Watanabe *et al. J Wireless Com Network* (2021) 2021:22
https://doi.org/10.1186/s13638-021-01901-x

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Shortened LDPC codes accelerate OSD decoding performance

Kohtaro Watanabe*, Ryusei Kaguchi and Toshiya Shinoda

*Correspondence:
wata@nda.ac.jp
Department of Computer Science, National Defense Academy of Japan, Hashirimizu, Yokosuka 2398686, Japan

Abstract

Medium-length LDPC codes are in demand in certain areas such as mobile environment (Wi-Fi and Mobile WiMAX) and in telecommand links from the ground to space because of their lower latency properties. However, because of the length of these codes is rather short, decoding error rates are worse than those of long-length codes. In this paper, we show that the combination of shortened LDPC codes, whose shortened positions are properly selected, and ordered statistic decoding (OSD) significantly improves the decoding error. For the best choice of shortening positions, we used the integer programming approach. In particular, we utilized Feldman–Wainwright–Karger code polytope for this purpose. Some studies have independently reported the efficiency of shortened LDPC codes and OSD methods. This paper emphasizes that their combination results in multiplicative effectiveness.

Keywords: LDPC codes, OSD method, Shortened code, Code polytope, Integer programming

1 Introduction

It is well known that long-length LDPC codes with the sum-product (SP) decoding algorithm show good decoding error properties near the Shannon limit; see sections 17.10–17.19 of [13]. However, medium-length LDPC codes (whose length are a hundred to a few thousand bits) are adopted in some actual communication standards, such as IEEE 802.11n (Wi-Fi), 802.16e (WiMAX) and telecommand (TC) links from ground to space [4, 5] because of their lower latency properties. Unfortunately, the error correction abilities of these medium-length codes are inferior to that of long-length codes. As a method that compensates for this gap, the Ordered Statistic Decoding (OSD) method [8] has been considered effective. This method uses outputs of SP decoding as reliability measure and reprocesses them to improve the output quality. However, for each reprocessing phase i, the method requires $\binom{k}{i}$ codewords to be processed, where k is the information bit length. Thus, up to phase p, a total number of $\sum_{i=0}^{p}\binom{k}{i}$ codewords will be generated. This procedure is called "Order-p reprocessing". Thus, in practice, the reprocessing procedure is limited to a relatively small number, for example at most order 4; see [2]. Due to this limitation on the reprocessing order, improvements from SP

decoding are not large enough as expected, especially if the order of reprocessing is small, such as $p = 1$ or 2. To resolve these contradictory requirements on decoding precision and decoding time, we propose a method, that is a concatenation of the iterative decoding method (SP) and OSD method. It is known that shortened LDPC codes improve the decoding error rate to a certain extent, e.g., see [16, 17] and references therein. Thus, by choosing shortened LDPC codes in some appropriate way, the number of erroneous bits in most reliable bits (MRB) are expected to be reduced, and this decreasing of errors in the MRB improves the total decoding error significantly even in the adoption of "Order-1" or "Order-2" reprocessing. Here, we used the term "-appropriately shortened LDPC codes-" for LDPC codes whose shortening positions were selected to decrease the decoding error. We compared three methods for the selection of shortening positions, namely (A): Every 8 bits, (B): Worst reliable bits [16], (C): Integer Programming based approach. We note that in method (C), the Feldman–Wainwright–Karger code polytope [7] is used for demonstrating integer programming (IP) procedure to enumerate a set of codewords with small Hamming weight. We exclude these codewords by suitably choosing shortening positions and for this choice of shortening positions, IP again plays an important role. To the best of our knowledge, the approach (C) is new in our current paper.

For the decoding time, it was suppressed to be reasonable, because we assumed a relatively small-order reprocessing scheme. We examine this in Sect. 6 with numerical experiments.

For an error-correcting performance, by appropriately choosing shortening positions of information bits, we found that Order-2 reprocessing is enough to achieve a codeword error rate (CER) of less than 10^{-5} at a signal-to-noise ratio $E_b/N_0 = 3.0$ dB. To be specific, the followings accomplish this property:

(1) IEEE 802.16e (WiMAX) LDPC code with $(n, k) = (576, 288)$ shortened 36 bits,
(2) IEEE 802.11n (Wi-Fi) LDPC code with $(n, k) = (648, 324)$ shortened 40 and 36 bits,
(3) Consultative Committee for Space Data System (CCSDS) TC link LDPC code with $(n, k) = (512, 256)$ shortened 32 bits,

where n is the code length and k the information bit length. We note that the codeword error rate level in TC links is required to be lower than 10^{-5}; see Baldi et al. [2]. Of three methods for the determination of shortening positions, method (C) seems to be effective especially in the case of a relatively high signal-to-noise ratio range.

Finally, we note that the method we are proposing here is a sort of concatenation of SP and OSD methods, analogous to [2]. Hence, further concatenation with a cyclic redundancy check (CRC) as [9, 14] may be possible by decreasing the number of shortening bits and using them for CRC.

This paper is organized as follows. In Sect. 3, we review the shortened LDPC code, the OSD method, and their concatenation. In Sect. 4, we propose three different methods for shortening positions. We evaluate these methods through a series of numerical experiments in Sect. 5. Section 6 discusses our evaluations from the execution time. Throughout the experiments, we assume relatively small order OSD, while keeping the ability of error correction at some satisfactory level. Thus, the execution time is also

relatively small in these OSD decoding class. Finally, in Sect. 7 we discuss some topics which we leave for future research.

2 Methods

We have examined the effectiveness of an encoding/decoding method via computer simulations, which were performed on an Intel(R) Xeon(R) E5-1660 3.70GHz processor host using gcc 4.4.7 -O3.

3 Shortened LDPC codes and the OSD method

In this section we review the shortened LDPC codes with SP decoding and the OSD method, and then we look for the way to combine them into a single coding algorithm. First, we introduce some notations. Let (n, k) denote code length and information bit length respectively. We assume an additive white Gaussian noise (AWGN) channel and $N_0/2$ to be the two-sided noise power spectral density, thus the standard deviation of a noise is $\sigma = \sqrt{N_0/2}$. As for modulation, binary phase-shift keying (BPSK) is assumed, whose signal energy per one information bit is denoted by E_b. Thus, over a noiseless channel, outputs of the matched filters are

$$\begin{cases} -\sqrt{RE_b}, & \text{if sending information is 1,} \\ \sqrt{RE_b}, & \text{if sending information is 0,} \end{cases} \tag{1}$$

where R is the code rate. Next, we briefly introduce the encoding algorithm, which is standard for shortened LDPC code. Assume a $(n+\alpha, k+\alpha)$ base LDPC code is available. Let α be the number of shortened bits from the base LDPC code. We denote the parity check matrix of base LDPC code by $H = (h_{i,j})$.

(Encoding Algorithm)

(1) Let $(x_1, \ldots, x_k)$, $x_j \in \{0,1\}, j \in \{1, \ldots, k\}$ be information bits. Embed α bits of all "0" sequence to information bits and obtain (pseudo) information sequence $\mathbf{x}' = (x'_1, \ldots, x'_{k+\alpha})$ as follows; see Figure 1. Let T be a set of embedded (true) bits positions.
for $j = 1$ **to** $k + \alpha$ **do**

$$x'_j = \begin{cases} 0, & \text{if } j \in T \\ x_{j-\beta_j}, \beta_j := |\{l \,|\, l \in T, l < j\}|, & \text{else} \end{cases}$$

end for

(2) Obtain parity part $(x'_{k+\alpha+1}, \ldots, x'_{n+\alpha})$ from

$$H(x'_1, \ldots, x'_{k+\alpha}, x'_{k+\alpha+1}, \ldots, x'_{n+\alpha})^T = \mathbf{0}^T.$$

(3) Transmit a sequence $(x_1, \ldots, x_k, x'_{k+\alpha+1}, \ldots, x'_{n+\alpha})$ by BPSK modulation.

We introduce the following three different methods to determine the shortening position T:

$$\begin{cases} \text{(A) } T = \{8, 16, \ldots, 8\alpha\} \text{ (equal interval)} \\ \text{(B) Worst reliable } \alpha \text{ positions according to [16]} \\ \text{(C) Integer Programming based approach} \end{cases}$$

We note that the length of $n+\alpha$ sequence is not transmitted in the above encoding process. Instead, we transmit the length of a n sequence, thus, the coding rate is k/n (Fig. 1). Because both sender and receiver know α bits of shortening position T and they are set to be zero, hence it is not necessary to transmit these all zero bits.

Next, we introduce a decoding algorithm, which is a concatenation of the SP-algorithm for shortened LDPC codes and the OSD method. We note that the following decoding algorithm is essentially the same as the hybrid decoding algorithm in [2], except for the adoption of shortened LDPC codes (for the applications of this hybrid decoding algorithm to non-binary LDPC codes, consult Baldi et al. [1]). However, as shown in the numerical experiments discussed in Sect. 5, adoption of appropriate shortened LDPC codes fairly improve error correction ability compared to the one that uses prime base LDPC codes for the OSD process.

In the following, L denotes a maximum iteration number of SP-algorithm, M is a large real number, and $\mathbf{y} = (y_1, \ldots, y_k, y_{k+\alpha+1}, \ldots, y_{n+\alpha})$ $(y_j \in \mathbb{R}, j \in \{1, \ldots, k, k+\alpha+1, \ldots, n+\alpha\})$, a received sequence. In step (2) of the following decoding procedure, we will use accumulative log-likelihood ratio (LLR) information as in [11] owing to its efficiency. In step (3), we apply hard decision process to the outputs of SP. If this hard decision satisfy parity condition, we adopt this hard decision as the final decoding result. In step (4), positions corresponding to set T are set to large value M, since these bits include no error. Step (5) and (6) are conventional OSD method.

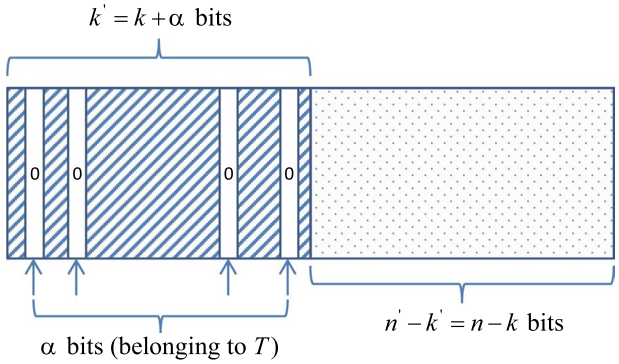

Fig. 1 Encoding process

(Decoding Algorithm)

(1) Compensate all α bits (positions corresponding to T) with value "$-2\sqrt{RE_b}$" and obtain the length of the $n+\alpha$ sequence; $\mathbf{y}' = (y'_1, \ldots, y'_{n+\alpha})$.

(2) Set $i = 1$ and $\gamma = (\gamma_1, \ldots, \gamma_{n+\alpha}) = (0, \ldots, 0)$.
Apply the (log-domain) SP-algorithm to $\mathbf{y}'$. Set $\gamma^{(\mathbf{i})} = (\gamma_1^{(i)}, \ldots, \gamma_{n+\alpha}^{(i)})$, an output of the SP-algorithm. Update $\gamma = \gamma + (|\gamma_1^{(i)}|, \ldots, |\gamma_{n+\alpha}^{(i)}|)$.

(3) (Hard Decision and Parity Check) Set $\mathbf{z} = (z_1, \ldots, z_{n+\alpha})$ $(z_j \in \{0,1\},\ j \in \{1, \ldots, n+\alpha\})$ which is a sequence obtained from the hard decision of γ. **if** $H\mathbf{z}^T = \mathbf{0}^T$ **then**
decode $\mathbf{x}' = (x'_1, \ldots, x'_{n+\alpha}) = \mathbf{z}$.
exit
end if
$i = i + 1$
if $i < L$ **then**
goto (2) (continue further iteration)
end if

(4) Exchange α bits of γ and $\mathbf{z}$ (positions corresponding to T) with M and "0" respectively. Define sequences γ' and $\mathbf{z}'$ respectively.

(5) (Gaussian elimination for OSD) Set γ'' as a reordering of γ' with ascending order (thus, $\gamma'' = \pi_1[\gamma']$, where π_1 is a corresponding permutation).
Set H' to be a reordering of columns of H with respect to π_1 (resultant matrix is denoted by $H' = \pi_1[H]$). Apply Gaussian elimination to H' for the transformation to systematic form H'' (the permutation corresponding to column exchange is denoted by π_2). Set $\pi = \pi_2 \circ \pi_1$.

(6) (Order-p OSD) Find $\mathbf{w} = (w_1, \ldots, w_{n+\alpha})$ which maximizes

$$C(\mathbf{w}) = \sum_{j=1}^{n+\alpha} w_j y'_j.$$

Set $C(\mathbf{w}) = 0$.
for $i = 0$ **to** p **do**
Flip different i positions of $(z'_{\pi(n-k+1)}, \ldots, z'_{\pi(n)})$.
Set $(z'_{\pi(1)}, \ldots, z'_{\pi(n-k)})$ to satisfy $H''\pi[z']^T = \mathbf{0}^T$.
if $C(\mathbf{z}') > C(\mathbf{w})$ **then**
$\mathbf{w} = \mathbf{z}'$
end if
end for
Decode $\mathbf{x}' = (x'_1, \ldots, x'_{n+\alpha}) = \mathbf{w}$.

Here we note that we do not flip any of $z'_{\pi(n+1)}, \ldots, z'_{\pi(n+\alpha)}$ in step (6) since this sequence does not contain any errors.

4 Selection of α-shortening positions

The selection of shortening positions in α information bits is important for improving CER; see [3, 16, 17] and their references. Shortening techniques do not depend on the code length, however, they result in significant progress in the error correction ability when they are adopted to medium-length codes with OSD decoding. This is because, a small refinement in error rate of MRB often results in large improvement in the process of OSD decoding, as we shall see in the numerical experiments in the following sections. In the experiments, we try the following three types of shortening methods. First method is a code independent, while other two methods are code dependent one.

(A) Shortening positons are every 8 bit as

$T = \{8, 16, \ldots, 8\alpha\}$.

(B) Worst α reliable bits selection [16]

(1) Assume that we send all zero sequences and put the received sequence as

$$(y_1, \ldots, y_{n+\alpha}) = (2\sqrt{RE_b}, \ldots, 2\sqrt{RE_b}).$$

(2) Iterate procedure (2) of the decoding algorithm M times (without parity check procedure at this stage).
(3) Reorder the output of procedure (2) in ascending order.
(4) Take the indices of the first α part of the reordered output of (3).

In the case of (B), we fixed $E_s = 0.5E_b = 0.5N_0 = 4$ and $M = 50$. By applying the above method (B), we obtained the following tables that describe the shortening positions for (1) $(n+\alpha, k+\alpha, \alpha) = (576, 288, 36)$ IEEE 802.16e (WiMAX) LDPC code, (2) $(n+\alpha, k+\alpha, \alpha) = (648, 324, 40)$ and (648,324,36) IEEE 802.11n (Wi-Fi) LDPC code, (3) $(n+\alpha, k+\alpha, \alpha) = (256, 128, 16)$ CCSDS LDPC code and (4) $(n+\alpha, k+\alpha, \alpha) = (512, 256, 32)$ CCSDS LDPC code (Table 1).

For the third method, we need some definitions.

Definition 1 Let

$$A_i := \{j \in \{1, 2, \ldots, n+\alpha\} : h_{i,j} = 1\} \quad \forall i \in \{1, 2 \ldots, n-k\} \tag{2}$$

$$T_i := \{S \subset A_i : |S| \text{ is odd}\} \tag{3}$$

and

Table 1 Shortening positions *T* of each LDPC code when method (B) is applied

(1) $(n+a, k+a, a) = (576, 288, 36)$ IEEE 802.16e	
T	551, 550, 549, 548, 547, 546, 545, 544, 543, 542, 541, 540, 539, 538, 537, 536, 535, 534, 533, 532, 531, 530, 529, 528, 347, 346, 345, 344, 343, 342, 341, 340, 339, 338, 337, 336
(2) $(n+a, k+a, a) = (648, 324, 40)$ and (648,324,36) IEEE 802.11n	
T	620, 619, 618, 617, 616, 615, 614, 613, 612, 611, 610, 609, 608, 607, 606, 605, 604, 603, 602, 601, 600, 599, 598, 597, 596, 595, 594, 431, 430, 429, 428, 427, 426, 425, 424, 423, 422, 421, 420, 419
(3) $(n+a, k+a, a) = (256, 128, 16)$ CCSDS	
T	175, 174, 173, 172, 171, 170, 169, 168, 167, 166, 165, 164, 163, 162, 161, 160
(4) $(n+a, k+a, a) = (512, 256, 32)$ CCSDS	
T	351, 350, 349, 348, 347, 346, 345, 344, 343, 342, 341, 340, 339, 338, 337, 336, 335, 334, 333, 332, 331, 330, 329, 328, 327, 326, 325, 324, 323, 322, 321, 320

$$1 + \sum_{t\in S}(x_t - 1) - \sum_{t\in A_i\setminus S} x_t \le 0, \quad \forall i \in \{1,\dots,n-k\},\ \forall S \in T_i \tag{4}$$

$$0 \le x_j \le 1, \quad \forall j \in \{1,\dots,n+\alpha\}. \tag{5}$$

The following polytope is called the Feldman–Wainwright–Karger code polytope [7]:

$$P(H) = \{\mathbf{x} \in \mathbb{R}^{n+\alpha} : \mathbf{x} \text{ satisfies (4) and (5)}\}. \tag{6}$$

(C) Integer Programming based approach (I)

(1) Set $C = \emptyset$, $P' = P(H)$, $j = 0$ and L be a large positive integer.
(2) **if** $j > L$ **then exit**
(2) Minimize: $\sum_{t=1}^{n+\alpha} x_t$ subject to $\mathbf{x} \in P'$.
(3) Let an optimal solution of Step (2) be $\mathbf{x}^*$ and optimal value z^*. Redefine P' and C as

$$P' = P' \cup \{\mathbf{x} \in \mathbb{R}^{n+\alpha} : \sum_{t=1}^{n+\alpha} x_t^* x_t \le z^* - 1\},$$
$$C = C \cup \{\mathbf{x}^*\}$$

and put $j = j + 1$.
(4) **goto** Step (2).

First, we determine a code set which has small Hamming weight.

The set C obtained through above process represents a collection of codewords that have a small Hamming weight. Table 2 shows (a part of) the positions that take $x_t^* = 1$, for (1) $(n+\alpha, k+\alpha, \alpha) = (576, 288, 36)$ IEEE 802.16e (WiMAX) LDPC code and (2) $(n+\alpha, k+\alpha, \alpha) = (648, 324, 40)$ IEEE 802.11n (Wi-Fi) LDPC code. We have computed 900 codewords in ascending order of the Hamming weight for case (1) and 800 codewords for case (2) by using MIP Solver Gurobi optimizer 8.1. We tried to avoid codewords of small Hamming weight appearing in set C as much as possible. To

decide the shortening positions of the code so as to achieve this objective, we make use of the integer programming technique again. We set a positive integer K as large as possible so that the following optimization problem is feasible. We put N as the total number of shortening positions.

(C) Integer Programming based approach (II)

(1) Let

$$\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n})$$

be an i-th element of the set C.

(2) Solve the following integer programming problem:

Minimize:

Subject to:

$$\sum_{j=1}^{n} x_{i,j} z_j \geq 1, \quad (1 \leq i \leq K)$$

$$\sum_{j=1}^{n} z_j \leq N$$

$$z_j \in \{0, 1\}, \quad (1 \leq j \leq n)$$

The vector $\mathbf{z} = (z_1, \ldots, z_n)$ above represents a position to be shortened. More precisely, if $z_i = 1$ then i-th position in original codeword is shortened and its value is set to be "0". Next proposition shows any codeword up to K-th position in the set C is prohibited in the proposed shortened code.

Proposition 1 *Assume* $\mathbf{z} = (z_1, \ldots, z_n)$ *be a feasible solution to the integer programming problem* (*C*). *Further, if* $z_i = 1$, *then we set* $i \in T$ *in "Encoding Algorithm". Then, any codeword up to K-th position in the set C never appears in the set of codewords designed by the "Encoding Algorithm".*

1 (Proof)

Set $C_K \subset C$ *be the set of codewords up to K-th position in C and* $\mathcal{C}$ *be the set of codewords designed by the "Encoding Algorithm". Let* $\mathbf{x}^*$ *be an element in* C_K *with minimum Hamming weight* d^*. *Then, from the definition of* $\mathbf{z}$, *at least one position of* $\mathbf{x}^*$ *whose bit has a value "1" is forced to be "0" in "Encoding Algorithm". Thus,* $\mathbf{x}^*$ *is never contained in* $\mathcal{C}$. *Assume any codeword of* C_K *with Hamming weight* d $(d > d^*)$ *is not contained in* $\mathcal{C}$. *Now let* $\mathbf{x}_{d+1} \in C_K$ *be an arbitrary codeword with Hamming weight* $d + 1$. *Then, again from the definition of* $\mathbf{z}$, *at least one position of* $\mathbf{x}_{d+1}$ *whose bit has a value "1" is forced to be "0" in "Encoding Algorithm"(thus modified* $\mathbf{x}_{d+1}$ *has a Hamming weight less than or equal to d). Hence, by induction hypothesis,* $\mathbf{x}_{d+1}$ *is never contained in* $\mathcal{C}$, *which proves the assertion.* □

Thus, the minimum free distance of the proposed shortened code is significantly increased compared to the original code. Hence, we can expect erroneous bits in the proposed shortened code decrease, especially in a relatively high signal-to-noise ratio environment. This minor improvement on erroneous bits accelerates total improvement on OSD decoding method.

Although we have experimented with four cases of different LDPC codes, we have demonstrated approach (C) for only the above two cases (1) and (2) (different two cases $\alpha = 40$ and $\alpha = 36$); see Table 3. This is mainly due to the limit of computation time (in our computing environment, it took about a month to obtain the result of Table 2). As we shall see in the following, method (C) shows a remarkable advantage in error correction ability, it seems worthwhile to try to explore some methods to obtain the result of (C)-I (as shown in Table 2) within more reasonable time. Combination of some probabilistic methods for weight distribution, e.g., [6, 10, 12, 15] and IP approach might prove promising. Specifically, an approach based on the "impulse method" by Hu et al. [10] and Declercq and Fossorier [6] is known to be effective for enumerating such low weight codewords.

5 Numerical experiments

For our experiments, we used four different LDPC codes, which have a systematic structure (WiMAX code, Wi-Fi code and two CCSDS codes). We evaluated error correction ability with codeword error rate (CER), which is defined as the ratio of the number of

Table 2 A set of codewords which have small Hamming weight

No.	$(n+\alpha, k+\alpha, \alpha) = (576, 288, 36)$ **IEEE 802.16e**
1	100, 158, 247, 261, 345, 409, 419, 423, 433, 443, 494, 504, 518
2	109, 167, 246, 256, 354, 408, 418, 428, 442, 452, 503, 513, 527
3	116, 150, 253, 263, 337, 411, 415, 425, 435, 449, 486, 510, 520
⋮	⋮
24	104,162, 241, 251, 349, 413, 423, 427, 437, 447, 498, 508, 522
25	96, 110, 144, 154, 243, 247, 341, 355, 409, 415, 439, 443, 480, 490, 504, 524
26	99, 113, 147, 157, 246, 250, 344, 358, 412, 418, 442, 446, 483, 493, 507, 527
⋮	⋮
899	1, 47, 101, 159, 251, 258, 334, 341, 346, 358, 365, 382, 389, 400, 406, 424, 430, 454, 471, 481, 508, 539, 563
900	11, 27, 59, 113, 154, 158, 217, 308, 314, 317, 338, 362, 382, 391, 428, 436, 466, 492, 494, 516, 517, 540, 541
No.	$(n+\alpha, k+\alpha, \alpha) = (648, 324, 40)$ **IEEE 802.11n**
1	84, 85, 263, 301, 355, 382, 409, 435, 452, 462, 488, 581, 582, 609, 636
2	100, 101, 252, 317, 371, 398, 425, 441, 451, 478, 504, 570, 571, 598, 625
3	92, 93, 244, 309, 363, 390, 417, 433, 443, 470, 496, 589, 590, 617, 644
⋮	⋮
27	90, 91, 269, 307, 361, 388, 415, 441, 458, 468, 494, 587, 588, 615, 642
28	33, 45, 85, 139, 168, 288, 355, 382, 396, 409, 468, 495, 522, 561, 565, 580, 582, 607
29	28, 43, 95, 149, 178, 271, 365, 379, 392, 419, 478, 505, 532, 544, 548, 590, 592, 617
⋮	⋮
799	73, 136, 163, 181, 188, 259, 307, 335, 352, 397, 404, 431, 448, 458, 475, 511, 524, 558, 560, 587, 631, 632
800	79, 187, 193, 265, 382, 403, 409, 436, 454, 463, 481, 488, 490, 515, 542, 557, 569, 584, 596, 611, 623, 638

Table 3 Shortening positions *T* of each LDPC code in case method (C) were applied

(1) $(n+\alpha, k+\alpha, \alpha) = (576, 288, 36)$ IEEE 802.16e	
T	150, 313, 317, 330, 345, 350, 355, 358, 362, 366, 370, 409, 410, 413, 417, 418, 419, 421, 422, 426, 429, 430, 431, 449, 470, 483, 486, 487, 490, 491,495, 499, 503, 506, 520, 574
(2) $(n+\alpha, k+\alpha, \alpha) = (648, 324, 40)$ IEEE 802.11n	
T	290, 372, 379, 381, 383, 387, 396, 402, 405, 407, 409, 411, 413, 424, 430, 445, 455, 471, 515, 516, 553, 568, 576, 579, 585, 586, 589, 594, 596, 598, 599, 600, 601, 602, 609, 615, 617, 618, 619, 620
(2') $(n+\alpha, k+\alpha, \alpha) = (648, 324, 36)$ IEEE 802.11n	
T	86, 379, 397, 398, 399, 407, 408, 409, 413, 420, 422, 429, 431, 441, 470, 555, 572, 581, 588, 594, 595, 596, 598, 601, 602, 603, 605, 607, 609, 612, 613, 614, 616, 618, 619, 620

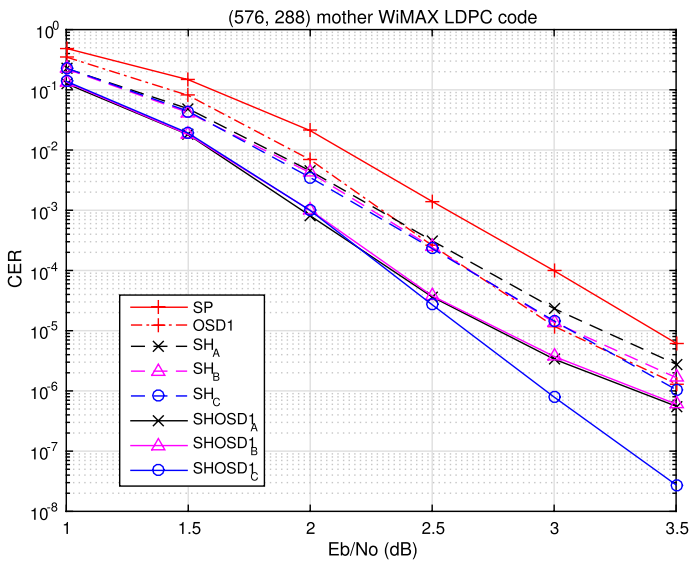

Fig. 2 IEEE 802.16e $(n+\alpha, k+\alpha) = (576, 288)$ base LDPC code. $M = 50, L = 10{,}000$

decoding failures to total number of received codewords. In each of the following figures, horizontal axis represents the signal to noise ratio: E_b/N_0. In all experiments we assumed $E_s = RE_b = 1$ and they were performed on Intel(R) Xeon(R) E5-1660 3.70 GHz host using gcc 4.4.7-O3.

(1) In this case, we used $(n+\alpha, k+\alpha, \alpha) = (576, 288, 36)$ IEEE 802.16e (WiMAX) LDPC code as the base code. Thus, $(n, k) = (540, 252)$, and hence coding rate is $R = k/n = 0.467$. The result for this case is shown in Fig. 2. In the figure, label SP shows the CER of the "original" (576,288) LDPC code with the SP-algorithm, and OSD1 stands for the result of Order-1 OSD method applied to base LDPC code. $\mathrm{SH_A}$, $\mathrm{SH_B}$ and $\mathrm{SH_C}$ are the results of α bit shortened codes whose shortening positions were determined by (A), (B) and (C) as described in the previous section, respectively. $\mathrm{SHOSD1_A}$, $\mathrm{SHOSD1_B}$ and $\mathrm{SHOSD1_C}$ are the proposed methods which apply the Order-1 OSD method to $\mathrm{SH_A}$, $\mathrm{SH_B}$ and $\mathrm{SH_C}$, respectively. $\mathrm{SH_B}$, $\mathrm{SH_C}$ and OSD1 showed almost the same CER abilities (whereas $\mathrm{SH_A}$ is inferior to those). However, we observed that $\mathrm{SHOSD1_A}$, $\mathrm{SHOSD1_B}$ and

SHOSD1$_C$ accomplished CER under 10^{-5} at $E_b/N_0 = 3.0$ dB and 3.5 dB and constantly overperformed the result of OSD-1 decoding. In particular, the SHOSD1$_C$ is superior even if compared to SHOSD1$_A$ and SHOSD1$_B$ at $E_b/N_0 = 3.0$ dB and 3.5 dB. Table 4 shows the effectiveness of the OSD effect for the base and shortened LDPC codes. As shown in Table 4, the SH$_A$/SHOSD1$_A$ and SH$_B$/SHOSD1$_B$ ratios are, in most cases, superior to the SP/OSD ratio, except at 3.0 dB and 3.5 dB. However, SH$_C$/SHOSD1$_C$ shows a far superior improvement rate compared with the other three cases. This indicates that if we properly select shortening positons, the OSD effect is accelerated by the corresponding shortened codes. Table 5 shows the number of excluded codewords, out of No.1 through No.900, appearing in Table 2 by methods (A), (B) and (C), respectively. From this, we see that the number of excluded codewords by method (C) is larger than those with methods (A) and (B). Figure 3 gives the distribution of the number of codewords which are not covered by each of the shortening methods (A), (B) and (C). The horizontal axis indicates the code weight, while the vertical gives the number of (log scaled) codewords uncovered by respective methods. Method (A) and (B) remain considerable amount of uncovered codewords (especially for a low weight distribution, such as 13 and 16), whereas the method (C) keeps only three uncovered codewords at a weight 23. The error correction ability at 3.0 dB and 3.5 dB appears to be affected by these numbers of uncovered numbers of codewords with small Hamming weight in Table 5.

(2)-1 In this case, we used $(n+\alpha, k+\alpha, \alpha) = (648, 324, 40)$ IEEE 802.11n (Wi-Fi) LDPC code as the base code. Therefore, $(n, k) = (608, 284)$, and hence coding rate is $R = k/n = 0.467$. The result for this case is shown in Fig. 4. All labels mean the same as in Fig. 2. We can see that SH$_A$, SH$_B$, SH$_C$ and OSD1 show almost the equivalent CER abilities. On the other hand, we observe that SHOSD1$_A$, SHOSD1$_B$ and SHOSD1$_C$ accomplish CER under 10^{-5} at $E_b/N_0 = 2.5$ dB and constantly overperform OSD-1 decoding. In particular, the SHOSD1$_B$ and SHOSD1$_C$ results at $E_b/E_0 = 3.0$ dB are strong compared to those of SHOSD1$_A$. Table 6 shows the efficiency of the OSD effect

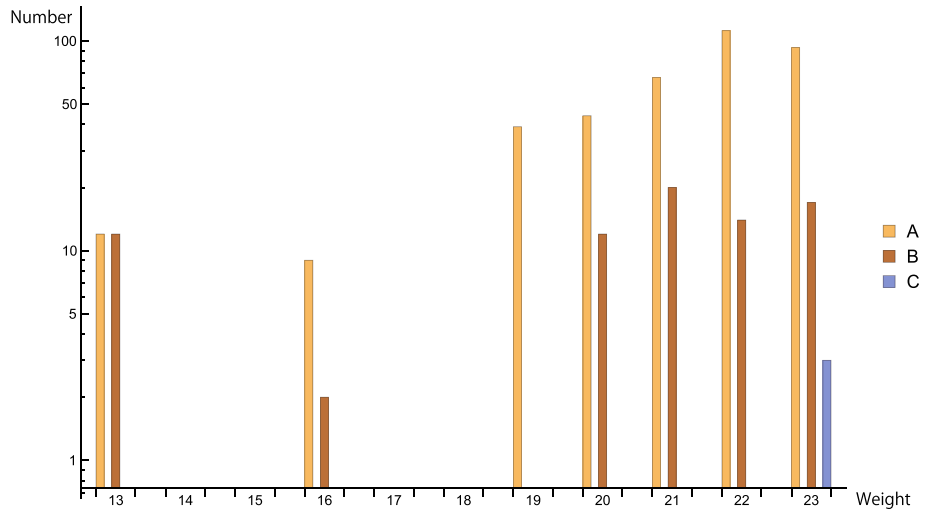

Fig. 3 Distribution of number of codewords which are not covered by each shortening methods for WiMAX $(n+\alpha, k+\alpha, \alpha) = (576, 288, 36)$ LDPC code

Table 4 OSD effect for base and shortened codes: case (1)

dB	SP/OSD	RA	RB	RC
1.0	1.39	1.78	1.66	1.66
1.5	1.83	2.69	2.31	2.30
2.0	3.04	5.56	4.20	3.54
2.5	5.38	8.57	6.41	8.57
3.0	8.35	6.91	3.68	18.35
3.5	4.73	4.99	2.75	38.1

Following abbreviation is used: RA $=$ SH_A/$SHOSD1_A$, RB $=$ SH_B/$SHOSD1_B$ and RC $=$ SH_C/$SHOSD1_C$

Table 5 Number of covered (inhibited) codewords appearing in Table 2 for IEEE 802.16e code by method (A), (B) and (C) respectively

Method	(A)	(B)	(C)
Number	524	823	897

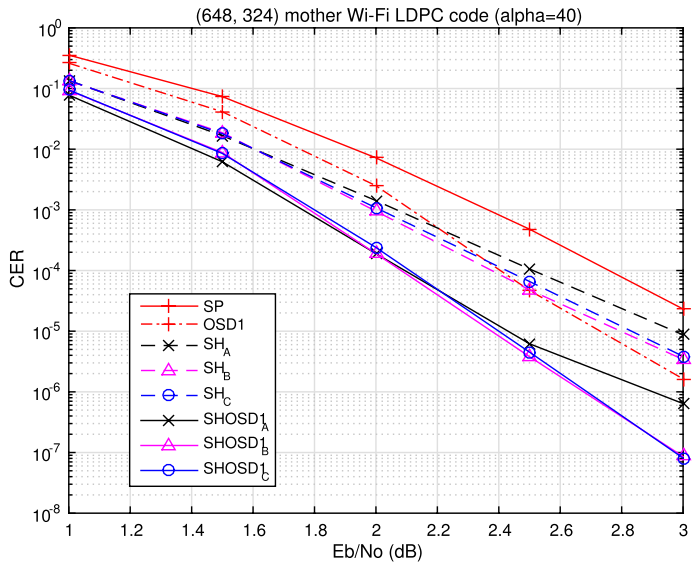

Fig. 4 IEEE 802.11n $(n+\alpha, k+\alpha) = (648, 324)$ base LDPC code $(\alpha = 40)$. $M = 50, L = 10{,}000$

Table 6 OSD effect for base and shortened codes: case (2) ($\alpha = 40$)

dB	SP/OSD	RA	RB	RC
1.0	1.36	1.75	1.44	1.43
1.5	1.78	2.69	2.13	2.12
2.0	2.88	7.47	5.00	4.5
2.5	10.21	17.1	12.63	14.62
3.0	14.20	13.59	39.73	47.53

Table 7 Number of covered (inhibited) codewords appearing in Table 2 for the IEEE 802.11n code ($\alpha = 40$) by method (A), (B) and (C) respectively

Method	(A)	(B)	(C)
Number	432	771	793

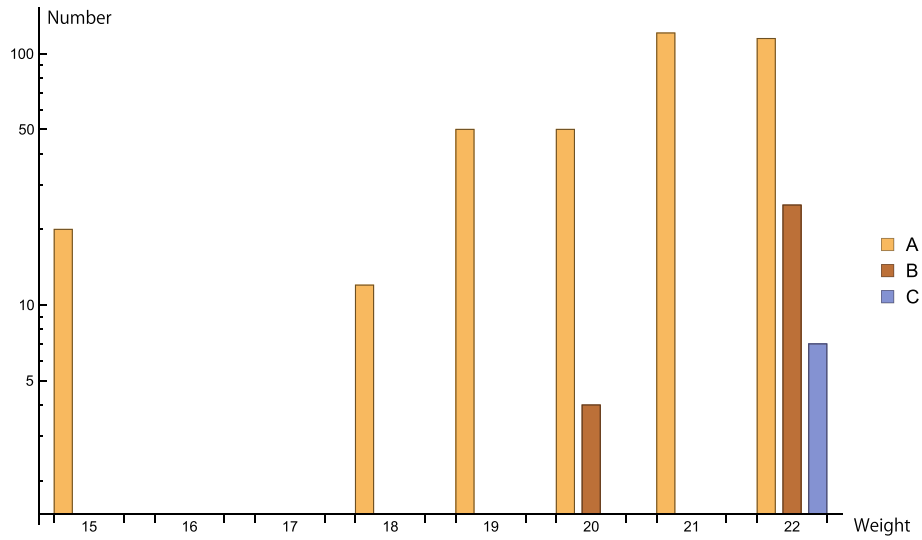

Fig. 5 Distribution of number of codewords which are not covered by each shortening method for Wi-Fi $(n+\alpha, k+\alpha, \alpha) = (648, 324, 40)$ LDPC code

for the base and shortened LDPC codes. As shown in Table 6, the ratios $\mathrm{SH_A/SHOSD1_A}$, $\mathrm{SH_B/SHOSD1_B}$ and $\mathrm{SH_C/SHOSD1_C}$, except for the case of 3.0 dB for $\mathrm{SH_A/SHOSD1_A}$, are superior to the ratio SP/OSD, which means that the proposed method accelerates the OSD effect. As a possible cause that $\mathrm{SHOSD1_A}$ loses its efficiency at $E_b/E_0 = 3.0\,\mathrm{dB}$, we mention the following. First, as shown in Table 7, the covered codewords for methods (B) and (C) are relatively large (771 and 793) compared to the number for method (A). Second, from Fig. 5, we observe similarity in the distribution of the number of uncovered codewords for the methods (B) and (C), although the distribution with method (B) has non-zero value at weight 20. On the other hand, the distribution of the number of the uncovered codewords for method (A) has non-zero value at low Hamming weight (at 15, 18, 19). This likely causes the degradation of CER at $E_b/E_0 = 3.0\,\mathrm{dB}$ for method $\mathrm{SHOSD1_A}$.

(2)-2 Next, we examine the same IEEE 802.11n (Wi-Fi) LDPC code as a base code for a different parameter, specifically, $(n+\alpha, k+\alpha, \alpha) = (648, 324, 36)$. Thus, $(n, k) = (612, 288)$ and hence $R = k/n = 0.471$. The results are shown in Fig. 6, with its efficiency in Table 8, the number of covered codewords for each method in Table 9, and the distribution of the number of uncovered codewords for each Hamming weight of codeword in Fig. 7. We observed almost the same tendency as in the case of $\alpha = 40$.

(3) In this case, we used $(n+\alpha, k+\alpha, \alpha) = (256, 128, 16)$ CCSDS LDPC code as the base code see; [4]. Thus, $(n, k) = (240, 112)$, and hence coding rate is $R = k/n = 0.467$.

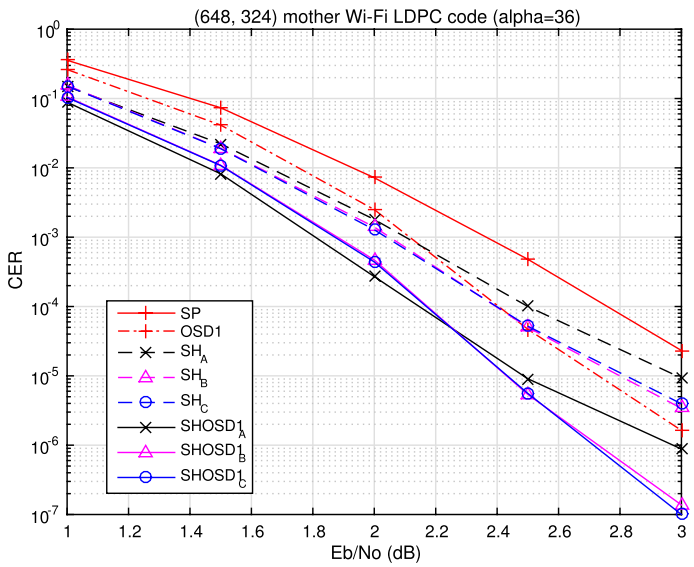

Fig. 6 IEEE 802.11n $(n+\alpha, k+\alpha) = (648, 324)$ base LDPC code $(\alpha = 36)$. $M = 50, L = 10{,}000$

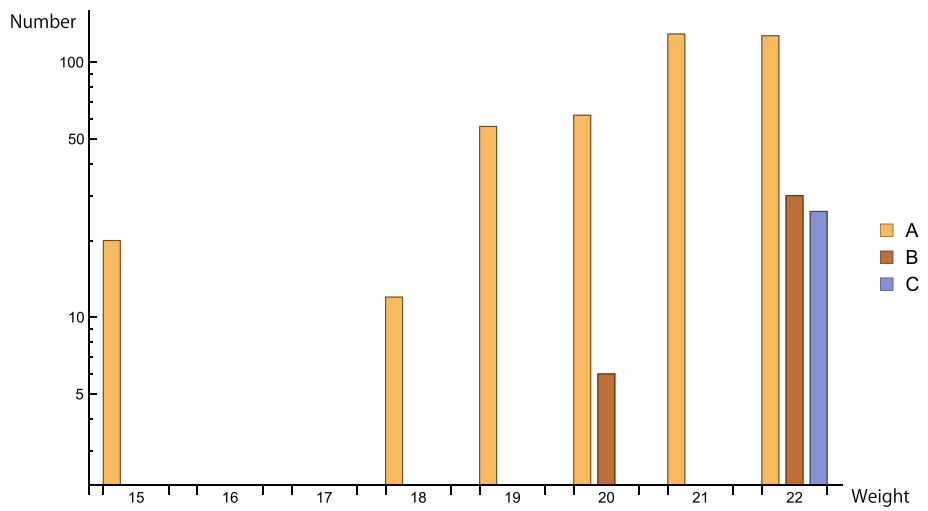

Fig. 7 Distribution of number of codewords which are not covered by each shortening methods for Wi-Fi $(n+\alpha, k+\alpha, \alpha) = (648, 324, 36)$ LDPC code

The result for this case is shown in Fig. 8. In the figure, the meanings of SP, SH_A and SH_B are the same as in Fig. 2. OSD2 shows the result from the Order-2 OSD method applied to the base LDPC code. $SHOSD2_A$ and $SHOSD2_B$ are proposed methods that apply the Order-2 OSD method to SH_A and SH_B respectively. We found that the error correction ability of OSD2 is superior to those of SH_A and SH_B. We observed that $SHOSD2_A$ and $SHOSD2_B$ accomplish CER under 10^{-5} at $E_b/N_0 = 3.5$ dB and constantly outperforms the result of OSD-2 decoding. As an interesting finding, we observe in this case CER of $SHOSD2_A$ constantly outperforms $SHOSD2_B$, although not so significant degree. Table 10 shows the efficiency of OSD effect for a base and shortened LDPC codes. From

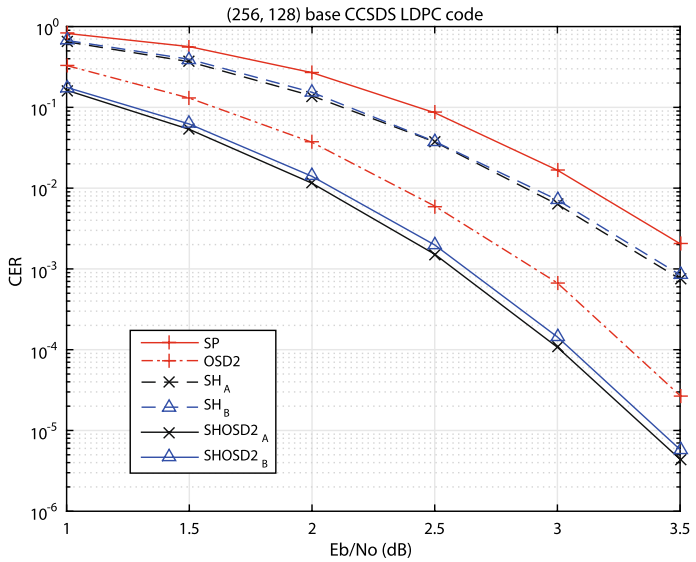

Fig. 8 CCSDS $(n+\alpha, k+\alpha) = (256, 128)$ base LDPC code. $M = 50, L = 10{,}000$

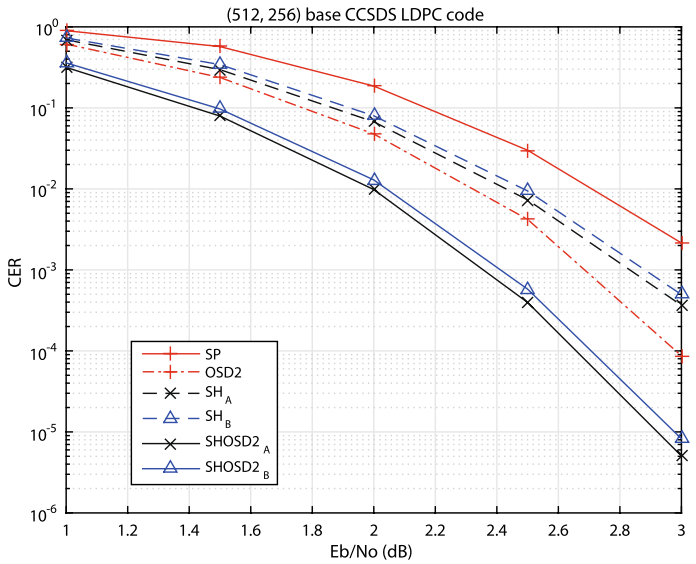

Fig. 9 CCSDS $(n+\alpha, k+\alpha) = (512, 256)$ base LDPC code. $M = 50, L = 10{,}000$

Table 8 OSD effect for base and shortened codes: case (2) ($\alpha = 36$)

dB	SP/OSD	RA	RB	RC
1.0	1.36	1.69	1.47	1.46
1.5	1.78	2.71	1.75	1.75
2.0	2.88	6.67	2.98	2.91
2.5	10.21	11.01	9.28	9.32
3.0	14.20	10.57	24.73	39.5

Table 9 Number of covered (inhibited) codewords appearing in Table 2 for IEEE 802.11n code ($\alpha = 36$) by method (A), (B) and (C) respectively

Method	(A)	(B)	(C)
Number	394	764	774

Table 10 OSD effect for base and shortened codes: case (3)

dB	SP/OSD	RA	RB
1.0	2.50	3.98	3.81
1.5	4.34	6.87	6.33
2.0	7.29	12.06	10.96
2.5	14.28	24.61	19.09
3.0	24.91	58.68	49.79
3.5	75.93	170.45	145.52

Table 11 OSD effect for base and shortened codes: case (4)

dB	SP/OSD	RA	RB
1.0	1.46	2.19	2.03
1.5	2.41	3.74	3.53
2.0	3.91	6.87	6.19
2.5	7.14	18.46	16.30
3.0	24.71	72.48	60.12

this table, we see that ratios $SH_A/SHOSD2_A$ and $SH_B/SHOSD2_B$ consistently improve the ratio of SP/OSD2. Hence, in this case, by shortening the base LDPC code, the OSD effect was accelerated.

(4) In this case we used $(n+\alpha, k+\alpha, \alpha) = (512, 256, 32)$ CCSDS LDPC code as the base code see; [4, 5]. Thus, $(n, k) = (480, 224)$, and hence coding rate is $R = k/n = 0.467$. The results for this case is shown in Fig. 9. All captions are the same as in Fig. 8. SH_A, SH_B and OSD2 show almost the same CER abilities. On the other hand, $SHOSD2_A$ and $SHOSD2_B$ accomplish CER under 10^{-5} at $E_b/N_0 = 3.0$ dB and consistently improve the OSD-2 decoding result. As in case (3), the CER of $SHOSD2_A$ slightly better than that of $SHOSD2_B$. Table 11 shows the efficiency of the OSD effect for the base and shortened LDPC codes. As shown in Table 11, we see that $SH_A/SHOSD2_A$ and $SH_B/SHOSD2_B$ constantly improve the ratio of SP/OSD2. Therefore, as in case (3), by shortening the base LDPC code, the OSD effect was accelerated.

6 Execution time evaluation

The decoding algorithm we presented in this paper is a kind of hybrid type decoding algorithms and its structure is analogous to that of Baldi et al. [2]. As explained in [2], most decoding trials end with procedure (3) of the decoding algorithm. Moreover, we assumed a relatively small order OSD reprocessing procedure (in the experiment discussed in the previous section order one and two reprocessing was used), thus, the

Table 12 Average execution time of Case (1)

dB/Method	1.0	2.0
SP (s)	0.011	0.0043
OSD1 (s)	0.014	0.014
OSD1/SP	1.27	3.26
$\mathrm{SHOSD1}_{\mathrm{A}}$ (s)	0.014	0.013
$\mathrm{SHOSD1}_{A}$/SP	1.27	3.02
OSD ratio (%)	48.7	2.08

Table 13 Average execution time of Case (2) ($\alpha = 40$)

dB/Method	1.0	2.0
SP (s)	0.012	0.0052
OSD1 (s)	0.018	0.019
OSD1/SP	1.5	3.65
$\mathrm{SHOSD1}_{\mathrm{A}}$ (s)	0.018	0.019
$\mathrm{SHOSD1}_{A}$/SP	1.5	3.65
OSD ratio (%)	35.6	0.71

average decoding time would not be so apart from that of SP. We demonstrated this via numerical experiments. Tables 12 and 13 show the average computing time of ratio for the experiment cases (1) and (2) ($\alpha = 40$) of the previous section respectively. Here "OSD1" represents the average execution time for Order-1 OSD process (without the time for Sum-product part) and similarly, "$\mathrm{SHOSD1}_{\mathrm{A}}$" shows the average execution time for OSD process with shortened LDPC code and shortening method (A). OSD ratio refers to the percentage of OSD trials, i.e., the ratio of SP decoding failure. The labels OSD1/SP and $\mathrm{SHOSD1}_{\mathrm{A}}$/SP refer to the execution time ratios. We note execution times shown in these tables are based on Intel(R) Xeon(R) E5-1660 3.70GHz processor host using gcc 4.4.7 -O3. Both Tables 12 and 13 show that the execution times for OSD and $\mathrm{SHOSD1}_{\mathrm{A}}$ do not depend on the signal to noise ratio E_b/N_0. From Table 12, we can observe that in the case, E_b/N_0 is relatively low (= 1.0 dB), execution times of SP and OSD1 (or $\mathrm{SHOSD1}_{\mathrm{A}}$) do not show a remarkable difference (OSD1/SP=$\mathrm{SHOSD1}_{\mathrm{A}}$ /SP = 1.27). On the other hand, in the case, E_b/N_0 is relatively high (= 2.0 dB), execution times of OSD1 and $\mathrm{SHOSD1}_{A}$ are approximately three times longer than that of SP. However, in the case $\mathrm{Eb/N_0} = 2.0$ dB, OSD ratio is only 2.08 %, so about 98 % of instances are sufficient for SP decoding, as we have noted at the beginning of this section, that most of decoding trials end at procedure (3) of decoding algorithm. Thus, even from the viewpoint of the total execution time, OSD-based decodings do not seem to lose their advantage (high precision decoding property) compared with SP decoding even in a relatively high E_b/N_0 circumstance. Almost same tendency can be observed in Table 13.

7 Results and discussion

An effective way to increase the OSD decoding ability was presented. As mentioned regarding the experiments described in Sect. 5, by determining T appropriately, CER can be reduced. In particular the method (C), which is based on mathematical programming,

seems to be effective. However, this method requires a collection of the codewords that have a small Hamming weight as shown in Table 2. Obtaining these tables via only mathematical programming as in method (C)–(I) is a computationally very hard task. Hence, hybrid methods with some heuristic approaches are desirable; see [6, 10, 12, 15].

Abbreviations
AWGN: Additive White Gaussian noise; BPSK: Binary phase-shift keying; CER: Codeword error rate; CRC: Cyclic redundancy check; LDPC: Low density parity check code; MRB: Most reliable bits; OSD: Ordered statistic decoding; SP: Sum-product; TC: Telecommand; IP: Integer programming.

Acknowledgements
The authors are grateful to the referees for their careful reading and invaluable comments. The first author (KW) is grateful to Professor Takeo Yamada for his careful reading and comments.

Authors' contributions
RK and TS carried out the simulation and tuned up the encoding/decoding algorithm and KW designed the encoding/decoding algorithm. All authors read and approved the final manuscript.

Data availability
Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Competing interests
The authors declare that they have no competing interests.

Received: 6 March 2020 Accepted: 12 January 2021
Published online: 05 February 2021

References

1. M. Baldi, F. Chiaraluce, N. Maturo, G. Liva, E. Paolini, A hybrid decoding scheme for short non-binary LDPC codes. IEEE Commun. Lett. **18**(12), 2093–2096 (2014)
2. M. Baldi, N. Maturo, E. Paolini, F. Chiaraluce, On the use of ordered statistics decoders for low-density parity-check codes in space telecommand links. EURASIP J Wirel Commun Netw **2016**, 272 (2016)
3. M. Beermann, T. Breddermann, P. Vary, Rate-compatible LDPC codes using optimized dummy bit insertion, in *8th International Symposium on Wireless Communication Systems* (2011). p. 447–451
4. CCSDS, Short Block Length LDPC Codes for TC Synchronization and Channel Coding, Orange Book. CCSDS 231.1-O-1 (2015)
5. CCSDS, TC Synchronization and Channel Coding, Blue Book. CCSDS 231.0-B-3 (2017)
6. D. Declercq, M.P.C. Fossorier, Improved impulse method to evaluate the low weight profile of sparse binary linear codes, in *2008 IEEE International Symposium on Information Theory, Toronto* (2008). p. 1963–1967
7. J. Feldman, M.J. Wainwright, D.R. Karger, Using linear programming to decode binary linear codes. IEEE Trans. Inf. Theory **51**(3), 954–972 (2005)
8. M.P.C. Fossorier, Iterative reliability-based decoding of low-density parity check codes. IEEE J. Sel. Areas Commun. **19**(5), 908–917 (2001)
9. S. Gounai, T. Ohtsuki, Lowering error floor of irregular LDPC codes by CRC and OSD algorithm. IEICE Trans. Commun. **E89–B**(1), 1–10 (2006)
10. X. Hu, M.P.C. Fossorier, E. Eleftheriou, On the computation of the minimum distance of low-density parity-check codes, in *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, **2** (2004). p. 767–771
11. M. Jiang, C. Zhao, E. Xu, L. Zhang, Reliability-based iterative decoding of LDPC codes using likelihood accumulation. IEEE Commun. Lett. **11**(8), 677–679 (2007)
12. J. Leon, A probabilistic algorithm for computing minimum weights of large error-correcting codes. IEEE Trans. Inf. Theory **IT–34**(5), 1354–1359 (1988)
13. S. Lin, D.J. Costello, *Error Control Coding*, 2nd edn. (Pearson, Hoboken, 2004)
14. J. Lim, D.J. Shin, A novel bit flipping decoder for systematic LDPC codes. IEICE Electron. Express **14**(2), 1–8 (2017)
15. J. Stern, A method for finding codewords of small weight, in *Coding Theory and Applications*, ed. by G. Cohen, J. Wolfmann (Springer, New York, 1989)
16. H. Wang, Q. Chen, Y. Zhang, On the LLR criterion based shortening design for LDPC codes, in *IEEE 2016 Annual Conference on Information Science and Systems* (2016). https://doi.org/10.1109/CISS.2016.7460482
17. A. Wongsriwor, V. Imtawil, P. Suttisopapan, Design of rate-compatible LDPC codes based on uniform shortening distribution. Eng. Appl. Sci. Res. **45**(2), 140–146 (2018)

Publisher's Note