A Probabilistic Algorithm for Computing the Weight Distribution of LDPC Codes

Hirotomo, Masanori

Mohri, Masami

Morii, Masakatu

(Citation)
IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences,E92-A(7):1677-1689

(Issue Date)
2009

(Resource Type)
journal article

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/90001304

PAPER

A Probabilistic Algorithm for Computing the Weight Distribution of LDPC Codes

Masanori HIROTOMO[†a)], **Masami MOHRI**[††], ***and*** **Masakatu MORII**[†], ***Members***

SUMMARY Low-density parity-check (LDPC) codes are linear block codes defined by sparse parity-check matrices. The codes exhibit excellent performance under iterative decoding, and the weight distribution is used to analyze lower error probability of their decoding performance. In this paper, we propose a probabilistic method for computing the weight distribution of LDPC codes. The proposed method efficiently finds low-weight codewords in a given LDPC code by using Stern's algorithm, and stochastically computes the low part of the weight distribution from the frequency of the found codewords. It is based on a relation between the number of codewords with a given weight and the rate of generating the codewords in Stern's algorithm. In the numerical results for LDPC codes of length 504, 1008 and 4896, we could compute the weight distribution by the proposed method with greater accuracy than by conventional methods.

key words: *LDPC codes, weight distribution, probabilistic method, minimum distance*

1. Introduction

Low-density parity-check (LDPC) codes are linear block codes defined by sparse parity-check matrices [1]. Interest in them has been revived since the codes exhibit excellent performance under iterative decoding [2]. The performance with low error probability in high signal noise ratio (SNR) region is theoretically analyzed by upper and lower bounds such as asymptote of union bound [3], [4]. In the analysis, the average weight distribution in the ensemble of LDPC codes is used instead of the exact weight distribution, and formulas of the average weight distribution have been derived for some LDPC code ensembles [5]–[7]. But the formulas cannot be applied to computing the weight distribution of an LDPC code.

The minimum distance of linear codes determines the number of errors correctable by the code, and the weight distribution is required to analyze the maximum-likelihood (ML) decoding performance. Unfortunately, it is difficult to compute the minimum distance and the weight distribution, since the complexity grows exponentially as the code length and the dimension are larger. It has been well known that no algorithm computes exactly the minimum distance and the weight distribution of long linear codes in polynomial time. Generally, the problem for any linear code is NP-hard [8], [9].

Manuscript received October 21, 2008.
Manuscript revised February 16, 2009.

[†]The authors are with the Graduate School of Engineering, Kobe University, Kobe-shi, 657-8501 Japan.
[††]The author is with the Information and Multimedia Center, Gifu University, Gifu-shi, 501-1193 Japan.
a) E-mail: hirotomo@eedept.kobe-u.ac.jp

DOI: 10.1587/transfun.E92.A.1677

To estimate the minimum distance of LDPC codes, R.M. Tanner presented a lower bound on the minimum distance of regular LDPC codes by analyzing the connectivity of the bipartite graph representing an LDPC code [10]. The minimum distance of LDPC codes with algebraic structures is actively investigated, for example, quasi-cyclic LDPC codes [11], [12], array LDPC codes [13]–[16]. In particular, the minimum distance of finite geometry LDPC codes is determined by parameters used to construct the codes [17]. These methods can apply to LDPC codes with restricted structures, but cannot evaluate the weight distribution.

X.-Y. Hu et al. proposed an algorithm, called nearest nonzero codeword search (NNCS) approach, to approximate the minimum distance of LDPC codes [18], [19]. In this method, the error impulse (EI) method [20] and the iterative reliability-based (IRB) decoding [21] are used to find minimum-weight codewords in an LDPC code. In the fundamental principle, the method perturbs the transmitted vector corresponding to the all-zero codeword with error impulse, and uses the IRB decoding to obtain nonzero codewords which are nearest to the all-zero codeword. The numerical results of applying the NNCS approach to evaluating the low part of the weight distribution for several LDPC codes have been reported in [22]. Furthermore, in [23], a modified algorithm based on the EI method has been presented. The algorithm uses error patterns constituted by two error impulses located in two different bit positions at two level search. However, these methods cannot necessarily find all of codewords included in the low-part weight distribution, because the accuracy of the found minimum distance and the computed weight distribution is dependent on the capability of the decoding algorithms in these methods. Thus, computing the exact weight distribution of an LDPC code constructed by an arbitrary structure has been hard problem yet.

As for the evaluation of the minimum distance of long binary linear codes, probabilistic algorithms for finding the minimum-weight codeword have been proposed in [25], [26] and [27]. These algorithms efficiently find low-weight codewords in a binary linear code of long length and high dimension. However it is unable to determine the number of codewords with a given weight only using these algorithms, so the weight distribution cannot be evaluated.

In this paper, we propose a probabilistic method for computing the weight distribution of LDPC codes. In the proposed method, we apply Stern's probabilistic algorithm [26] to finding low-weight codewords in a given LDPC

code, and the low part of the weight distribution is stochastically computed from the frequency of the found low-weight codewords. It is based on a relation between the number of codewords with a given weight and the rate of generating the codewords in Stern's algorithm. Using the proposed method, we can compute the low-part weight distribution with high accuracy. Additionally, as numerical experiments, we computed the weight distribution of several LDPC codes. In these results, we found a minimum-weight codeword of the LDPC codes which has not been reported in [18], [22] and [23], and newly found a lot of low-weight codewords included in the weight distribution. It implies that the weight distribution of these codes computed by our method is greater reliability than by the conventional methods.

This paper is organized as follows. In Sect. 2, we describe the definition of LDPC codes and weight distribution. In Sect. 3, we explain the concept of probabilistic methods and review Stern's algorithm. In Sect. 4, we propose a probabilistic method for computing the weight distribution of LDPC codes. In Sect. 5, we show numerical results obtained by applying the proposed method to several LDPC codes. In Sect. 6, we conclude this paper.

2. LDPC Codes and Weight Distribution

LDPC codes in their broader definition are linear block codes whose parity-check matrices have fewer nonzeros than zeros. Let C be a binary linear code of length n and dimension k. The parity-check matrix is denoted by

$$\mathbf{H} = [h_{ji}]_{\substack{1 \le j \le m \\ 1 \le i \le n}}, \tag{1}$$

where $m \ge n-k$. Any codeword $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ satisfies

$$\mathbf{H}\mathbf{c}^T = \mathbf{0}, \tag{2}$$

and the binary (n, k) linear code C is a set of 2^k codewords of length n where $k = n - \mathrm{rank}(\mathbf{H})$. The weight (Hamming weight) of $\mathbf{c}$ is denoted by $w_h(\mathbf{c})$, and the number of codewords of weight w in C is denoted by A_w. Then, the weight distribution $\{A_0, A_1, \ldots, A_n\}$ of C is defined as

$$\begin{aligned} A_w &= \#\{w_h(\mathbf{c}) = w : \mathbf{c} \in C\} \\ &= \#\{w_h(\mathbf{c}) = w : \mathbf{H}\mathbf{c}^T = \mathbf{0}\}. \end{aligned} \tag{3}$$

The minimum weight (minimum distance) of C is the smallest positive integer d such that $A_d \neq 0$.

3. Probabilistic Method

The minimum distance of linear codes determines the number of errors correctable by the code, and the weight distribution is required for analyzing the decoding performance. Unfortunately, for a general code of length n and dimension k, it is necessary to examine the weight of 2^k codewords in order to compute the weight distribution. Even if we perform the procedure of verifying whether all vectors of length n and weight $w = 1, 2, \ldots, d$ is satisfied with Eq. (2) in order to determine the minimum weight, we require the complexity $\sum_{w=1}^{d} \binom{n}{w}$. Thus it is difficult to compute the minimum weight and the weight distribution of long practical LDPC codes. For the problem of computing the minimum weight of long linear codes, probabilistic algorithms are proposed in [25], [26] and [27]. The general probabilistic method is performed as follows:

1. Given ε, small positive real number.
2. Repeat the random choice of codewords from C and the examination of its weight. Then, suppose d to be lowest weight in the examinations.
3. It can be concluded that, if C were to contain codewords of nonzero weight less than d, at least one such codeword would have been found with probability at least $1 - \varepsilon$.

When the above process is concluded, the probability of failing to determine the minimum weight of C is not more than ε. So, we shall say that the minimum weight of C is d with probability $1 - \varepsilon$. ε denotes a failure probability of the minimum weight of C.

In [26], J. Stern has presented a probabilistic algorithm to find low-weight codewords of binary linear codes with high probability. Stern's algorithm performs as follows. The columns of $\mathbf{H}$ are randomly permuted, and the Gaussian elimination is applied to the column-permuted matrix in order to obtain a matrix $[\mathbf{E}_{n-k}\mathbf{P}]$ where $\mathbf{E}_{n-k}$ is an $(n-k)\times(n-k)$ identity matrix. Then, the columns of $\mathbf{P}$ is randomly split into two submatrices of same size denoted by $\mathbf{X}$ and $\mathbf{Y}$, so that $\mathbf{H}' = [\mathbf{E}_{n-k}\mathbf{X}\mathbf{Y}]$. The random column permutation implies that each permutation is occurred with same probability, and the random index selection implies that each index is aimlessly selected with same probability. Sets of indices whose columns of $\mathbf{H}$ are permutated into certain parts of $\mathbf{H}'$ is denoted as follows:

- I denotes the set of column indices of $\mathbf{H}$ permutated into $\mathbf{P}$, and is called an information set.
- I_X and I_Y denotes sets of column indices of $\mathbf{H}$ permutated into $\mathbf{X}$ and $\mathbf{Y}$ respectively.
- I'_X denotes a subset of I_X with p elements, and I'_Y denotes a subsets of I_Y with p elements.
- R denotes the set of column indices of $\mathbf{H}$ permutated into $\mathbf{E}_{n-k}$, and is called a redundancy set.
- L denotes a set containing l indices which are selected from the redundancy set R randomly.

In Stern's algorithm, vectors $\mathbf{c}' = (c'_1, c'_2, \ldots, c'_n)$ satisfying $\mathbf{H}'\mathbf{c}'^T = \mathbf{0}$ are generated under the following conditions and the weight of $\mathbf{c}'$ is examined:

$$w_h(\mathbf{c}'_{|I_X}) = w_h(\mathbf{c}'_{|I_Y}) = p, \tag{4}$$

$$w_h(\mathbf{c}'_{|L}) = 0, \tag{5}$$

where $\mathbf{c}'_{|I}$ is a vector consisting of elements of $\mathbf{c}'$ whose indices are contained in I. Since $\mathbf{H}'$ is in the systematic form, $(n-k)$-tuple of bits of $\mathbf{c}'$ corresponding to the redundancy

set R is obtained by adding up the columns of $\mathbf{H}'$ whose indices are contained in the information set I. The $(n-k)$-tuple of bits of $\mathbf{c}'$ is denoted by $\mathbf{v} = (c'_1, c'_2, \ldots, c'_{n-k})$. The vector $\mathbf{c}'$ satisfying Eq. (4) is obtained by the linear combination of $2p$ columns whose indices are chosen from I_X and I_Y. These indices are represented by elements of I'_X and I'_Y. For example, when $I'_X = \{i_1, i_2\}$ and $I'_Y = \{i_3, i_4\}$, the part of $\mathbf{c}'$ corresponding to R are calculated by $\mathbf{v} = \mathbf{h}'_{i_1} + \mathbf{h}'_{i_2} + \mathbf{h}'_{i_3} + \mathbf{h}'_{i_4}$ where $\mathbf{h}_i$ is a column of $\mathbf{H}'$. Furthermore, when $\mathbf{c}'$ satisfies Eq. (5), $\mathbf{v}$ holds the condition of $\mathbf{v}_{|L} = \mathbf{0}$. Then we obtain $\mathbf{h}'_{i_1|L} + \mathbf{h}'_{i_2|L} = \mathbf{h}'_{i_3|L} + \mathbf{h}'_{i_4|L}$. The vector $\mathbf{c}'$ satisfying Eqs. (4) and (5) can efficiently be found by the following algorithm:

Stern's Algorithm [26]
Input: Parity-check matrix $\mathbf{H}$, parameters p and l
Output: Weight w of codewords in C
Step1: Permute columns of $\mathbf{H}$ randomly, and apply the Gaussian elimination to the column-permuted matrix so that $[\mathbf{E}_{n-k}\mathbf{P}]$. Then, if no pivot is found in the selected column, swap the column for one of unselected columns. Let I be an information set containing indices whose columns were permutated into $\mathbf{P}$, and let R be a redundancy set containing indices whose columns were permutated into $\mathbf{E}_{n-k}$. Then, split I to two subset I_X and I_Y of size $k/2$ in order to obtain the following matrix:

$$\mathbf{H}' = [\mathbf{E}_{n-k}\mathbf{X}\mathbf{Y}],$$
$$\mathbf{X} = [\mathbf{x}_i]_{i\in I_X},$$
$$\mathbf{Y} = [\mathbf{y}_i]_{i\in I_Y}.$$

Step2: Select l indices from the redundancy set R randomly. The set is denoted by $L = \{j_1, j_2, \ldots, j_l\}$, and the indices point to the rows of $\mathbf{H}'$.
Step3: Let I'_X and I'_Y be subsets containing p column indices of I_X and I_Y respectively. Choose every subset I'_X of I_X with p elements, and compute the l-bit linear combination

$$\mathbf{\Lambda}_{I'_X|L} = \sum_{i\in I'_X} \mathbf{x}_{i|L},$$

and add I'_X into the entry with key $\mathbf{\Lambda}_{I'_X|L}$ in the hash table of 2^l entries. Then, choose every subset I'_Y of I_Y with p elements, and compute the l-bit linear combination

$$\mathbf{\Lambda}_{I'_Y|L} = \sum_{i\in I'_Y} \mathbf{y}_{i|L},$$

and search for I'_X with key $\mathbf{\Lambda}_{I'_Y|L}$ in the hash table.
Step4: Using the hash table, consider all pairs (I'_X, I'_Y) such that $\mathbf{\Lambda}_{I'_X|L} = \mathbf{\Lambda}_{I'_Y|L}$ and compute $\mathbf{v} = (v_1, v_2, \ldots, v_{n-k})$ as follows:

$$\mathbf{v} = \mathbf{\Lambda}_{I'_X} + \mathbf{\Lambda}_{I'_Y} = \sum_{i\in I'_X} \mathbf{x}_i + \sum_{i\in I'_Y} \mathbf{y}_i.$$

If $w_h(\mathbf{v}) = w - 2p$, a vector $\mathbf{c}' = (c'_1, c'_2, \ldots, c'_n)$ such that

$$c'_i = \begin{cases} 1, & \text{if } i \in I'_X \cup I'_Y \text{ or } v_i = 1 \\ 0, & \text{otherwise} \end{cases}$$

is exactly of weight w and satisfies $\mathbf{H}'\mathbf{c}'^T = \mathbf{0}$. Then, return the weight w of the vectors $\mathbf{c}'$ for the pairs (I'_X, I'_Y) above. ■

The vectors $\mathbf{c}'$ is a codeword whose elements are permuted since $\mathbf{c}'$ is satisfied with $\mathbf{H}'\mathbf{c}'^T = \mathbf{0}$ for the column-permuted parity-check matrix $\mathbf{H}'$. Therefore, if $w_h(\mathbf{c}'_{|R}) = w - 2p$, C contains a codeword of weight w.

The capability that Stern's probabilistic algorithm find the minimum weight codeword is analyzed in [26] and [28]. In the literature, in order to evaluate the minimum weight of C, the algorithm investigates whether C contains at least one codeword of weight w. If C contains a codeword $\mathbf{c}$ of weight w, the probability to find the codeword in a process from Step1 to Step4 of Stern's algorithm is given by

$$\pi_{w,p,l} = \frac{\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p}}{\binom{n}{k/2}\binom{n-k/2}{k/2}} \frac{\binom{n-k-w+2p}{l}}{\binom{n-k}{l}}. \quad (6)$$

The codeword $\mathbf{c}$ is found as the vector $\mathbf{c}'$ in Step4 of Stern's algorithm. The first part of right hand of Eq. (6) is a probability that p ones and $k/2 - p$ zeros of $\mathbf{c}$ are permutated into the bit positions corresponding to I_X and I_Y respectively. The numerator of the first part implies the number of such permutations, and the denominator implies the number of all permutations to choose $k/2$ column indices into both I_X and I_Y among n column indices. The second part is a probability that l zeros among the remaining $n - k - w + 2p$ zeros are permutated into the bit positions corresponding to L. The numerator of the second part implies the number of such permutations, and the denominator implies the number of all permutations to choose l indices into L among $n - k$ indices of R. Strictly speaking, we cannot obtain the permutations that R contains indices of linear dependent columns when the Gaussian elmination is applied to $\mathbf{H}$. However, the number of the permutations is much smaller than the number of all permutations, so the probability to find a codeword of weight w is estimated by Eq. (6).

To find a codeword of minimum weight efficiently, we can use Stern's algorithm instead of the random codewords choice, and repeat the algorithm until the failure probability is smaller than ε. The events that the vector $\mathbf{c}'$ is found are independent for each iteration of Stern's algorithm. Thus, the probability of failing to find the vector $\mathbf{c}'$ of weight w after r iterations is $(1 - \pi_{w,p,l})^r$. Given a small positive real number ε, the number of iterations required to find $\mathbf{c}'$ with failure probability ε or less is given by $r_w = \lceil \log(\varepsilon)/\log(1 - \pi_{w,p,l}) \rceil$. When we find a vector $\mathbf{c}'$ of weight d and find no vector of weight less than d after r_d iterations of Stern's algorithm, the failure probability of minimum weight is ε at most. Because $r_w \leq r_d$ for $w < d$, and no vector of weight less than d is found after r_d iterations. Hence, when we find a codeword of weight d and find no codeword of weight less than d after r iteration of Stern's algorithm, we shall say that the minimum weight is d with probability $1 - \varepsilon$ such that

$$\varepsilon = (1 - \pi_{d,p,l})^r. \quad (7)$$

Next, we consider the time complexity of Stern's algorithm for (n, k) linear codes. The complexity is dominated by Step1, Step3 and Step4, and the complexity of each step is analyzed in [26] and [27] as follows:

- The Gaussian elimination performed in Step1 requires a number of bit operations of order $\frac{1}{2}(n-k)^3 + k(n-k)^2$.
- In Step3, there are $\binom{k/2}{p}$ linear combinations of p columns of $\mathbf{X}$ and $\mathbf{Y}$ respectively. Computing each of them on a l-bit selection and putting it in the hash table require pl binary additions. Additionally, we need $K\left(2^l + p\binom{k/2}{p}\right)$ more operations to perform the dynamic memory allocation where K is the size of a computer word ($K = 32$ or 64).
- The average number of collisions, i.e., the average number of pairs (I'_X, I'_Y) such that $\mathbf{\Lambda}_{I'_X|L} = \mathbf{\Lambda}_{I'_Y|L}$ is equal to $\frac{\binom{k/2}{p}^2}{2^l}$. For each collision, we perform $2p-1$ additions of $(n-k)$-bit columns for computing $\mathbf{\Lambda}_{I'_X} + \mathbf{\Lambda}_{I'_Y}$ and a weight checking.

Hence the average number of elementary operations performed at each iteration of Stern's algorithm is

$$\Omega_{p,l} = \frac{1}{2}(n-k)^3 + k(n-k)^2 + 2pl\binom{k/2}{p} + K\left(2^l + p\binom{k/2}{p}\right) + 2p(n-k)\frac{\binom{k/2}{p}^2}{2^l}. \quad (8)$$

We call $\Omega_{p,l}$ a running cost.

For the hard problem of computing the minimum weight d of linear codes with large parameters n and k, we can stochastically evaluate the minimum weight with small failing probability ε by using Stern's algorithm. However, even if we evaluate the minimum weight d, we cannot know the number of codewords of minimum weight d since the algorithm has no scheme to compute the number of codewords of a given weight. It is unable to compute the weight distribution only using the probabilistic algorithm.

4. Probabilistic Method for Computing the Weight Distribution of LDPC Codes

In this section, we propose a probabilistic method for computing the weight distribution for low-weight codewords of LDPC codes. It is based on Stern's algorithm. In this method, first, we find low-weight codewords in C by using Stern's algorithm, then stochastically compute the low part of the weight distribution from the frequency of found codewords. The method focuses on a relation between the number of codewords of weight w in C and the rate of generating the codewords in the iterations of Stern's algorithm. The minimum weight and the weight distribution have been evaluated for several codes defined by sparse parity-check matrices with random constructions in the literature, for example, [1], [5]–[7], [18], [22], [23]. It have been reported that the minimum weight is larger as the length is longer. On the other hand, the codes have small minimum weight and few minimum-weight codewords. Since the complexity of our method for computing the low-part weight distribution depends on the minimum weight of a given code, our method can efficiently compute the low-part weight distribution of these LDPC codes. In the remainder of this paper, A_w denotes the exact weight distribution of C, and $\tilde{A}_w$ denotes the weight distribution computed by the proposed method. B_w also denotes the frequency of found codewords of weight w in the iterations of Stern's algorithm.

4.1 Relation Between the Number of Codewords with a Given Weight and the Rate of Generating the Codewords in Stern's Algorithm

The probability to find a codeword of weight w by Stern's algorithm is given by $\pi_{w,p,l}$ in Eq. (6). It is used to estimate the failure probability of the minimum weight of C. In the estimation, we determine whether C contains at least one codewords of each weight. In [26], the probability $\pi_{w,p,l}$ is given under the condition that C contains only one codewords of weight w, i.e., the probability is not considered under the condition that C contain A_w codewords of weight w. On the other hand, in the case that C contains several codewords of weight w, the rate of generating the codewords becomes larger than Eq. (6). We focus on this property and develop a probabilistic method for computing the weight distribution.

Theorem 1: If C contains A_w codewords of weight w, the average rate of generating the codewords at each iteration of Stern's algorithm is given by

$$E_{w,A_w,p,l} = A_w \pi_{w,p,l} ,$$

where $\pi_{w,p,l}$ is given as Eq. (6).

Proof: In the case of $A_w = 1$, the probability to find a codeword of weight w is given by Eq. (6). Then, the average rate of generating the codeword is equal to Eq. (6), i.e.,

$$E_{w,1,p,l} = \frac{\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p}}{\binom{n}{k/2}\binom{n-k/2}{k/2}} \frac{\binom{n-k-w+2p}{l}}{\binom{n-k}{l}}. \quad (9)$$

On the other hand, if $A_w \geq 2$, $E_{w,A_w,p,l}$ is not equal to Eq. (6). That is, if C contains several codewords of weight w, it effects the following part of Eq. (9):

$$\frac{\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p}}{\binom{n}{k/2}\binom{n-k/2}{k/2}}. \quad (10)$$

Let $\mathbf{c}$ be the codeword of weight w in C, i.e., $\mathbf{H}\mathbf{c}^T = \mathbf{0}$. In Step4 of Stern's algorithm, the vector $\mathbf{c}'$ is generated by $\mathbf{H}'$ which is obtained by the column permutation of $\mathbf{H}$, and $\mathbf{c}'$ has p ones and $k/2 - p$ zeros in the bit positions corresponding to I_X and I_Y respectively. Thus, when p ones and $k/2 - p$ zeros of $\mathbf{c}$ are permuted into the bit positions corresponding to I_X and I_Y by the column permutation of $\mathbf{H}$ in Step1, the codeword $\mathbf{c}$ is generated as the permuted codeword $\mathbf{c}'$ by $\mathbf{H}'$, so that $\mathbf{H}'\mathbf{c}'^T = \mathbf{0}$. $\binom{n}{k/2}\binom{n-k/2}{k/2}$ in

Eq. (10) is the number of all permutations to choose $k/2$ column indices into both I_X and I_Y among n column indices. $\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p}$ in Eq. (10) is the number of permutations to choose p ones and $k/2-p$ zeros into both I_X and I_Y among w ones and $n-w$ zeros of $\mathbf{c}$. Therefore, we obtain Eq. (10), and the average rate of finding $\mathbf{c}$ is given by Eq. (9).

Next, we consider the case that C contains two codewords of weight w. Let $\mathbf{c}_1$ and $\mathbf{c}_2$ be the two codewords respectively. When we choose p ones and $k/2-p$ zeros into both I_X and I_Y among w ones and $n-w$ zeros of $\mathbf{c}_1$, there are $\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p}$ permutations. Similarly, there are $\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p}$ permutations for $\mathbf{c}_2$. Some of these permutations to choose from $\mathbf{c}_2$ are the same as the permutations to choose from $\mathbf{c}_1$. This implies that both $\mathbf{c}_1$ and $\mathbf{c}_2$ can be generated by all of $\mathbf{H}'$ with these permutations. Therefore, we only have to add the number of permutations to choose for each codeword. Consequently, in the case of $A_w = 2$, the vector $\mathbf{c}'$ of weight w is generated

$$2\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p}. \tag{11}$$

times from $\mathbf{H}'$ obtained by all column permutation.

In the case that C contains A_w codewords of weight w, the frequency of generating the vector $\mathbf{c}'$ of weight w from $\mathbf{H}'$ with all permutations is

$$A_w\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p}. \tag{12}$$

Consequently, the average rate of generating the codewords of weight w is given by

$$E_{w,A_w,p,l} = \frac{A_w\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p}}{\binom{n}{k/2}\binom{n-k/2}{k/2}} \cdot \frac{\binom{n-k-w+2p}{l}}{\binom{n-k}{l}}. \tag{13}$$

□

Example 1: Consider a $(7,4)$ code C defined by the following parity-check matrix:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

The code contains two codewords of weight 2, i.e., $\mathbf{c}_1 = (1,0,0,1,0,0,0)$, $\mathbf{c}_2 = (0,0,0,0,1,1,0)$. The number of all permutations to choose two column indices of $\mathbf{H}$ into both I_X and I_Y is

$$\binom{n}{k/2}\binom{n-k/2}{k/2} = \binom{7}{2}\binom{5}{2} = 210.$$

When $p = 1$, the number of permutations to choose an one and a zero into both I_X and I_Y among 2 ones and 4 zeros of $\mathbf{c}_1$ is

$$\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p} = \binom{2}{1}\binom{4}{1}\binom{1}{1}\binom{3}{1} = 40.$$

The permutations are as follows:

$$\begin{aligned}(I_X, I_Y) =& (\{1,2\},\{3,4\}), (\{1,2\},\{4,5\}), (\{1,2\},\{4,6\}),\\ & (\{1,2\},\{4,7\}), (\{1,3\},\{2,4\}), (\{1,3\},\{4,5\}),\\ & (\{1,3\},\{4,6\}), (\{1,3\},\{4,7\}), (\{1,5\},\{2,4\}),\\ & (\{1,5\},\{3,4\}), (\{1,5\},\{4,6\}), (\{1,5\},\{4,7\}),\\ & (\{1,6\},\{2,4\}), (\{1,6\},\{3,4\}), (\{1,6\},\{4,5\}),\\ & (\{1,6\},\{4,7\}), (\{1,7\},\{2,4\}), (\{1,7\},\{3,4\}),\\ & (\{1,7\},\{4,5\}), (\{1,7\},\{4,6\}), (\{2,4\},\{1,3\}),\\ & (\{2,4\},\{1,5\}), (\{2,4\},\{1,6\}), (\{2,4\},\{1,7\}),\\ & (\{3,4\},\{1,2\}), (\{3,4\},\{1,5\}), (\{3,4\},\{1,6\}),\\ & (\{3,4\},\{1,7\}), (\{4,5\},\{1,2\}), (\{4,5\},\{1,3\}),\\ & (\{4,5\},\{1,6\}), (\{4,5\},\{1,7\}), (\{4,6\},\{1,2\}),\\ & (\{4,6\},\{1,3\}), (\{4,6\},\{1,5\}), (\{4,6\},\{1,7\}),\\ & (\{4,7\},\{1,2\}), (\{4,7\},\{1,3\}), (\{4,7\},\{1,5\}),\\ & (\{4,7\},\{1,6\}).\end{aligned}$$

It implies that $\mathbf{c}_1$ can be generated by all matrices $\mathbf{H}'$ obtained by these column permutations. Similarly, for $\mathbf{c}_2$, there are 40 permutations to choose 1 one and 1 zero into both I_X and I_Y. Among the number, both $\mathbf{c}_1$ and $\mathbf{c}_2$ can be generated in case of the following permutations:

$$\begin{aligned}(I_X, I_Y) =& (\{1,5\},\{4,6\}), (\{1,6\},\{4,5\}),\\ & (\{4,5\},\{1,6\}), (\{4,6\},\{1,5\}).\end{aligned}$$

Let us consider the case of $(I_X, I_Y) = (\{1,5\},\{4,6\})$. Then $\mathbf{c}' = (c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7)$ is satisfied with the condition of $c'_4 = c_1$, $c'_5 = c_5$, $c'_6 = c_4$ and $c'_7 = c_6$. $\mathbf{c}_1$ is generated by $\mathbf{H}'$ when $\mathbf{c}'$ becomes $c'_4 = 1$, $c'_5 = 0$, $c'_6 = 1$ and $c'_7 = 0$, and $\mathbf{c}_2$ is generated by $\mathbf{H}'$ when $\mathbf{c}'$ becomes $c'_4 = 0$, $c'_5 = 1$, $c'_6 = 0$ and $c'_7 = 1$. Thus both $\mathbf{c}_1$ and $\mathbf{c}_2$ are generated under the condition of the permutation of $(I_X, I_Y) = (\{1,5\},\{4,6\})$. Similarly, both $\mathbf{c}_1$ and $\mathbf{c}_2$ can be generated in the above four cases. Consequently the number of finding $\mathbf{c}_1$ and $\mathbf{c}_2$ for all permutations is

$$A_w\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p} = 2\binom{2}{1}\binom{4}{1}\binom{1}{1}\binom{3}{1} = 80,$$

and the average rate of generating the codewords of weight 2 at each iteration is equal to

$$\frac{A_w\binom{w}{p}\binom{n-w}{k/2-p}\binom{w-p}{p}\binom{n-w-k/2+p}{k/2-p}}{\binom{n}{k/2}\binom{n-k/2}{k/2}} = \frac{80}{210} = 0.381.$$

4.2 Probabilistic Method for Computing the Weight Distribution

In the case that C contains A_w codewords of weight w, the average rate of generating the codewords at each iteration of Stern's algorithm is given by $A_w\pi_{w,p,l}$. We count up the frequency B_w of generating the vectors $\mathbf{c}'$ of weight w through r iterations of Stern's algorithm. Since the average frequency at each iteration is more close to $E_{w,A_w,p,l}$ as the number of iterations r is larger, we obtain the following relation:

$$E_{w,A_w,p,l} \approx \frac{B_w}{r} . \tag{14}$$

From Theorem 1, the weight distribution can be computed as follows:

$$\tilde{A}_w = \frac{B_w}{r\pi_{w,p,l}} . \tag{15}$$

Proposed Method

Input: Parity-check matrix $\mathbf{H}$, parameters used in Stern's algorithm p, l, and number of iterations r

Output: Weight distribution $\tilde{A}_0, \tilde{A}_1, \ldots, \tilde{A}_n$

Initialization: $B_0 = B_1 = \cdots = B_n = 0$

1. Perform Stern's algorithm r times, and count up B_w for all vectors $\mathbf{c}'$ found in Step4 of Stern's algorithm.

$$B_{w_h(\mathbf{c}')} := B_{w_h(\mathbf{c}')} + 1$$

2. Calculate the weight distribution by the following equation:

$$\tilde{A}_w = \frac{B_w}{r\pi_{w,p,l}} \qquad w = 1, 2, \ldots, n. \qquad \blacksquare$$

4.3 Running Cost of Stern's Algorithm for LDPC Codes

We estimate the running cost of Stern's algorithm for LDPC codes. The running cost for any linear code is given by Eq. (8). But, in the case that Stern's algorithm is applied to LDPC codes, it effects the first and the second parts of Eq. (8). Because the number of elementary row operations in Step1 of the algorithm is reduced when the Gaussian elimination is applied to sparse matrices. Note, however, that the number of ones in each row of the matrix becomes close to $\frac{n-m}{2} + 1$ through the process of the Gaussian elimination. Hence, the probability to occur the event of adding the pivoted row to eliminate the ones in the matrix through the process of the Gaussian elimination.

Assuming that the Gaussian elimination is applied to the parity-check matrix $\mathbf{H}$ whose ones are assigned to each row randomly, we consider the expected weight of the rows of $\mathbf{H}$ at each step of the Gaussian elimination. Let $w_j^{(t)}$ be a weight of the j-th row of $\mathbf{H}$ when the pivot in the forward elimination belongs to (t,t)-th component. $w_j^{(1)}$ denotes the weight of the j-th row of the given matrix $\mathbf{H}$. When the (j,t)-th component is eliminated from j-th row by adding the t-th row, the weight of the j-th row can be estimated as follows:

$$w_j^{(t+1)} = \begin{cases} w_j^{(t)} & \text{if } j \le t \\ w_j^{(t)} + \frac{w_j^{(t)}}{n-t+1}\left(w_t^{(t)} - 2 - \frac{2(w_j^{(t)}-1)(w_t^{(t)}-1)}{n-t}\right) & \text{otherwise} \end{cases} , \tag{16}$$

for $t = 1, 2, \ldots, m-1$. $\frac{w_j^{(t)}}{n-t+1}$ in Eq. (16) is a probability that there exists one in the (j,t)-th component. Since the j-th row is added to the t-th row with the probability, $w_j^{(t)}$ is added to $w_t^{(t)} - 2$ with the probability. $\frac{(w_j^{(t)}-1)(w_t^{(t)}-1)}{n-t}$ in Eq. (16) is the number of collisions of ones by the linear combination of the j-th and the t-th rows. Thus twice the number is subtracted from $w_j^{(t)}$ with probability $\frac{w_j^{(t)}}{n-t+1}$. In the forward elimination, the t-th row is swapped with other rows if the pivot is not in the (t,t)-th component. But, we can ignore the swap operations under the estimation of the weight of rows of $\mathbf{H}$, since the expected weight of the $(t+1)$-th to m-th rows is equal to the weight of t-th row.

Let $w_j'^{(t)}$ be a weight of the j-th row of $\mathbf{H}$ when the pivot in the back substitution belongs to the (t,t)-th component. $w_j'^{(m)}$ denotes the weight of the j-th row after the forward elimination, i.e., $w_j'^{(m)} = w_j^{(m)}$. When the (t,t)-th componet is sustituted for the (j,t)-th component by adding the t-th row, the weight of the j-th row can be estimated as follows:

$$w_j'^{(t-1)} = \begin{cases} w_j'^{(t)} & \text{if } j \ge t \\ w_j'^{(t)} + \frac{w_j'^{(t)}-1}{n-m+t-j}\left(w_t'^{(t)} - 2 - \frac{2(w_j'^{(t)}-2)(w_t'^{(t)}-1)}{n-m}\right) & \text{otherwise} \end{cases} , \tag{17}$$

for $t = m, m-1, \ldots, 2$. $\frac{w_j'^{(t)}-1}{n-m+t-j}$ in Eq. (17) is a probability that $\mathbf{H}$ has one in the (j,t)-th component. When the j-th row is added to the t-th row with the probability, $w_j^{(t)}$ is added to $w_t^{(t)} - 2$. Furthermore, since $\frac{(w_j'^{(t)}-2)(w_t'^{(t)}-1)}{n-m}$ ones in the t-th row collide with ones in the j-th, twice the number is subtracted from $w_j^{(t)}$.

In the forward elimination of the Gaussian elimination, when the (j,t)-th component is eliminated from the j-th row by adding the t-th row, we perform the addition of $(n-t+1)$-bit rows with probability $\frac{w_j^{(t)}}{n-t+1}$. Therefore, the number of bit operations for the forward elimination is approximately $\sum_{t=1}^{m-1}\sum_{j=t+1}^{m} w_j^{(t)}$. In the back substitution, when the (t,t)-th component is substituted for the (j,t)-th component by adding the t-th row, we perform the addition of $(n-m+1)$-bit rows with probability $\frac{w_j'^{(t)}-1}{n-m+t-j}$. Therefore, the number of bit operations for the back substitution is approximately $\sum_{t=2}^{m}\sum_{j=1}^{t-1}\frac{(w_j'^{(t)}-1)(n-m+1)}{n-m+t-j}$. Consequently, the running cost of Stern's algorithm for LDPC codes is given by

$$\Omega_{p,l} = \sum_{t=1}^{m-1}\sum_{j=t+1}^{m} w_j^{(t)} + \sum_{t=2}^{m}\sum_{j=1}^{t-1} \frac{(w_j'^{(t)} - 1)(n-m+1)}{n-m+t-j} + 2pl\binom{k/2}{p} + K\left(2^l + p\binom{k/2}{p}\right) + 2p(n-k)\frac{\binom{k/2}{p}^2}{2^l}. \quad (18)$$

4.4 Complexity of the Proposed Method

We evaluate the complexity for finding a codeword of minimum weight of several LDPC codes by the proposed method, and show the optimal parameters p and l used in the proposed method. Using the proposed method with the optimal parameters, we can efficiently generate low-weight codewords many times, and can accurately evaluate the weight distribution for the low-weight codewords.

The probability to find a codeword of weight w is $\pi_{w,p,l}$, so the number of iterations for finding the codeword is approximately $\frac{1}{\pi_{w,p,l}}$. Therefore, the complexity for finding each codeword of weight w is given by

$$W_{w,p,l} = \frac{\Omega_{p,l}}{\pi_{w,p,l}}. \quad (19)$$

We call $W_{w,p,l}$ a work factor of the proposed method. Since $\Omega_{p,l}$ and $\pi_{w,p,l}$ can be explicitly calculated by Eqs. (6) and (18), we can determine the parameters p and l which minimize the work factor when the size of code and the targeted weight w are given. Thus, we can efficiently find codewords of weight w using the proposed method with the optimal parameters.

Now we evaluate the work factors $W_{w,p,l}$ to find a minimum-weight codeword of the following widely-known LDPC codes:

- C_1: MacKay's high-rate regular $(495, 433)$ code [29]
- C_2: MacKay's $(3, 6)$-regular $(504, 252)$ code [29]
- C_3: MacKay's $(3, 6)$-regular $(1008, 504)$ code [29]
- C_4: Ramanujan-Margulis $(17, 5)$ code with length 4896 and dimension 2474 [30]

where C_1, C_2, C_3 and C_4 are labeled as 495.62.3.2915, 252.252.3.252, 504.504.3.504 and ram17.5 in [29] respectively. The numerical results of evaluating the minimum weight of these codes have shown in [18]. The results have reported that the minimum weight of C_1, C_2, C_3 and C_4 is 4, 20, 32 and 24 respectively. We show the work factor for finding these minimum-weight codewords by the proposed method in Table 1.

As you can see Table 1, the probability $\pi_{w,p,l}$ becomes larger as the parameter p is larger, although the running cost $\Omega_{p,l}$ becomes larger. Furthermore, the probability $\pi_{w,p,l}$ becomes larger as the parameter l is larger, but a certain parameter l minimizes the running cost $\Omega_{p,l}$. Thus we minimize the work factor $W_{w,p,l}$ by optimizing the parameters p and l.

Table 1 Work factor for finding a minimum-weight codeword in MacKay's $(495, 433)$ code C_1, $(504, 252)$ code C_2, $(1008, 504)$ code C_3 and $(4896, 2474)$ Ramanujan-Margulis code C_4.

code n, m, k	weight w	parameters p, l	probability $\pi_{w,p,l}$	running cost $\Omega_{p,l}$	work factor $W_{w,p,l}$
C_1	4	$p=1, l=5$	3.110×10^{-2}	$2^{19.58}$	$2^{24.58}$
$n=495$		$p=1, l=6$	3.003×10^{-2}	$2^{19.40}$	$2^{24.46}$
$m=62$		$p=1, l=7$	2.897×10^{-2}	$2^{19.31}$	$2^{24.42}$
$k=433$		$p=1, l=8$	2.794×10^{-2}	$2^{19.28}$	$2^{24.44}$
		$p=1, l=9$	2.692×10^{-2}	$2^{19.29}$	$2^{24.51}$
		$p=2, l=13$	2.184×10^{-1}	$2^{24.38}$	$2^{26.58}$
		$p=2, l=14$	2.184×10^{-1}	$2^{23.76}$	$2^{25.95}$
		$p=2, l=15$	2.184×10^{-1}	$2^{23.42}$	$2^{25.61}$
		$p=2, l=16$	2.184×10^{-1}	$2^{23.43}$	$2^{25.63}$
		$p=2, l=17$	2.184×10^{-1}	$2^{23.80}$	$2^{25.99}$
C_2	20	$p=1, l=2$	6.145×10^{-5}	$2^{22.81}$	$2^{36.80}$
$n=504$		$p=1, l=3$	5.702×10^{-5}	$2^{22.60}$	$2^{36.70}$
$m=252$		$p=1, l=4$	5.290×10^{-5}	$2^{22.49}$	$2^{36.69}$
$k=252$		$p=1, l=5$	4.906×10^{-5}	$2^{22.42}$	$2^{36.74}$
		$p=1, l=6$	4.549×10^{-5}	$2^{22.39}$	$2^{36.82}$
		$p=2, l=13$	6.408×10^{-4}	$2^{23.83}$	$2^{34.44}$
		$p=2, l=14$	5.979×10^{-4}	$2^{23.48}$	$2^{34.18}$
		$p=2, l=15$	5.577×10^{-4}	$2^{23.37}$	$2^{34.18}$
		$p=2, l=16$	5.200×10^{-4}	$2^{23.52}$	$2^{34.43}$
		$p=2, l=17$	4.848×10^{-4}	$2^{23.91}$	$2^{34.92}$
C_3	32	$p=1, l=2$	3.481×10^{-8}	$2^{25.81}$	$2^{50.59}$
$n=1008$		$p=1, l=3$	3.273×10^{-8}	$2^{25.60}$	$2^{50.46}$
$m=504$		$p=1, l=4$	3.077×10^{-8}	$2^{25.48}$	$2^{50.43}$
$k=504$		$p=1, l=5$	2.892×10^{-8}	$2^{25.42}$	$2^{50.46}$
		$p=1, l=6$	2.719×10^{-8}	$2^{25.38}$	$2^{50.52}$
		$p=2, l=15$	9.995×10^{-7}	$2^{26.74}$	$2^{46.68}$
		$p=2, l=16$	9.423×10^{-7}	$2^{26.32}$	$2^{46.34}$
		$p=2, l=17$	8.882×10^{-7}	$2^{26.12}$	$2^{46.22}$
		$p=2, l=18$	8.372×10^{-7}	$2^{26.13}$	$2^{46.32}$
		$p=2, l=19$	7.889×10^{-7}	$2^{26.37}$	$2^{46.64}$
C_4	24	$p=1, l=5$	6.102×10^{-6}	$2^{32.14}$	$2^{49.46}$
$n=4896$		$p=1, l=6$	6.046×10^{-6}	$2^{32.10}$	$2^{49.44}$
$m=2448$		$p=1, l=7$	5.991×10^{-6}	$2^{32.08}$	$2^{49.43}$
$k=2474$		$p=1, l=8$	5.937×10^{-6}	$2^{32.07}$	$2^{49.44}$
		$p=1, l=9$	5.883×10^{-6}	$2^{32.07}$	$2^{49.44}$
		$p=2, l=21$	1.641×10^{-4}	$2^{32.80}$	$2^{45.38}$
		$p=2, l=22$	1.627×10^{-4}	$2^{32.55}$	$2^{45.13}$
		$p=2, l=23$	1.613×10^{-4}	$2^{32.45}$	$2^{45.05}$
		$p=2, l=24$	1.600×10^{-4}	$2^{32.50}$	$2^{45.11}$
		$p=2, l=25$	1.587×10^{-4}	$2^{32.70}$	$2^{45.32}$

In Table 1, the parameters $p = 1$ and $l = 7$ minimize the work factor $W_{4,p,l}$ for $(495, 433)$ code C_1 where $K = 64$. Thus we understand that the proposed method can generate a codeword of weight 4 in C_1 with complexity $2^{24.42}$. Furthermore, the work factor $W_{20,p,l}$ is $2^{34.18}$ when we find a codeword of weight 20 in $(504, 252)$ code C_2 using the

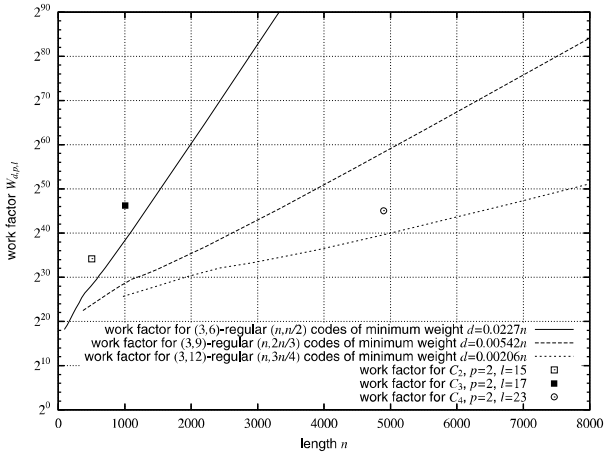

Fig. 1 Work factor for finding a minimum-weight codeword of $(n, n/2)$, $(n, \frac{2}{3}n)$ and $(n, \frac{3}{4}n)$ codes by the proposed method with optimized parameters.

proposed method with $p = 2$ and $l = 15$, the work factor $W_{32,p,l}$ is $2^{46.22}$ when we find a codeword of weight 32 in $(1008, 504)$ code C_3 using the proposed method with $p = 2$ and $l = 17$, and the work factor $W_{24,p,l}$ is $2^{45.05}$ when we find a codeword of weight 24 in $(4896, 2474)$ code C_4 using the proposed method with $p = 2$ and $l = 23$. These work factors are as small as we can compute in actual time. In fact, we can compute the problem with complexity 2^{50} in 300 hours using current computers. Thus we can find the low-weight codewords by using the proposed method with the optimal parameters, and can compute the low part of the weight distribution of these codes.

Next, we show the relation between the work factor $W_{d,p,l}$ and the length n for some codes of minimum weight d. We assume that these codes are $(3, 6)$, $(3, 9)$ and $(3, 12)$-regular Gallager's LDPC codes with typical minimum distance $0.0227n$, $0.00542n$ and $0.00206n$ respectively. Figure 1 shows the work factor $W_{d,p,l}$ of finding a codeword of minimum weight d by the proposed method with optimized parameters p and l. In Fig. 1, the work factor $W_{d,p,l}$ is larger as the length n is longer. However, we confirm that the proposed method can find a codeword of minimum weight $d = 0.0227n$ of $(n, n/2)$ codes within work factor 2^{50} if $n \leq 1500$. Furthermore, within work factor 2^{50}, the proposed method can find a codeword of minimum weight $d = 0.00542n$ of $(n, \frac{2}{3}n)$ codes if $n \leq 3900$, and can find a codeword of minimum weight $d = 0.00206n$ of $(n, \frac{3}{4}n)$ codes if $n \leq 7700$. The work factors $W_{d,p,l}$ for C_2 and C_3 are larger than that of $(n, n/2)$ codes in Fig. 1 since the minimum distance of C_2 and C_3 is larger than the typical minimum distance of $(n, n/2)$ codes.

5. Numerical Experiments

We computed the weight distribution for low-weight codewords of MacKay's $(495, 433)$ code C_1, $(504, 252)$ code C_2, $(1008, 504)$ code C_3, and $(4896, 2474)$ Ramanujan-Margulis code C_4 using the proposed method. Table 2 shows results of the weight distribution obtained by applying our method to C_1, C_2, C_3 and C_4.

The parameters p and l were used as the optimal parameters shown in Sect. 4.3 and the number of iterations r is set as follows:

- C_1: $p = 1, l = 7, r = 1.0 \times 10^7$
- C_2: $p = 2, l = 15, r = 1.0 \times 10^7$
- C_3: $p = 2, l = 17, r = 5.0 \times 10^7$
- C_4: $p = 2, l = 23, r = 1.0 \times 10^6$

We used the computers of CPU Core 2 Duo E6700 2.66 GHz and RAM 2 GB, and it took 2.4 hours, 15 hours, 543 hours and 1024 hours to obtain results of C_1, C_2, C_3 and C_4 respectively. In these numerical experiments, we recorded all the distinct low-weight codewords found by the proposed method. In Table 2, B'_w denotes the number of distinct codewords of weight w among B_w codewords. Since C_1, C_2, C_3 and C_4 contain at least B'_w codewords of weight w, we understand that the exact weight distribution A_w is not less than B'_w. On the other hand, in [18] and [22], the minimum distance and the weight distribution of the same codes are evaluated by the NNCS approach. These are shown as A_w^{NNCS} in Table 2.

As you can see Table 2, for C_3, we found a codeword of weight 30 which have not been found by the NNCS approach. Although the minimum distance of C_3 has been reported to be 32 in [22], it was newly confirmed that it was not 32 but 30 in the result of reestimate by our method. For C_1, C_2 and C_4, the minimum distance computed by our method is equal to that estimated in [18] and [22]. But, since we evaluated the weight distribution by our method which based on Stern's algorithm, we can determine that the exact minimum distance is equal to the lowest weight shown in Table 2 with the failure probability given by Eq. (7). Thus we shall say that the minimum weight of C_1, C_2, C_3 and C_4 is the lowest weight shown in Table 2 with the following failure probabilities:

- C_1: $\varepsilon = (1 - \pi_{4,1,7})^{10000000} = 3.232 \times 10^{-12447}$
- C_2: $\varepsilon = (1 - \pi_{20,2,15})^{10000000} = 1.701 \times 10^{-2483}$
- C_3: $\varepsilon = (1 - \pi_{32,2,17})^{50000000} = 1.893 \times 10^{-68}$
- C_4: $\varepsilon = (1 - \pi_{24,2,23})^{1000000} = 4.945 \times 10^{-71}$

We discuss the accuracy of the computed weight distribution $\tilde{A}_w$. In Table 2, $\tilde{A}_w$ is almost equal to B'_w for low weight w. When we obtain $B'_w \ll B_w$ after many iterations of Stern's algorithm, we expect that almost all codewords of weight w were found and that B'_w is almost equal to the exact weight distribution A_w. Thus, we confirm that $\tilde{A}_w$ is approximately equal to A_w for low weight in Table 2. On the other hand, there is some difference between $\tilde{A}_w$ and B'_w except for low weight w. But it is smaller than difference between A_w^{NNCS} and B'_w. We expect that B'_w is almost equal to A_w when $B'_w \ll B_w$ and that $\tilde{A}_w$ is closer to A_w than A_w^{NNCS}. We have given the evaluation of the reliability of our computed weight distribution in [31]. In addition, we discuss the accuracy of the computed weight distribution of the following irregular LDPC codes:

Table 2 Weight distribution for low-weight codewords of LDPC codes computed by the proposed method and the conventional methods.

MacKay's (495, 433) code C_1

	Proposed method			NNCS approach	
				[18]	with 2-bit noise pattern [22]
w	$\pi_{w,1,7}$	B_w (B'_w)	$\tilde{A}_w$	A_w^{NNCS}	A_w^{NNCS}
4	2.914×10^{-2}	16634711 (60)	59	60	60
6	8.666×10^{-4}	32446849 (4439)	3992	320	4439
8	1.798×10^{-5}	77477469 (597947)	475969	1157	92286
10	2.991×10^{-7}	173568445 (79584904)	66490530	4074	472606

MacKay's (504, 252) code C_2

	Proposed method			NNCS approach	
				[18]	with 2-bit noise pattern [22]
w	$\pi_{w,2,15}$	B_w (B'_w)	$\tilde{A}_w$	A_w^{NNCS}	A_w^{NNCS}
20	5.715×10^{-4}	9992 (2)	2	2	2
22	1.802×10^{-4}	32549 (22)	18	—	18
24	5.413×10^{-5}	48589 (117)	90	—	66
26	1.559×10^{-5}	54387 (481)	349	—	190
28	4.327×10^{-6}	77096 (2587)	1782	—	526

MacKay's (1008, 504) code C_3

	Proposed method			NNCS approach	
				[18]	with 2-bit noise pattern [22]
w	$\pi_{w,2,17}$	B_w (B'_w)	$\tilde{A}_w$	A_w^{NNCS}	A_w^{NNCS}
30	3.119×10^{-6}	116 (1)	1	0	0
32	9.101×10^{-7}	30 (1)	1	0	2
34	2.598×10^{-7}	86 (11)	7	1	2
36	7.272×10^{-8}	99 (39)	27	—	11
38	1.998×10^{-8}	84 (64)	84	—	16

(4896, 2474) Ramanujan-Margulis code C_4

	Proposed method			NNCS approach [18]	Modified EI algorithm [23]
w	$\pi_{w,2,23}$	B_w (B'_w)	$\tilde{A}_w$	A_w^{NNCS}	A_w^{MEI}
24	1.619×10^{-4}	25443 (204)	157	204	198

- C_5: Irregular (504, 252) PEG code [32]
- C_6: Irregular (1008, 504) PEG code [32]

where C_5 and C_6 are labeled as PEGirReg252x504 and PEGirReg504x1008 in [29] respectively. Table 3 shows the weight distribution of these codes computed by the proposed method. In the numerical experiments for C_5 and C_6, we used the same parameters p and l as the optimal parameters for C_2 and C_3, and set $r = 1.0 \times 10^7$. In Table 3, $\tilde{A}_w$ is almost equal to B'_w for low weight w. We expect that B'_w is almost equal to the exact weight distribution A_w because of $B'_w \ll B_w$. Therefore, we obtained the computed weight distribution $\tilde{A}_w$ which is approximately equal to the exact weight distribution A_w for low-weight codewords of C_5 and C_6. On the other hand, in Tables 2 and 3, since the number of codewords of weight w is larger as weight w is larger, it is infeasible to find all codewords of the weight. Then we cannot evaluate the weight distribution from B'_w. However, we can approximately evaluate the weight distribution by $\tilde{A}_w$ computed by Eq. (15).

6. Conclusion

In this paper, we proposed a probabilistic method for computing the weight distribution of LDPC codes. The proposed method efficiently finds low-weight codewords in a given LDPC code by using Stern's algorithm, and stochastically computes the low-part weight distribution from the relation between the number of codewords with a given weight and the rate of generating the codewords in Stern's algorithm. Our method enables to compute the low-part weight distribution of LDPC codes with the widely-used random constructions, since such codes contain few low-weight codewords and our method can find the codewords for many iterations. Even if the code contains many codewords of the given weight, our method can approximately evaluate the weight distribution.

Table 3 The weight distribution for low-weight codewords of irregular LDPC codes computed by the proposed method.

Irregular (504, 252) PEG code C_5

w	$\pi_{w,2,15}$	B_w (B'_w)	$\tilde{A}_w$
13	1.877×10^{-2}	184820 (1)	1
14	1.223×10^{-2}	119853 (1)	1
15	7.742×10^{-3}	377117 (5)	5
16	4.780×10^{-3}	510167 (11)	11
17	2.888×10^{-3}	445115 (16)	15
18	1.712×10^{-3}	460350 (28)	27
19	9.968×10^{-4}	569941 (60)	57
20	5.715×10^{-4}	654276 (121)	114

Irregular (1008, 504) PEG code C_6

w	$\pi_{w,2,17}$	B_w (B'_w)	$\tilde{A}_w$
13	2.404×10^{-2}	236934 (1)	1
16	6.784×10^{-3}	197852 (3)	3
17	4.252×10^{-3}	123029 (3)	3
18	2.618×10^{-3}	100835 (4)	4
19	1.586×10^{-3}	45508 (3)	3
20	9.469×10^{-4}	80826 (9)	9
21	5.582×10^{-4}	58089 (11)	10
22	3.252×10^{-4}	112956 (37)	35

We computed the weight distribution of MacKay's (495, 433), (504, 252), (1008, 504) codes, irregular (504, 252), (1008, 504) PEG codes, and (4896, 2474) Ramanujan-Margulis code by using the proposed method. In these results, for MacKay's (1008, 504) code, we found a minimum-weight codeword which has not been reported in [18], [22] and [23]. Additionally, for other codes, we found a lot of low-weight codewords which have not been found by the NNCS approach and the modified EI algorithm.

Our method can give good approximate weight distribution for low-weight codewords of LDPC codes. It can be used for the evaluation of the ML decoding performance of LDPC codes, and is effective in the performance evaluation of LDPC codes.

References

[1] R.G. Gallager, Low Density Parity Check Codes, MIT Press, Cambridge, 1963.
[2] D.J.C. MacKay, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inf. Theory, vol.45, no.2, pp.399–431, March 1999.
[3] G. Miller and D. Burshtein, "Bounds on the maximum-likelihood decoding error probability of low-density parity-check codes," IEEE Trans. Inf. Theory, vol.47, no.7, pp.2696–2710, Nov. 2001.
[4] R. Ikegaya, K. Kasai, T. Shibuya, and K. Sakaniwa, "Performance of standard irregular LDPC codes under maximum likelihood decoding," IEICE Trans. Fundamentals, vol.E90-A, no.7, pp.1432–1443, July 2007.
[5] S. Litsyn and V. Shevelev, "On ensembles of low-density parity-check codes: Asymptotic distance distributions," IEEE Trans. Inf. Theory, vol.48, no.4, pp.887–908, April 2002.
[6] S. Litsyn and V. Shevelev, "Distance distributions in ensembles of irregular low-density parity-check codes," IEEE Trans. Inf. Theory, vol.49, no.12, pp.3140–3159, Dec. 2003.
[7] C. Di, T.J. Richardson, and R.L. Urbanke, "Weight distribution of low-density parity-check codes," IEEE Trans. Inf. Theory, vol.52, no.11, pp.4839–4855, Nov. 2006.
[8] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems," IEEE Trans. Inf. Theory, vol.24, no.3, pp.384–386, May 1978.
[9] A. Vardy, "The intractability of computing the minimum distance of a code," IEEE Trans. Inf. Theory, vol.43, no.6, pp.1757–1766, Nov. 1997.
[10] R.M. Tanner, "Minimum-distance bounds by graph analysis," IEEE Trans. Inf. Theory, vol.47, no.2, pp.808–821, Feb. 2001.
[11] D.J.C. MacKay and M. Daver, "Evaluation of Gallager codes for short block length and high rate applications," Codes, Systems, and Graphical Models, ed. B. Marcus and J. Rosenthal, The IMA volumes in Mathematics and its Applications, vol.123, pp.113–130, Springer-Verlag, New York, 2001.
[12] M.P.C. Fossorier, "Quasi-cyclic low density parity check codes from circulant permutation matrices," IEEE Trans. Inf. Theory, vol.50, no.8, pp.1788–1793, Aug. 2004.
[13] T. Mittelholzer, "Efficient encoding and minimum distance bounds of Reed-Solomon-type array codes," Proc. 2002 IEEE Int. Symposium on Information Theory, p.282, 2002.
[14] K. Yang and T. Helleseth, "On the minimum distance of array codes as LDPC codes," IEEE Trans. Inf. Theory, vol.49, no.12, pp.3268–3271, Dec. 2003.
[15] K. Sugiyama and Y. Kaji, "On the minimum weight of simple full-length array LDPC codes," IEICE Trans. Fundamentals, vol.E91-A, no.6, pp.1502–1508, June 2008.
[16] Y. Kaji, "On the number of minimum weight codewords of SFA-LDPC codes with column weight three and five," Proc. 31th Symposium on Information Theory and Its Applications, pp.1–6, Oct. 2008.
[17] Y. Kou, S. Lin, and M.P.C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," IEEE Trans. Inf. Theory, vol.47, no.7, pp.2711–2736, Nov. 2001.
[18] X.-Y. Hu, M.P.C. Fossorier, and E. Eleftheriou, "On the computation of the minimum distance of low-density parity-check codes," Proc. 2004 IEEE Int. Conference on Communications, pp.767–771, June 2004.
[19] X.-Y. Hu, M.P.C. Fossorier, and E. Eleftheriou, "Approximate algorithms for computing the minimum distance of low-density parity-check codes," Proc. 2004 IEEE Int. Symposium on Information Theory, p.475, Chicago, USA, June/July 2004.
[20] C. Berrou, S. Vaton, M. Jézéquel, and C. Douillard, "Computing the minimum distance of linear codes by the error impulse method," Proc. IEEE Global Telecommunications Conference 2002, pp.1017–1020, Taiwan, R.O.C., Nov. 2002.
[21] M.P.C. Fossorier, "Iterative reliability-based decoding of low-density parity-check codes," IEEE J. Sel. Areas Commun., vol.19, no.5, pp.908–917, May 2001.
[22] C. Tissieres, Exploring algorithms for finding low-weight codewords, Diploma Thesis, Univ. of Hawaii and EPFL Lausanne, Sept. 2004.
[23] F. Daneshgaran, M. Laddomada, and M. Mondin, "An algorithm for the computation of the minimum distance of LDPC codes," European Trans. Telecommunications, vol.17, no.1, pp.57–62, Jan. 2006.
[24] M. Hirotomo, M. Mohri, and M. Morii, "A probabilistic computation method for the weight distribution of low-density parity-check codes," Proc. 2005 IEEE Int. Symposium on Information Theory, pp.2166–2170, Adelaide, Australia, Sept. 2005.
[25] J.S. Leon, "A probabilistic algorithm for computing minimum weights of large error-correcting codes," IEEE Trans. Inf. Theory, vol.34, no.5, pp.1354–1359, Sept. 1988.
[26] J. Stern, "A method for finding codewords of small weight," Coding Theory and Applications, ed. G. Cohen and J. Wolfmann, LNCS 388, pp.106–113, Springer-Verlag, New York, 1989.

[27] A. Canteaut and F. Chabaud, "A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511," IEEE Trans. Inf. Theory, vol.44, no.1, pp.367–378, Jan. 1998.
[28] F. Chabaud, "On the security of some cryptosystems based on error-correcting codes," Advances in Cryptology — EUROCRYPT'94, ed. A. De Santis, LNCS 950, pp.131–139, Springer-Verlag, New York, 1994.
[29] D.J.C. MacKay, Encyclopedia of sparse graph codes, http://www.inference.phy.cam.ac.uk/mackay/codes/data.html
[30] J. Rosenthal and P.O. Vontobel, "Construction of LDPC codes based on Ramanujan graphs and ideas from Margulis," Proc. 38th Annual Allerton Conf. on Communication, Control, and Computing, pp.248–257, Monticello, IL, Oct. 2000.
[31] M. Hirotomo, M. Mohri, and M. Morii, "On the capability of the probabilistic method for computing the weight distribution of LDPC codes," Proc. 30th Symposium on Information Theory and its Applications, pp.556–561, Nov. 2007.
[32] X.-Y. Hu, E. Eleftheriou, and D.M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," IEEE Trans. Inf. Theory, vol.51, no.1, pp.386–398, Jan. 2005.
[33] T. Richardson, "Error floors of LDPC codes," Proc. 41th Annual Allerton Conf. on Communication, Control, and Computing, pp.1426–1435, Monticello, IL, Oct. 2003.
[34] C.A. Cole, S.G. Wilson, E.K. Hall, and T.R. Giallorenzi, "A general method for finding low error rates of LDPC codes," Available: http://arxiv.org/abs/cs.IT/0605051

Appendix: Block Error Probability of C_1, C_2, C_3, C_4, C_5 and C_6

We evaluate decoding error probabilities of MacKay's $(495, 433)$ code C_1, $(504, 252)$ code C_2, $(1008, 504)$ code C_3, $(4896, 2474)$ Ramanujan-Margulis code C_4, irregular $(504, 252)$ PEG code C_5 and irregular $(1008, 504)$ PEG code C_6 using the union bound computed from the weight distribution shown in Table 2. The purpose is to compare the decoding error probability estimated by the weight distribution obtained by our method with by the conventional methods. The union bound is well known that the block error probability of a linear block code of rate $R = \frac{k}{n}$ with maximum-likelihood decoding in AWGN channel, and it can be upper-bounded by the following expression:

$$P_e \leq \frac{1}{2} \sum_{w=d}^{n} A_w Q\left(\sqrt{\frac{2wRE_b}{N_0}} \right), \qquad (\text{A}\cdot 1)$$

where $Q(\cdot)$ is the Gaussian integral function defined as $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{z^2}{2}} dz$. The union bound may not be the only way to estimate the performance under iterative decoding at low error rate, because the performance is discussed by pseudo-codewords and trapping sets [33]. But it still is a useful tool as it may suggest error floors.

Figures A·1, A·2, A·3, A·4, A·5 and A·6 show union bounds on block error probabilities of C_1, C_2, C_3, C_4, C_5 and C_6 respectively. These are computed from the weight distribution $\tilde{A}_w$, A_w^{NNCS} and A_w^{MEI} shown in Tables 2 and 3. In addition, the block error probabilities of C_1, C_2, C_3, C_4, C_5 and C_6 simulated by the sum-product decoder are shown in Figs. A·1–A·6. In Figs. A·1, A·2 and A·4, the union

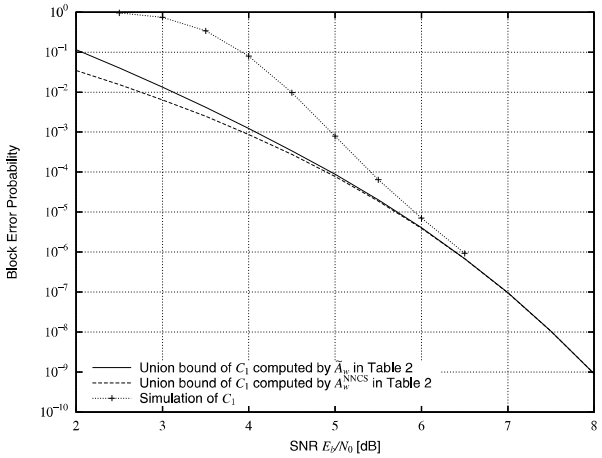

Fig. A·1 The union bound computed from the weight distribution shown in Table 2 and the block error probability of simulated performance of MacKay's $(495, 433)$ code C_1.

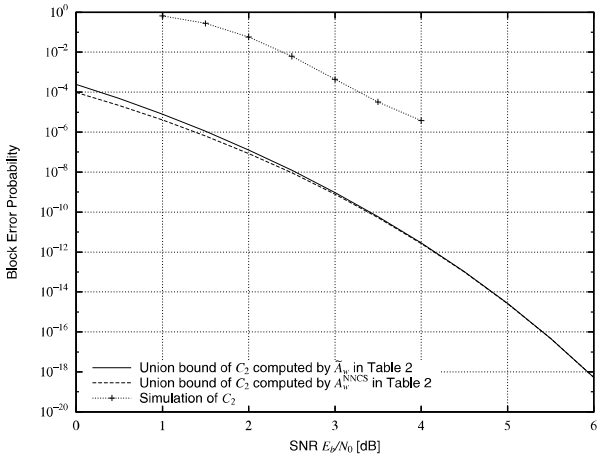

Fig. A·2 The union bound computed from the weight distribution shown in Table 2 and the block error probability of simulated performance of MacKay's $(504, 252)$ code C_2.

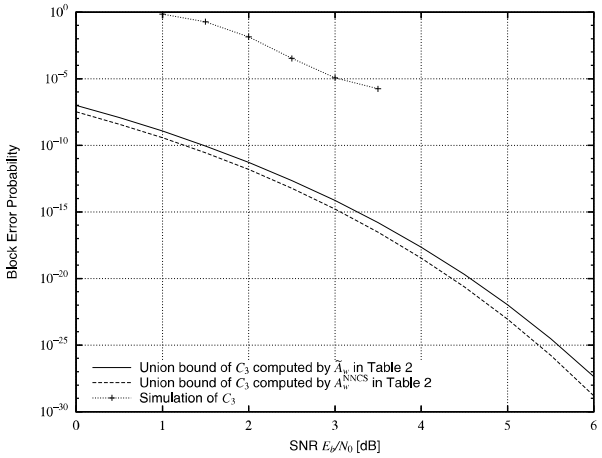

Fig. A·3 The union bound computed from the weight distribution shown in Table 2 and the block error probability of simulated performance of MacKay's $(1008, 504)$ code C_3.

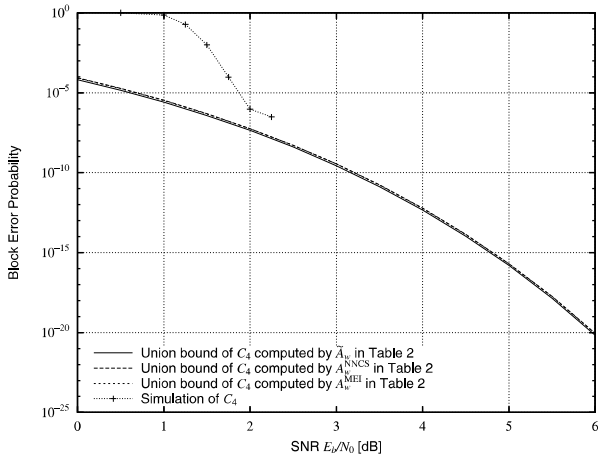

Fig. A· 4 The union bound computed from the weight distribution shown in Table 2 and the block error probability of simulated performance of (4896, 2474) Ramanujan-Margulis code C_4.

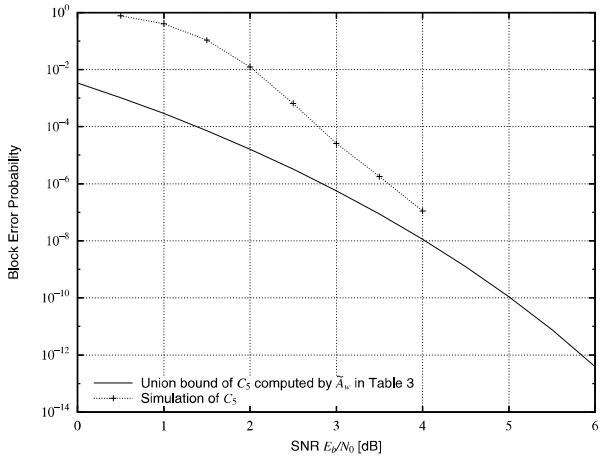

Fig. A· 5 The union bound computed from the weight distribution shown in Table 3 and the block error probability of simulated performance of irregular (504, 252) PEG code C_5.

bound computed from $\tilde{A}_w$ is almost equal to that computed by A_w^{NNCS} and A_w^{MEI}, since the weight distribution computed by the proposed method $\tilde{A}_w$ is almost equal to that computed by the NNCS approach A_w^{NNCS} and the modified EI algorithm A_w^{MEI} for low weight w. On the other hand, in Fig. A· 3, there is some difference between the error probability computed by $\tilde{A}_w$ and that computed by A_w^{NNCS}, since though our method could find a codeword of weight 30, the NNCS approach could not find the codeword. Additionally, the simulated error probabilities converge the union bound at high SNR region in Figs. A· 1, A· 2, A· 5 and A· 6. But, in Figs. A· 2 and A· 3, there is difference between the simulated error probabilities and the union bound, since the performance of LDPC codes decoded by iterative algorithms may have influence on pseudo-codeword and trapping sets of C_2 and C_3 [34].

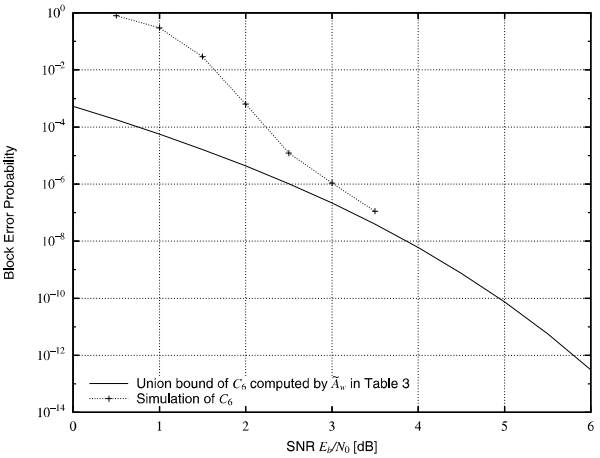

Fig. A· 6 The union bound computed from the weight distribution shown in Table 3 and the block error probability of simulated performance of irregular (1008, 504) PEG code C_6.

Masanori Hirotomo received the B.E., M.E. and D.E. degrees in information engineering from the University of Tokushima, Japan, in 2000, 2002, and 2006 respectively. From 2005 to 2006 he was a Research Associate at the Department of Intelligent Systems and Information Science, Faculty of Engineering, the University of Tokushima, Japan. From 2006 to 2008 he was a Researcher at the Hyogo Institute of Information Education Foundation, Japan. Since 2008, he has been an Assistant Professor at the Graduate School of Engineering, Kobe University, Japan. His research interests are in coding theory and information security. He is a member of the IEEE and the Society of Information Theory and Its Applications.

Masami Mohri received the B.E. degree in Information Science and the M.E. degree in Information Science from Ehime University, Ehime, Japan, in 1993 and 1995 respectively. And she received the Ph.D. degree in Information Science and Intelligent Systems from the University of Tokushima, Tokushima, Japan, in 2002. From 1995 to 1998 she was an assistant professor at the Department of Management and Information Science, Kagawa junior college, Japan. From 1998 to 2002 she was a research associator of the Department of Intelligent Systems and Information Science, Faculty of Engineering at the University of Tokushima, Japan. From 2003 to 2008 she was a lecturer of the Department of Intelligent Systems and Information Science, at the University of Tokushima, Japan. Since 2008, she has been an associate professor at the Information and Multimedia Center, Gifu University, Japan. Her research interests are in coding theory and network security. She is a member of the IEEE and the Society of Information Theory and Its Applications.

Masakatu Morii received the B.E. degree in electrical engineering and the M.E. degree in electronics engineering from Saga University, Saga, Japan, and the D.E. degree in communication engineering from Osaka University, Osaka, Japan, in 1983, 1985, and 1989, respectively. From 1989 to 1990 he was an Instructor in the Department of Electronics and Information Science, Kyoto Institute of Technology, Japan. From 1990 to 1995 he was an Associate Professor at the Department of Computer Science, Faculty of Engineering at Ehime University, Japan. From 1995 to 2005 he was a Professor at the Department of Intelligent Systems and Information Science, Faculty of Engineering at the University of Tokushima, Japan. Since 2005, he has been a Professor at the Department of Electrical and Electronics Engineering, Faculty of Engineering at the Kobe University, Japan. His research interests are in error correcting codes, cryptography, discrete mathematics and computer networks and information security. Dr. Morii is a member of the IEEE, the Information Processing Society of Japan and the Society of Information Theory and Its Applications.