Enumeration of the Degree Distribution Space for Finite Block Length LDPC Codes

Spencer Giddens[‡], Marco A. C. Gomes[*], João P. Vilela[†], José L. Santos[§], and Willie K. Harrison[‡]
[*]Instituto de Telecomunicações, Department of Electrical and Computer Engineering, University of Coimbra, Portugal
[†]CRACS/INESCTEC and CISUC, Department of Computer Science, University of Porto, Portugal
[‡]Department of Electrical and Computer Engineering, Brigham Young University, UT, USA
[§]University of Coimbra, CMUC, Department of Mathematics, ORCiD 0000-0002-2727-6774
Emails: giddens2spencer@gmail.com, marco@co.it.pt, jvilela@fc.up.pt, zeluis@mat.uc.pt, willie.harrison@byu.edu

***Abstract*—Current methods for optimization of low-density parity-check (LDPC) codes analyze the degree distribution pair asymptotically as block length approaches infinity. This effectively ignores the discrete nature of the space of valid degree distribution pairs for LDPC codes of finite block length. While large codes are likely to conform reasonably well to the infinite block length analysis, shorter codes have no such guarantee. We present and analyze an algorithm for completely enumerating the space of all valid degree distribution pairs for a given block length, code rate, maximum variable node degree, and maximum check node degree. We then demonstrate this algorithm on an example LDPC code of finite block length. Finally, we discuss how the result of this algorithm can be utilized by discrete optimization routines to form novel methods for the optimization of small block length LDPC codes.**

I. Introduction

Low-density parity-check (LDPC) codes were introduced by Robert Gallager in the 1960s [1], though not popularized until rediscovered by MacKay in the 1990s [2], [3], [4]. These codes correct errors in transmitted codewords via belief propagation [5], [6], a message-passing decoding algorithm that has been shown to perform very close to the theoretical Shannon capacity [7], [8], making LDPC codes very powerful in many applications [9]. Today, their use is widespread and well-documented. As a result, there have been a number of methods developed to try to optimize the code construction process (e.g., [9], [10], [11]).

As we tend towards next generation wireless machine-to-machine communication, short-packet ultrareliable and low-latency transmissions become crucial [12]. Most of the methods developed to date for the optimization of LDPC codes, consider code design asymptotically as the block length approaches infinity [9], [13]. Other methods rely on a curve-fitting approach using EXIT functions that does not explicitly take block length into account, but relies on large block length to attain reliable performance estimates for the EXIT curves [10]. Even code modification techniques involving the removal of trapping sets [11], and alternate construction techniques such as protograph-based constructions [14] rely on infinite block length analysis to some degree. Although these optimization methods have yielded positive results for large codes, they ignore the underlying structure of LDPC code degree distributions, and carry no guarantees for short block length code design. There has also been significant work on short block length LDPC code design (e.g., [15], [16], [17]), and yet these works have not explored the enumeration of all possible degree distribution pairs within fixed size constraints on the codes.

In this paper, we show how to enumerate the finite block length LDPC code degree distributions explicitly. Section II defines LDPC codes in terms of bipartite graphs, presents the constraints a finite block length LDPC code must satisfy to be considered valid, and introduces the space of degree distribution pairs over which optimization is performed. Section III then describes and analyzes the main result of the paper, an algorithm that generates all possible valid degree distribution pairs for a given block length, code rate, and maximum variable and check node degrees. We visualize the space for one particular example in Section IV, then discuss the potential this enumeration has for affecting the way LDPC code optimization is performed at finite block lengths, before concluding in Section V.

II. Background

First we include a brief note about notation used throughout the paper. If x is a vector, we denote the ith element of x by x_i. To denote the ith through jth elements of x, we write $x_{i:j}$. If j is not written explicitly in this notation (i.e. $x_{i:}$), we mean all elements of x from the ith one to the end of the vector.

LDPC codes are commonly described using bipartite Tanner graphs [18]. In these graphs, one set of nodes, called the variable nodes, represents the codeword, while the other set of nodes, called the check nodes, represents the parity-check constraints. A block of n bits is a codeword if for each check node, the mod-2 sum over all adjacent variable nodes is zero. A distribution on the variable nodes, denoted by λ, is defined to be a vector of length equal to the maximum variable node

This work was partially funded by the following entities and projects: the US National Science Foundation (Grant Award Number 1761280) the MobiWise project: From mobile sensing to mobility advising (P2020 SAICTPAC/0011/2015), co-financed by COMPETE 2020, Portugal 2020 - Operational Program for Competitiveness and Internationalization (POCI), European Union's ERDF (European Regional Development Fund), and the Portuguese Foundation for Science and Technology (FCT) through project UID/EEA/50008/2019, as well as Centre for Mathematics of the University of Coimbra - UIDB/00324/2020, funded by the Portuguese Government through FCT/MCTES.

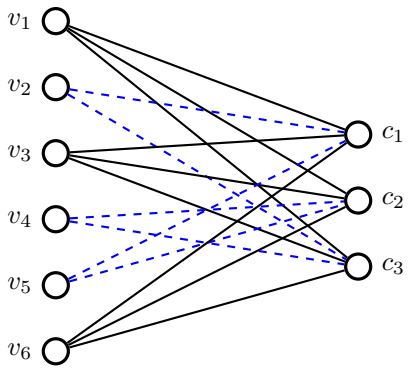

Fig. 1. An example of the type of bipartite graph commonly used to describe LDPC codes. The variable nodes are on the left and the check nodes are on the right. The variable nodes correspond to bits in the received message, and have maximum degree 3. Notice that 6 of the 15 total edges (blue, dashed) are connected to variable nodes of degree 2, while the other 9 of the 15 (black, solid) are connected to variable nodes of degree 3. So, $\lambda = (0, \frac{6}{15}, \frac{9}{15})^T$ for this LDPC code. All 15 edges are connected to check nodes of degree 5, so $\rho = (0, 0, 0, 0, \frac{15}{15})^T$.

degree, where the ith component of λ is the proportion of edges connected to variable nodes of degree i. A distribution on the check nodes is defined analogously and is denoted by ρ. A tuple containing λ and ρ is commonly referred to as a degree distribution pair. See Fig. 1 for an example.

An LDPC code can be completely determined by its parity-check matrix. However, these matrices can be large, and slight variations in matrices usually do not yield significant differences in code performance. Thus, working with an LDPC code *ensemble* or *family*, the collection of codes with the same degree distribution pair, is more common than working with specific code realizations.

Let d_v be the maximum variable node degree, d_c be the maximum check node degree, n be the block length of the code, n_e be the number of edges present in the bipartite graph, and $m = n(1-r)$, where r is the rate of the code. For a degree distribution pair to be valid, it must adhere to the following:

$$\sum_{i=1}^{d_v} \lambda_i = 1, \qquad \sum_{i=1}^{d_c} \rho_i = 1, \tag{1}$$

$$\sum_{i=1}^{d_v} \frac{\lambda_i}{i} = \frac{n}{n_e}, \qquad \sum_{i=1}^{d_c} \frac{\rho_i}{i} = \frac{m}{n_e}, \tag{2}$$

$$\lambda_i \geq 0, \qquad \rho_i \geq 0. \tag{3}$$

For the purposes of this paper, we require $\lambda_1 = \rho_1 = 0$ to ensure there are no nodes of degree 1. We recognize that degree 1 nodes can sometimes be acceptable, but care must be taken in constructing the code in that case so as to avoid hindering the decoding process [6].

Most methods for optimizing LDPC codes search the space of valid degree distribution pairs for the pair that achieves performance closest to the theoretical Shannon capacity. However, as mentioned in the introduction, this optimization is performed asymptotically as block length approaches infinity. This allows the range of potential values for the fractions on the right-hand side of the constraints in (2) to become effectively continuous. At finite block lengths, those constraints have the effect of discretizing the space of valid LDPC code degree distributions, and the discretized space is progressively more sparse as the block length decreases. Thus, the degree distribution pairs produced by common optimization methods are not, rigorously speaking, likely to be in the space of valid degree distribution pairs for finite block lengths. For small codes, many of the assumptions made in infinite block length LDPC code optimization are much farther from the truth than for large codes. Furthermore, smaller codes require more "rounding" with the degree distributions to find a valid code than do larger codes. It is also worth mentioning here that asymptotic optimization assumes there are no cycles in the graph, which are well known to hinder performance [11]. Though there are methods to reduce the number of small cycles in finite length LDPC codes [11], [14], even these techniques require *good* degree distributions as a starting point, which can be better accomplished for small codes through discrete code enumeration.

III. Methods

Using constraints (1) through (3), we show it is possible to completely enumerate every valid degree distribution pair. We assume a fixed block length, a fixed code rate and fixed maximum variable and check node degrees, then enumerate every possible degree distribution pair at that block length and code rate. For convenience, we break up the analysis into steps.

1) We show that, given a fixed number of edges in the bipartite graph and one particular valid degree distribution pair, we can find all other valid degree distribution pairs with the same number of edges.
2) We demonstrate that there are a minimum and maximum number of edges possible for a fixed block length, fixed code rate, and fixed maximum variable and check node degrees.
3) We explain a way to easily find one particular valid degree distribution pair given any number of edges between the minimum and maximum number possible.

The combination of these three steps yields an algorithm that can enumerate every valid degree distribution pair for a finite block length LDPC code. At the end of this section, we provide a complexity analysis for the algorithm.

A. Step 1: Finding Remaining Valid Pairs

For the first step, assume a fixed number of edges, n_e, and a given valid degree distribution pair, (λ, ρ). We show that all other valid degree distribution pairs can be found from these assumptions. Let $\hat{\lambda}$ be any other valid variable node degree distribution. Then we can write $\hat{\lambda} = \lambda + h$ for some step h.

Since equations (1) and (2) must hold for λ and $\hat{\lambda}$ (note $\lambda_1 = \hat{\lambda}_1 = 0$, so $h_1 = 0$), we have the constraints for h

$$\sum_{i=2}^{d_v} h_i = 0, \qquad \sum_{i=2}^{d_v} \frac{h_i}{i} = 0. \tag{4}$$

This set of constraints can be represented in matrix form as

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{d_v} \\ 1 & 1 & \cdots & 1 \end{bmatrix} h_{2:} = 0, \tag{5}$$

$$\implies Ch = \begin{bmatrix} 1 & 0 & -\frac{2}{4} & -\frac{4}{5} & \cdots & \frac{2(3-d_v)}{d_v} \\ 0 & 1 & \frac{6}{4} & \frac{9}{5} & \cdots & \frac{3d_v-6}{d_v} \end{bmatrix} h_{2:} = 0, \tag{6}$$

where C is the constraint matrix obtained by Gaussian elimination.

From (6), we see that for h to be in the null space of C as required, we can lower the degrees of freedom by two, so

$$h_2 = \sum_{i=4}^{d_v} \frac{2(i-3)}{i} h_i, \tag{7}$$

$$h_3 = \sum_{i=4}^{d_v} \frac{6-3i}{i} h_i. \tag{8}$$

Next we need to make sure that constraint (3) is met for both λ and $\hat{\lambda}$. To do this, we require

$$0 \leq \hat{\lambda}_i \implies 0 \leq \lambda_i + h_i \implies -h_i \leq \lambda_i. \tag{9}$$

These constraints can be combined with (7) and (8) to produce the matrix inequality

$$\begin{bmatrix} -\frac{2}{4} & -\frac{4}{5} & \cdots & \frac{2(3-d_v)}{d_v} \\ \frac{6}{4} & \frac{9}{5} & \cdots & \frac{3d_v-6}{d_v} \\ -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ & & \cdots & \\ 0 & 0 & \cdots & -1 \end{bmatrix} h_{4:} = Ah_{4:} \leq \lambda_{2:}, \tag{10}$$

where the $\leq$ operates element-wise. Note A is dependent on d_v, but not λ or n_e.

We know by (9) and (3) applied to the given valid λ that $-1 \leq h_i \leq 1$ for all i. In addition, we can reason that if we are going to change the given valid distribution λ by some amount h, that h_i must be an integer multiple of $\frac{i}{n_e}$. This is because the number of edges connected to variable nodes of degree i must always be an integer multiple of i, and we assume n_e is fixed.

Finally, we can summarize how we can find all other valid degree distribution pairs given a fixed number of edges, n_e, and one valid pair in particular, (λ, ρ). First, for all $i \geq 4$, create a set of values, denoted S_i for h_i by taking all integer multiples of $\frac{i}{n_e}$ in the interval $[-1, 1]$. Use these lists to construct a set of vectors

$$H = \Bigg\{ h \Big| h_i \in S_i, i \geq 4; h_1 = 0; h_2 = \sum_{i=4}^{d_v} \frac{2(i-3)}{i} h_i; \\ h_3 = \sum_{i=4}^{d_v} \frac{6-3i}{i} h_i \Bigg\}. \tag{11}$$

Now, remove all vectors from H that fail to satisfy (10) to form a new set,

$$\hat{H} = \Big\{ h \Big| h \in H, Ah_{4:} \leq \lambda_{2:} \Big\}. \tag{12}$$

The result of adding λ to all elements in $\hat{H}$ is the set of all valid variable node degree distributions for a fixed number of edges. Notice all previously done analysis applies for ρ as well as λ, so to find all valid degree distribution pairs, simply repeat the described steps with ρ in place of λ.

B. Step 2: Bounds on Number of Edges

For the second step, assume without loss of generality a fixed block length, n, fixed code rate, r, and fixed maximum variable and check node degrees, d_v and d_c, respectively. There must be a minimum number of edges possible for a valid LDPC code. To find this, we consider that there are n variable nodes and $m = n(1-r)$ check nodes. Each variable and check node must have at least two edges by assumption. So, the minimum number of edges possible based on the variable nodes is $2n$, and the minimum number possible based on the check nodes is $2m$. Since $0 < r < 1$, we have $m < n$ by construction, so $2n$ forms a tighter bound on the minimum number of edges. Thus, $n_e^{\min} = 2n$.

The maximum number of edges possible for a valid LDPC code can be found as follows. For the belief propagation algorithm, there cannot be more than one edge connecting any two nodes, meaning we require $d_v \leq m$ and $d_c \leq n$. Since there are n variable nodes and d_v represents the maximum number of edges for each node, the maximum number of edges possible based on the variable nodes is nd_v. Likewise, the maximum number of edges possible based on the check nodes is md_c. The actual maximum number of edges possible, $n_e^{\max}$, is thus given by the formula $n_e^{\max} = \min\{nd_v, md_c\}$.

Algorithm 1: Minimum edges pair generation

Result: Generate valid pair for min number of edges
Fix block length n;
Fix code rate r;
Fix max variable node degree d_v;
Fix max check node degree d_c;
$m := n(1-r)$;
$n_e^{\min} := 2n$;
$\lambda := (0, 2n, 0, \ldots, 0)^T$;
$\rho := (0, 2m, 0, \ldots, 0)^T$;
$k := 2m$;
while $k < 2n$ **do**
  $i :=$ idx of first non-zero elt. of ρ;
  Decrement ρ_i by i;
  Increment ρ_{i+1} by $i+1$;
  $k = k + 1$;
end
$\lambda = \lambda / n_e^{\min}$;
$\rho = \rho / n_e^{\min}$;
return (λ, ρ)

C. Step 3: Obtaining a Valid Pair for Each Number of Edges

For the third step, we explain an algorithm to find a valid degree distribution pair for every number of edges between the minimum and maximum number possible, beginning with

the minimum. First, fix a block length, a code rate, and maximum variable and check node degrees. Second, use step 2 to determine the minimum and maximum number of edges for these fixed values. Then, set $\lambda = (0, 2n, 0, \ldots, 0)^T$ and $\rho = (0, 2m, 0, \ldots, 0)^T$. Set $k = 2m$ and while $k < 2n$, do the following things. First, let i be the index of the first non-zero element of ρ. Second, decrement ρ_i by i. Third, increment ρ_{i+1} by $i+1$. Finally, increment k by 1. When $k = 2n$, divide both λ and ρ by $n_e^{\min}$ to obtain a valid degree distribution pair for the minimum possible number of edges. This algorithm is detailed in Algorithm 1.

To generate valid degree distribution pairs for the remaining possible number of edges, set $k = n_e^{\min}$ and while $k \leq n_e^{\max}$, follow the same steps described in the previous paragraph, but do them for both λ and ρ simultaneously. The degree distribution pair produced at each step k forms a valid degree distribution pair for k edges.

Algorithm 2: Valid pair generation

Result: Generate all valid degree distribution pairs
Run Algorithm 1;
Initialize dictionary Λ for variable node distributions;
Initialize dictionary R for check node distributions;
$n_e^{\max} := \min\{nd_v, md_c\}$;
$k := n_e^{\min}$;
while $k \leq n_e^{max}$ **do**
  Construct $\hat{H}_\lambda$ as in (12) for λ;
  Construct $\hat{H}_\rho$ as in (12) for ρ;
  Set $\Lambda[k] = \lambda + \hat{H}_\lambda$;
  Set $R[k] = \rho + \hat{H}_\rho$;
  $\lambda = k\lambda$;
  $\rho = k\rho$;
  $i :=$ idx of first non-zero elt. of λ;
  Decrement λ_i by i;
  Increment λ_{i+1} by $i+1$;
  $i :=$ idx of first non-zero elt. of ρ;
  Decrement ρ_i by i;
  Increment ρ_{i+1} by $i+1$;
  $k = k + 1$;
  $\lambda = \lambda/k$;
  $\rho = \rho/k$;
end
return (Λ, R);

To summarize this and the two preceding subsections, we have shown that it is possible to completely enumerate the entire space of valid degree distribution pairs and provided analysis leading to a rigorous algorithm for doing so. First, we fix all necessary variables and run Algorithm 1. Then, we generate one particular valid degree distribution pair for each number of edges between the minimum and maximum. Finally, we use the generated pairs to generate all valid degree distribution pairs for all possible number of edges. This algorithm is detailed in pseudocode in Algorithm 2. For a given number of edges, all valid variable node degree distributions for that number of edges and all valid check node degree distributions for that same number of edges can together form a valid pair. So, the algorithm returns two dictionaries of sets of valid variable and check node distributions, indexed by the number of edges.

D. Complexity Analysis

We now present a complexity analysis for the number of floating-point operations (FLOPs) required for Algorithm 1 and Algorithm 2. For Algorithm 1, 8 FLOPs are required prior to the **while** loop. The **while** loop runs $2n - 2m = 2n - 2(n(1-r)) = 2nr$ times and requires 6 FLOPs each time, for a total of $12nr$ FLOPs. Finally, the divisions at the end of the algorithm require d_v and d_c FLOPs, respectively. All together, Algorithm 1 requires $12nr + d_v + d_c + 8 = \mathcal{O}(nr)$ FLOPs.

A similar, though far more complex, analysis can be performed for Algorithm 2. Due to space considerations, we will only provide a summary of the analysis and present upper and lower bounds on the resulting complexity. The dominating contribution to the number of FLOPs required for the algorithm comes from the construction of $\hat{H}_\lambda$ and $\hat{H}_\rho$. It can be shown that for each k, the construction of these two sets requires

$$2k \log\left(\frac{d_v d_c}{9}\right) + (2d_v^2 - 5d_v - 7)\frac{(2k)^{d_v-3}(3!)}{d_v!} + (2d_c^2 - 5d_c - 7)\frac{(2k)^{d_c-3}(3!)}{d_c!} \tag{13}$$

FLOPs. Within the **while** loop, k ranges from $n_e^{\min}$ to $n_e^{\max}$, so a lower bound on the number of FLOPs in (13) can be found by replacing k with $n_e^{\min} = 2n$, and an upper bound can be found by replacing k with $n_e^{\max}$, which for simplicity we will assume is nd_v. As a lower bound, constructing $\hat{H}_\lambda$ and $\hat{H}_\rho$ requires

$$\mathcal{O}\left(d_v^2\frac{(4n)^{d_v-3}}{d_v!} + d_c^2\frac{(4n)^{d_c-3}}{d_c!}\right) \tag{14}$$

FLOPs, while as a upper bound the construction requires

$$\mathcal{O}\left(d_v^2\frac{(2nd_v)^{d_v-3}}{d_v!} + d_c^2\frac{(2nd_v)^{d_c-3}}{d_c!}\right) \tag{15}$$

FLOPs.

The **while** loop runs $n_e^{\max} - n_e^{\min} = nd_v - 2n = \mathcal{O}(nd_v)$ times, so the total number of FLOPs required for Algorithm 2 is bounded below by

$$\mathcal{O}\left(nd_v^3\frac{(4n)^{d_v-3}}{d_v!} + nd_v d_c^2\frac{(4n)^{d_c-3}}{d_c!}\right) \tag{16}$$

and bounded above by

$$\mathcal{O}\left(nd_v^3\frac{(2nd_v)^{d_v-3}}{d_v!} + nd_v d_c^2\frac{(2nd_v)^{d_c-3}}{d_c!}\right). \tag{17}$$

We note that these bounds are polynomial in the block length n, but exponential in the maximum variable and check node degrees, d_v, and d_c. As a result, the algorithm becomes computationally intractable if you allow large maximum degrees. Fortunately, the enumeration algorithms are useful primarily

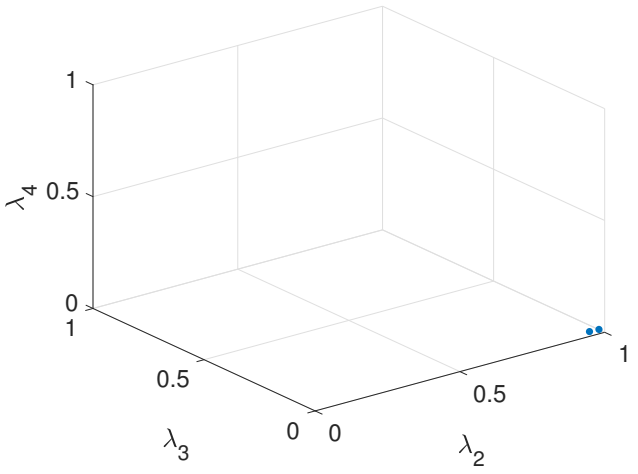

Fig. 2. A visualization of the valid space of variable node degree distributions for $n_e = 202$. Note there are only two valid distributions, located at the bottom right of the figure.

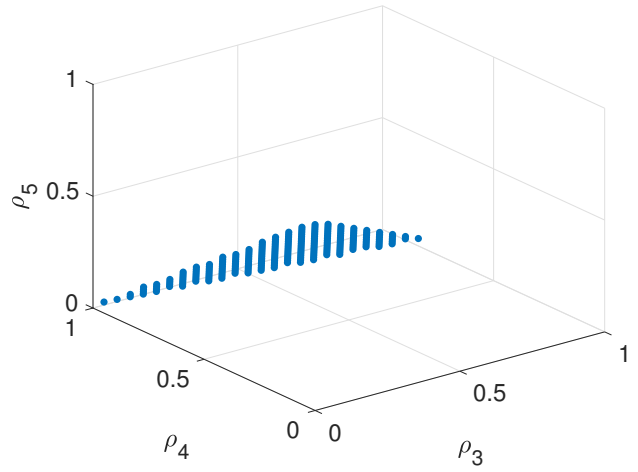

Fig. 3. A visualization of the valid space of check node degree distributions for $n_e = 202$. Note the set forms a subset of a plane in $\mathbb{R}^4$.

for small block lengths and the low density requirement for LDPC codes ensures reasonably small d_v and d_c parameters.

IV. Results

To demonstrate this algorithm, we provide an example. We choose exceptionally small values for block length and maximum variable and check node degrees to facilitate visualization of the discrete nature of the space. Fix the block length $n = 100$, the code rate $r = 1/2$, the maximum variable node degree $d_v = 4$, and the maximum check node degree $d_c = 5$. We run Algorithm 2 with these fixed values, and plot a visualization of all of the corresponding pairs for $n_e = 202$ edges.

The three dimensional plot in Fig. 2 represents the set of all valid variable node distributions for 202 edges. Since $\lambda_1 = 0$ and $d_v = 4$, there are only three possible degrees of freedom, plotted as the three axes in the plot. Algorithm 2 produces only two valid degree distributions for $n_e = 202$.

The plot shown in Fig. 3 represents the corresponding set of all valid check node distributions. Since $d_c = 5$ and $\rho_1 = 0$, there are four degrees of freedom, but we plot only three, ρ_3, ρ_4, and ρ_5. Based on the construction of $\hat{H}$ in (12), we would expect the degrees of freedom to be reduced by two, and therefore expect the valid space of check node distributions to be a discrete subset of a plane in $\mathbb{R}^4$. We see from the plot of three of the degrees of freedom that this is the case. We note that any combination of a variable node distribution from Fig. 2 and a check node distribution from Fig. 3 forms a valid degree distribution pair for $n_e = 202$.

The complete enumeration of degree distribution pairs presented in this paper allows for LDPC code optimization that is both flexible and accurate at small block lengths. Asymptotic optimization methods by nature are only capable of optimizing objective functions that can be evaluated theoretically for infinite block length codes. In contrast, the enumeration of all valid pairs combined with standard discrete optimization methods, such as those found in [19], allows for the optimization of any objective function that takes as an input a valid degree distribution pair. This opens the door for the optimization of LDPC codes with respect to objective functions that utilize simulation of real codes, among other things. In addition, since the search space for these methods will consist entirely of valid degree distribution pairs of finite block length, any optimal pair obtained from such optimization is guaranteed optimal for the finite case, rather than only for the asymptotic case.

To conclude this section, we present an example of an objective function that can be optimized using our enumeration method. Again, we fix the block length $n = 100$, rate $r = 1/2$, maximum variable node degree $d_v = 4$, and maximum check node degree $d_c = 5$. Our goal is to find the degree distribution pair with these constraints that produces the LDPC code family that minimizes the average bit-error rate (BER) through a simulated additive white Gaussian noise (AWGN) channel at a fixed $E_b/N_0 = 5$ dB. We note the generation of individual code realizations within a code family is handled independently of the enumeration algorithms presented in this paper. We define the objective function to accept a degree distribution pair, run the simulations, and output the average BER for that pair.

In Fig. 4 and Fig. 5, we visualize the value of this objective function for every possible degree distribution pair shown in Fig. 2 and Fig. 3. The plot in Fig. 4 is constructed by plotting the two degrees of freedom (h_4 and h_5) for the check node distributions against the objective function value for the left-most valid variable node distribution seen in Fig. 2, while Fig. 5 is constructed similarly, but for the right-most valid variable node distribution in Fig. 2. There is a clearly discernible pattern to the BER as a function of the degree distribution pair that can be exploited by discrete optimization methods to produce the optimal pair for this objective function. We note that for $n_e = 202$ like in the visualizations, the optimal pair is $(\lambda, \rho) = \left((0, \frac{196}{202}, \frac{6}{202}, 0)^T, (0, 0, 0, \frac{192}{202}, \frac{10}{202})^T\right)$, which

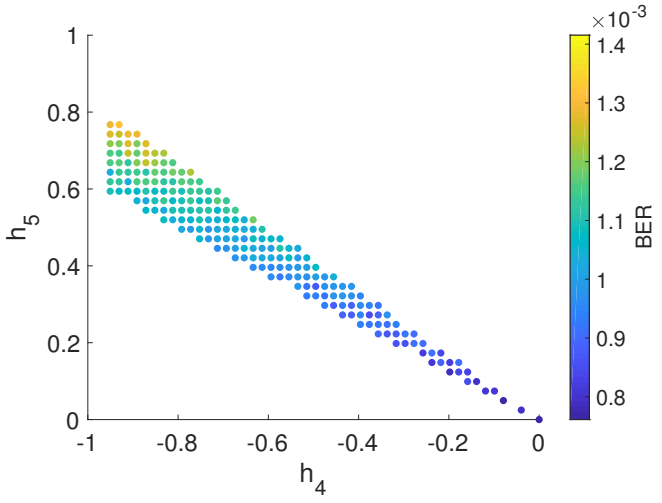

Fig. 4. A visualization of the continuous dependence of BER on location in the space of valid check node degree distributions for the left-most variable node degree distribution.

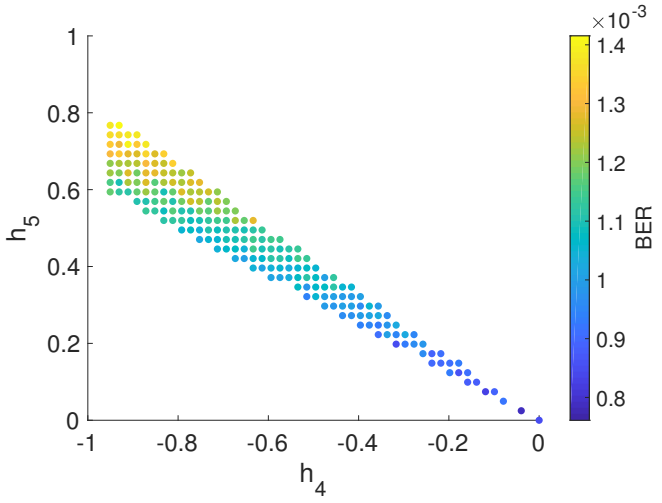

Fig. 5. A visualization of the continuous dependence of BER on location in the space of valid check node degree distributions for the right-most variable node degree distribution.

corresponds to $(h_4, h_5) = (0, 0)$ for the left-most variable node degree distribution. However, a true optimization would also need to account for degree distribution pairs with $n_e \neq 202$ when optimizing the objective function.

V. Conclusion

In this paper, we have described an algorithm for completely enumerating the valid space of degree distribution pairs for LDPC codes of finite block length. We have performed the necessary mathematical analysis to demonstrate the validity and complexity of our method and have run the algorithm on an example and visualized the results. Finally, we discussed the possibility of using this algorithm for better optimization of LDPC codes of small block lengths. In the future, we plan to utilize the enumeration algorithm to find the best degree distribution pairs satisfying a given objective function.

References

[1] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[2] D. J. C. MacKay and R. M. Neal, "Good codes based on very sparse matrices," in *Cryptography and Coding 5th IMA Conference*, ser. Lecture Notes in Computer Science, C. Boyd, Ed. Berlin, Germany: Springer, 1995, no. 1025, pp. 100–111.

[3] ——, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, Mar. 1997.

[4] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[5] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[6] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY: Cambridge University Press, 2008.

[7] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[8] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ: John Wiley & Sons, Inc., 2006.

[9] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, Feb 2001.

[10] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 670–678, 2004.

[11] M. Ivkovic, S. K. Chilappagari, and B. Vasic, "Eliminating trapping sets in low-density parity-check codes by using Tanner graph covers," *IEEE Transactions on Information Theory*, vol. 54, no. 8, pp. 3763–3768, 2008.

[12] G. Durisi, T. Koch, and P. Popovski, "Toward massive, ultrareliable, and low-latency wireless communication with short packets," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1711–1726, 2016.

[13] Sae-Young Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58–60, Feb 2001.

[14] S. Abu-Surra, D. Divsalar, and W. E. Ryan, "Enumerators for protograph-based ensembles of LDPC and generalized LDPC codes," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 858–886, 2011.

[15] X. Zheng, F. C. M. Lau, and C. K. Tse, "Constructing short-length irregular LDPC codes with low error floor," *IEEE Transactions on Communications*, vol. 58, no. 10, pp. 2823–2834, 2010.

[16] H. Park, S. Hong, J. No, and D. Shin, "Design of multiple-edge protographs for QC LDPC codes avoiding short inevitable cycles," *IEEE Transactions on Information Theory*, vol. 59, no. 7, pp. 4598–4614, 2013.

[17] T. V. Nguyen and A. Nosratinia, "Rate-compatible short-length protograph LDPC codes," *IEEE Communications Letters*, vol. 17, no. 5, pp. 948–951, 2013.

[18] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[19] C. Audet and W. Hare, *Derivative-Free and Blackbox Optimization*. Gewerbestrasse 11, 6330 Cham, Switzerland: Springer International Publishing AG, 2017.