Ensemble weight enumerators for protographs: a proof of Abu Surra's conjecture and a continuous relaxation for a faster enumeration

Tarik Benaddi*†‡, Charly Poulliat†‡,
Marie-Laure Boucheret †‡, Benjamin Gadat§ and Guy Lesthievent*
*CNES †University of Toulouse, ENSEEIHT/IRIT ‡TéSA - Toulouse §Thales Alenia Space

***Abstract*—In this paper, we provide a proof for the conjecture made by Abu Surra *et al.* [1] to simplify the computation of ensemble input output weight enumerators for protograph-based low density parity check (LDPC) codes. Furthermore, we propose a new method to compute more efficiently the ensemble weight enumerator. This approach can be applied particularly to lighten the computations for high rate codes, generalized LDPC codes or spatially coupled LDPC codes.**

I. Introduction

Low density parity check (LDPC) codes are known to achieve very good performance both in the waterfall and the error floor region. In order to predict the performance in the error floor region, one can evaluate the average codeword weight enumerators: good code ensembles are those which have a minimum distance that grows linearly with the code length. Inspired from concatenated schemes [2], the derivation of the ensemble weight enumerators for protograph-based LDPC codes are well explained [3], [4]. However, the simulations become quickly unfeasible when dealing with large code lengths. The bottleneck is the evaluation of the check node enumerators, especially for high rate codes (highly connected check nodes) and for generalized check nodes. In order to alleviate the computations, [1] proposed a conjecture based on the edge types. In this paper, we intend to provide a proof of the Abu Surra's conjecture. Additionally, we introduce a new method that computes more efficiently the enumerator of the check nodes. Based on a continuous relaxation optimization, this method gives very accurate results and reduces greatly the complexity even for very large code lengths or large degree check nodes.

This paper is organized as follows: after the introduction of the main notations and a review of the propograph code ensemble weight enumerators in Section II, we provide the proof of the Abu Surra's conjecture in Section III. In Section IV, we introduce our new method for evaluating the check node ensemble weight enumerators. Finally, some numerical results are presented in Section V.

II. Protograph ensemble weight enumerators

A. Notations

A *protograph* [5] is a relatively small bipartite graph described by the tuple (V, C, E). The set of variable nodes V (of cardinality n_v) is connected to the set of check nodes C (of cardinality n_c) through edges in E (the set of edges with cardinality $|E|$). The protograph is usually described by its protomatrix $\boldsymbol{B}$ where $B(j,i) \geq 0$ is the number of connections between the variable node (VN) v_i and the check node (CN) c_j.

Let q_{v_i} (q_{c_j}) be the degree of the VN v_i (resp. CN c_j). The corresponding protograph based LDPC code is constructed by "copy-and-permute" operations [5]: it consists of making a certain number of copies of the protograph, say N times, and merging them by permuting the endpoints of each edge copies. Figure 1 depicts the 'vectorized' protograph code associated with the following base matrix:

$$\boldsymbol{B} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

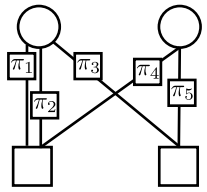

Fig. 1: The 'vectorized' protograph ldpc code generated by "copy-and-permute" operations. Each interleaver $\pi_i|_{1\leq i\leq 5}$ permutes the bundle of the N replicas of each edge.

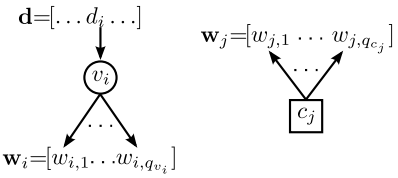

Fig. 2: Main notations considered for VNs and CNs

Let Ω_t (reps. Ω_p) denote the set of transmitted (resp. punctured) VNs of cardinality n_t (resp. n_p). The set operator $|.|$ returns the number of elements of an ensemble, the matrix operator $.^T$ denotes the transposition and the vector function $\|.\|_1$ (resp. $\|.\|_2$) is the L^1 (resp. L^2) norm.

The overall input weight vector is denoted by $\boldsymbol{d} = [d_i]_i$: *i.e.*, to each VN v_i, we associate its input weight d_i ($0 \leq d_i \leq N$) and output weight vector $\boldsymbol{w}_i = [w_{i,1}, \ldots, w_{i,q_{v_i}}]^T$. For CNs, we denote by $\boldsymbol{w}_j = [w_{j,1}, \ldots, w_{j,q_{c_j}}]^T$ the input weight vector of a CN c_j,and a fictitious edge output with weight 0. Observe that $\boldsymbol{w}_j$ is directly deduced from $\boldsymbol{d}$ and the graph connectivity, *i.e.* $w_{i,k} = w_{j,l}$ if $B(l,k) \neq 0$. Also, note that each component of the vector $\boldsymbol{w}_j$ is equal to a component d_i of the vector $\boldsymbol{d}$ if v_i is a neighbor of c_j for some i.

Figure 2 summarizes the main notations. In the following, for ease of notations, we omit the subscripts i and j as long as it is clear from the context that we refer to an arbitrary VN or CN.

B. Weight enumerators

To derive the expression of the ensemble weight enumerator of protograph-based LDPC codes, we consider that the size-N interleavers $\{\pi_e\}_{1\leq e\leq |E|}$ are uniform. Hence, based on the result for serially concatenated codes [2], the average

weight enumerator can be obtained when we reinterpret the protograph LDPC code as a serial concatenation of VNs constituent codes and CNs constituent codes. It is shown in [1] that the average number of codewords of weight-vector $\boldsymbol{d} = [d_i]_i$ is written as:

$$A(\boldsymbol{d}) = \frac{\prod_{j=1}^{n_c} A^{c_j}(\boldsymbol{w}_j)}{\prod_{i=1}^{n_v} \binom{N}{d_i}^{q_{v_i}-1}} \quad (1)$$

where $A^{c_j}(\boldsymbol{w}_j)$ denotes the input weight-vector-$\boldsymbol{w}_j$ enumerator for the c_j constituent CN code. Writing Eq. (1) in the log-domain gives:

$$\log A(\boldsymbol{d}) = \sum_{j=1}^{n_c} \log A^{c_j}(\boldsymbol{w}_j) - \sum_{i=1}^{n_v} (q_{v_i}-1) \log \binom{N}{d_i} \quad (2)$$

For sufficiently large N, using Stirling's approximation and denoting $\delta_i = d_i/N$, we get:

$$\log \binom{N}{d_i} \approx N.H(\delta_i) \quad (3)$$

where $H(.)$ is the binary entropy function.

In order to derive the expression of $a^{c_j}(\boldsymbol{w}_j)$, [6] derived the explicit expression of $a^{c_j}(\boldsymbol{w}_j)$ when $q_{c_j} = 3$. For higher degree CNs, one can perform a check splitting operation to write $a^{c_j}(\boldsymbol{w}_j)$ as the maximization over $q_{c_j} - 3$ variables of the sum of $q_{c_j} - 2$ degree-3 constituent CNs code weight enumerators [6, Eq. 14]. For highly connected CNs or for generalized CNs, this method introduces many dummy VNs and the maximization becomes complex. Therefore, [1] proposed a more efficient method for generalized CNs.

Consider a degree-q_c constituent CN code c. We want to compute $A^c(\boldsymbol{w})$ where $\boldsymbol{w} = \{w_1, ..., w_{q_c}\}^T$. We have:

$$A^c(\boldsymbol{w}) = \sum_{\{\mathscr{C}\}} C(N; \mathscr{C}) \quad (4)$$

where $\{\mathscr{C}\}$ is the family of sets $\mathscr{C}$, each contains N valid codewords of the CN constituent code c, such that the i^{th} component, $\forall i \in [\![1, q_c]\!]$, of all these codewords constitute a partition of the input weight w_i. $C(N; \mathscr{C})$ is the number of distinct permutations of the elements in $\mathscr{C}$.

Let $\boldsymbol{M}^c$ be the $q_c \times K$ matrix formed by the codewords of the CN c, denoted $\{C_1, \ldots, C_K\}$, as its columns ($K = 2^{q_c-1}$). Equation (4) can be rewritten as:

$$A^c(\boldsymbol{w}) = \sum_{\{\boldsymbol{n}\}} C(N; \boldsymbol{n}) \quad (5)$$

where $\{\boldsymbol{n}\} = \{(n_1, \ldots, n_K)\}^T$ is the set of solutions of $\boldsymbol{w} = \boldsymbol{M}^c.\boldsymbol{n}$ such that $n_i \geq 0$ and $\sum_{i=1}^{K} n_i = N$. $C(N; \boldsymbol{n})$ is the multinomial coefficient [7].

Rewriting $A^c(\boldsymbol{w})$ in the logarithm domain gives:

$$\begin{aligned} \log\left(A^c(\boldsymbol{w})\right) &= \log\left(\sum_{\{\boldsymbol{n}\}} C(N; \boldsymbol{n})\right) \\ &= \max_{\{\boldsymbol{n}\}}{}^{*}\left(\log C(N; \boldsymbol{n})\right) \end{aligned} \quad (6)$$

where the pairwise $\max^*$ operator is defined as $\max^*(x, y) = \max(x, y) + \log(1 + e^{-|x-y|})$.

As the number of combinations $C(N; \boldsymbol{n})$ can be very large, Eq. (6) can be approximated by:

$$\log\left(A^c(\boldsymbol{w})\right) = \max_{\{\boldsymbol{n}\}}\left(\log C(N; \boldsymbol{n})\right) \quad (7)$$

inducing an error of at most $\max_{x,y} \log(1 + e^{-|x-y|}) = \log 2$.

Lemma 1: For sufficiently large N:

$$\log C(N; \boldsymbol{n}) \approx N.H\left(\frac{n_1}{N}, \ldots, \frac{n_K}{N}\right)$$

where $H(.)$ is the multivariate entropy function.

Inserting Lemma 1 in Eq. (7) gives:

$$\log\left(A^c(\boldsymbol{w})\right) = N \max_{\{\boldsymbol{n}\}}\left(H\left(\frac{n_1}{N}, \ldots, \frac{n_K}{N}\right)\right)$$

Thus, the generic enumerator of the constituent CN code c with the N-normalized input weight vector $\boldsymbol{\delta} = \frac{\boldsymbol{w}}{N}$ can be written as

$$a^c(\boldsymbol{\delta}) = \limsup_{N\to\infty} \frac{\log\left(A^c(\boldsymbol{w})\right)}{N} \approx \max_{\{\boldsymbol{p}\}}\left(H(p_1, \ldots, p_K)\right) \quad (8)$$

under the constraints $\|\boldsymbol{p}\|_1 = 1$ and $\boldsymbol{\delta} = \boldsymbol{M}^c.\boldsymbol{p}$, where $\boldsymbol{p} = \{p_i\}_i$ and $p_i = \frac{n_i}{N}$.

III. A PROOF OF ABU SURRA'S CONJECTURE

The average number of the protograph's codewords of weight d is given by

$$A_d = \sum_{\{d_i/v_i \in \Omega_t\}} \sum_{\{d_k/v_k \in \Omega_p\}} A(\boldsymbol{d})$$

where $\sum_{d_i/v_i\in\Omega_t} d_i = d$. Let n be the number of transmitted VNs in the code (*i.e.* $n_t.N$). The n-normalized logarithmic asymptotic weight enumerator can be then written as [1]:

$$r\left(\frac{d}{n}\right) = \frac{1}{n_t}\tilde{r}(\delta)$$

where $\delta = d/N$ and

$$\begin{aligned} \tilde{r}(\delta) &= \limsup_{N\to+\infty} \frac{A_d}{N} \\ &= \max_{\delta_t : v_t \in \Omega_t} \max_{\delta_p : v_p \in \Omega_p} \left\{ \sum_{j=1}^{n_c} a_j^c(\boldsymbol{\delta}_j) - \sum_{i=1}^{n_v} (q_{v_i}-1) H(\delta_i) \right\} \end{aligned} \quad (9)$$

The computation complexity of $\tilde{r}(\delta)$ is limited by the complexity of spanning all partitions of size n_t of the weight d, combined with all possible input weights of the punctured VNs. To alleviate this difficulty, [1] proposed the following steps: the protograph's VNs are partitioned into subsets according to their types. Two VNs are of the same type when their neighborhoods are identical, in other words, if one switches the input weights between the VNs of the same type, the resulting input weight vector forms a valid codeword and no difference is noticed from all CNs point of view. By labeling each edge of the protograph by its adjacent VN type, each CN's adjacent edge can be labeled with its weight and its type. The *subset-weight vector* (SWV) of a CN codeword is defined as the vector whose entries are the weights of bit subsets of the CN codeword.

[1] conjectured that *"in the maximization of Eq.* (8)*, the optimal point occurs when codewords of equal SWV have the same proportion of occurence"*. This is equivalent to say that the maximum is reached when edges that belongs to the same type carry the same weight. In general, this is not always possible since the total weight carried by the edges of the same type is not necessarily a multiple of the number of edges of this type. In the following we will prove the generalized form of this conjecture:

Theorem 1: The maximization of Eq. (8) occurs when the weights number of occurrences $\{p_i\}_i$ of the check node codewords, that belongs to the same VNs type composition are as uniform as possible.

A. Proof when CN has 3 edges of the same type

Without loss of generality, let us consider a degree-q_c CN where 3 edges, $\alpha < \beta < \gamma$ are of the same type. The corresponding N-normalized input weight vector is $\boldsymbol{\delta} = (\delta_1 \dots \delta_\alpha \dots \delta_\beta \dots \delta_\gamma \dots \delta_{q_c})^T$ such that $\delta_\alpha, \delta_\beta$ and δ_γ are not all equal.

A triplet (f_1, f_2, f_3), of total weight $s = Nf_1 + Nf_2 + Nf_3$ is called as *as-uniform-as-possible* (AUAP) if a permutation of (Nf_1, Nf_2, Nf_3) has the form:

$$\begin{cases} (k,k,k) \text{ if } s = 3k \\ (k,k,k+1) \text{ if } s = 3k+1 \\ (k+1,k+1,k) \text{ if } s = 3k+2 \end{cases} \tag{10}$$

N.B.: In the asymptotic case, *i.e.* as N goes to infinity, the weights will be equal (dealing with normalized weights k/N asymptotically equals $(k+1)/N$, when N goes to infinity), *as-uniform-as-possible* could then be substituted simply with *uniform*.

Since $\boldsymbol{M}^c$ is not necessarily unimodular [8], $\boldsymbol{M}^c.\boldsymbol{p} = \boldsymbol{\delta}$ does exist only for some $\boldsymbol{\delta}$.

Definition 1: $\boldsymbol{\delta}$ is called *admissible* if $\boldsymbol{M}^c.\boldsymbol{p} = \boldsymbol{\delta}$ admits a solution .

Theorem 2: If $\boldsymbol{\delta}$ is admissible, then the related AUAP version $\boldsymbol{\delta}^*$ is also admissible

Proof: By circular permutation of the weights $\delta_\alpha, \delta_\beta$ and δ_γ, we form the following vectors:

$$\begin{aligned} \boldsymbol{\delta}_1^T &= (\delta_1 \dots \delta_\alpha \dots \delta_\beta \dots \delta_\gamma \dots \delta_{q_c}) \\ \boldsymbol{\delta}_2^T &= (\delta_1 \dots \delta_\gamma \dots \delta_\alpha \dots \delta_\beta \dots \delta_{q_c}) \\ \boldsymbol{\delta}_3^T &= (\delta_1 \dots \delta_\beta \dots \delta_\gamma \dots \delta_\alpha \dots \delta_{q_c}) \end{aligned}$$

Because of the symmetry of VNs of the same type, it is clear that $\boldsymbol{\delta}_1^T, \boldsymbol{\delta}_2^T$ and $\boldsymbol{\delta}_3^T$ are also valid CN input weight-vector (see introduction of Section III). Let $\boldsymbol{p}_1, \boldsymbol{p}_2$ and $\boldsymbol{p}_3$ be solutions of the equality contraint of Eq. (8) such that :

$$\boldsymbol{M}^c.\boldsymbol{p}_1 = \boldsymbol{\delta}_1 \tag{11}$$
$$\boldsymbol{M}^c.\boldsymbol{p}_2 = \boldsymbol{\delta}_2 \tag{12}$$
$$\boldsymbol{M}^c.\boldsymbol{p}_3 = \boldsymbol{\delta}_3 \tag{13}$$
$$\|\boldsymbol{p}_1\|_1 = \|\boldsymbol{p}_2\|_1 = \|\boldsymbol{p}_3\|_1 = 1$$

Let us find three variables a, b and c such that $a\boldsymbol{\delta}_1 + b\boldsymbol{\delta}_2 + c\boldsymbol{\delta}_3$ is AUAP wih respect to $\delta_\alpha, \delta_\beta$ and δ_γ. To this end, we have to solve the system:

$$\underbrace{\begin{pmatrix} \delta_\alpha & \delta_\gamma & \delta_\beta \\ \delta_\beta & \delta_\alpha & \delta_\gamma \\ \delta_\gamma & \delta_\beta & \delta_\alpha \end{pmatrix}}_{\boldsymbol{\Delta}} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \tag{14}$$

Lemma 2: Δ is invertible $\forall(\delta_\alpha, \delta_\beta, \delta_\gamma)$

Proof: The determinant of $\boldsymbol{\Delta}$ is:

$$\begin{aligned} |\boldsymbol{\Delta}| &= \frac{1}{2}(\delta_\alpha + \delta_\beta + \delta_\gamma)\left((\delta_\alpha - \delta_\beta)^2 + (\delta_\alpha - \delta_\gamma)^2 + (\delta_\beta - \delta_\gamma)^2\right) \\ &\neq 0 \qquad (\delta_\alpha, \delta_\beta \text{ and } \delta_\gamma \text{ are not all null by definition}) \end{aligned}$$

■

The determinant of Eq. (14) is not null, it exists then a unique solution (a, b, c).

Lemma 3: $(a, b, c) \in [0,1]^3$ and $a + b + c = 1$

Proof: • By summing the three equations of the system Eq. (14), we obtain:

$$(\delta_\alpha + \delta_\beta + \delta_\gamma)(a + b + c) = f_1 + f_2 + f_3$$

Since $f_1 + f_2 + f_3 = \delta_\alpha + \delta_\beta + \delta_\gamma \neq 0$, then $a + b + c = 1$.

• Since $|\boldsymbol{\Delta}| > 0$, by Cramer's rule we have $a = |\boldsymbol{\Delta}_{\boldsymbol{a}}|/|\boldsymbol{\Delta}|$, where $\boldsymbol{\Delta}_{\boldsymbol{a}}$ is formed by replacing the first column of $\boldsymbol{\Delta}$ by $(f_1, f_2, f_3)^T$. By studying the sign of $|\boldsymbol{\Delta}_{\boldsymbol{a}}|$ in the cases depicted in Eq. (10), one can show that $a > 0$. Similar result can be shown for b and c.

So far, we have shown that a, b and c are positive and $a + b + c = 1$, which lead to $(a, b, c) \in [0,1]^3$. ■

Summing Eqs. (11) to (13) weighted by (a, b, c) gives:

$$\begin{aligned} \boldsymbol{M}^c.(a\boldsymbol{p}_1 + b\boldsymbol{p}_2 + c\boldsymbol{p}_3) &= a\boldsymbol{\delta}_1 + b\boldsymbol{\delta}_2 + c\boldsymbol{\delta}_3 \\ &= (\ \delta_1 \dots f_1 \dots f_2 \dots f_3 \dots \delta_{q_c}\ )^T \\ &\triangleq \boldsymbol{\delta}^* \end{aligned} \tag{15}$$

From Lemma 3, we get $\boldsymbol{p}^* \triangleq a\boldsymbol{p}_1 + b\boldsymbol{p}_2 + c\boldsymbol{p}_3 \in [0,1]^K$ and by construction:

$$\sum_{i:v_i \in \Omega_t} \boldsymbol{\delta}^*(i) = \sum_{i:v_i \in \Omega_t} \boldsymbol{\delta}_1(i) = \sum_{i:v_i \in \Omega_t} \boldsymbol{\delta}_2(i) = \sum_{i:v_i \in \Omega_t} \boldsymbol{\delta}_3(i) = \delta$$

i.e $\boldsymbol{\delta}^*$ is within the same search space in Eq. (9) as $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$ and $\boldsymbol{\delta}_3$. ■

Theorem 3: $H(\boldsymbol{p}_1) = H(\boldsymbol{p}_2) = H(\boldsymbol{p}_3)$ and $H(\boldsymbol{p}^*) \geq H(\boldsymbol{p}_1)$

Proof: Recall $\boldsymbol{M}^c = (C_1, \dots C_K)$ where $\{C_i\}_i$ are all CN codewords. Let L_j be the j^{th} row of the q_c rows of $\boldsymbol{M}^c$. Substituting the rows of Eq. (12) gives:

$$\boldsymbol{M}^c.\boldsymbol{p}_2 = \boldsymbol{\delta}_2 \Rightarrow \begin{pmatrix} L_1 \\ \vdots \\ L_\alpha \\ \vdots \\ L_\beta \\ \vdots \\ L_\gamma \\ \vdots \\ L_{q_c} \end{pmatrix} .\boldsymbol{p}_2 = \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_\gamma \\ \vdots \\ \delta_\alpha \\ \vdots \\ \delta_\beta \\ \vdots \\ \delta_{q_c} \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} L_1 \\ \vdots \\ L_\beta \\ \vdots \\ L_\gamma \\ \vdots \\ L_\alpha \\ \vdots \\ L_{q_c} \end{pmatrix} .\boldsymbol{p}_2 = \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_\alpha \\ \vdots \\ \delta_\beta \\ \vdots \\ \delta_\gamma \\ \vdots \\ \delta_{q_c} \end{pmatrix} \tag{16}$$

Concerning single parity check codes, notice that the necessary and sufficient condition of any CN codeword C_i is that it must have an even number of $'1's$. Permuting the rows

of $\boldsymbol{M}^c$ clearly preserves the number of $'1's$ in each column (for general parity check nodes, particular attention must be given to the form of the matrix $\boldsymbol{M}^c$ and to the potential edges of the same type). Hence, the most left hand matrix in Eq. (16) is just a column-wise permuted version of $\boldsymbol{M}^c$. It exists then a permutation matrix Π_2 such that:

$$(\ L_1 \dots L_\beta \dots L_\gamma \dots L_\alpha \dots L_{q_c}\ )^T = \boldsymbol{M}^c.\Pi_2$$

Consequently, Eq. (16) becomes:

$$\boldsymbol{M}^c.\Pi_2.\boldsymbol{p}_2 = \boldsymbol{\delta}_1$$

By definition of $\boldsymbol{p}_1$ and the symmetry of the entropy function:

$$H(\boldsymbol{p}_1) \geq H(\Pi_2 \boldsymbol{p}_2) = H(\boldsymbol{p}_2)$$

Similarly, we can show that $H(\boldsymbol{p}_2) \geq H(\boldsymbol{p}_1)$. This leads to $H(\boldsymbol{p}_1) = H(\boldsymbol{p}_2)$. Same conclusion can be drawn for $\boldsymbol{p}_3$. Consequently:

$$H(\boldsymbol{p}_1) = H(\boldsymbol{p}_2) = H(\boldsymbol{p}_3)$$

Using Lemma 3 and the concavity of H, Jensen's inequality gives:

$$\begin{aligned} H(\boldsymbol{p}^*) &= H(a\boldsymbol{p}_1 + b\boldsymbol{p}_2 + c\boldsymbol{p}_3) \\ &\geq aH(\boldsymbol{p}_1) + bH(\boldsymbol{p}_2) + cH(\boldsymbol{p}_3) \\ &\geq (a+b+c)H(\boldsymbol{p}_1) \\ &\geq H(\boldsymbol{p}_1) \end{aligned}$$

■

To summarize, for a fixed δ, from an admissible input-weight vector $\boldsymbol{\delta}$, we built an admissible input-weigth vector $\boldsymbol{\delta}^*$ which is AUAP such that $a^c(\boldsymbol{\delta}^*) \geq a^c(\boldsymbol{\delta})$. Therefore, solutions of the maximization in Eq. (9) belongs to the set of AUAP $\boldsymbol{\delta}$ weight-vectors. Theorem 1 and the conjecture in [1] are hence proved. Q.E.D.

N.B.:A similar proof, with great simplifications, can be made for CN with only 2 edges of the same type.

B. CN has more than three edges of the same type

As in the previous section, let us consider the normalized CN input-weight vector $\boldsymbol{\delta} = (\delta_i)_i^T$. In this case, the corresponding vector $\boldsymbol{\delta}^* = (\zeta_1, ...\zeta_n)$ is said to be AUAP when the weights of all the edges that belong to the same type, *i.e.* a subset of $\{\zeta_i\}_i$, are as uniformly distributed as possible.

Consider first the particular case when the CN has 4 edges of the same type. $\boldsymbol{\Delta}$ in Eq. (14) is a size-4 right-hand circulant matrix. Consider its *associated polynomial* [9]:

$$f(x) = \delta_\alpha + \delta_\beta x + \delta_\gamma x^2 + \delta_\kappa x^3$$

It follows that the 4 eigenvalues of $\boldsymbol{\Delta}$ are given by [9]:

$$\lambda_1 = f(1),\ \lambda_2 = f(z),\ \lambda_3 = f(z^2),\ \lambda_4 = f(z^3)$$

where $z = exp(\frac{2\pi i}{4})$ is the 4-th root of unity and i the imaginary unit. We have:

$$\lambda_2 = f(z) = (\delta_\alpha - \delta_\gamma) + i(\delta_\beta - \delta_\kappa)$$

If $\delta_\alpha = \delta_\gamma$ and $\delta_\beta = \delta_\kappa$, then $\lambda_2 = 0$, therefore, $\boldsymbol{\Delta}$ is not invertible. Consequently the previous proof cannot be applied. A strategy to generalize our approach when more than 3 edges of the same type exist is as follows:

1) Pick the first leftmost edge, say edge i, whose weight δ_i is different from its AUAP value ζ_i.
2) Select two adequate edges j and k within the same type such that: $\delta_i + \delta_j + \delta_k \geq \zeta_i$.
3) Give to δ_i its final value ζ_i by putting $f_1 = \zeta_i$.
4) Form the new vector $\boldsymbol{\delta}$, where $\delta_i = \zeta_i$, and its corresponding $\boldsymbol{p}$.
5) Go back to step 1 if $\boldsymbol{\delta} \neq \boldsymbol{\delta}^*$.

At the end, we obtain the configuration $\boldsymbol{\delta}^*$ and its corresponding $\boldsymbol{p}^*$ that maximizes H.

IV. Efficient computation of $\tilde{r}(\delta)$ through countinuous relaxation

In order to solve Eq. (8), [7] enumerates all the solutions of the equality $\boldsymbol{M}^c.\boldsymbol{p} = \boldsymbol{\delta}$ where $N.p_i \in \mathbb{N}$. Even if some reduction of the search space can be made (cf. [7, Appendix A]), Theorem 1 does not simplify necessarily the computation of Eq. (8) for protographs where not much VNs can be gathered as belonging to the same type. If besides that, we have a generalized or a highly connected CN, the computation of Eq. (8) becomes very complex. Moreover, for convolutional LDPC codes [10], this simplification cannot be applied since the VNs belongs to different time instants [11].

A. Continuous relaxation

The constituent CN c weight vector enumerator in Eq. (8) is computed by solving the following optimization:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & H(\boldsymbol{x}) \\ \text{subject to:} \quad & \boldsymbol{M}^c.\boldsymbol{x} = \boldsymbol{\delta} \\ & \sum_{i=0}^{K} x_i = 1 \\ & x_i \in \{\frac{k}{N} / k \in [\![0, N]\!]\}, \quad \forall i \in [\![1, K]\!] \end{aligned}$$

By putting together the equality constraints, which defines $\boldsymbol{M'^c}.\boldsymbol{x} = \boldsymbol{b}$, and noting $\mathscr{L} = \{\frac{k}{N} / k \in [\![0, N]\!]\}$, we get the following discrete optimization programming:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & H(\boldsymbol{x}) \\ \text{subject to:} \quad & \boldsymbol{M'^c}.\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \in \mathscr{L}^K \end{aligned} \tag{17}$$

Eq. (17) is a concave but nonlinear discrete optimization with equality constraints. To our knowledge, no method is given to solve efficiently this kind of problems. Consequently, we have to enumerate all possible solutions in the search space, which grows in exponential time with the CN degree q_c and polynomial time with the lifting factor N.

By relaxing Eq. (17), let us consider the following concave nonlinear continuous programming:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & H(\boldsymbol{y}) \\ \text{subject to:} \quad & \boldsymbol{M'^c}.\boldsymbol{y} = \boldsymbol{b} \\ & \boldsymbol{y} \in [0, 1]^K \end{aligned} \tag{18}$$

Eq. (18) can be solved easily and efficiently with different well-known optimization methods [12], even for large values of K. Moreover, the solution, when it exists, is optimal.

Theorem 4: Given a solution of Eq. (18), one can construct a solution in $\mathscr{L}^K$ of Eq. (17) given any arbitrary small penalty ε on the maximum value of

$H(\boldsymbol{x})$ and the constraint equality in Eq. (17).

Proof: It is easy to show that $\mathscr{L}^K$ is dense in $[0,1]^K$, in other words, for any element $\boldsymbol{y}$ in $[0,1]^K$, we can form a sequence with elements $\boldsymbol{x}_N$ in $\mathscr{L}^K$ such that $\boldsymbol{x}_N \xrightarrow[N\to\infty]{\|.\|_2} \boldsymbol{y}$. As an example, one can take $\boldsymbol{x}_N = (\frac{\lfloor Ny_i \rfloor}{N})_{1\leq i\leq K}$.

Let $\boldsymbol{y}^{opt} = (y_i^{opt})_{1\leq i\leq K}$ be the solution of Eq. (18) and $\boldsymbol{x}_N^{opt}$ the sequence of elements in $\mathscr{L}^K$ that tends to $\boldsymbol{y}^{opt}$. Using the formal definition of the limit and the continuity of the entropy function H, we have, $\forall \varepsilon \geq 0$:

$$\begin{aligned}
&\exists N_1 \in \mathbb{N}, \forall N \geq N_1, \|\boldsymbol{x}_N^{opt} - \boldsymbol{y}^{opt}\|_2 \leq \varepsilon \\
&\exists N_2 \in \mathbb{N}, \forall N \geq N_2, |H(\boldsymbol{x}_N^{opt}) - H(\boldsymbol{y}^{opt})| \leq \varepsilon \\
&\exists N_3 \in \mathbb{N}, \forall N \geq N_3, \|\boldsymbol{M'}^c.\boldsymbol{x}_N^{opt} - \boldsymbol{b}\|_2 \leq \varepsilon
\end{aligned}$$

Taking $N_0 = \max(N_1, N_2, N_3)$, we get:

$$\forall \varepsilon \geq 0, \exists N_0 \in \mathbb{N}, \forall N \geq N_0, \quad \begin{cases} \|\boldsymbol{x}_N^{opt} - \boldsymbol{y}^{opt}\|_2 \leq \varepsilon \\ |H(\boldsymbol{x}_N^{opt}) - H(\boldsymbol{y}^{opt})| \leq \varepsilon \\ \|\boldsymbol{M'}^c.\boldsymbol{x}_N^{opt} - \boldsymbol{b}\|_2 \leq \varepsilon \end{cases}$$

Consequently, $\boldsymbol{x}_N^{opt} \in \mathscr{L}^K$ is a solution to the optimization in Eq. (17) that satisfies both the maximization of H and the equality constraints for an arbitrarily small correction term ε as $N \to \infty$. ∎

B. On numerical implementations

To solve Eq. (18), we can apply different iterative optimization methods such as internal point (IP) or sequential quadratic programming (SQP) [12]. For this king of algorithms, we need to provide an initial guess, denoted $\boldsymbol{y}_0$, that satisfies the constraints (here $\boldsymbol{M'}^c.\boldsymbol{y} = \boldsymbol{b}$), and is within the feasible region (here, $[0,1]^K$).

First, let us put aside the feasible region and try to find at least one solution in the following affine space:

$$\boldsymbol{M'}^c.\boldsymbol{y} = \boldsymbol{b}$$

One can compute the Moore−Penrose pseudoinverse [13] (not necessarily unique) of $\boldsymbol{M'}^c$, denoted $\boldsymbol{M'}^{c+}$. Hence, $\boldsymbol{y}_0 = \boldsymbol{M'}^{c+}\boldsymbol{b}$.

In Eq. (17), since $\boldsymbol{M'}^c$ is not necessarily unimodular [14], $\boldsymbol{M'}^c.\boldsymbol{x} = \boldsymbol{b}$ does not have a solution $\forall \boldsymbol{b}$, *i.e.* for any value of the CN input-weight vector $\boldsymbol{w}$. When it is not possible, it means that $\mathscr{C}$ is an empty set and no codeword of the protograph with such Hamming weight exists.

When the system is solvable, Moore−Penrose pseudoinverse does not provide necessarily a solution that remains within the domain of the multivariate entropy function H, (some entries of $\boldsymbol{y}_0$ may be < 0 or > 1). We can solve this problem by considering an adequate analytic continuation of H. Since H is concave, it has one global maximum, then substituting in Eq. (18) $H(.)$ with any analytic continuation $H^*(.)$ which does not induce local maxima in the regions $]-\infty,\ 0]$ and $[1,\ +\infty[$ is transparent for the optimization routines.

V. Numerical results

Figure 3 gives the asymptotic codeword weight enumerator for different regular LDPC codes and their corresponding d_{min}. The numerical results are obtained using $N = 10^5$.

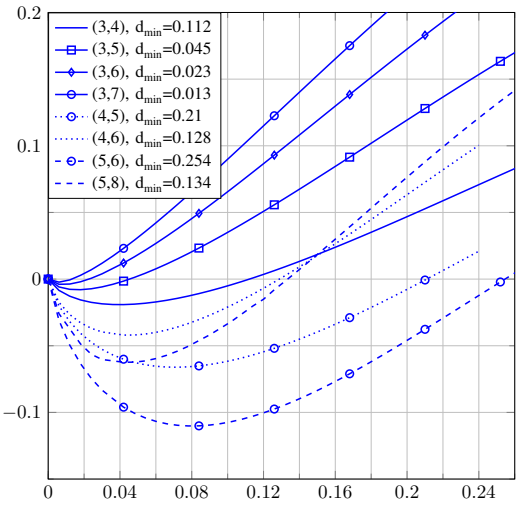

Fig. 3: Asymptotic weight enumerators for different protograph codes

VI. Conclusion

In this paper, we provided a proof of Abu Surra's conjecture used to lighten the computation of ensemble weight enumerators of protograph-based LDPC codes. Moreover, we proposed a new method based on a continuous relaxation to compute more efficiently the ensemble weight enumerator especially for CNs with large $\boldsymbol{M'}^c$ such as highly connected CNs of generalized CNs. The results of this paper also might be usefull for enumeration of other non-codeword objects in LDPC codes such as trapping sets, stopping sets and pseudocodewords.

References

[1] S. Abu-Surra, D. Divsalar, and W. E. Ryan, "Enumerators for protograph-based ensembles of ldpc and generalized ldpc codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 858–886, 2011.

[2] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 909–926, 1998.

[3] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for" turbo-like" codes," in *Proceedings of the annual Allerton Conference on Communication control and Computing*, vol. 36. UNIVERSITY OF ILLINOIS, 1998, pp. 201–210.

[4] S. Fogal, R. McEliece, and J. Thorpe, "Enumerators for protograph ensembles of ldpc codes," in *International Symposium on Information Theory, 2005. ISIT 2005.* IEEE, 2005, pp. 2156–2160.

[5] J. Thorpe, "Low-density parity-checks codes (ldpc) constructed from protographs," *IPN Progress Report*, pp. 42–154, 2003.

[6] D. Divsalar, "Ensemble weight enumerators for protograph ldpc codes," in *IEEE International Symposium on Information Theory, 2006*, 2006, pp. 1554–1558.

[7] S. A. Abu-Surra, "Protograph-based generalized ldpc codes: Enumerators, design, and applications," Ph.D. dissertation, The University of Arizona, 2009.

[8] D. S. Hochbaum and A. Pathria, "Can a system of linear diophantine equations be solved in strongly polynomial time?" *Citeseer*, 1994.

[9] P. J. Davis, *Circulant matrices*. American Mathematical Soc., 1979.

[10] D. G. Mitchell, M. Lentmaier, and D. J. Costello Jr, "Spatially coupled ldpc codes constructed from protographs," *arXiv preprint arXiv:1407.5366*, 2014.

[11] D. G. Mitchell, M. Lentmaier, and D. J. Costello, "On the minimum distance of generalized spatially coupled ldpc codes," in *IEEE International Symposium on Information Theory Proceedings (ISIT), 2013*. IEEE, 2013, pp. 1874–1878.

[12] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[13] R. Penrose, "A generalized inverse for matrices," in *Mathematical proceedings of the Cambridge philosophical society*, vol. 51, no. 03. Cambridge Univ Press, 1955, pp. 406–413.

[14] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[15] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

APPENDIX A
PROOF OF LEMMA 1

[1] proved Lemma 1 using the method of types [15, Thm. 12.1.3]. An alternative proof is as follows: For sufficiently large N and even if some particular n_i are not large we have:

$$\begin{aligned}
C(N;\boldsymbol{n}) &= C(N;n_1,\ldots,n_K)\\
&= \frac{N!}{n_1!n_2!\ldots n_K!}\\
&\approx \frac{\left(\frac{N}{e}\right)^N}{\left(\frac{n_1}{e}\right)^{n_1}\ldots\left(\frac{n_K}{e}\right)^{n_K}} \quad \text{(Striling: } n!\approx(\tfrac{n}{e})^n\ \text{)}\\
&= \frac{\left(\frac{N}{e}\right)^{n_1}\ldots\left(\frac{N}{e}\right)^{n_K}}{\left(\frac{n_1}{e}\right)^{n_1}\ldots\left(\frac{n_K}{e}\right)^{n_K}} \quad (\textstyle\sum_{i=1}^K n_i = N)\\
&= \prod_{i=1}^K \left(\frac{N}{n_i}\right)^{n_i}\\
&= \exp\left(\sum_{i=1}^K n_i \log\left(\frac{N}{n_i}\right)\right)\\
&= \exp\left(N.H(\frac{n_1}{N},\ldots,\frac{n_K}{N})\right)
\end{aligned}$$

APPENDIX B
PROOF OF LEMMA 3

$$\begin{aligned}
|\boldsymbol{\Delta}| &= \begin{vmatrix}\delta_\alpha & \delta_\gamma & \delta_\beta\\ \delta_\beta & \delta_\alpha & \delta_\gamma\\ \delta_\gamma & \delta_\beta & \delta_\alpha\end{vmatrix}\\
&= \delta_\alpha^3+\delta_\beta^3+\delta_\gamma^3-3\delta_\alpha\delta_\beta\delta_\gamma\\
&= (\delta_\alpha+\delta_\beta+\delta_\gamma)\left(\delta_\alpha^2+\delta_\beta^2+\delta_\gamma^2\right.\\
&\qquad\qquad \left.-\delta_\alpha\delta_\beta-\delta_\alpha\delta_\gamma-\delta_\beta\delta_\gamma\right)\\
&= \frac{1}{2}(\delta_\alpha+\delta_\beta+\delta_\gamma)\Big((\delta_\alpha-\delta_\beta)^2+(\delta_\alpha-\delta_\gamma)^2\\
&\qquad\qquad +(\delta_\beta-\delta_\gamma)^2\Big)\\
&\neq 0 \qquad (\delta_\alpha,\delta_\beta \text{ and } \delta_\gamma \text{ are not all null by definition})
\end{aligned}$$

APPENDIX C
PROOF OF EQ. (15)

$$\begin{aligned}
\boldsymbol{M}^c.(a\boldsymbol{p}_1+b\boldsymbol{p}_2+c\boldsymbol{p}_3) &= a\boldsymbol{\delta}_1+b\boldsymbol{\delta}_2+c\boldsymbol{\delta}_3\\
&= \begin{pmatrix}
(a+b+c)\delta_1\\
\vdots\\
a\delta_\alpha+b\delta_\gamma+c\delta_\beta\\
\vdots\\
a\delta_\beta+b\delta_\alpha+c\delta_\gamma\\
\vdots\\
a\delta_\gamma+b\delta_\beta+c\delta_\alpha\\
\vdots\\
(a+b+c)\delta_{q_c}
\end{pmatrix}\\
&= (\ \delta_1\ldots f_1\ldots f_2\ldots f_3\ldots\delta_{q_c}\ )^T\\
&\triangleq \boldsymbol{\delta}^*
\end{aligned}$$