
Toward the Weight Enumerator Function of Pseudo-codewords

Assane Gueye
Department of Electrical Engineering
University of California, Berkeley
agueye@eecs.berkeley.edu

1 Introduction

LDPC codes are (with Turbo Codes) class of codes that approaches the Shannon limit on capacity.

They were invented by Gallager ([5]) in the early 1960's and were almost ignored for about two decades before Tanner [10] provided a graphical interpretation of LDPC codes in 1981. Then started another period of "ignorance about LDPC" codes. It was only in the late 1990's that Mc Kay [9], and Wiberg [11], have at the same time "rediscovered" them again. Since then , LDPC codes have gained a lot of interest.

LDPC codes are known to provide very good decoding performance. Chung *et al.* [1] have given a family of LDPC codes that come within 0.0045 dB of the capacity of the channel. Belief propagation is the most commonly used decoder for LDPC codes. In this algorithm, messages are iteratively sent across a factor graph modeling the structure of the code. Analysis of this *message passing algorithm* consider the asymptotical case where the block length of the code approaches infinity. However, the behavior of this algorithm for the case of finite length code is at present not well understood. Nevertheless, in some cases, techniques have been provided to analysis the performance of finite length LPDC codes. In [3], Di *et al.* analyze the case of the binary Erasure channel (BEC) and show that the performance of the message passing algorithm is tightly related to the presence of *stopping sets* in the factor graph. Wiberg, in [11], developed a finite length analysis based on *computation tree* and *pseudo-codewords*.

Recently, a new algorithm, based on the method of *linear programming (LP) relaxation*, for decoding not only LDPC but an arbitrary linear code, was introduced [4]. In this paper, Wainwright *et al.* design a polytope that contains all valid codewords, and an objective function for which the maximum-likelihood (ML) codeword is the optimum point with integral coordinates. The ML decoding is equivalent to running a linear program on the codeword polytope. It has been reported that experiments on LDPC codes shows that the performance of LP decoding is better than the usual min-sum algorithm.

LP relaxation leads to the notion of pseudo-codeword. This notion was earlier used by Wiberg in [11]. Koeter and Vontobel [7] showed that the performance of iterative decoding is largely dominated not by the minimum distance considerations but by the minimum weight of the pseudo-codeword. In both [4] and [7], it was observed that for the BEC, pseudo-codewords are essentially stopping sets, therefore the minimal pseudo-codeword weight is equal to the minimal stopping set size.

Hence, analyzing iterative message passing algorithm for finite length code is equivalent to analyzing the properties of the pseudo-codewords. One way to completely describe the properties of the (pseu)-codewords of a given code if to compute its weight enumerator function (WEF). In this project, we aim to study the properties of pseudo-codewords with hope to derive results about the WEF.

In this very first step of the project, we will review the basic idea behind LP decoding and present the pseudo-codeword concept. As our goal is to determine the weight enumerator function (WEF) of these pseudo-codewords we will present some result about known WEF and the techniques used to derived them. We will then state a certain

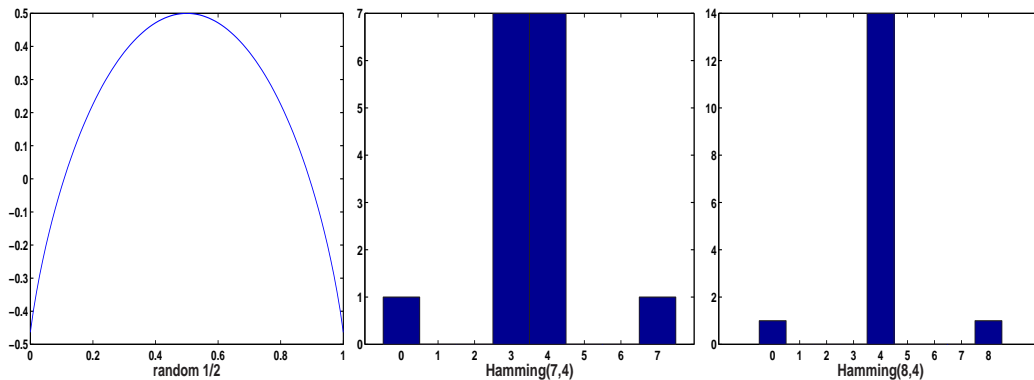


Figure 1: WEF of random rate 1/2 code, Hamming(7,4) code, and extended Hamming(8,4) code

number of questions we would like to answer and give some motivation for these questions.

This report is organized as follow. Section 2 gives a brief introduction on linear codes and their weight enumerator function. In Section 3, we present Gallager's derivation of the WEF for a certain family of LDPC codes. The LP decoding algorithm and the notion of pseudo-codeword are presented in section 4. In section 5 we give some motivation on WEF of pseudo-codeword and conclude this report by a certain number of questions we would like to answer.

2 Linear Code and the WEF

The *weight enumerator function (WEF)* $A(W)$ is the number of codewords \mathbf{c} of weight W . It is a powerful tool to study the structure of error-correcting codes. Note that for linear code, the WEF is, for any code, the number of codewords at distance W . To explain how the WEF can help to understand the structure of a code, let's consider some simple examples.

Figure 1 shows the WEF for the random rate $\frac{1}{2}$ code, the Hamming(7,4) code, and the extended Hamming(8,4) code. Note that for the random code, we have plotted the normalized log WEF. Observing the WEF of Hamming (8,4) and (7,4) codes, one can see that most of the codewords have weight 3 or 4 (i.e. concentrated around what is called minimum distance of the code). In particular for the Hamming(8,4) code, besides the all-zero and the all-one codewords, all codewords have weight 4. This is a feature we can find in all efficient codes. The 'gap' in the weight distribution corresponds to the zones where, for most of the codes, there will not be any codeword (or the average number of codeword vanishes as the length of the code goes to infinity). The bigger the gap is, the better the code is. An optimal decoder will select the codeword which is nearest to the received vector. If we assume that the all-zero codeword was sent (which has been shown to be reasonable if the code is linear), we clearly see that the amount of acceptable noise will be proportional to the size of the gap.

In the general case, the errors of any error-correcting code depend on the properties of the weight enumerator function. Also the WEF determines entirely the probability of an

undetected for a code C_j via the formula:

$$P_u(E|C_j) = \sum_{i=1}^n A_{ij} p^i (1-p)^{n-i} \quad (1)$$

Theoretically, we can compute the weight enumerator of an (n, k) linear code by examining its 2^k codewords or by examining the 2^{n-k} codewords of its dual code and then applying the MacWilliams identity ([8] pp-92). However this computation becomes practically impossible if n, k , and $n - k$ are large. Other techniques for computing the WEF have been presented in the literature. Gallager's method of compute the average WEF of LDPC code is presented in section 3. The weight distribution of the dual and triple-error correcting primitive BCH codes have been completely determined by Kasami [6]. This was done by first computing the WEF of the dual code, and then applying McWilliams identity. For convolutional codes, the modified state diagram and the Masson's formula provide a complete description of the weight enumerator function.

3 Gallager Method *

This method was originally presented in [5].

We will derive the average of the weight enumerator function $A(w)$. We restrict ourself to the ensemble of LDPC codes defined as follow:

$$H = \begin{pmatrix} H_1 \\ H_2 \\ \vdots \\ H_J \end{pmatrix}$$

Where H an $M \times N$ parity-check matrix, H_1 is an $\frac{N}{K} \times N$ matrix of the form:

$$H_1 = \begin{pmatrix} 1111 & 0000 & \cdots & 0000 \\ 0000 & 1111 & \ddots & 0000 \\ \vdots & \ddots & \ddots & 0000 \\ 0000 & \cdots & 0000 & 1111 \end{pmatrix}$$

Where the length of each block of ones or zeros is K . The matrices H_2 to H_J are obtained by random permutation of the columns of H_1 . The resulting matrix is an $M \times N$ matrix with K ones per rows and J ones per column.

Let's first compute the number $A_1(W)$ of sequences of length W that satisfies any one of the J block of $\frac{N}{K}$ parity-checks. Since each column in A_1 has only one 1, no two rows (in any block) have common 1 in the same column. Thus the $\frac{N}{K}$ parity-checks in each block are mutually exclusive and exhaust all the digits. Consider now the set of sequence of length K containing an even number of ones and construct an ensemble E_1 from these sequences by assigning the same probability to each. The total number of sequences in this ensemble

*Source: [2]

is 2^{K-1} , and the probability of a sequence containing k bits (k even), is $\binom{K}{k} 2^{1-K}$. The moment generating function for the number of (even) ones in a sequence is given by:

$$g(s) = \sum_{i \text{ even}} \binom{K}{k} 2^{1-K} e^{si} \quad (2)$$

$$= 2^{-k} [(1 + e^s)^k + (1 - e^s)^k] \quad (3)$$

To go from 2 to 3, we use the binomial expansion and observe that the odd terms cancel.

Now from a new ensemble E_2 of N -length sequence in the following way. For each of the $\frac{N}{K}$ parity-check sets, the corresponding bits (in the N -length sequence) are equal to those of a randomly and independently chosen K -length sequence from E_1 . E_2 is an ensemble of equiprobable N -length sequences, each satisfying the $\frac{N}{K}$ parity checks. The number of ones in each sequence in E_2 is equal to the sum of $\frac{N}{K}$ independent random variables each having moment generating function $g(s)$ in the equation 3. Consequently the moment generating function of the number of ones in an element of E_2 is $(g(s))^{\frac{N}{K}}$. We will now use this quantity to derive an upper bound for the probability $Q(W)$ that a sequence has W ones. By definition,

$$(g(s))^{\frac{N}{K}} = \sum_{W=0}^N e^{sW} Q(W) \quad (4)$$

$$\geq e^{sW} Q(W) \quad (5)$$

Where equation 5 holds for any s and W .

Thus,

$$Q(W) \leq \exp\left(\frac{N}{K}\mu(s) - sW\right) \quad (6)$$

Where

$$\mu(s) = \ln(2^{-k} [(1 + e^s)^k + (1 - e^s)^k]) \quad (7)$$

Finally, $A_1(W)$ equals $Q(W)$ times the number of sequences in E_2 . Since there are $2^{(K-1)}$ sequences in E_1 , and each element in E_2 is composed with $\frac{N}{K}$ randomly chosen elements of E_1 (with repetition allowed), $|E_2| = 2^{\frac{N(K-1)}{K}}$. So that:

$$A_1(W) \leq \exp\left(\frac{N}{K}\mu(s) - sW + \frac{N(K-1)}{K}\ln(2)\right) \quad (8)$$

When we set the derivative of the exponential in equation 8 to zero, we get $W = \frac{N}{K}\mu'(s)$, and we obtain the following equation:

$$A_1\left(\frac{N}{K}\mu'(s)\right) \leq \exp\left(\frac{N}{K}(\mu(s) - s\mu'(s) + (K-1)\ln(2))\right) \quad (9)$$

Now we can use this result to find the probability $P(W)$ of the set of codes for which some particular sequence of weight W is a codeword. Since all permutation of a code

are equally likely, $P(W)$ is independent of the particular W -length sequence chosen. If we choose a W -length at random, then for any code in the ensemble (of code having H as parity check matrix) the probability is $\frac{A_1(w)}{\binom{K}{k}}$ that the chosen sequence will satisfy any

particular block of $\frac{N}{K}$ parity-checks. Since all J blocks have to be satisfied and the block are chosen independently (random permutation of H_1),

$$P(W) = \left(\frac{A_1(w)}{\binom{K}{k}} \right)^J \quad (10)$$

Since we have $\binom{K}{k}$ different codewords of length W , we have:

$$\langle A(w) \rangle \leq \binom{N}{wN} P(W) = \binom{N}{wN}^{1-J} (A_1(W))^J \quad (11)$$

Finally we obtained:

$$\frac{1}{N} \log_2(\langle A(w) \rangle) \approx -(J-1)H(w) + \frac{J}{K(\mu(s) - s\mu'(s) + (K-1)\ln 2)} \quad (12)$$

Where

$$w = \frac{\mu'(s)}{K} \quad (13)$$

and $\mu(s)$ is given in equation 7.

Figure 2 shows plots of the the expected weight enumerator function of different codes.

4 LP decoding of LPDC and Pseudo-codewords

In this section we present the LP decoding algorithm and the notion of pseudo-codewords. We also present the notion of pseudo-codeword as it was defined in [4] and in [7].

Let's start by briefly recalling the ML decoding problem and show how it can be solved by linear programming (LP). This was shown in [4].

4.1 For ML decoding to LP decoding

The basic communication problem is to transmit an information (i.e. a codeword of length n from an alphabet χ) y through a noisy channel, and the decoding process consists to "recover" the sent information from a noisy observation of the channel output \hat{y} . The ML decoding problem is to find the codeword y_{ml} that maximizes $Pr[y|\hat{y}]$ for all y in χ . If we consider discrete memoryless channel and assume that all codewords are equally likely a

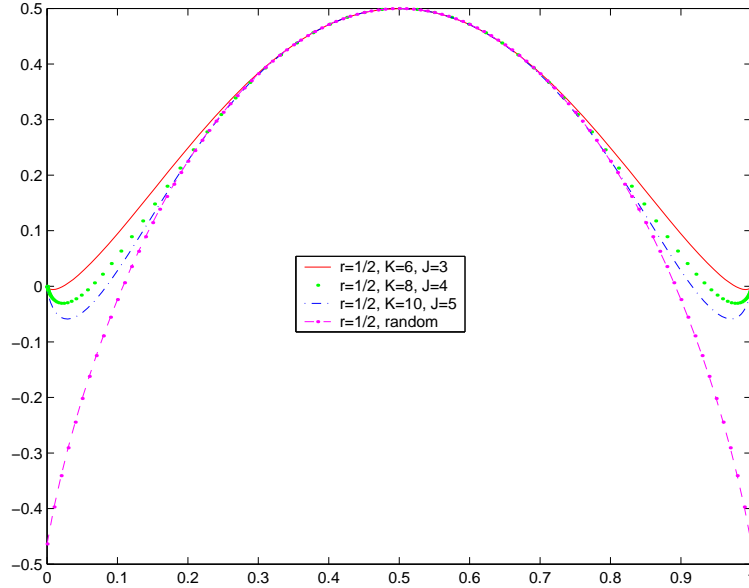


Figure 2: Expected WEF for different Gallager (LDPC) code at rate 1/2. Here we plot $\frac{1}{N} \log_2(\langle A(w) \rangle)$ against the normalized weight w . Curves: solid (red) $K=6, J=3$; dots (green) $K=8, J=4$; dash-dot $K=10, J=5$; dashed-dot random

priori, y_{ml} can be written as:

$$y_{ml} = \arg \max_{y \in \mathcal{X}} Pr[y|\hat{y}] \quad (14)$$

$$= \arg \max_{y \in \mathcal{X}} Pr[\hat{y}|y] \quad (15)$$

$$= \arg \max_{y \in \mathcal{X}} \prod_{i=1}^n Pr[y_i|\hat{y}_i] \quad (16)$$

Where we go from equation 14 to equation 15 by using the equally likely *a priori* assumption and Bayes's rule. The memoryless assumption explains the next step.

Now y_{ml} is the sequence that minimizes the cost function $\sum_{i=1}^n \gamma_i y_i^\dagger$ where

$$\gamma_i = \log\left(\frac{Pr[\hat{y}_i|y_i = 0]}{Pr[\hat{y}_i|y_i = 1]}\right) \quad (17)$$

In other words, our ML problem consists now to minimize a cost function subject to the constraint that the variable is a codeword in some alphabet. This is "almost" the definition of a linear programming. To make it "exact" all we need is to formulate the constraints in the way of linear programming. For that let's define, for a given code, the *codeword polytope* as the *convex* hull of all possible codewords

$$poly(C) = \left\{ \sum_{y \in C} \lambda_y y : \lambda_y \geq 0, \sum_{y \in C} \lambda_y = 1 \right\} \quad (18)$$

Figure 3 shows the codeword polytope of the parity-check code of length 3.

[†]Details of this can be found in [4]

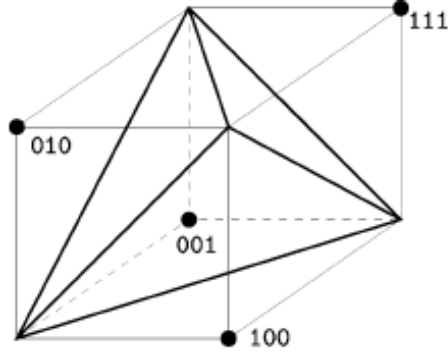


Figure 3: Codeword polytope for the parity-check code of length 3

This polytope is contained within the n -hypercube $[0, 1]^n$, and the vertices correspond to the codewords in C . A key factor is that any linear program attains its optimum at a vertex of the polytope. Thus the ML decoding is now "exactly" equivalent to the problem of minimizing the cost function $\sum_{i=1}^n \gamma_i y_i$ subject to the constraint that the codeword $y = (y_1, \dots, y_n) \in \text{Poly}(C)$.

4.2 LP relaxation and pseudo-codewords

Recall that in the previous section, we have reduced the ML problem into the problem of solving an LP problem over the codeword polytope. However this does not simplify the problem. In fact the NP-completeness of the ML problem has just "changed its hat". In the LP decoding described, the number of constraints ($y = (y_1, \dots, y_n) \in \text{Poly}(C)$) is exponential in the code length n . Therefore, one still needs to find an efficient algorithm for the LP decoding. One possible solution is to define a *relaxed* polytope that contains all the codewords, but has a more manageable representation.

The relaxation strategy proposed in [4] is based on the factor graph of the code and works as follows. Each check node defines a *local code* which is the set of binary vectors that satisfy the check relation (i.e. the bits that correspond to that check must have even weight). The global code corresponds to the intersection of all local codes. In LP terminology, each check node defines a local codeword polytope and the global relaxation polytope will be the intersection of all these polytopes.

The *relaxed polytope* is again contained within the n -hypercube. However, this polytope contains not only the codewords in the underlying code (which correspond to those vertex points in the polytope that have integer values [[4], prop1] called *integral point*), but also other (non-integer) vertex points. These points introduce sequences that satisfy some check relations and violate others. Note that a codeword satisfies all check relations in the factor graph. Since the linear program can attain its optimum at any vertex point, LP decoding over the "relaxed polytope" is no longer equivalent to ML decoding. However, if we ignore non-integral solutions, we obtain an algorithm that will either output the ML solution or acknowledge an error. This is called the *ML certificate property*.

Now let's go back to our "relaxed polytope" and present two different definitions of the concept of pseudo-codewords. The definition in [7] by Koetter *et al.* was inspired by graph

covering while the one given in [4] by Wainwright *et al.* was inspired by the LP relaxation.

Koetter-Vontobel's definition

Source [7].

Let a graph $G = (V, E)$ be given with vertex set $V = (v_0, v_1, \dots, v_{l-1})$ and edge set E . A finite m -cover of G is a graph \hat{G} with vertex set $\hat{V} = \cup_{i=0}^{l-1} \hat{V}_i$ where each set $\hat{V}_i = (v_{i,0}, v_{i,1}, \dots, v_{i,m-1})$ contains exactly m copies of the vertex v_i . The edge set \hat{E} of \hat{G} is chosen as follow.

For each v_i and v_j that are connected in G , connect one copy of v_i to one copy of v_j

- each copy of v_i has the same degree as v_i
- the neighborhood of each copy of v_i contains exactly on copy the v_j

If G is a Tanner graph for a code C of length n , \hat{G} is a Tanner graph a code \hat{C} of length nm . The codewords in \hat{C} are of two kinds:

- all those codewords obtained by assigning the value of a particular variable node in G to all its replicates in \hat{G}
- all other mn -length sequences that satisfy the checks relations

For each variable node $v_i \in G$ and for each codeword $\hat{c} \in \hat{C}$, let's define $w_i(\hat{c})$ as the fraction of times a replicate of v_i assumes the value one. We have:

$$w_i(\hat{c}) = \frac{|l : v_{i,l} = 1|}{m} \quad (19)$$

The vector $w(\hat{c})(w_0(\hat{c}), w_1(\hat{c}), \dots, w_n(\hat{c}))$ is called a pseudo-codeword of C .

Note that any codeword of C is a pseudo-codeword. However there are pseudo-codewords that are not codeword in C .

Figure 4 illustrates this with a trivial code $C = 000$. We see that the 3-cover of the Tanner graph generates a pseudo-codeword $(2/3, 2/3, 2/3)$ which is not a codeword in C . A key point in graph covering is that any locally operating decoding algorithm (e.g max-product) cannot distinguish if it is operating on a Tanner graph or on any finite cover of this graph. Thus any locally operating algorithm will automatically take into account all possible codewords in all possible covers (i.e. all possible pseudo-codewords)

Feldman-Wainwright's definition

Source [4].

Let's first give a definition of a codeword motivates the notion of a pseudo-codeword. Let E_j be the set of non-empty even-sized subsets of the neighborhood of check node j . Let h be a vector in $(0, 1)^n$, and let u be a setting of nonnegative integer weights, one weight

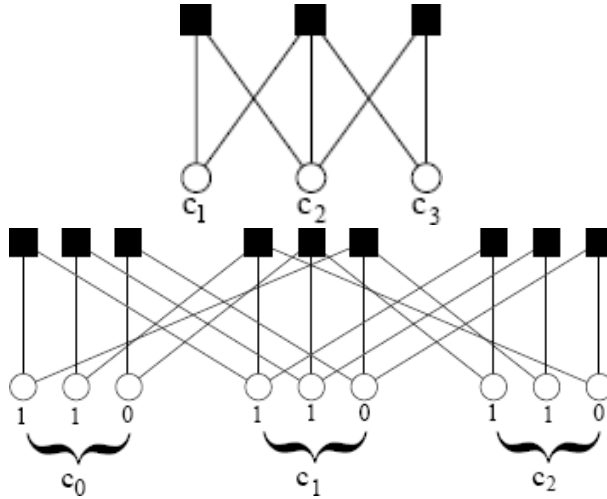


Figure 4: Graph cover illustration

$u_{j,S}$ for each check j and $S \in E_j$. We say that (h, u) is a codeword if, for all edges (i, j) in the factor graph G , $h_i = \sum_{S \in E_j, i \in S} u_{j,i}$.

We obtain the definition of a pseudo-codeword by removing the restriction $h_i \in \{0, 1\}$, and instead allowing each h_i to take on arbitrary nonnegative integer values. In other words, a pseudo-codeword is a vector $h = (h_1, \dots, h_n)$ of nonnegative integers such that, for every parity check $j \in J$, the neighborhood $h_i : i \in N(j)$ is a sum of local codewords (incidence vectors of even-sized sets in \mathcal{C}). Here $N(j)$ is the set of all neighbors of j .

With this definition, any codeword is (trivially) a pseudo-codeword. However, there are pseudo-codewords that do not correspond to codewords in the original code.

5 Conclusion and Future work

The LP solution of the ML problem proposed in [4] has been reviewed. For the iterative decoding of finite length LDPC code, it was shown [[7] that [4]], the performance of the decoding scheme is dominated by the presence of pseudo-codewords. Hence, understanding the properties of the pseudo-codewords is crucial for the analysis of the performance of iterative decoding of LDPC codes. For this project we have set the *ambitious* goal to compute the weight enumerator function of the pseudo-codewords. However, before being able to compute this WEF, one should answer several questions. We list some of them in the next sections.

What are the typical properties of a pseudo-codeword?

In answering this question, we are trying to find any non evident structure of the pseudo-codewords. For example, we are trying to answer the following question: does the pseudo-codewords set have any group (ring, field) structure, linear space? For what operation? This can be helpful for the analysis. For example a linear space of pseudo-codewords suggests that there is a generator matrix.

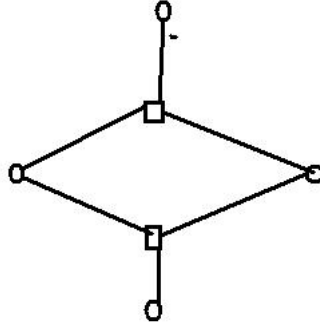


Figure 5: Example of code whose pseudo-codewords are generated by a finite cover

Is there any "optimal" relaxation strategy?

The presence of pseudo-codewords reduces a lot the probability of decoding. A relaxation method that minimizes the number of pseudo-codewords will definitely provide better decoding performance. However, it seems that there is a tradeoff here because the optimal strategy would not do any relaxation and we are back to the ML decoding problem which is NP-hard.

Is there an m -cover that generate all possible pseudo-codewords?

It was shown in [7] that if G is a Tanner graph for a code C of length n , an m -cover \hat{G} is a Tanner graph of a code \hat{C} . Thus if all possible codewords are generated by a certain m -cover, we can base our study in such cover. One example is the Tanner graph given in figure 5. In this case we can compute all the possible pseudo-codewords $(0,1,1,0; 1,0,1,1; 1,1,0,1; 0,1,1,0; 0,1/2,1/2,1; 1,1/2,1/2,0)$. There exists a 2-cover of the graph that generate all possible pseudo-codewords. Does this hold for the general case? This is question is still open.

References

- [1] Sea-Young Chun, David Forney, , Thomas J. Richardson, and Rudiger L. Urbanke. On the design of low density parity check codes within 0.0045 db of the shannon limit. In *IEEE Communication Letters*, volume 5, Feb 2001.
- [2] Sylvain Condamin. Study of the weight enumerator function of a gallager code. In *Cavendish Laboratory Press, Cambridge*, June 2002.
- [3] Changyan Di, David Prooietti, I. Emre Telatar, Thomas J. Richardson, and Rudiger L. Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. In *IEEE Transactions on Information Theory*, volume 48, pages 1570–1579, June 2002.

- [4] J. Feldman, Martin J. Wainwright, and David R. Karger. Using linear programming to decode binary linear codes. In *IEEE Transactions on Information Theory*, pages 954–972, March 2005.
- [5] R.G. Gallager. Low density parity check (ldpc) codes. In *IEEE Transactions on Information Theory*, pages 21–28, 1962.
- [6] T. Kasami. Weight distributions of bose-chaudhuri-hoquenghem codes. In *Conference on Combinatorial Mathematics and its Applications*, 1968.
- [7] R. Koetter and P.O. Vontobel. Graph-covers and iterative decoding of finite length codes. In *International Symposium on Turbo Codes*, pages 75–82, September 2003.
- [8] Shu Lin and Jr. Daniel J. Costello. Error control coding. In *Pearson Prentice Hall*.
- [9] D. McKay. Good error-correcting codes based on very sparse matrices. In *IEEE Transactions on Information Theory*, pages 399–431, March 1999.
- [10] R.M. Tanner. A recursive approach to low complexity codes. In *IEEE Transaction on Information Theory*, 1981.
- [11] N. Wiberg. Codes and decoding on general graphs. In *Ph.D. dissertation, Linkoping University, Sweden*, 1996.