BEC with erasure probability ϵ . The DE is often used in predicting the performance of LDPC codes in an asymptotic setting such as infinite code lengths and infinite numbers of iterations. Let $x_u^{(\ell)}$ and $y_v^{(\ell)}$ denote the average erasure probabilities of messages at iteration ℓ emitted from the variable nodes at position u and the check nodes at position v , respectively. Set the initial conditions as $x_u^{(0)} = \epsilon$ for all u . Then, the evolution of $x_u^{(\ell)}$ can be expressed as

$$\begin{aligned} y_v^{(\ell)} &= 1 - \left(1 - \frac{\sum_j T_{v,j} x_j^{(\ell)}}{M_c r_v} \right)^{r_v - 1} \\ x_u^{(\ell+1)} &= \epsilon \lambda_u \left(\frac{\sum_i T_{i,u} y_i^{(\ell)}}{\sum_i T_{i,u}} \right). \end{aligned} \quad (2)$$

With the DE equations in (2), we can obtain the BP threshold which is defined as the maximum ϵ for which $x_u^{(\ell)}$ goes to zero for all u . Additionally, we define the overall number of required iterations I_r for successful decoding as the minimum ℓ such that $x_u^{(\ell)} = 0$ for all u and define the average convergence speed as L/I_r [18].

C. Expected Graph Evolution of SC-LDPC Ensembles

In [15], [25], the scaling law of SC-LDPC codes is derived to predict the finite-length performance of SC-LDPC codes. The scaling law of SC-LDPC ensembles depends on the scaling parameters derived by analyzing the expected graph evolution of the peeling decoding. The peeling decoder sequentially recovers the value of an unknown variable node which is connected to a degree-one check node and removes the recovered variable node and the degree-one check node connected to it from the graph. As the peeling decoding progresses, the number of nodes in the graph decreases and becomes zero if the codeword is successfully decoded. In other words, successful decoding is achieved if at least one degree-one check node remains alive in the remaining graph until all unknown variable nodes are recovered. Thus, the finite-length performance of an ensemble is related to the expected evolution of the remaining graph, more specifically, the expected number of degree-one check nodes in the remaining graph.

In [25], a system of coupled differential equations to compute the expected number of degree-one check nodes is derived for regular SC-LDPC ensembles. Because the SC-LDPC ensemble considered in this paper has the non-uniform degree distributions, the differential equations to compute the expected number of degree-one check nodes should be modified as follows.

Consider the degree distributions of the original graph before the peeling decoder is initialized. First, consider the degree distribution of the variable nodes at position u . Define the type of a variable node at position u using the vector $\underline{x} = (x_1, \dots, x_w)$, where x_t represents the number of edges connected to the check nodes at position $u + t - 1$. Although the profile of degrees of each variable node is sufficient to represent the DE equations in (2), the type of each variable node should be considered when calculating the expected graph evolution. Let $\mathbb{P}_u(\underline{x})$ be the probability that a variable node chosen at random from position u in the original graph is of type \underline{x} . Considering the construction method of the SC-LDPC ensemble, we can see that the vector $\underline{x} = (x_1, \dots, x_w)$ for the variable nodes at position u follows a multinomial distribution with probabilities $\underline{p} = (p_1, \dots, p_w)$, where $p_t = T_{u+t-1,u} / \sum_i T_{i,u}$. Thus, the probability $\mathbb{P}_u(\underline{x})$ is given as

$$\mathbb{P}_u(\underline{x}) = \frac{|\underline{x}|!}{x_1! \cdots x_w!} \prod_{t=1}^w \left(\frac{T_{u+t-1,u}}{\sum_i T_{i,u}} \right)^{x_t}$$

where $|\underline{x}| = \sum_i x_i$. Next, consider the check nodes at position v . Let $\rho_{m,v}$ be the probability that a check node chosen at random from position v in the original graph is of degree m , which is

given as

$$\rho_{m,v} = \begin{cases} \binom{r_v}{m} \left(\frac{\sum_j T_{v,j}}{M_c r_v} \right)^m \left(1 - \frac{\sum_j T_{v,j}}{M_c r_v} \right)^{r_v - m}, & \text{if } v \in [1, w-1] \\ 1, & \text{if } m = r_v, v \in [w, L] \\ 0, & \text{if } m < r_v, v \in [w, L] \\ \rho_{m, L+w-v}, & \text{if } v \in [L+1, L+w-1]. \end{cases}$$

For iteration ℓ of the peeling decoder, let τ be the number of iterations normalized by M , i.e., $\tau = \ell/M$. At time τ , let $R_{j,v}(\tau)$ be the number of edges connected to the check nodes of degree j , $j = 1, \dots, r_v$, at position v , $v \in [1, L+w-1]$. Likewise, let $U_{\underline{x},u}(\tau)$ be the number of edges that are connected to variable nodes of type \underline{x} at position u , $u \in [1, L]$. After the initialization of the peeling decoder, the expected value of $R_{j,v}(0)$ is expressed as

$$\mathbb{E}[R_{j,v}(0)] = j M_c \sum_{m \geq j}^{r_v} \rho_{m,v} \binom{m}{j} \epsilon^j (1 - \epsilon)^{m-j}.$$

In addition, the initial values of the expected value of $U_{\underline{x},u}(\ell)$ can be computed as

$$\mathbb{E}[U_{\underline{x},u}(0)] = \begin{cases} \epsilon \lambda_{u,|\underline{x}|} \frac{M}{\int_0^1 \lambda_u(z) dz} \mathbb{P}_u(\underline{x}), & u \in [1, L] \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathbb{E}[\Delta R_{j,v}(\tau)] = \mathbb{E}[R_{j,v}(\tau+1/M) - R_{j,v}(\tau)]$ and $\mathbb{E}[\Delta U_{\underline{x},u}(\tau)] = \mathbb{E}[U_{\underline{x},u}(\tau+1/M) - U_{\underline{x},u}(\tau)]$, where the expectation is determined given the degree distributions in the remaining graph at time τ . In order to compute the expectation of $R_{j,v}(\tau)$ and $U_{\underline{x},u}(\tau)$ from the initial values, $\mathbb{E}[\Delta R_{j,v}(\tau)]$ and $\mathbb{E}[\Delta U_{\underline{x},u}(\tau)]$ should be obtained to solve the system of differential equations described as [15], [25].

$$\begin{aligned} \frac{\partial R_{j,v}(\tau)}{\partial \tau} &= \frac{\mathbb{E}[\Delta R_{j,v}(\tau)]}{1/M} \\ \frac{\partial U_{\underline{x},u}(\tau)}{\partial \tau} &= \frac{\mathbb{E}[\Delta U_{\underline{x},u}(\tau)]}{1/M}. \end{aligned}$$

The procedure used to obtain $\mathbb{E}[\Delta R_{j,v}(\tau)]$ and $\mathbb{E}[\Delta U_{\underline{x},u}]$ is described as follows. Let $\phi_{m,\underline{x},u}(\tau)$ be the probability that a variable node of type \underline{x} connected to a degree-one check node at position m belongs to position u . Then, we have

$$\phi_{m,\underline{x},u}(\tau) = \begin{cases} \frac{\frac{x_{m-u+1}}{|\underline{x}|} U_{\underline{x},u}(\tau)}{\sum_{i \in S(m)} \left(\sum_{\underline{x}'} \frac{x'_{m-i+1}}{|\underline{x}'|} U_{\underline{x}',i}(\tau) \right)}, & \text{if } u \in S(m) \\ 0, & \text{otherwise} \end{cases}$$

where $S(m) = \{j \mid \min(m - (w - 1), 1) \leq j \leq m\}$.

When a degree-one check node from position m and the variable node connected to it are removed, we define $\xi_{m,v,t}(\tau)$ as the probability that t edges of the removed variable node are connected to the check nodes other than the removed check node at position v . Then, we have

$$\xi_{m,v,t}(\tau) = \begin{cases} \sum_{i \in S(v)} \left(\sum_{\underline{x}: x_{v-i+1}=t} \phi_{m,\underline{x},i}(\tau) \right), & \text{if } m \neq v \\ \sum_{i \in S(v)} \left(\sum_{\underline{x}: x_{v-i+1}=t+1} \phi_{m,\underline{x},i}(\tau) \right), & \text{if } m = v \end{cases}$$

for $t \leq l_{\max} - 1$, where l_{\max} denotes the maximum degree of variable nodes.

The average number of degree- j check nodes losing one edge when t edges are randomly removed from check nodes at position v is given as

$$F_{j,v,t}(\tau) = \sum_{k=1}^t k \binom{t}{k} \delta_{j,v}^k(\tau) (1 - \delta_{j,v}(\tau))^{t-k}$$

where $\delta_{j,v}(\tau) = \frac{R_{j,v}(\tau)}{\sum_{q=1}^{r_v} R_{q,v}(\tau)}$ for $j \leq r_v$ and $\delta_{r_v+1,v}(\tau) = 0$.

Then, we have

$$\begin{aligned} \mathbb{E}[\Delta U_{\underline{x},u}(\tau) \mid \text{pos}(\tau) = m] &= -|\underline{x}| \phi_{m,\underline{x},u}(\tau) \\ \mathbb{E}[\Delta R_{j,v}(\tau) \mid \text{pos}(\tau) = m] &= \begin{cases} j \sum_{t=1}^{l_{\max}-1} \xi_{m,v,t}(\tau) \left(F_{j+1,v,t}(\tau) - F_{j,v,t}(\tau) \right) - 1, & \text{if } v = m, j = 1 \\ j \sum_{t=1}^{l_{\max}-1} \xi_{m,v,t}(\tau) \left(F_{j+1,v,t}(\tau) - F_{j,v,t}(\tau) \right), & \text{otherwise} \end{cases} \end{aligned}$$

where $\text{pos}(\tau)$ is the position at which a degree-one check node is removed at time τ .

Finally, the expectations of $\Delta R_{j,v}(\tau)$ and $\Delta U_{\underline{x},u}(\tau)$ are described as

$$\begin{aligned} \mathbb{E}[\Delta R_{j,v}(\tau)] &= \sum_{m=1}^{L+w-1} \mathbb{E}[\Delta R_{j,v}(\tau) \mid \text{pos}(\tau) = m] p_m(\tau) \\ \mathbb{E}[\Delta U_{\underline{x},u}(\tau)] &= \sum_{m=1}^{L+w-1} \mathbb{E}[\Delta U_{\underline{x},u}(\tau) \mid \text{pos}(\tau) = m] p_m(\tau) \end{aligned}$$

where $p_m(\tau) = R_{1,m}(\tau) / \sum_{v=1}^{L+w-1} R_{1,v}(\tau)$.

Then, the expectation of the total number of degree-one check nodes $\mathbb{E}[R_1(\tau)]$ in the remaining graph at τ becomes $\sum_v \mathbb{E}[R_{1,v}(\tau)]$ and the normalized number of degree-one check nodes $r_1(\tau)$ is computed as $r_1(\tau) = \mathbb{E}[R_1(\tau)]/M$.

III. PROPOSED DESIGN METHODS OF SC-LDPC ENSEMBLES

In this section, we propose new design methods of code parameters $\lambda_u(x)$, ρ_v , and $T_{v,u}$ for the SC-LDPC ensembles.

A. Pre-Computation of $\delta_u(z)$

Before describing the proposed design methods of the SC-LDPC ensemble, we introduce a pre-computation algorithm, as in Algorithm 1. This algorithm is firstly used in [23], and it is essential for the proposed design methods.

Algorithm 1 Pre-computation of $\delta_u(z)$ [23]

Input: $\{\lambda_1(x), \dots, \lambda_L(x)\}, \{r_1, \dots, r_{L+w-1}\}, \mathbf{T}, Q, \epsilon, u$

- 1: **for** $q = 1 : Q$ **do**
- 2: **Set** $z_q = \epsilon \frac{q}{Q}$
- 3: Update $x_i^{(\ell)}$ for all i except u and $L - u + 1$ by DE equations (2) until the messages are saturated while fixing $x_u^{(\ell)} = z_q, x_{L-u+1}^{(\ell)} = z_q$ for all ℓ .
- 4: Let $\delta_u(z_q)$ be the incoming message to variable nodes at position u , that is,

$$\delta_u(z_q) = \frac{\sum_i T_{i,u} y_i^{(\ell)}}{\sum_i T_{i,u}}.$$

- 5: **end for**
-

In Algorithm 1, the graph can be considered as two parts, which are the variable nodes at positions u and $L - u + 1$ and the remaining graph, and then the input/output message relationship is obtained. First, the input message z to the remaining graph is passed from the variable nodes at positions u and $L - u + 1$ to the remaining graph. And then the messages in the remaining graph are updated using the DE equations (2) until all of the messages are saturated while the messages from the variable nodes at positions u and $L - u + 1$ to the remaining graph are fixed at z . Finally, let the incoming message to the variable nodes at positions u and $L - u + 1$ be $\delta_u(z)$. In other words, Algorithm 1 calculates the output message $\delta_u(z)$ corresponding to the input message z from the perspective of the remaining graph.

Consider a message update scheduling such that the message x_u is updated only after the other messages are saturated to their fixed values. Under this message update scheduling, x_u is

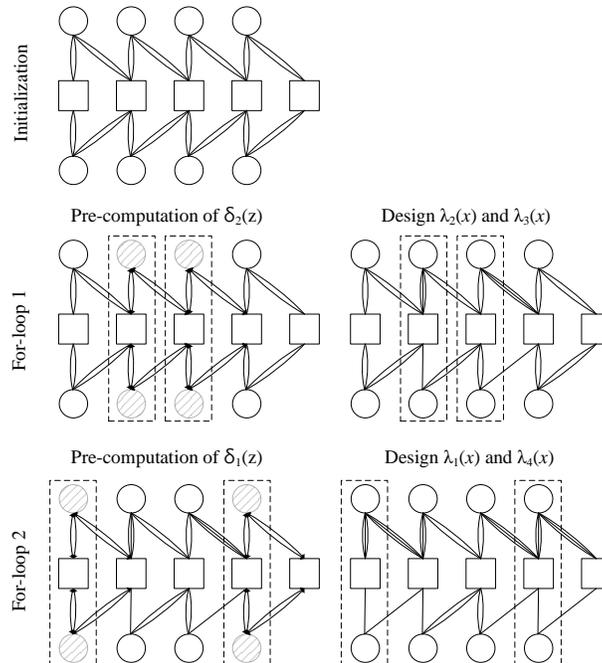


Fig. 2. Design procedure of the proposed algorithms with $L = 4$, $w = 2$, and $M = 2$.

updated from z to $\epsilon\lambda_u(\delta_u(z))$ because the incoming message to the variable nodes at position u is $\delta_u(z)$. In other words, pre-computation of $\delta_u(z)$ gives a one-dimensional DE equation $z^{(\ell+1)} = \epsilon\lambda_u(\delta_u(z^{(\ell)}))$ for position u with an initial value of $z^{(0)} = \epsilon$ under the message update scheduling described before. The one-dimensional DE makes it possible to design the SC-LDPC ensemble by the methods used to optimize uncoupled LDPC codes, such as maximizing the design rate [21] and minimizing the number of required iteration [22], [24].

B. Maximizing Design Rate

First, a design method to maximize the design rate is described in Algorithm 2. Maximizing the design rate can be achieved by solving a LP problem with the objective function to minimize the average variable node degree $1/\int_0^1 \lambda_u(x)dx$ under the constraint of the successful decoding $\epsilon^{\text{BP}}\lambda_u(\delta_u(z_q)) < z_q$ for $q \in [1, Q]$. Unlike the codes defined in Section II-A, where the number of variable nodes at each position is fixed to M , there are $Ml\int_0^1 \lambda_u(x)dx$ variable nodes at position u because Ml edges heading to the variable nodes at each position are distributed to the variable nodes with the average variable node degree $1/\int_0^1 \lambda_u(x)dx$. Thus, minimizing the average degree

Algorithm 2 Design method for maximizing design rate

Input: $l, r, L, w, l_{\min}, l_{\max}, Q, I_{\max}$

- 1: **Initialization:** Code parameters of the regular SC-LDPC ensemble
 - 2: **for** Iter = 1 to I_{\max} **do**
 - 3: **for** $u = L/2$ to 1 **do**
 - 4: Calculate the BP threshold ϵ^{BP} by the DE equations (2)
 - 5: Pre-computation of $\delta_u(z)$ by **Algorithm 1**
 - 6: Obtain $\lambda^*(x) = \sum_{k=l_{\min}}^{l_{\max}} \lambda_k^* x^{k-1}$ by solving the following LP problem

$$\begin{aligned} & \text{minimize } \frac{1}{\int_0^1 \lambda^*(x) dx} \\ & \text{subject to } \epsilon^{\text{BP}} \lambda^*(\delta(z_q)) < z_q \text{ for } 1 \leq q \leq Q, \lambda^*(1) = 1 \end{aligned}$$
 - 7: $\lambda_u(x) = \lambda^*(x), \lambda_{L-u+1}(x) = \lambda^*(x)$
 - 8: **end for**
 - 9: **end for**
-

$1/\int_0^1 \lambda_u(x) dx$ corresponds to maximizing the number of variable nodes $Ml \int_0^1 \lambda_u(x) dx$ and the design rate.

In Algorithm 2, the graph is initialized with the regular SC-LDPC ensemble. Then, the degree distribution $\lambda_u(x)$ is designed starting from $u = L/2 \in \mathbb{N}$ to maximize the design rate and $\lambda_{L-u+1}(x)$ is determined by the designed degree distribution because we assume a symmetric code structure $\lambda_u(x) = \lambda_{L-u+1}(x)$. To design $\lambda_{L/2}(x)$, the value $\delta_{L/2}(z)$ is pre-calculated; this is used to design $\lambda_{L/2}(x)$ to maximize the design rate. The degree distribution for the next position $\lambda_{L/2-1}(x)$ is designed in the same manner after designing $\lambda_{L/2}(x)$. This local design is conducted until the number of iterations for the algorithm reaches I_{\max} or $\lambda_u(x)$ does not change with Iter. For the other algorithms to be introduced, we apply the same design procedure, in which the local design of the degree distributions is conducted for each pair of positions one at a time, as in Fig. 2.

For the input values $l = 4, r = 8, L = 10, w = 3, l_{\min} = 3, l_{\max} = 10, Q = 1000$, and $I_{\max} = 10$, the design rate is increased from 0.4 to 0.4360 by Algorithm 2. Likewise, the design

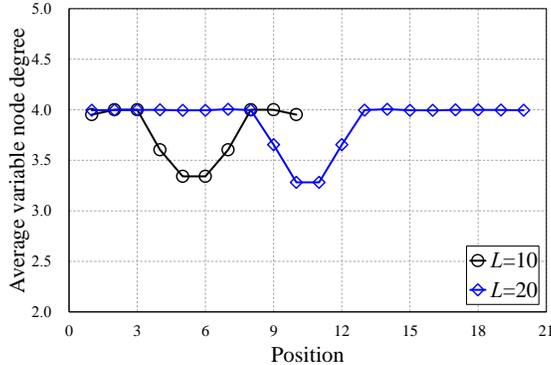


Fig. 3. Evolution of design rate of the SC-LDPC ensembles obtained by Algorithm 2 as the for-loop proceeds.

rate is increased from 0.45 to 0.4670 by Algorithm 2 for $L = 20$. Fig. 3 shows the average variable node degree $1/\int_0^1 \lambda_u(x)dx$ of each position of the SC-LDPC ensemble obtained by Algorithm 2 for $L = 10$ and 20. It shows that the degree distributions of the center positions 5, 6 for $L = 10$ and 10, 11 for $L = 20$ are properly designed to increase the design rate by minimizing the average variable node degree. However, the variable node degrees of the positions far from the center positions remain nearly identical to the initial degree $l = 4$. Thus, the fraction of the number of positions to be improved by Algorithm 2 becomes smaller as L becomes larger. To achieve an improvement in all positions for large L , a different objective function for the LP problem is required instead of maximizing the design rate.

C. Minimizing the Number of Required Iterations

Another objective function described in Algorithm 3 is minimizing the number of required iterations. Consider a non-increasing iterative function $f(x)$. Given an initial value a , the iterative process is represented by $x^{(0)} = a$ and $x^{(\ell)} = f(x^{(\ell-1)})$. The number of required iterations $I(b)$ for the target value b , $b < a$, is defined as the minimum ℓ such that $x^\ell \leq b$. In [22], the number of required iterations $I(b)$ is approximated as

$$\begin{aligned}
 I(b) &\approx \int_b^a \frac{1}{x - f(x)} dx \\
 &\approx \sum_{i=2}^{n-1} \frac{\Delta x_i}{x_i - f(x_i)}
 \end{aligned}$$

where $x_0 = b$, $x_n = a$, $x_i = b + i\frac{a-b}{n}$, and $\Delta x_i = \frac{x_{i+1} - x_{i-1}}{2}$.

Algorithm 3 Design method for minimizing the number of required iterations

Input: $l, r, L, w, l_{\min}, l_{\max}, Q, I_{\max}$

- 1: **Initialization:** Code parameters of the regular SC-LDPC ensemble
 - 2: **for** Iter = 1 to I_{\max} **do**
 - 3: **for** $u = L/2$ to 1 **do**
 - 4: Calculate the BP threshold ϵ^{BP} by the DE equations (2)
 - 5: Pre-computation of $\delta_u(z)$ by **Algorithm 1**
 - 6: Obtain $\lambda^*(x) = \sum_{k=l_{\min}}^{l_{\max}} \lambda_k^* x^{k-1}$ by solving the following LP problem

$$\begin{aligned} & \text{minimize} \quad \sum_{q=2}^{Q-1} \frac{\Delta z_q}{z_q - \epsilon^{\text{BP}} \lambda_u(\delta_u(z_q))} \\ & \text{subject to} \quad \frac{1}{\int_0^1 \lambda^*(x) dx} = l \\ & \quad \epsilon^{\text{BP}} \lambda^*(\delta_u(z_q)) < z_q \text{ for } 1 \leq q \leq Q, \lambda^*(1) = 1 \end{aligned}$$
 - 7: $\lambda_u(x) = \lambda^*(x), \lambda_{L-u+1}(x) = \lambda^*(x)$
 - 8: **end for**
 - 9: **end for**
-

In the case of the one-dimensional DE $z^{(\ell+1)} = \epsilon \lambda_u(\delta_u(z^{(\ell)}))$ of the SC-LDPC ensemble, the iterative function is given as $f(x) = \epsilon \lambda_u(\delta_u(x))$. Thus, the number of required iterations from the initial value $z_Q = \epsilon$ to the target value $z_1 = \epsilon/Q$ is approximated as

$$\sum_{q=2}^{Q-1} \frac{\Delta z_q}{z_q - \epsilon \lambda_u(\delta_u(z_q))} \quad (3)$$

where $\Delta z_q = 1/Q$. Thus, the design method that minimizes the number of iterations can be summarized as Algorithm 3, where the objective function is to minimize the equation (3). In Algorithm 3, there is the additional constraint $1/\int_0^1 \lambda^*(x) dx = l$, which implies that the average variable node degree should be equal to the variable node degree l of the initial regular SC-LDPC ensemble to maintain the design rate. For example, coefficients of $\lambda_u(x)$ obtained by Algorithm 3 with $l = 4, r = 8, L = 20, w = 3, l_{\min} = 3, l_{\max} = 10, Q = 1000$, and $I_{\max} = 10$ are shown in Table II. Due to the assumption of symmetry, we only express the degree distributions

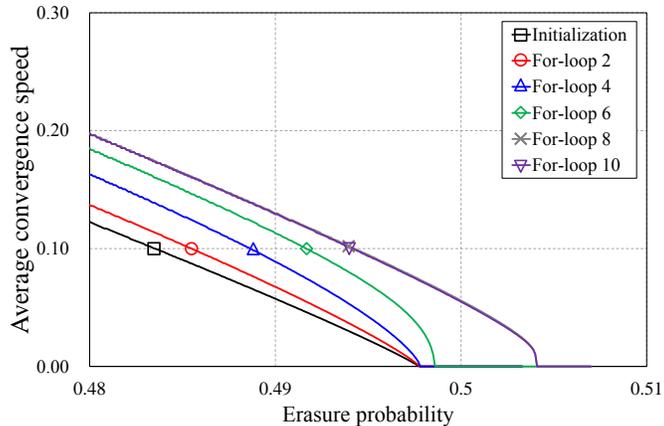


Fig. 4. Evolution of average convergence speed as the for-loop in Algorithm 3 proceeds.

TABLE II
COEFFICIENTS OF $\lambda_u(x)$ FOR THE SC-LDPC ENSEMBLE OBTAINED BY ALGORITHM 3 FOR $L = 20$

u	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9
1	0	1	0	0	0	0	0	0
2	0.191645	0.488946	0.319409	0	0	0	0	0
3	0.385168	0	0.576933	0.036234	0	0	0.001665	0
4	0.458708	0	0.206459	0.334833	0	0	0	0
5	0.526000	0	0	0.291995	0.182005	0	0	0
6	0.518531	0.128315	0	0	0.052264	0.300890	0	0
7	0.612828	0	0	0	0	0.264387	0.035736	0.087049
8	0.642857	0	0	0	0	0	0	0.357143
9	0.642857	0	0	0	0	0	0	0.357143
10	0.642857	0	0	0	0	0	0	0.357143

of half of the positions. Note that, to maintain the advantages of the regular SC-LDPC codes whose minimum variable node degree is larger than 2 [26], we generally initialize the algorithm from the regular SC-LDPC ensemble with $l = 4$ and a minimum degree constraint as $l_{\min} = 3$. Instead, Algorithm 3 is not initialized from the regular SC-LDPC ensemble with $l = 3$ because a non-zero fraction of degree-two variable nodes is unavoidably required to introduce a fraction of variable nodes with degree larger than 3.

In order to investigate the change of the overall number of required iterations in the DE equations (2) at each local design, the evolution of the average convergence speed is shown in

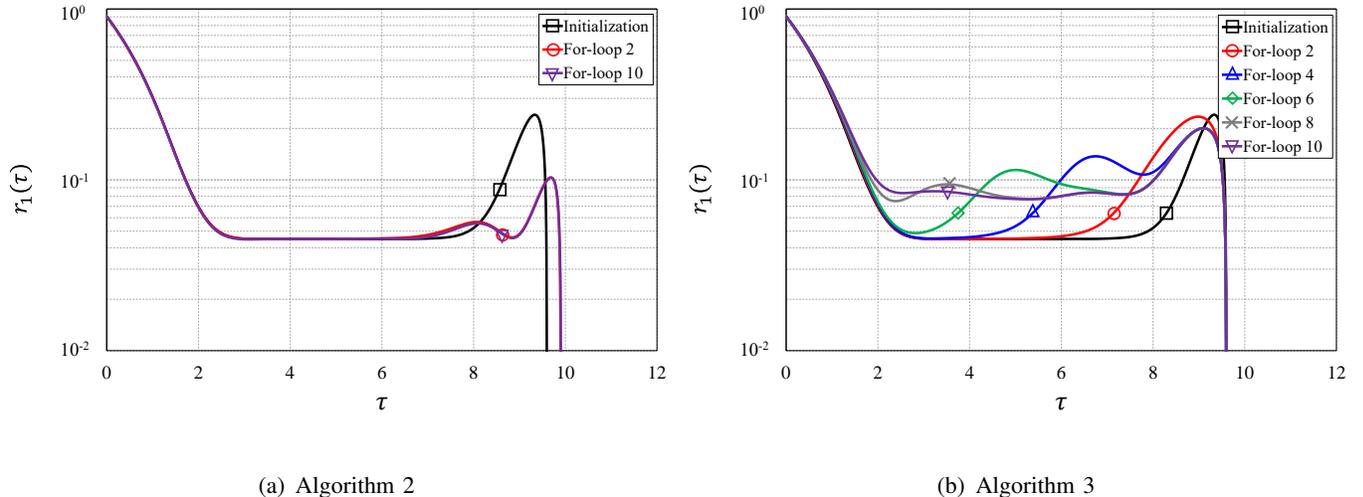


Fig. 5. Evolution of $r_1(\tau)$ for the SC-LDPC ensembles obtained by Algorithms 2 and 3 as the for-loop proceeds.

Fig. 4 as the for-loop in Algorithm 3 proceeds. It shows that the average convergence speed increases steadily as the local design of each position proceeds. In addition, we note an increase in the BP threshold at for-loop 6, where the BP threshold corresponds to the erasure probability that the average convergence speed becomes zero.

To observe the change in the SC-LDPC ensemble according to the local design of Algorithms 2 and 3 graphically, the expected graph evolution can be used. With the differential equations in Section II-C, Fig. 5 shows the evolutions of $r_1(\tau)$ at $\epsilon = 0.48$ for the SC-LDPC ensembles obtained by the algorithms as the for-loop in the algorithms proceeds. The evolution of $r_1(\tau)$ at the initialization corresponds to $r_1(\tau)$ for the regular SC-LDPC ensemble that is analyzed in [25]. According to the analysis in [25], $r_1(\tau)$ remains constant at the local minimum for a certain interval of τ , referred to as the critical phase. On the basis of the critical phase, the evolution of $r_1(\tau)$ is divided into three phases: the initial phase, the critical phase, and the third phase. Most of the variable nodes located in the boundary positions have already been recovered in the initial phase. Then, in the critical phase, two decoding waves emerge and travel along the inner positions while recovering the variable nodes through which the decoding waves pass. If the local performance of each position is defined as the number of degree-one check nodes when the decoding waves pass through the position, the steady value of $r_1(\tau)$ in the critical phase implies that the local performances of the inner positions are all the same. Moreover, the

TABLE III

BP THRESHOLDS OF THE REGULAR SC-LDPC ENSEMBLES AND THE SC-LDPC ENSEMBLES OBTAINED BY ALGORITHM 3

	ϵ^{BP}			
	$L = 10$	$L = 20$	$L = 30$	$L = 40$
Regular SC-LDPC	0.4981	0.4977	0.4977	0.4977
SC-LDPC, Alg. 3	0.5241	0.5069	0.5027	0.5014

BP threshold of the SC-LDPC ensemble is determined by the local performance of the inner positions. Finally, in the third phase, the decoding waves meet around the center and the variable nodes located at the center positions are recovered, which corresponds to the increase in the $r_1(\tau)$ value.

Fig. 5(a) shows the evolution of $r_1(\tau)$ as the for-loop in Algorithm 2 proceeds. After performing for-loop 2 in Algorithm 2, the degree distributions of the center positions from 9 to 12 are locally designed to decrease their average variable node degree. However, the value of $r_1(\tau)$ is decreased in the third phase at the expense of lowering the average variable node degree of the center positions as in Fig. 5(a). In addition, it is observed that the minimum value of $r_1(\tau)$ in the third phase is nearly identical to the value of $r_1(\tau)$ in the critical phase due to the constraint that the BP threshold should be maintained in Algorithm 2. If the minimum value of $r_1(\tau)$ in the third phase becomes smaller than the value of the critical phase, the BP threshold is degraded. In other words, because the value of $r_1(\tau)$ in the third phase is larger than the critical phase value of the initial regular SC-LDPC ensemble, there is a room for improvement in the design rate for the center positions. However, an improvement in the design rate of the inner positions cannot be achieved while maintaining the BP threshold. Accordingly, the average variable node degree of the positions except the center positions does not change, as indicated in Fig. 3, and the evolution of $r_1(\tau)$ is unchanged from for-loops 2 to 10.

Compared to Algorithm 2, Fig. 5(b) shows the evolution of $r_1(\tau)$ as the for-loop in Algorithm 3 proceeds. After for-loop 2, where the local design has been performed from center positions 9–12, $r_1(\tau)$ in the right part of the critical phase increases. This means that the local design improves the local performance of the center positions and accordingly the length of the critical phase is shortened and the overall number of required iterations is decreased. Likewise, from for-loops 2 to 4, the length of the critical phase is steadily shortened. After for-loop 6, the

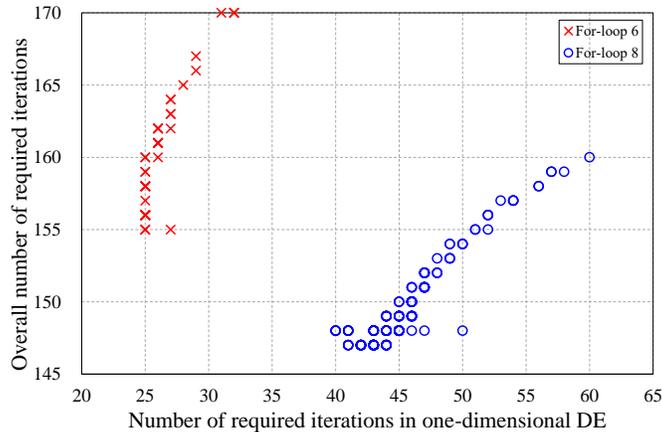


Fig. 6. Correlation between the two numbers of required iterations in the DE equations of the SC-LDPC ensemble and one-dimensional DE equation.

critical phase observed in the initial regular SC-LDPC ensemble disappears and only a single local minimum point remains. The local minimum value is greater than the value of $r_1(\tau)$ in the critical phase, which means that the BP threshold is increased. Further, in for-loop 8, the critical phase observed in the initialization completely disappears, which implies that the BP threshold is increased significantly. In other words, the BP threshold is increased in for-loop 6 because the local performances of positions 7–14 have already been improved. Conversely, the BP threshold does not increase up to for-loop 4 because the local performances of positions 1–6 and 15–20, which are not yet designed at this point, become a bottleneck despite the fact that the local performances of positions 7–14 are improved by the local design. On the other hand, the BP threshold is increased when the bottleneck positions 5 and 16 are designed in for-loop 6. In summary, the BP threshold can be increased if the degree distributions of all positions are properly designed by Algorithm 3. In Table III, the BP thresholds of the regular SC-LDPC ensemble and the SC-LDPC ensembles obtained by Algorithm 3 are shown for various values of L .

It is important to remark that the degree distributions obtained by Algorithm 3 are not the global optimum that minimizes the overall number of required iterations in the DE equations (2) because the local design is performed instead of the global design and the one-dimensional DE assumes different decoding scheduling from the conventional parallel scheduling as mentioned in

Section III-A. Therefore, minimizing the number of required iterations in the one-dimensional DE equation does not guarantee minimization of the overall number of required iterations in parallel scheduling. However, we numerically observe that these two numbers of required iterations are generally proportional. Fig. 6 shows the correlation between the numbers of required iterations in the one-dimensional DE equation $z^{(\ell+1)} = \epsilon \lambda_u(\delta_u(z^{(\ell)}))$ and DE equations in (2) at for-loops 6 and 8 in Algorithm 3. Each point in Fig. 6 is the numbers of required iterations measured at $\epsilon = \epsilon^{\text{BP}} - 0.01$ for a randomly generated degree distribution. According to Fig. 6, it can be seen that both numbers of required iterations are generally proportional and thus the objective function of Algorithm 3 is proper for reducing the overall number of required iterations.

D. Minimizing the Number of Required Iterations for the SC-LDPC Ensemble with Non-Uniform Check Node Degrees

In Algorithm 3, we consider the SC-LDPC ensemble with a uniform check node degree. However, additional improvements can be achieved by permitting non-uniform check node degrees for the SC-LDPC ensemble, which is considered in Algorithm 4. For example, while designing $\lambda_u(x)$, the degree r_u is changed from r_{\min} to r_{\max} and the optimal r_u value is selected such that the approximated value of the number of required iterations in (3) is minimized. Such comparison is performed for all degrees of check nodes connected to the variable nodes at position u , that is, r_u, \dots, r_{u+w-1} and then the optimal result is obtained. Note that the corresponding entries $T_{u,u}, \dots, T_{u+w-1,u}$ of the connectivity matrix \mathbf{T} should be modified according to the selected check node degree. This algorithm permits non-uniform check node degrees for each position and a different connectivity matrix from that of the regular SC-LDPC ensemble. Because it adds a degree of freedom in the design of $\lambda_u(x)$, Algorithm 4 has higher complexity than Algorithm 3 but gives code parameters.

Unlike Algorithm 3, Algorithm 4 can be initialized with the regular SC-LDPC ensemble with $(l, r) = (3, 6)$ and the minimum variable node degree $l_{\min} = 3$ because the average variable node degree $1/\int_0^1 \lambda_u(x) dx$ can be higher than $l = 3$ if the degrees of the connected check nodes are increased. For example, consider when the positions $u = L/2$ and $L/2 + 1$ are firstly designed for $(l, r) = (3, 6)$ and $w = 3$. Recall that the number of edges $\sum_i T_{i,u}$ coming into the variable nodes at position u should be equal to the number of sockets $M/\int_0^1 \lambda_u(x) dx$ and that there are $Ml/w = M$ edges between two connected positions. Thus, the average variable node degree

TABLE IV
BP THRESHOLDS OF THE SC-LDPC ENSEMBLES FOR $L = 20$ AND $w = 3$

	ϵ^{BP}
Regular SC-LDPC	0.4977
SC-LDPC, Alg. 3	0.5069
SC-LDPC, Alg. 4, $(l, r) = (4, 8)$	0.5170
SC-LDPC, Alg. 4, $(l, r) = (3, 6)$	0.5247

TABLE V
COEFFICIENTS OF $\lambda_u(x)$ FOR THE SC-LDPC ENSEMBLE OBTAINED BY ALGORITHM 4 FOR $L = 20$

i	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9
1	0.121726	0.675397	0.202877	0	0	0	0	0
2	0.779293	0	0	0	0	0.102748	0.117960	0
3	0.571231	0	0	0	0.335636	0.093134	0	0
4	0.460899	0	0	0	0.067294	0.471807	0	0
5	0.588881	0	0	0	0.129720	0.281399	0	0
6	0.286159	0.199364	0.237952	0	0	0	0.157692	0.118833
7	0.171139	0	0.810973	0.017888	0	0	0	0
8	0	1	0	0	0	0	0	0
9	0.642857	0	0	0	0	0	0	0.357143
10	1	0	0	0	0	0	0	0

$1/\int_0^1 \lambda_u(x)dx$ of the variable nodes at position u should be equal to 3 because the number of edges coming to the variable nodes at position u is $\sum_i T_{i,u} = 3M$ for the regular SC-LDPC ensemble. However, if the degree of check nodes at position u is increased from 6 to 7 and the number of edges between variable nodes and check nodes at position u is accordingly increased from M to $\frac{3}{2}M$ as an example, the number of edges connected to variable nodes at position u becomes $\frac{7}{2}M$. Accordingly, the average variable node degree becomes $\frac{7}{2} \geq 3$. Therefore, variable nodes of higher degrees can be introduced while maintaining the minimum variable node degree l_{\min} as 3.

Table IV summarizes the BP thresholds of the SC-LDPC ensembles obtained by the algorithms proposed so far, which shows that the ensemble obtained by Algorithm 4 with $(l, r) = (3, 6)$ is the best in terms of the BP threshold. Table V shows the degree distributions obtained by Algorithm

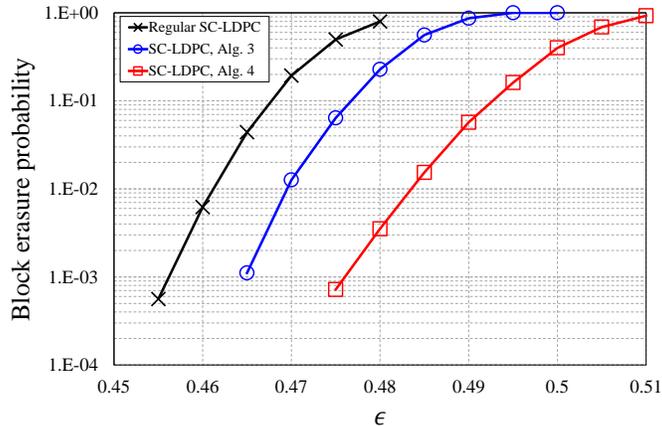


Fig. 7. Block erasure probability of the SC-LDPC codes for $L = 20$.

B. Performance Improvement by Multi-Edge Type Check Nodes

The SC-LDPC ensemble defined in Section II-A is referred to as the randomly constructed ensemble because the variable node sockets at position u are connected to the check nodes at position v at random with probability $T_{v,u}/\sum_i T_{i,u}$. However, as mentioned in previous works [12], [15], [25], randomly constructed SC-LDPC codes are inferior to SC-LDPC codes with specific structures such as protograph-based SC-LDPC codes in terms of the finite-length code performance. Protograph-based SC-LDPC codes have the MET structure [21] for both variable and check nodes. However, in this paper, we consider the SC-LDPC ensemble having the MET structure only for check nodes in the design of the degree distributions because it should be assumed that the edges of variable nodes are connected randomly to utilize the one-dimensional DE.

It is assumed that the edges connected to a check node consist of w edge types. An edge of type t , which is connected to a check node at position v , is connected to a variable node at position $v - t + 1$ for $1 \leq t \leq w$. A degree type of check nodes is represented by the vector $\underline{d} = (d_1, \dots, d_w)$ such that d_t is the number of edges of type t . Also, at position v , it is assumed that there are w degree types represented by each row of a $w \times w$ degree type matrix \mathbf{S}^v . To be specific, $M_c \frac{1}{w}$ check nodes have a degree type represented by $\underline{d} = \underline{S}_k^v \triangleq (S_{k,1}^v, \dots, S_{k,w}^v)$, where \underline{S}_k^v denotes the k th row vector of \mathbf{S}^v for $k \in [1, w]$. For example, the MET regular SC-LDPC

ensemble with $r = 6$ and $w = 3$ has

$$\mathbf{S}^v = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix},$$

which means that all check nodes have the same degree type vector $(2, 2, 2)$ similar to the structure of the protograph-based SC-LDPC codes [12]. For the regular SC-LDPC ensemble when r/w is not an integer, degree types of check nodes are not yet defined in the protograph-based ensemble. To define these degree types as closely as possible to the structure in the protograph-based ensemble, the first row \underline{S}_1^v of the degree type matrix \mathbf{S}^v for position v is defined as

$$\underline{S}_1^v = \left(\overbrace{[\lceil r/w \rceil], \dots, [\lceil r/w \rceil]}^{r \bmod w}, \overbrace{[\lfloor r/w \rfloor], \dots, [\lfloor r/w \rfloor]}^{w - (r \bmod w)} \right).$$

In addition, the k th row \underline{S}_k^v for $2 \leq k \leq w$ is obtained by right circular shifting the first row \underline{S}_1^v by k times. For example, the regular SC-LDPC ensemble with $r = 8$ and $w = 3$ has the following degree type matrix:

$$\begin{bmatrix} 3 & 3 & 2 \\ 2 & 3 & 3 \\ 3 & 2 & 3 \end{bmatrix}.$$

The number of edges between the check nodes at position v and the variable nodes at position $v - t + 1$ becomes $M_c \frac{1}{w} \sum_k S_{k,t}^v$, which corresponds to $T_{v,v-t+1}$. Generally, the construction method of MET SC-LDPC codes is described as follows. First, the placement of variable and check nodes is identical to that of the randomly constructed SC-LDPC codes in Section II-A. We label the variable and check node sockets at each position and assign the degree type vector \underline{S}_k^v to some $M_c \frac{1}{w}$ check nodes at position v for $k \in [1, w]$. Then, the number of check node sockets connected to variable nodes at position $v - t + 1$ is $M_c \frac{1}{w} \sum_k S_{k,t}^v = T_{v,v-t+1}$ and let these check node sockets be group t for $t \in [1, w]$. Let π_u be a random permutation on $[1, \sum_i T_{i,u}]$ for position u . Divide π_u into w disjoint subsets denoted by π_u^1, \dots, π_u^w such that the size of π_u^t becomes $T_{u+t-1,u}$. Then, the j th check node socket in group t at position $u + t - 1$ is connected to the $\pi_u^t(j)$ th variable node socket at position u .

Because the $1/w$ fraction of the check nodes at position v have check node degree type \underline{S}_k^v , the node perspective degree distribution [21] of the check nodes at position v is represented as

$$R_v(\underline{x}) = \sum_{k=1}^w \frac{1}{w} \underline{x}^{\underline{S}_k^v}$$

where $\underline{x} = (x_1, \dots, x_w)$ and $\underline{x}^{\underline{S}_k^v}$ denotes $\prod_{t=1}^w x_t^{S_{k,t}^v}$. From the node perspective degree distribution, the edge perspective degree distribution of the check nodes at position v is obtained as

$$\begin{aligned} \rho_v(\underline{x}) &= \left(\frac{\partial R_v(\underline{x})}{\partial x_1} / \frac{\partial R_v(\underline{1})}{\partial x_1}, \dots, \frac{\partial R_v(\underline{x})}{\partial x_w} / \frac{\partial R_v(\underline{1})}{\partial x_w} \right) \\ &= \left(\sum_{k=1}^w \frac{S_{k,1}^v}{r_v} \underline{x}^{\underline{S}_k^v} / x_1, \dots, \sum_{k=1}^w \frac{S_{k,w}^v}{r_v} \underline{x}^{\underline{S}_k^v} / x_w \right). \end{aligned}$$

Similar to the DE equations in (2), let $x_u^{(\ell)}$ denote the average erasure probability of messages at iteration ℓ emitted from the variable nodes at position u . In addition, let the t th element of a vector $\underline{y}_v^{(\ell)} = (y_{v,1}^\ell, \dots, y_{v,w}^\ell)$ be the average erasure probability of messages at iteration ℓ emitted from the check nodes at position v to the variable nodes at position $v - t + 1$. Set the initial conditions as $x_u^{(0)} = \epsilon$ for all u . For simple expression of the DE equations, it is assumed that there exist variable nodes at position u for $u < 0$ with the assumption $x_u^{(\ell)} = 0$ for all ℓ . Then, the DE equations for the MET SC-LDPC ensembles are described as

$$\begin{aligned} \underline{y}_v^{(\ell)} &= 1 - \rho_v(1 - x_{v-(w-1)}^{(\ell)}, \dots, 1 - x_v^{(\ell)}) \\ x_u^{(\ell+1)} &= \epsilon \lambda_u \left(\frac{\sum_{t=1}^w T_{u+t-1,u} y_{u+t-1,t}^{(\ell)}}{\sum_i T_{i,u}} \right). \end{aligned} \quad (4)$$

Algorithms 3 and 4 can be applied to the MET SC-LDPC ensemble using the DE equations in (4). Note that especially for Algorithm 4, the degree type matrix \mathbf{S}^v should be changed according to changes of check node degrees and the connectivity matrix. Because the DE equations are changed as in (4), the resulting degree distributions differ from those obtained by the randomly constructed SC-LDPC ensemble. Table VI compares the BP thresholds between the randomly constructed and MET SC-LDPC ensembles, indicating that imposing the MET structure improves the BP threshold.

TABLE VI
COMPARING BP THRESHOLDS OF THE RANDOMLY CONSTRUCTED AND MET SC-LDPC ENSEMBLES

	ϵ^{BP}	
	Random	MET
Regular SC-LDPC	0.4977	0.4977
SC-LDPC, Alg. 3	0.5069	0.5079
SC-LDPC, Alg. 4, $(l, r) = (4, 8)$	0.5170	0.5211
SC-LDPC, Alg. 4, $(l, r) = (3, 6)$	0.5247	0.5290

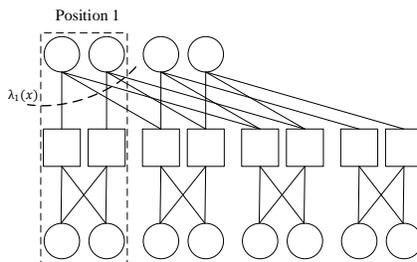


Fig. 8. A Tanner graph of the SC-RA ensemble for $q = 3$, $L = 2$, and $M = 2$.

C. Applying the Proposed Algorithms to SC-RA Codes

In this subsection, it is shown that the proposed algorithms can be applied to other coupled code structures. Especially, the SC-RA code ensemble [16] is considered due to its superior decoding performance reported in [15]. It is known that SC-RA codes outperform regular SC-LDPC codes with the same design rate, code length, and decoding complexity [15], [16]. The SC-RA ensemble is constructed by coupling the L disjoint uncoupled (q, a) -regular RA ensemble. Specifically, the SC-RA ensemble with $q = a$ is considered in this paper. Fig. 8 shows a Tanner graph of the SC-RA ensemble for $L = 2$, $q = 3$, and $M = 4$. There are $M/2$ variable nodes of degree q at each position from 1 to L in the upper side and $M/2$ variable nodes of degree 2 at each position from 1 to $L + q - 1$ in the lower side. The variable nodes of degree q at position u are connected to the check nodes at positions $u, \dots, u + q - 1$, which implies that the possible number w of positions to be connected is equal to the variable node degree q . On the other hand, the variable nodes of degree 2 are connected to the check nodes at the same position. We consider the SC-RA ensemble with the MET structure whose check nodes at position v are connected to variable nodes of degree q based on a $q \times q$ degree type matrix \mathbf{S}^v . Since w is equal

TABLE VII
BP THRESHOLDS OF THE REGULAR SC-RA AND THE SC-RA ENSEMBLES OBTAINED BY ALGORITHM 3

	ϵ^{BP}			
	$L = 10$	$L = 20$	$L = 30$	$L = 40$
Regular SC-RA	0.5107	0.4949	0.4946	0.4946
SC-RA, Alg. 3	0.5399	0.5128	0.5051	0.5017

to q , \mathbf{S}^v becomes all one matrix for all v . The edge connection between the variable nodes of degree q and the check nodes is established using a method similar to that of the MET SC-LDPC ensemble. The remaining construction method follows the process described in Section IV-B.

The SC-RA ensemble with non-uniform degree distributions has an irregular degree distribution $\lambda_u(x)$ with a minimum degree of 3 for the variable nodes in the upper side at position u , while the variable nodes of degree 2 remain unchanged in the lower side. Using the design algorithm of Algorithm 3, the degree distributions of the SC-RA ensemble with non-uniform degree distributions can be obtained. Table VII shows that the BP threshold of the SC-RA ensemble with non-uniform degree distributions designed by Algorithm 3 is superior to that of the regular SC-RA ensemble. Note that the result obtained by Algorithm 4 is not included because the performance improvement is insignificant compared to the results by Algorithm 3.

The design method of degree distributions of the SC-RA ensemble is conceptually identical to that of the SC-LDPC ensemble. However, when we generate a code instance from the designed ensemble, small stopping sets [21] consisting of variable nodes of degree lower than q can be made with a high probability due to the existence of variable nodes of degree 2. Therefore, it is practically important to avoid such small stopping sets for finite-length codes. We consider two types of stopping sets to be avoided, as presented in Fig. 9, where the stopping sets consist of the hatched variable nodes. Type 1 stopping sets consist of one variable node of degree i , $3 \leq i < q$, with a set of variable nodes of degree 2 and all edges of a variable node degree i have at least one edge pair connected to check nodes at the same position. Similarly, type 2 stopping sets consist of multiple variable nodes of degree i , $3 \leq i < q$, with a set of variable nodes of degree 2 and all edges of the multiple variable nodes have at least one edge pair connected to check nodes at the same position. We disregard other stopping sets such as stopping sets consisting of variable nodes whose degrees are greater than or equal to q because the probability of occurrence

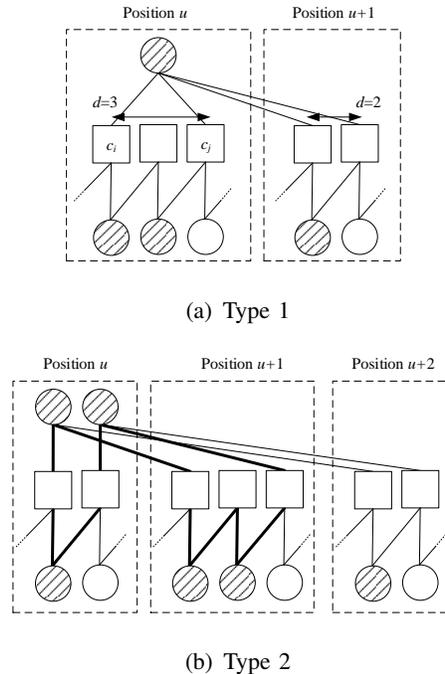


Fig. 9. Two types of stopping sets produced by low degree variable nodes.

of such stopping sets is identical to that in the regular SC-RA ensemble.

Consider a loop consisting of M_c check nodes and M_c variable nodes of degree 2 at a position in a Tanner graph. If the edges between check nodes and variable nodes of degree 2 are properly connected, only one loop of length $2M_c$ is produced at each position. On the loop, define *distance* d in the loop of two check nodes as the number of check nodes between two check nodes including themselves. For example, the distance in the loop of check nodes c_i and c_j in Fig. 9(a) is 3. Consider a case in which two edges of a variable node are connected to two check nodes that are separated by distance d in the loop at the same position. This inevitably produces a cycle of length $2d$. Further, if all edges of a variable node are contained in such cycles, a type 1 stopping set is produced.

However, type 1 stopping sets are easily avoided by imposing the MET structure on variable nodes as well as check nodes. For an ensemble with randomly connected variable nodes, the edges of a variable node of degree i at position u can be connected to check nodes located at the same position, which can produce type 1 stopping sets. If all edges of a variable node are connected to check nodes at different positions, type 1 stopping sets are avoided accordingly.

Thus, we impose the MET structure on variable nodes such that each edge of a variable node is connected to the check node at different positions. To be specific, the first row $\underline{S}_1^{u,i}$ of the $q \times q$ degree type matrix $\mathbf{S}^{u,i}$ for variable nodes of degree i at position u is defined as

$$\underline{S}_1^{u,i} = \left(\overbrace{(\lceil i/q \rceil, \dots, \lceil i/q \rceil)}^{i \bmod q}, \overbrace{(\lfloor i/q \rfloor, \dots, \lfloor i/q \rfloor)}^{q - (i \bmod q)} \right)$$

and the remaining rows are obtained by circularly shifting the first row. For example, the degree type matrix for degree $i = 3$ and $q = 5$ is given as

$$\mathbf{S}^{u,3} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix},$$

which implies that all edges of variable nodes of degree 3 are connected to check nodes located at different positions and thus type 1 stopping sets can be avoided.

Even if all edges of a variable node are connected to check nodes at different positions, type 2 stopping sets can be made. To avoid type 2 stopping sets, we use the PEG algorithm [27] with an additional constraint. Let \mathcal{N}_v^l be the neighborhood of variable node v within depth l , which denotes the set of check nodes reached by a computation tree spreading from variable node v within depth l . In the conventional PEG algorithm, a cycle of length $2l + 2$ is avoided by connecting the edges of variable node v to the check nodes not in \mathcal{N}_v^l . However, the conventional PEG algorithm cannot prevent the production of type 2 stopping sets if the girth of the PEG algorithm is lower than the length of the cycles in the stopping set. For example, the length of the cycle denoted by the bold line in Fig. 9(b) is 10 and thus it is not avoided by the PEG algorithm with a girth such as 6 or 8. Thus, it is difficult to prohibit connecting the check nodes in \mathcal{N}_v^l using the conventional computational tree. Instead, we consider the *expanded computation tree* by d , which is defined as follows. At depth l of the conventional computation tree from variable node v , there exist check nodes directly connected to variable node v within depth $l - 1$, while the depth l of the expanded tree contains not only the directly connected check nodes but also the set of check nodes within distance d in the loop from the directly connected check nodes. Also, only the variable nodes of degree lower than q are included without other higher

degree variable nodes in the expanded computation tree, as we are dealing with stopping sets consisting of such low degree variable nodes. Let $\mathcal{N}_v^{l,d}$ be the set of check nodes within depth l in the expanded computation tree by d from a variable node v . Then, by connecting a new edge of variable node v to the check nodes not in $\mathcal{N}_v^{l,d}$, a cycle of a length up to $(2l+2) + (l+1)d$ is avoided in the sub-graph consisting of check nodes and low degree variable nodes. Because the expanded tree contains only low degree variable nodes, this additional constraint can be applied for large l and d to avoid small size of type 2 stopping sets. For example, the cycle in Fig. 9(b) is avoided by prohibiting a connection between variable node v and a check node in $\mathcal{N}_v^{1,3}$ and thus the type 2 stopping set can be avoided. Note that because this constraint aims to avoid stopping sets consisting of low degree variable nodes, the conventional PEG algorithm should also be applied at the same time.

D. Comparing Finite-Length Performances of the SC-LDPC and SC-RA Codes

Fig. 10 shows the block erasure probability of the SC-LDPC codes for $w = 3$ and $L = 10, 20, 30$. Each code instance for the SC-LDPC ensembles is obtained by the PEG algorithm with $M = 990$. The code lengths for $L = 10, 20$, and 30 are 9,900, 19,800, and 29,700, respectively. For $L = 20$ in Fig. 10 (b), the dashed lines correspond to the randomly constructed SC-LDPC codes and the solid lines correspond to the MET SC-LDPC codes. According to the simulation result, it is confirmed that the MET SC-LDPC codes have better finite-length performance than the randomly constructed SC-LDPC codes. Moreover, the improvement by Algorithms 3 and 4 from the regular SC-LDPC codes is observed for both the randomly constructed and the MET SC-LDPC codes. For $L = 10$, we present the finite-length performances of the MET SC-LDPC codes in Fig. 10(a), which shows the performance improvement by the proposed algorithms. It is also shown that the SC-LDPC code obtained by Algorithm 4 outperforms the code in [20] for $L = 10$. The code length of the code in [20] is 10,000, which is slightly larger than those of the other codes. The performance improvement by the proposed algorithms is also shown for $L = 30$ in Fig. 10(c).

We include the simulation results for the SC-RA codes for $q = 5$ in Fig. 10. To match their code lengths with those of the SC-LDPC codes, the SC-RA code instances are obtained from their ensembles with $M = 900$. In terms of the code rate, the SC-RA codes have an advantage because the code rate of the SC-RA codes with $q = 5$ is slightly higher than that of the SC-

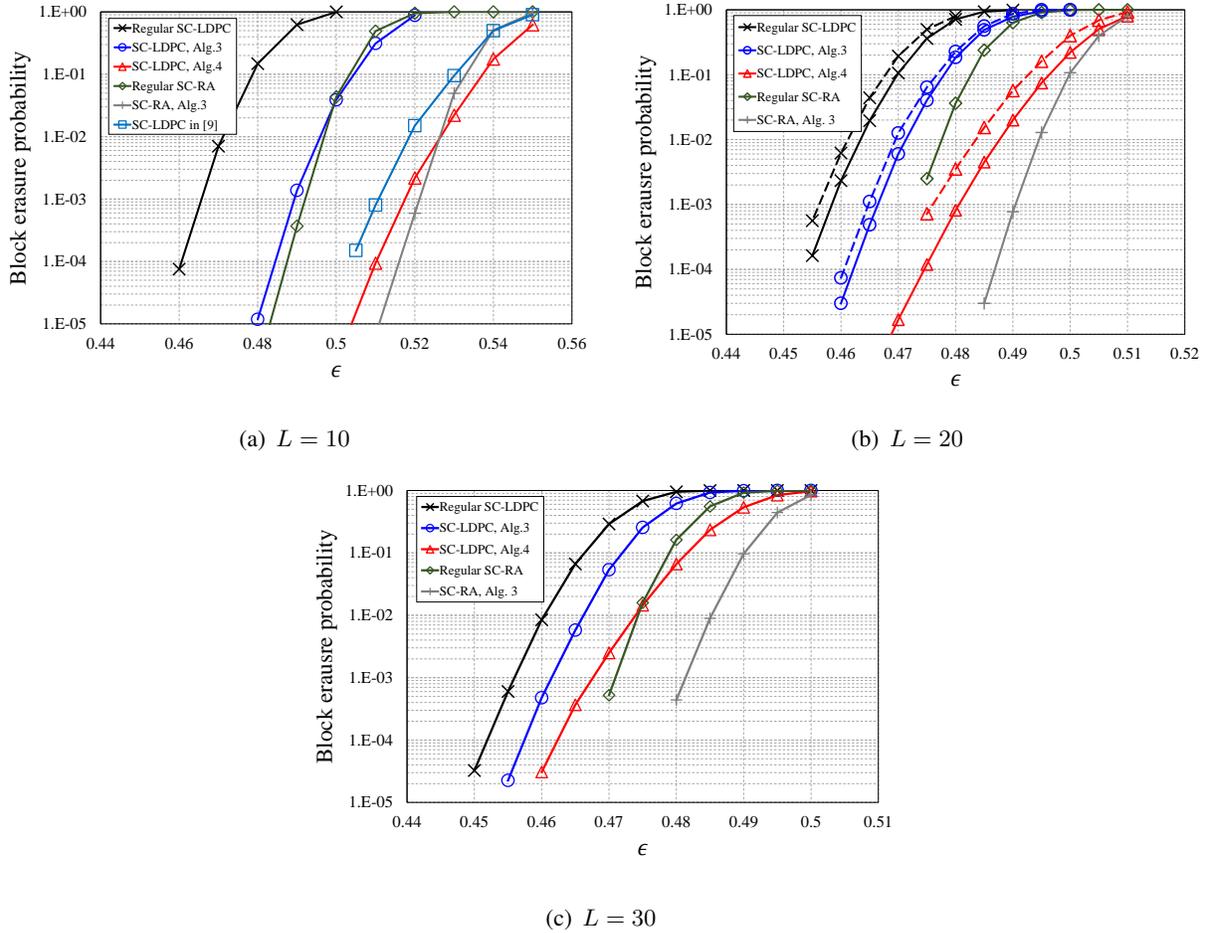


Fig. 10. Block erasure probability of the SC-LDPC and SC-RA codes for $L = 10, 20$, and 30 .

LDPC codes with $w = 3$ for the same L [15], [16]. Each code instance of the SC-RA codes with the designed non-uniform degree distributions is obtained using the PEG algorithm with an additional constraint avoiding a connection between variable node v and check nodes in $\mathcal{N}_v^{1,7}$. For $L = 20$, the code length of all the codes in Fig. 10(b) is 19,800. As in the SC-LDPC codes, the SC-RA codes obtained by Algorithm 3 show better performance than the regular SC-RA codes for $L = 10, 20$, and 30 . Comparing all the results in Fig. 10, we can confirm that the SC-RA codes obtained by Algorithm 3 are the best in terms of the finite-length performance.

V. CONCLUSION

In this paper, we proposed design methods for SC-LDPC codes with non-uniform degree distributions. The proposed methods are based on the local design of degree distribution at each

position by solving LP problems, which is enabled by the pre-computation of the input/output message relationship. It was found that the objective function of the LP problem should be carefully selected, where the proper objective function is minimizing the number of required iterations. It was shown that the designed degree distributions obtained by the proposed methods substantially improve the performance of SC-LDPC codes. We also presented the methods which improve the finite-length performance further such as imposing the MET structure and applying the design methods to the SC-RA code structure. Simulation results confirm that the proposed methods improve the finite-length performance for both SC-LDPC and SC-RA codes.

REFERENCES

- [1] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 7761–7813, Dec. 2013.
- [2] M. Lentmaier, A. Sridharan, D. J. Costello, Jr., and K. Sh. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.
- [3] 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.
- [4] S. Kudekar, C. Measson, T. J. Richardson, and R. L. Urbanke, "Threshold saturation on BMS channels via spatial coupling," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Brest, France, Sep. 2010, pp. 309–313.
- [5] A. Yedla, Y.-Y. Jian, P. S. Nguyen, and H. D. Pfister, "A simple proof of threshold saturation for coupled scalar recursions," in *Proc. 7th Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Gothenburg, Sweden, Aug. 2012, pp. 51–55.
- [6] —, "A simple proof of Maxwell saturation for coupled scalar recursions," *IEEE Trans. Inf. Theory*, vol. 60, no.11, pp. 6943–6965, Nov. 2014.
- [7] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," Jet Propulsion Lab., Pasadena, CA, USA, INP Progress Report 42–154, 2003.
- [8] D. Divsalar, H. Jin, and R. McEliece, "Coding theorems for turbo-like codes," in *Proc. 36th Allerton Conf. Commun. Control Comput.*, Monticello, IL, USA, 1998, pp. 201–210.
- [9] D. Divsalar, S. Dolinar, C. Jones, and K. Andrews, "Capacity approaching protograph codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009.
- [10] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
- [11] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, Jr., "New Families of LDPC block codes formed by terminating irregular protograph-based LDPC convolutional codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Austin, Texas, USA, June 2010, pp. 824–828.
- [12] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, Jr., "Spatially coupled LDPC codes constructed from protographs," *IEEE Trans. Inf. Theory*, vol. 61, no. 9, pp. 4866–4889, Sep. 2015.
- [13] K. Kasai and K. Sakaniwa, "Spatially-coupled MacKay-Neal codes and Hsu-Anastasopoulos codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, St. Petersburg, Russia, 2011, pp. 747–751.

- [14] N. Obata, Y.-Y. Jian, K. Kasai, and H. D. Pfister, "Spatially-coupled multi-edge type LDPC codes with bounded degrees that achieve capacity on the BEC under BP decoding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Turkey, 2013, pp. 2433–2437.
- [15] M. Stinner and P. M. Olmos, "On the waterfall performance of finite-length SC-LDPC codes constructed from protographs," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 345–361, Feb. 2016.
- [16] S. Johnson and G. Lechner, "Spatially coupled repeat-accumulate codes," *IEEE Commun. Lett.*, vol. 17, no. 2, pp. 373–376, Feb. 2013.
- [17] W. Nitzold, G. P. Fettweis, and M. Lentmaier, "Spatially-coupled nearly regular LDPC code ensembles for rate-flexible code design," in *Proc. IEEE ICC*, Sydney, NSW, Australia, Jun. 2014, pp. 2027–2032.
- [18] V. Aref, L. Schmalen, and S. ten Brink, "On the convergence speed of spatially coupled LDPC ensembles," in *Proc. 51st Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Monticello, IL, USA, Oct. 2013, pp. 342–349.
- [19] L. Schmalen, V. Aref, J. Cho, D. Suikat, D. Rosener, and A. Leven, "Spatially coupled soft-decision error correction for future lightwave systems," *J. Lightwave Technology*, vol. 33, No. 5, pp. 1109–1116, Mar. 2015.
- [20] Y. Koganei, M. Yofune, C. Li, T. Hoshida, and Y. Amezawa "SC-LDPC code with nonuniform degree distribution optimized by using genetic algorithm," *IEEE Commun. Lett.*, vol. 20, no. 5, pp. 874–877, May 2016.
- [21] T. J. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.
- [22] V. Jamali, Y. Karimian, J. Huber, and M. A. Attari, "On the design of fast convergent LDPC codes for the BEC: An optimization approach," *IEEE Trans. Commun.*, vol. 63, no. 2, pp. 351–363, Feb. 2015.
- [23] M. R. Sanatkar and H. D. Pfister, "Increasing the rate of spatially-coupled codes via optimized irregular termination," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, Sep. 2016, pp. 31–35.
- [24] B. Smith, M. Ardakani, W. Yu, and F. R. Kschischang, "Design of irregular LDPC codes with optimized performance-complexity tradeoff," *IEEE Trans. Commun.*, vol. 58, no. 2, pp. 489–499, Feb. 2010.
- [25] P. M. Olmos and R. 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.
- [26] D. J. Costello, Jr., L. Dolecek, T. E. Fuja, J. Klierer, D. G. M. Mitchell, and R. Smarandache, "Spatially coupled sparse codes on graphs: Theory and practice," *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 168–176, Jul. 2014.
- [27] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.

Algorithm 4 Design method for minimizing the number of required iterations for the SC-LDPC ensemble with non-uniform check node degrees

Input: $l, r, L, w, M, l_{\min}, l_{\max}, r_{\min}, r_{\max}, Q, I_{\max}$

- 1: **Initialization:** Code parameters of the regular SC-LDPC ensemble
 - 2: **for** Iter = 1 to I_{\max} **do**
 - 3: **for** $u = L/2$ to 1 **do**
 - 4: Calculate the BP threshold ϵ^{BP} by the DE equations (2) and set $s = 1$
 - 5: **for** $v = u$ to $u + w - 1$ **do**
 - 6: **for** $r_v = r_{\min}$ to r_{\max} **do**
 - 7: $T_{v,u} = r_v M \frac{l}{r} - \sum_{j \neq u} T_{v,j}$, $r_{L+w-v} = r_v$, $T_{L+w-v, L-u+1} = T_{v,u}$
 - 8: $v^s = v$, $r^s = r_v$, $\mathbf{T}^s = \mathbf{T}$
 - 9: Pre-computation of $\delta_u(z)$ by **Algorithm 1**
 - 10: Obtain $\lambda_s^*(x) = \sum_{k=l_{\min}}^{l_{\max}} \lambda_{s,k}^* x^{k-1}$ by solving the following LP problem

$$\begin{aligned} &\text{minimize } I(s) = \sum_{q=2}^{Q-1} \frac{\Delta z_q}{z_q - \epsilon^{\text{BP}} \lambda_s^*(\delta_u(z_q))} \\ &\text{subject to } \frac{1}{\int_0^1 \lambda_s^*(x) dx} = \sum_i T_{i,u} / M, \\ &\epsilon^{\text{BP}} \lambda_s^*(\delta_u(z_q)) < z_q \text{ for } 1 \leq q \leq Q, \lambda_s^*(1) = 1 \end{aligned}$$
 - 11: $s = s + 1$
 - 12: **end for**
 - 13: **end for**
 - 14: $s^* = \operatorname{argmin} I(s)$
 - 15: $\lambda_u(x) = \lambda_{s^*}^*(x)$, $\lambda_{L-u+1}(x) = \lambda_{s^*}^*(x)$, $r_v = r^{s^*}$, $r_{L+w-v} = r^{s^*}$, $\mathbf{T} = \mathbf{T}^{s^*}$
 - 16: **end for**
 - 17: **end for**
-