PAPER *Special Section on Information Theory and Its Applications*

An Improved Method of Reliability-Based Maximum Likelihood Decoding Algorithms Using an Order Relation among Binary Vectors*

Hideki YAGI[†a)], ***Student Member***, **Manabu KOBAYASHI**[††], **Toshiyasu MATSUSHIMA**[†], ***Members***, ***and*** **Shigeichi HIRASAWA**[†], ***Fellow***

SUMMARY Reliability-based maximum likelihood decoding (MLD) algorithms of linear block codes have been widely studied. These algorithms efficiently search the most likely codeword using the generator matrix whose most reliable and linearly independent *k* (dimension of the code) columns form the identity matrix. In this paper, conditions for omitting unnecessary metrics computation of candidate codewords are derived in reliability-based MLD algorithms. The proposed conditions utilize an order relation of binary vectors. A simple method for testing if the proposed conditions are satisfied is devised. The method for testing proposed conditions requires no real number operations and, consequently, the MLD algorithm employing this method reduces the number of real number operations, compared to known reliability-based MLD algorithms.

key words: *maximum likelihood decoding, soft decision decoding, reliability measure, linear block codes, order relation*

1. Introduction

Maximum likelihood decoding (MLD) minimizes the block error probability of decoding when each codeword is equally likely to be transmitted. Since the complexity for performing MLD of block codes becomes impractically large as the code length becomes larger, many researchers have been devoted to reduce the complexity of MLD algorithms.

There are, in general, two types of efficient MLD algorithms. The first type is trellis-based MLD algorithms such as the Viterbi algorithm [11] or the recursive MLD algorithm [5]. Trellis-based MLD algorithms are "breadth-first" search algorithm [7] which mainly reduces the maximum number of computations. The latter type of efficient MLD algorithms is reliability-based MLD algorithms which iteratively generate candidate codewords. Reliability-based MLD algorithms are "depth-first" search algorithm which reduces the average number of computations and they are known to be efficient at moderate or high signal to noise ratio (SNR). One of well-known reliability-based MLD algorithms uses the bounded distance decoder (BDD) [1], [8], [10], [16] to generate candidate codewords. Objects of MLD algorithms using the BDD are codes with algebraic structure such as the BCH codes or the Goppa codes. Another reliability-based MLD algorithm uses the permuted generator matrix (PGM) of the code [6], [7], [12], [14], [17] to generate candidate codewords (Sub-optimum versions are found in [2]–[4], [13], [15]). MLD algorithms using the PGM are applicable to any binary linear block codes [15]. In this paper, we will focus on the reliability-based MLD algorithms using the PGM and we will call them, simply, the reliability-based MLD algorithms.

In the reliability-based MLD algorithms, test error patterns are iteratively generated to construct candidate codewords. Each time a new candidate codeword is constructed, metrics computation of it is carried out. In these algorithms, implicitly or explicitly, a sufficient condition for the optimality is tested. A sufficient condition for eliminating unnecessary test error patterns is also applied before they are encoded with the PGM. As a result, the reliability-based MLD algorithms require the relatively small number of candidate codewords and of their metrics computations. At low to moderate SNRs and for long codes, however, the number of candidate codewords for which the algorithm searches is still large. Therefore, the number of real number additions, subtractions and comparisons (hereafter, they will be called real number operations) is impractically large as the number of computing metrics of candidate codewords increases. We note that the total number of real number operations is one of the typical measures to evaluate the efficiency of MLD algorithms [4], [7].

In order to reduce the complexity of the reliability-based decoding algorithm where a large number of iterations (one iteration step consists of constructing candidate codewords and computing their metrics) are processed, we can consider the following two approaches: (1) reducing the number of iterations and (2) reducing the complexity for each iteration step. In this paper, we will concentrate ourselves on the latter approach. First, we define an order relation among binary vectors. Then we derive a sufficient condition for omitting unnecessary metrics computations of candidate codewords by using the defined order relation. A simple method for testing if the proposed conditions are satisfied is devised so that the test of proposed

Manuscript received January 19, 2004.
Manuscript revised June 3, 2004.
Final manuscript received June 28, 2004.

[†]The authors are with the Department of Industrial and Management Systems Engineering, School of Science and Engineering, Waseda University, Tokyo, 169-8555 Japan.

[††]The author is with the Department of Information Science, School of Engineering, Shonan Institute of Technology, Kanagawa, 251-8511 Japan.

a) E-mail: yagi@hirasa.mgmt.waseda.ac.jp

[*]The content of this work is partially based on [18].

conditions is implemented with increments of an integer or shift operations. In accordance with more likely codewords obtained, an *adaptive procedure* of the proposed condition, in which the codeword referenced by it is adaptively altered, is considered to make the proposed condition more effective. Testing the proposed conditions requires no real number operations and, as a result, the total number of real number operations for MLD is reduced. Finally, we show the effectiveness of the proposed conditions while the decoding algorithm employing the proposed conditions has no degradation of the error performance.

This paper is organized as follows. In Sect. 2, the general framework of the reliability-based MLD algorithm is briefly reviewed as a preliminary. In Sect. 3, a sufficient condition for omitting unnecessary metrics computation of candidate codewords is derived. Then, an adaptive procedure for implementing the proposed method is presented. Some simulation results are shown in Sect. 4 to demonstrate the effectiveness of the proposed conditions and concluding remarks are stated in Sect. 5

2. The Reliability-Based MLD Algorithm

2.1 Preliminary

For integers j_1 and j_2 such that $j_1 \leq j_2$, let $[j_1, j_2]$ denote the set of positive integers from j_1 to j_2. For binary vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_\alpha)$ of finite length α, let $w_H(\boldsymbol{x})$ and $supp(\boldsymbol{x})$ be, respectively, the Hamming weight of $\boldsymbol{x}$ and the support of $\boldsymbol{x}$ defined as $supp(\boldsymbol{x}) = \{j \,|\, x_j = 1\}$. For a set X, let $|X|$ be the cardinality of X.

Let $\mathcal{V}^n$ denote a set of all binary n-dimensional vectors. Let $C \subseteq \mathcal{V}^n$ be a binary linear (n, k, d) block code with length n, dimension k and minimum distance d. Let G be a generator matrix of C. Assume that each codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in C$ has equal probability to be transmitted over the Additive White Gaussian Noise (AWGN) channel with the signal to noise ratio (SNR) E_b/N_0 [dB]. The detector projects the received sequence $\boldsymbol{r} = (r_1, r_2, \ldots, r_n) \in \mathcal{R}^n$ into a sequence $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n) \in \mathcal{R}^n$ such that $\theta_j = \ln \frac{P(r_j|c_j=0)}{P(r_j|c_j=1)}$, $j \in [1, n]$, and delivers $\boldsymbol{\theta}$ into the decoder. Let $\boldsymbol{z} = (z_1, z_2, \ldots, z_n) \in \mathcal{V}^n$ be the hard decision received sequence of $\boldsymbol{\theta}$ such that

$$z_j = \begin{cases} 0, & \text{if } \theta_j \geq 0; \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

The decoder estimates a transmitted codeword from both $\boldsymbol{\theta}$ and $\boldsymbol{z}$. For $j \in [1, n]$, an error probability of the symbol z_j, $P(z_j \neq c_j | r_j)$, is smaller as the value $|\theta_j|$ becomes larger. Therefore, we call $|\theta_j|$ *reliability measure* of j-th symbol.

For any $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{V}^n$, let $L(\boldsymbol{x})$ be the *reliability loss* with respect to $\boldsymbol{z}$ defined as

$$L(\boldsymbol{x}) = \sum_{j=1}^{n} (x_j \oplus z_j)|\theta_j|, \tag{2}$$

where $\oplus$ represents the exclusive OR operation. For $\boldsymbol{x} \in \mathcal{V}^n$, $L(\boldsymbol{x})$ is also known as *correlation discrepancy* [9], [10], [15]. For a subspace X of $\mathcal{V}^n$, let $\underline{L}[X]$ be defined as

$$\underline{L}[X] = \min_{\boldsymbol{x} \in X} L(\boldsymbol{x}). \tag{3}$$

Then $L(\boldsymbol{c}_{ML}) = \underline{L}[C]$ if and only if $\boldsymbol{c}_{ML} \in C$ is the most likely (ML) codeword [9], [10], [12]. i.e., a codeword which has the smallest reliability loss is the closest codeword from $\boldsymbol{\theta}$.

2.2 General Framework of the Reliability-Based MLD Algorithm

After receiving $\boldsymbol{\theta}$, the decoder reorders positions of $\boldsymbol{\theta}$ in the non-increasing order of reliability measure. We denote the resultant sequence with $\overline{\boldsymbol{\theta}} = \lambda(\boldsymbol{\theta})$ where λ is the permutation function from $\boldsymbol{\theta}$ to $\overline{\boldsymbol{\theta}}$. i.e., $|\overline{\theta}_{j_1}| \geq |\overline{\theta}_{j_2}|$, $1 \leq j_1 < j_2 \leq n$. Let G' be the column-permuted generator matrix given by the same ordering of $\overline{\boldsymbol{\theta}}$.

For a location set $X \subseteq [1, n]$, let G'_X be the $k \times |X|$ matrix which consists of columns of G' over X. Define

$$\mathcal{M} = \arg\max_{X} \Big\{ \sum_{j \in X} |\overline{\theta}_j| \,\Big|\, |X| = k, rank(G'_X) = k \Big\}. \tag{4}$$

Then $\mathcal{M}$ is called the *k most reliable and linearly independent* (MRI) positions, i.e., the sum of reliability measures of the k MRI positions are the largest among that of any other k linearly independent positions. For G', the elementary row operations are carried out so that the k MRI columns form the identity matrix. The resultant generator matrix is denoted with $\overline{G}$. Let $\overline{C}$ be the code given by $\overline{G}$ which is equivalent to C. Furthermore, let $\overline{z} = \lambda(z)$. Let $\overline{\mathcal{V}}^n$ denote the set of binary vectors such that $\overline{\mathcal{V}}^n = \{\overline{\boldsymbol{x}} = \lambda(\boldsymbol{x}) \,|\, \boldsymbol{x} \in \mathcal{V}^n\}$, i.e., any $\overline{\boldsymbol{x}} \in \overline{\mathcal{V}}^n$ is permuted in the non-increasing order of reliability.

Define that $\boldsymbol{u} = (u_1, u_2, \ldots, u_k) \in \{0, 1\}^k$ consists of the k MRI symbols of $\overline{z}$ in non-increasing order of reliability. The sequence $\boldsymbol{u}$ is regarded as an information sequence and the decoder constructs the initial codeword $\overline{\boldsymbol{c}}_0$ by $\overline{\boldsymbol{c}}_0 = \boldsymbol{u}\overline{G}$. Remark that $\overline{\boldsymbol{c}}_0$ is the ML codeword if $\overline{\boldsymbol{c}}_0 = \overline{z}$ [10], [12]. If $L(\overline{\boldsymbol{c}}_0) > 0$, the decoder iteratively constructs candidate codewords by $\overline{G}$ and searches the ML codeword which minimizes Eq. (2).

Definition 1: For $0 \leq i \leq 2^k$, k-dimensional vector $\boldsymbol{t}_i \in \{0, 1\}^k$ is called i-th *test error pattern*. A codeword $\overline{\boldsymbol{w}}_i = (\overline{w}_{i,1}, \overline{w}_{i,2}, \ldots, \overline{w}_{i,n}) = \boldsymbol{t}_i\overline{G}$ is called a *test error codeword* which gives a candidate codeword $\overline{\boldsymbol{c}}_i = \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}_i$. A candidate codeword $\overline{\boldsymbol{c}}_i$ (or a test error codeword $\overline{\boldsymbol{w}}_i$) is said to be *better* than $\overline{\boldsymbol{c}}_{i'}$ (or $\overline{\boldsymbol{w}}_{i'}$), if and only if $L(\overline{\boldsymbol{c}}_i) < L(\overline{\boldsymbol{c}}_{i'})$. For a subset $\overline{C}'$ of $\overline{C}$, a candidate codeword $\overline{\boldsymbol{c}}_i$ and a test error codeword $\overline{\boldsymbol{w}}_i$ are said to be the *best* in $\overline{C}'$ if and only if $L(\overline{\boldsymbol{c}}_i) = \underline{L}[\overline{C}']$.

Let $\boldsymbol{t}_0 = 0^k$ where 0^k is k-dimensional all zero vector. Then $\boldsymbol{t}_0$ can be regarded as the test error pattern corresponding to the initial codeword $\overline{\boldsymbol{c}}_0$ since $\overline{\boldsymbol{c}}_0 = \overline{\boldsymbol{c}}_0 \oplus \boldsymbol{t}_0\overline{G} = \overline{\boldsymbol{c}}_0 \oplus 0^n$. For given $\overline{G}$ and $\overline{\boldsymbol{c}}_0$, it is obvious that there is one to one correspondence between $\boldsymbol{t}_i$ and $\overline{\boldsymbol{c}}_i$. Then the order of searching candidate codewords depends on that of generating test error patterns. Efficient orders of generating test error patterns have been devised [6], [7], [14].

Let $\overline{C}_s$ be a set of codewords which includes all candidate codewords $\overline{\boldsymbol{c}}_i = \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}_i$ such that $0 \le i < s$ at a decoding stage of generating $\boldsymbol{t}_s$, i.e.,

$$\overline{C}_s = \{\overline{\boldsymbol{c}}_i = \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}_i \,\big|\, \overline{\boldsymbol{w}}_i = \boldsymbol{t}_i\overline{G}, 0 \le i < s\}. \qquad (5)$$

For a test error pattern $\boldsymbol{t}_i$, let $F(\boldsymbol{t}_i)$ express arbitrary evaluation function of $\boldsymbol{t}_i$ satisfying

$$0 \le F(\boldsymbol{t}_i) \le L(\overline{\boldsymbol{c}}_i), \qquad (6)$$

where $\overline{\boldsymbol{c}}_i = \overline{\boldsymbol{c}}_0 \oplus \boldsymbol{t}_i\overline{G}$. Several evaluation functions have been proposed [2], [6], [7], [12].

At a decoding stage of generating $\boldsymbol{t}_i$, we need not to encode $\boldsymbol{t}_i$ if

$$\underline{L}[\overline{C}_i] \le F(\boldsymbol{t}_i), \qquad (7)$$

since $\overline{\boldsymbol{c}}_i$ cannot be better than the best candidate codeword obtained so far. i.e., if Eq. (7) holds, $\boldsymbol{t}_i$ cannot give the best candidate codeword.

For $0 \le \forall i \le 2^k$, let $\overline{\boldsymbol{e}}_i = (\overline{e}_{i,1}, \overline{e}_{i,2}, \dots, \overline{e}_{i,n})$ be such that $\overline{\boldsymbol{e}}_i = \overline{\boldsymbol{z}} \oplus \overline{\boldsymbol{c}}_i$. For $\overline{\boldsymbol{w}}_i \in \overline{C}$, let $\Lambda(\overline{\boldsymbol{w}}_i)$ be defined as

$$\Lambda(\overline{\boldsymbol{w}}_i) = \sum_{j \in supp(\overline{\boldsymbol{w}}_i)} (1 - 2\overline{e}_{0,j})|\overline{\theta}_j|. \qquad (8)$$

Then, for $\overline{\boldsymbol{c}}_i (= \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}_i)$, we can compute $L(\overline{\boldsymbol{c}}_i)$ by

$$L(\overline{\boldsymbol{c}}_i) = L(\overline{\boldsymbol{c}}_0) + \Lambda(\overline{\boldsymbol{w}}_i), \qquad (9)$$

since from $\overline{\boldsymbol{z}} \oplus \overline{\boldsymbol{c}}_i = \overline{\boldsymbol{e}}_0 \oplus \overline{\boldsymbol{w}}_i$,

$$\begin{aligned} L(\overline{\boldsymbol{c}}_i) &= \sum_{j=1}^{n} (\overline{e}_{0,j} \oplus \overline{w}_{i,j})|\overline{\theta}_j| \\ &= \sum_{j=1}^{n} \overline{e}_{0,j}|\overline{\theta}_j| + \sum_{j=1}^{n} (1 - 2\overline{e}_{0,j})\overline{w}_{i,j}|\overline{\theta}_j| \\ &= L(\overline{\boldsymbol{c}}_0) + \sum_{j \in supp(\overline{\boldsymbol{w}}_i)} (1 - 2\overline{e}_{0,j})|\overline{\theta}_j|. \qquad (10) \end{aligned}$$

By Eq. (9), for a fixed $\overline{\boldsymbol{c}}_0$, searching $\overline{\boldsymbol{c}}_i$ which minimizes $L(\overline{\boldsymbol{c}}_i)$ is equivalent to searching $\overline{\boldsymbol{w}}_i$ which minimizes $\Lambda(\overline{\boldsymbol{w}}_i)$.

We describe a general version of the reliability-based MLD algorithm below. For an integer α, let α++ denote the increment operation of α.

[The reliability-based MLD Algorithm]

```
1) Generate $\overline{\boldsymbol{c}}_0 := \boldsymbol{u}\overline{G}$, and set $\underline{L} := L(\overline{\boldsymbol{c}}_0)$, $\overline{\boldsymbol{w}}^* := 0^n$, $\underline{\Lambda} := 0$
   and $i := 1$.
2) Generate $\boldsymbol{t}_i$ and compute $F(\boldsymbol{t}_i)$. If $\underline{L} \le F(\boldsymbol{t}_i)$, then go to
   4).
3) Generate $\overline{\boldsymbol{w}}_i := \boldsymbol{t}_i\overline{G}$ and compute $\Lambda(\overline{\boldsymbol{w}}_i)$. If $\Lambda(\overline{\boldsymbol{w}}_i) < \underline{\Lambda}$,
   then $\underline{\Lambda} := \Lambda(\overline{\boldsymbol{w}}_i)$, $\underline{L} := L(\overline{\boldsymbol{c}}_0) + \underline{\Lambda}$ and $\overline{\boldsymbol{w}}^* := \overline{\boldsymbol{w}}_i$.
4) Set $i$++. If $i \le 2^k$ and a certain terminating criterion does
   not hold, then go to 2), otherwise output $\overline{\boldsymbol{c}}_{ML} := \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}^*$
   and stop.                                                      □
```

As for a terminating criterion of the decoding algorithm at step 4), several criteria have been proposed [2], [3], [6], [7], [12].

We here state the complexity of the reliability-based decoding algorithm. The time complexity of permuting $\boldsymbol{\theta}$ in the non-increasing order is $O(n \log n)$ and of constructing $\overline{G}$ is $O(n \times \kappa^2)$ where $\kappa = \min\{k, n-k\}$ [2], [6], [7]. These steps are carried out only once in a decoding procedure. Contrary to the above steps, generating $\boldsymbol{t}_i$ and constructing $\overline{\boldsymbol{w}}_i = \boldsymbol{t}_i\overline{G}$ are carried out iteratively, where each encoding requires binary operations of $O(kn)$ by conventional encoding method [2], [12]. For each test error codeword constructed, computing Eq. (8) costs real number operations of $O(n)$. Therefore, both encoding test error patterns and the real number operations of step 3) dominate mainly the whole decoding complexity [4], [7], [12]. As for the space complexity, storing $\overline{G}$ requires $O(kn)$. In some MLD algorithms [7], [12], [14], the test error patterns are stored in a list before encoded by $\overline{G}$. In these algorithms, denoting the maximum list size for decoding $\boldsymbol{r}$ by $N(\boldsymbol{r})$, the space complexity is $O(\gamma)$ where $\gamma = \max\{kn, N(\boldsymbol{r})\}$.

3. Proposed Methods Using an Order Relation

3.1 Conditions for Omitting Unnecessary Metrics Computations

We will develop the method for reducing the complexity of the reliability-based decoding algorithms by exploiting the following two properties of the decoding algorithm: (1) Every n-dimensional sequence is permuted in the non-increasing order of reliability measure, (2) at least one codeword (the initial codeword) is obtained before generating each test error codeword.

We consider the case in which $\boldsymbol{t}_i$ does not satisfy Eq. (7) and is encoded to $\overline{\boldsymbol{w}}_i$ in a decoding procedure. If we find out $\overline{\boldsymbol{w}}_i$ cannot give the best codeword, then the computation of Eq. (8) (which is the metrics computation of $\overline{\boldsymbol{w}}_i$) can be omitted. Roughly speaking, we measure a distance[†] (defined over $\overline{\mathcal{V}}^n$ like the Hamming distance) between $\overline{\boldsymbol{c}}_i$ and $\overline{\boldsymbol{\theta}}$ and that between $\overline{\boldsymbol{c}}_0$ and $\overline{\boldsymbol{\theta}}$. If $\overline{\boldsymbol{c}}_i$ is obviously farther from $\overline{\boldsymbol{\theta}}$ than $\overline{\boldsymbol{c}}_0$, we eliminate $\overline{\boldsymbol{c}}_i$ from consideration without computing its metrics. We will derive a condition that guarantees a test error codeword $\overline{\boldsymbol{w}}_i$ which cannot give the ML codeword.

We define the following order relations:

Definition 2: (**The Order Relation for Supports**) For two location sets $X = \{j_1, j_2, \dots, j_m\}$ and $X' = \{j'_1, j'_2, \dots, j'_{m'}\}$ such that $j_1 < j_2 < \cdots < j_m$ and $j'_1 < j'_2 < \cdots < j'_{m'}$, we write "$X' <_S X$" if $m' \le m$ and $j_h \le j'_h, \forall h \in [1, m']$.

Definition 3: (**The Order Relation for Binary Vectors**) For two vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$, we write "$\boldsymbol{x}' <_V \boldsymbol{x}$" if and only if $supp(\boldsymbol{x}') <_S supp(\boldsymbol{x})$.

For two vectors $\overline{\boldsymbol{x}} = (\overline{x}_1, \overline{x}_2, \dots, \overline{x}_n) \in \overline{\mathcal{V}}^n$ and $\overline{\boldsymbol{x}}' =$

[†] It will be defined in Definition 2 and 3, although it does not satisfy an axiom of the distance measure.

$(\overline{x}'_1, \overline{x}'_2, \ldots, \overline{x}'_n) \in \overline{\mathcal{V}}^n$, define two location sets as

$$\mathcal{D}_0(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = \{j \,|\, \overline{x}_j = 1 \text{ and } \overline{x}'_j = 0\}, \tag{11}$$

$$\mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = \{j \,|\, \overline{x}_j = 1 \text{ and } \overline{x}'_j = 1\}. \tag{12}$$

Theorem 1: For a test error codeword $\overline{\boldsymbol{w}}_i$, assume that there is an order relation such that

$$\mathcal{D}_1(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) <_S \mathcal{D}_0(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0). \tag{13}$$

Then $\overline{\boldsymbol{c}}_i (= \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}_i)$ cannot be better than $\overline{\boldsymbol{c}}_0$.

Proof: Assume that $\mathcal{D}_0(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = \{j_1, j_2, \ldots, j_m\}$ and $\mathcal{D}_1(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = \{j'_1, j'_2, \ldots, j'_{m'}\}$ such that $j_1 < j_2 < \cdots < j_m$ and $j'_1 < j'_2 < \cdots < j'_{m'}$. For each element of $\mathcal{D}_\alpha(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0)$, $\alpha \in \{0, 1\}$ satisfies

$$|\overline{\theta}_{j_1}| \geq |\overline{\theta}_{j_2}| \geq \cdots \geq |\overline{\theta}_{j_m}|, \tag{14}$$

$$|\overline{\theta}_{j'_1}| \geq |\overline{\theta}_{j'_2}| \geq \cdots \geq |\overline{\theta}_{j'_{m'}}|. \tag{15}$$

By the assumption of Eq. (13), $m' \leq m$ and

$$|\overline{\theta}_{j_h}| \geq |\overline{\theta}_{j'_h}|, \qquad \text{for } h \in [1, m']. \tag{16}$$

Equation (8) is now

$$\begin{aligned} \Lambda(\overline{\boldsymbol{w}}_i) &= 6 \sum_{j=1}^{n} (1 - 2\overline{e}_{0,j}) \overline{w}_{i,j} |\overline{\theta}_j| \\ &= \sum_{j \,|\, \overline{e}_{0,j}=0} \overline{w}_{i,j} |\overline{\theta}_j| - \sum_{j \,|\, \overline{e}_{0,j}=1} \overline{w}_{i,j} |\overline{\theta}_j| \\ &= \sum_{j \in \mathcal{D}_0(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0)} |\overline{\theta}_j| - \sum_{j \in \mathcal{D}_1(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0)} |\overline{\theta}_j|. \end{aligned} \tag{17}$$

Therefore $\Lambda(\overline{\boldsymbol{w}}_i) \geq 0$ by Eq. (16). Hence $L(\overline{\boldsymbol{c}}_i) = L(\overline{\boldsymbol{c}}_0) + \Lambda(\overline{\boldsymbol{w}}_i) \geq L(\overline{\boldsymbol{c}}_0)$ and $\overline{\boldsymbol{c}}_i$ cannot be better than $\overline{\boldsymbol{c}}_0$. □

If the order relation of Eq. (13) is satisfied, $\overline{\boldsymbol{c}}_i$ given by $\overline{\boldsymbol{w}}_i$ is farther from $\overline{\boldsymbol{\theta}}$ than $\overline{\boldsymbol{c}}_0$. i.e., $\overline{\boldsymbol{c}}_i$ cannot be the ML codeword. Equation (13) can be used for judging if the metrics of a candidate codeword need not to be computed. Hereafter, we call this order relation of Eq. (13) *Omitting Criterion A*.

We now present a method for testing if an order relation

$$\mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') <_S \mathcal{D}_0(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}'), \tag{18}$$

holds for $\overline{\boldsymbol{x}} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n) \in \overline{\mathcal{V}}^n$ and $\overline{\boldsymbol{x}}' = (\overline{x}'_1, \overline{x}'_2, \ldots, \overline{x}'_n) \in \overline{\mathcal{V}}^n$. For $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \{0, 1\}^n$, let $\boldsymbol{a} \gg$ and $\boldsymbol{a} \ll$ be the right and the left shift operation by one bit, respectively. The algorithm can be performed by increments of an integer and shift operations of a binary array.

[Procedure $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$]

```
1) Set $\boldsymbol{a} := (0, 1, 0, \ldots, 0)$ and $\tau := 1$.
2) If $\overline{x}_\tau = 1$ and $\overline{x}'_\tau = 0$, then $\boldsymbol{a} \gg$. If $\overline{x}_\tau = 1$ and $\overline{x}'_\tau = 1$,
   then $\boldsymbol{a} \ll$.
3) If $a_1 = 1$, then $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') := 1$ and stop. If $\tau = n$, then
   $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') := 0$ and stop. Otherwise, set $\tau{+}{+}$ and go to
   2). □
```

Note that for an integer α, $\alpha{+}{+}$ denotes the increment of α. In the above algorithm, we denote a returned value with $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') \in \{0, 1\}$. If Eq. (18) is satisfied, the algorithm returns $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = 0$ (the validity of the algorithm will be given below). Otherwise, it returns $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = 1$. Remark that $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ denotes either testing Eq. (18) or a returned value of the test.

At step 2) in the above algorithm, τ such that $\overline{x}_\tau = 1$ and $\overline{x}'_\tau = 0$ is an element of $\mathcal{D}_0(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$. Similarly, τ such that $\overline{x}_\tau = 1$ and $\overline{x}'_\tau = 1$ is an element of $\mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$. i.e., the procedure of step 2) means:

(1) if we find $\tau \in \mathcal{D}_0(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$, then we set $\boldsymbol{a} \gg$,
(2) if we find $\tau \in \mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$, then we set $\boldsymbol{a} \ll$.

Remark that we can realize shift operations of $\boldsymbol{a} \gg$ and $\boldsymbol{a} \ll$ much easier than ordinary shift operations of binary array of size n since $w_H(\boldsymbol{a}) = 1$. It is enough that the element one is moved by one bit and this shift can be accomplished by two exclusive OR operations. We also remark that we can describe Procedure $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ using a variable ρ, which keeps the position of the element 1 in $\boldsymbol{a}$, instead of using the vector $\boldsymbol{a}$. In that case, we initially set $\rho = 2$. The left and right shift operation in Procedure $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ can be expressed by the decrement and increment of ρ, respectively.

We will show the validity of the above algorithm.

Theorem 2: For $\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}' \in \overline{\mathcal{V}}^n$, the returned value is $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = 0$ if and only if Eq. (18) holds.

Proof: First, we will prove if part. We assume that Eq. (18) holds. Let $\mathcal{D}_0(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = \{j_1, j_2, \ldots, j_m\}$ and $\mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = \{j'_1, j'_2, \ldots, j'_{m'}\}$ such that $j_1 < j_2 < \cdots < j_m$ and $j'_1 < j'_2 < \cdots < j'_{m'}$. The algorithm searches $j_h \in \mathcal{D}_0(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ or $j'_h \in \mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ from left position to right one. For any $h \in [1, m']$, before we encounter $\tau = j'_h \in \mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$, we have already found $j_h \in \mathcal{D}_0(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ since $j_h < j'_h$, $\forall h \in [1, m']$, from Eq. (18). Therefore, after we encounter $\tau = j'_h \in \mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ for $h \in [1, m']$, j such that $a_j = 1$ is necessarily greater than one, i.e., $j > 1$. The condition $a_1 = 1$ does not hold for all positions $\tau, \forall \tau \in [1, n]$. Since τ is incremented up to n, the returned value is $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = 0$.

Next, we will prove only if part. We assume $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = 0$. We here assume there exist a certain h^* such that $j'_{h^*} < j_{h^*}$ and we will prove the theorem by contradiction. If $h^* = 1$, then we set $\boldsymbol{a} \ll$ and $a_1 = 1$ holds when encountering $\tau = j'_1 \in \mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ at step 2). Then h^* should be greater than one. If $h^* > 1$, then $j_{h^*-1} < j'_{h^*-1} < j'_{h^*} < j_{h^*}$. When we encounter $\tau = j'_{h^*-1} \in \mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ at step 2), we set $\boldsymbol{a} \ll$ and $a_2 = 1$ holds since we have already found exactly $h^* - 1$ elements of each $\mathcal{D}_0(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ and $\mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$. Therefore, when encountering $\tau = j'_{h^*} \in \mathcal{D}_1(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ at step 2), we set $\boldsymbol{a} \ll$ and $a_1 = 1$ holds. At step 3), the algorithm returns $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = 1$. This contradicts the assumption, $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}') = 0$. Hence $j_h < j'_h$ must be satisfied for $\forall h \in [1, m']$ and Eq. (18) holds. □

Testing Omitting Criterion A is denoted with $OT(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0)$.

Corollary 1: For $\overline{\boldsymbol{w}}_i$ and $\overline{\boldsymbol{e}}_0$, the returned value is $OT(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = 0$ if and only if Eq. (13) holds. When $OT(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = 0$, $\overline{\boldsymbol{c}}_i (= \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}_i)$ cannot be better than $\overline{\boldsymbol{c}}_0$.

By Corollary 1, if $OT(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = 0$ for $\overline{\boldsymbol{w}}_i$, we can omit the computation of Eq. (17) for $\overline{\boldsymbol{w}}_i$. It is obvious that the time complexity for performing $OT(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0)$ is increments of an integer and shift operations of $O(n)$. Note that increments of an integer are also necessary for encoding and computing metrics if we realize them serially by software. Since the computation of Eq. (17) costs real number operations of $O(n)$, the time complexity of testing Omitting Criterion A is fairly small. As for the space complexity, we need to store $\overline{\boldsymbol{e}}_0$ and this requires a binary array of size $O(n)$. In the reliability-based MLD algorithm, this number is negligible since storing only $\overline{G}$ requires a binary array of size $O(kn)$.

Example 1: Let $\overline{\boldsymbol{w}}_i = (00011011)$ and $\overline{\boldsymbol{e}}_0 = (00001010)$. Then $\mathcal{D}_0(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = \{4, 8\}$ and $\mathcal{D}_1(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = \{5, 7\}$. First, we find $\tau = 4 \in \mathcal{D}_0(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0)$, then we set $a_2 := 0$ and $a_3 := 1$ by $\boldsymbol{a} \gg$. Next, we find $\tau = 5 \in \mathcal{D}_1(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0)$, then we set $a_2 := 1$ and $a_3 := 0$ by $\boldsymbol{a} \ll$. When we find $\tau = 7 \in \mathcal{D}_1(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0)$, we set $a_1 := 1$ and $a_2 := 0$ by $\boldsymbol{a} \ll$. Since $a_1 = 1$, $OT(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = 1$ is returned.

Example 2: Let $\overline{\boldsymbol{w}}_i = (00101110)$ and $\overline{\boldsymbol{e}}_0 = (00001010)$. Then $\mathcal{D}_0(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = \{j_1 = 3, j_2 = 6\}$ and $\mathcal{D}_1(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = \{j'_1 = 5, j'_2 = 7\}$. Therefore, from $j_1 < j'_1$ and $j_2 < j'_2$, Eq. (13) holds. When we increment τ up to $n = 8$, the algorithm returns $OT(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = 0$.

We describe the reliability-based MLD algorithm employing the test of Omitting Criterion A. We will call this decoding algorithm the proposed decoding algorithm A.

[The Proposed Decoding Algorithm A]

```
1) Generate $\overline{\boldsymbol{c}}_0 := \boldsymbol{u}\overline{G}$, and set $\underline{L} := L(\overline{\boldsymbol{c}}_0)$, $\overline{\boldsymbol{e}}_0 := \overline{\boldsymbol{z}} \oplus \overline{\boldsymbol{c}}_0$,
   $\overline{\boldsymbol{w}}^* := 0^n$, $\underline{\Lambda} := 0$ and $i := 1$.
2) Generate $\boldsymbol{t}_i$ and compute $F(\boldsymbol{t}_i)$. If $\underline{L} \leq F(\boldsymbol{t}_i)$, then go to 4).
3) a) Generate $\overline{\boldsymbol{w}}_i := \boldsymbol{t}_i\overline{G}$. If $OT(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0) = 0$, then go to 4).
   b) Compute $\Lambda(\overline{\boldsymbol{w}}_i)$. If $\Lambda(\overline{\boldsymbol{w}}_i) < \underline{\Lambda}$, then $\underline{\Lambda} := \Lambda(\overline{\boldsymbol{w}}_i)$, $\underline{L} :=$
      $L(\overline{\boldsymbol{c}}_0) + \underline{\Lambda}$ and $\overline{\boldsymbol{w}}^* := \overline{\boldsymbol{w}}_i$.
4) Set $i$++. If $i \leq 2^k$ and a certain terminating criterion does not hold, then go to 2), otherwise output $\overline{\boldsymbol{c}}_{ML} := \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}^*$
   and stop. □
```

In the above algorithm, step 1) and 3) is modified to the original reliability-based decoding algorithm. At step 3)a), if $OT(\overline{\boldsymbol{w}}_i) = 0$, real number operations at step 3)b), which include the computation of Eq. (17) and one addition and comparison, are omitted.

3.2 Adaptive Procedure of the Proposed Conditions

Theorem 1 implies that Omitting Criterion A compares $\overline{\boldsymbol{c}}_i$ with $\overline{\boldsymbol{c}}_0$ and judges if $\overline{\boldsymbol{c}}_i$ is farther from $\overline{\boldsymbol{\theta}}$ than $\overline{\boldsymbol{c}}_0$. In a decoding procedure, let $\overline{\boldsymbol{c}}^*$ denote the best candidate codeword obtained so far such that $\overline{\boldsymbol{c}}^* = \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}^*$. At a decoding stage of constructing $\overline{\boldsymbol{w}}_i$, the best candidate codeword $\overline{\boldsymbol{c}}^*$ is not necessarily equal to $\overline{\boldsymbol{c}}_0$ and such $\overline{\boldsymbol{c}}^*$ is closer to $\overline{\boldsymbol{\theta}}$ than $\overline{\boldsymbol{c}}_0$. If we can compare $\overline{\boldsymbol{c}}_i$ with $\overline{\boldsymbol{c}}^*$ (not with $\overline{\boldsymbol{c}}_0$) and we test whether $\overline{\boldsymbol{c}}_i$ is farther from $\overline{\boldsymbol{\theta}}$ than $\overline{\boldsymbol{c}}^*$, a sufficient condition for omitting unnecessary metrics computation may be more effective. We will consider an *adaptive procedure* in which $\overline{\boldsymbol{c}}^*$ referenced by the proposed condition is adaptively altered.

At a decoding stage of constructing $\overline{\boldsymbol{w}}_i$, let $\overline{\boldsymbol{e}}^* = (\overline{e}^*_1, \overline{e}^*_2, \ldots, \overline{e}^*_n)$ be such that $\overline{\boldsymbol{e}}^* = \overline{\boldsymbol{z}} \oplus \overline{\boldsymbol{c}}^*$. Furthermore, let $\overline{\boldsymbol{v}}_i = \overline{\boldsymbol{w}}^* \oplus \overline{\boldsymbol{w}}_i$.

Lemma 1: Using $\overline{\boldsymbol{c}}^*$ and $\overline{\boldsymbol{v}}_i = \overline{\boldsymbol{w}}^* \oplus \overline{\boldsymbol{w}}_i$, $L(\overline{\boldsymbol{c}}_i)$ is expressed as follows:

$$L(\overline{\boldsymbol{c}}_i) = L(\overline{\boldsymbol{c}}^*) + \sum_{j \in supp(\overline{\boldsymbol{v}}_i)} (1 - 2\overline{e}^*_j)|\overline{\theta}_j|. \tag{19}$$

Equation (19) shows the relation between $L(\overline{\boldsymbol{c}}^*)$ and $L(\overline{\boldsymbol{c}}_i)$.

Proof: Since $\overline{\boldsymbol{c}}_0 = \overline{\boldsymbol{c}}^* \oplus \overline{\boldsymbol{w}}^*$ and $\overline{\boldsymbol{e}}^* = \overline{\boldsymbol{z}} \oplus \overline{\boldsymbol{c}}^*$, the left hand side (l.h.s.) of Eq. (19) expands in the following way:

$$\begin{aligned} L(\overline{\boldsymbol{c}}_i) &= L(\overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}_i) = L(\overline{\boldsymbol{c}}^* \oplus \overline{\boldsymbol{w}}^* \oplus \overline{\boldsymbol{w}}_i) \\ &= \sum_{j=1}^{n} (\overline{e}^*_j \oplus \overline{w}^*_j \oplus \overline{w}_{i,j})|\overline{\theta}_j| \\ &= \sum_{j=1}^{n} \overline{e}^*_j|\overline{\theta}_j| + \sum_{j=1}^{n} (1 - 2\overline{e}^*_j)(\overline{w}^*_j \oplus \overline{w}_{i,j})|\overline{\theta}_j| \\ &= L(\overline{\boldsymbol{c}}^*) + \sum_{j \,|\, \overline{w}^*_j \oplus \overline{w}_{i,j} = 1} (1 - 2\overline{e}^*_j)|\overline{\theta}_j|. \end{aligned} \tag{20}$$

Hence we have Eq. (19). □

Theorem 3: For a test error codeword $\overline{\boldsymbol{w}}_i$, assume that there is an order relation such that

$$\mathcal{D}_1(\overline{\boldsymbol{v}}_i, \overline{\boldsymbol{e}}^*) <_S \mathcal{D}_0(\overline{\boldsymbol{v}}_i, \overline{\boldsymbol{e}}^*). \tag{21}$$

Then $\overline{\boldsymbol{c}}_i (= \overline{\boldsymbol{c}}_0 \oplus \boldsymbol{w}_i)$ cannot be better than $\overline{\boldsymbol{c}}^*$.

Proof: We can prove the theorem in a similar way of proving Theorem 1 by using Lemma 1. □

Theorem 3 implies that $\overline{\boldsymbol{c}}_i$, given by $\overline{\boldsymbol{w}}_i$, is farther from $\overline{\boldsymbol{\theta}}$ than $\overline{\boldsymbol{c}}^*$ if Eq. (21) holds for $\overline{\boldsymbol{w}}_i$. Then $\overline{\boldsymbol{w}}_i$ cannot give the ML codeword. Therefore we need not compute metrics of $\overline{\boldsymbol{w}}_i$ which satisfies Eq. (21).

In general, for $\overline{\boldsymbol{e}}^* \neq \overline{\boldsymbol{e}}_0$, $w_H(\overline{\boldsymbol{e}}^*)$ tends to be smaller than $w_H(\overline{\boldsymbol{e}}_0)$ [3]. This implies that $|\mathcal{D}_1(\overline{\boldsymbol{v}}_i, \overline{\boldsymbol{e}}^*)|$ tends to be smaller than $|\mathcal{D}_1(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0)|$. Consequently, Eq. (21) can be a more effective condition than Omitting Criterion A since Eq. (13) or Eq. (21) is satisfied more often, as the cardinality of its l.h.s. is smaller. We will call the order relation of Eq. (21) *Omitting Criterion B*. For $\overline{\boldsymbol{w}}_i$ and $\overline{\boldsymbol{w}}^*$, testing Omitting Criterion B is denoted with $OT(\overline{\boldsymbol{v}}_i, \overline{\boldsymbol{e}}^*)$.

We describe the reliability-based MLD algorithm employing Omitting Criterion B in which step 1) and 3) is modified to the original MLD algorithm. We will call this decoding algorithm the proposed decoding algorithm B.

[The Proposed Decoding Algorithm B]

```
1) Generate $\overline{\boldsymbol{c}}_0 := \boldsymbol{u}\overline{G}$, and set $\underline{L} := L(\overline{\boldsymbol{c}}_0)$, $\overline{\boldsymbol{e}}_0 := \overline{\boldsymbol{z}} \oplus \overline{\boldsymbol{c}}_0$, $\overline{\boldsymbol{w}}^* := 0^n$, $\overline{\boldsymbol{e}}^* := 0^n$, $\underline{\Lambda} := 0$ and $i := 1$.
2) Generate $\boldsymbol{t}_i$ and compute $F(\boldsymbol{t}_i)$. If $\underline{L} \leq F(\boldsymbol{t}_i)$, then go to 4).
3) a) Generate $\overline{\boldsymbol{w}}_i := \boldsymbol{t}_i\overline{G}$ and set $\overline{\boldsymbol{v}}_i := \overline{\boldsymbol{w}}^* \oplus \overline{\boldsymbol{w}}_i$. If $OT(\overline{\boldsymbol{v}}_i, \overline{\boldsymbol{e}}^*) = 0$, then go to 4).
   b) Compute $\Lambda(\overline{\boldsymbol{w}}_i)$. If $\Lambda(\overline{\boldsymbol{w}}_i) < \underline{\Lambda}$, then $\underline{\Lambda} := \Lambda(\overline{\boldsymbol{w}}_i)$, $\underline{L} := L(\overline{\boldsymbol{c}}_0) + \underline{\Lambda}$, $\overline{\boldsymbol{w}}^* := \overline{\boldsymbol{w}}_i$ and $\overline{\boldsymbol{e}}^* := \overline{\boldsymbol{e}}_0 \oplus \overline{\boldsymbol{w}}^*$.
4) Set $i$++. If $i \leq 2^k$ and a certain terminating criterion does not hold, then go to 2), otherwise output $\overline{\boldsymbol{c}}_{ML} := \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}^*$ and stop. □
```

In the proposed decoding algorithm B, we construct $\overline{\boldsymbol{v}}_i$ each time $\overline{\boldsymbol{w}}_i$ is obtained at step 3)a). For $\overline{\boldsymbol{w}}^* \neq 0^n$, constructing $\overline{\boldsymbol{v}}_i$ costs just n binary operations which is smaller than the complexity of encoding each test error pattern (ordinarily that costs binary operations of $O(kn)$). We also update $\overline{\boldsymbol{e}}^*$ in order to implement $OT(\overline{\boldsymbol{v}}_i, \overline{\boldsymbol{e}}^*)$ each time a new best candidate codeword is obtained at step 3)b). Since $\overline{\boldsymbol{e}}^* = \overline{\boldsymbol{z}} \oplus \overline{\boldsymbol{c}}^*$ and $\overline{\boldsymbol{e}}_0 = \overline{\boldsymbol{z}} \oplus \overline{\boldsymbol{c}}_0$, we can obtain $\overline{\boldsymbol{e}}^*$ by

$$\overline{\boldsymbol{e}}^* = \overline{\boldsymbol{z}} \oplus \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}^* = \overline{\boldsymbol{e}}_0 \oplus \overline{\boldsymbol{w}}^*. \tag{22}$$

For $\overline{\boldsymbol{w}}^* \neq 0^n$, the computation of the right hand side (r.h.s.) of Eq. (22) requires just n binary operations. Furthermore, the space complexity for storing $\overline{\boldsymbol{v}}_i$ and $\overline{\boldsymbol{e}}^*$ is two binary arrays of size $O(n)$. We remark again that storing $\overline{G}$ requires $O(kn)$ and the increased space complexity is small.

We can also consider the following modification: Either Omitting Criterion A and B is selectively tested for $\overline{\boldsymbol{w}}_i$ in accordance with the order relation between $\boldsymbol{t}_i$ and $\boldsymbol{t}^*$ such that $\overline{\boldsymbol{w}}^* = \boldsymbol{t}^*\overline{G}$. Remark that, for $\overline{\boldsymbol{w}}_i$ and $\overline{\boldsymbol{w}}^*$, Omitting Criterion B holds only if $\boldsymbol{t}^* <_V \boldsymbol{t}_i$. Therefore, for $\overline{\boldsymbol{w}}_i$, we adopt Omitting Criterion A if $\boldsymbol{t}^* \not<_V \boldsymbol{t}_i$, and adopt Omitting Criterion B otherwise[†]. In this case, testing if $\boldsymbol{t}^* <_V \boldsymbol{t}_i$ can be carried out in a similar way of the test $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ for $\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}' \in \overline{\mathcal{V}}^n$. The time complexity for testing the order relation of test error patterns of length k is smaller than the test $OT(\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}')$ for $\overline{\boldsymbol{x}}, \overline{\boldsymbol{x}}'$ of length n.

3.3 Performance of the Proposed Decoding Algorithms

In this subsection, we state the performance of the proposed decoding algorithms.

Theorem 4: The both proposed decoding algorithms A and B achieve MLD.

Proof: Test error codewords constructed in the both proposed decoding algorithms A and B are the same as that constructed in the original MLD algorithm. The proposed decoding algorithms eliminate codewords which cannot be the ML codeword. For the ML codeword, its metrics is necessarily computed. □

We summarize the additional complexity of the proposed decoding algorithms A and B to the original reliability-based MLD algorithm. First, we state the additional complexity of the proposed decoding algorithm A.

(1) **Time complexity:** For testing $OT(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{e}}_0)$, at most $n-1$ increments of an integer and at most n shift operations are required.

(2) **Space complexity:** For storing $\overline{\boldsymbol{e}}_0$, we allocate a binary array of size n.

Next, we state the additional complexity of the proposed decoding algorithm B.

(1) **Time complexity:** For testing $OT(\overline{\boldsymbol{v}}_i, \overline{\boldsymbol{e}}^*)$, we construct $\overline{\boldsymbol{v}}_i$ each time $\overline{\boldsymbol{w}}_i$ is constructed. Furthermore, we construct $\overline{\boldsymbol{e}}^*$ each time $\overline{\boldsymbol{w}}^*$ is obtained. Constructing $\overline{\boldsymbol{v}}_i$ or $\overline{\boldsymbol{e}}^*$ costs n binary operations. For implementation of $OT(\overline{\boldsymbol{v}}_i, \overline{\boldsymbol{e}}^*)$, at most $n-1$ increments of an integer and at most n shift operations are required.

(2) **Space complexity:** For storing $\overline{\boldsymbol{v}}_i$ and $\overline{\boldsymbol{e}}_0$, we allocate two binary arrays of size n.

Theorem 5: The numbers of real number operations for both proposed decoding algorithms A and B are smaller than that for the original reliability-based MLD algorithm.

Proof: The total number of generating candidate codewords is the same for each decoding algorithm. The proposed decoding algorithm A (B) omits the computation of Eq. (17) if Omitting Criterion A (B) is satisfied for a test error codeword. Therefore, in proposed decoding algorithms A and B, the numbers of computation of Eq. (17) are reduced or at most the same as that in the original MLD algorithm. □

4. Simulation Results

4.1 Conditions for Simulations

In this section, we present simulation results for the binary (63,30,13) BCH code and the binary (127,64,21) BCH code in order to evaluate effectiveness of the proposed conditions. We assume each codeword is transmitted over the AWGN channel with the SNR E_b/N_0 [dB]. Although the proposed conditions can be applicable to any reliability-based decoding algorithms, we adopt the Gazelle and Snyders (GS) decoding algorithm [6] which is well-known for its efficiency with small space complexity.

The GS decoding algorithm [6] employs a simple evaluation function of $\boldsymbol{t}_i$ defined as

$$\Delta(\boldsymbol{t}_i) = \sum_{j=1}^{k} t_{i,j}|\tilde{\theta}_j|, \tag{23}$$

where $\tilde{\boldsymbol{\theta}} = (\tilde{\theta}_1, \tilde{\theta}_2, \ldots, \tilde{\theta}_n)$ is permuted sequence of $\overline{\boldsymbol{\theta}}$ such that the leftmost k positions are the k MRI positions in the non-increasing order of reliability measure. The function

[†] For $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{x}}'$, the order relation $\overline{\boldsymbol{x}} \not<_V \overline{\boldsymbol{x}}'$ means that the order relation $\overline{\boldsymbol{x}}' <_V \overline{\boldsymbol{x}}$ never holds.

$\Delta(\boldsymbol{t}_i)$ satisfies Eq. (6) (i.e., $\Delta(\boldsymbol{t}_i) \leq L(\overline{\boldsymbol{c}}_i)$) since

$$L(\overline{\boldsymbol{c}}_i) = \Delta(\boldsymbol{t}_i) + \sum_{j\in[1,n]\setminus\mathcal{M}} (\overline{z}_j \oplus \overline{c}_{i,j})|\overline{\theta}_j|. \tag{24}$$

Another evaluation function of $\boldsymbol{t}_i$ is used by MLD algorithms in [2], [6], [7]. The evaluation function $f(\boldsymbol{t}_i)$ gives a tighter lower bound of $L(\overline{\boldsymbol{c}}_i)$ than the function $\Delta(\boldsymbol{t}_i)$. The function $f(\boldsymbol{t}_i)$ uses the fact that the Hamming distance between some codeword $\overline{\boldsymbol{c}}_{\text{seed}}$ and any codeword $\overline{\boldsymbol{c}}_i \neq \overline{\boldsymbol{c}}_{\text{seed}}$ is no less than d, which is the minimum distance of the code C. Here, as in [2], [6], we consider the case $\overline{\boldsymbol{c}}_{\text{seed}} = \overline{\boldsymbol{c}}_0$[†]. We define $\mathcal{B}(\overline{\boldsymbol{c}}_0)$ as the set of positions where element of $\overline{\boldsymbol{e}}_0(= \overline{\boldsymbol{z}} \oplus \overline{\boldsymbol{c}}_0)$ is 0. Furthermore, for $\boldsymbol{t}_i$, let $\mathcal{A}(\overline{\boldsymbol{c}}_0, \boldsymbol{t}_i)$ be the set of $d - w_H(\overline{\boldsymbol{e}}_0) - w_H(\boldsymbol{t}_i)$ least reliable positions in $\mathcal{B}(\overline{\boldsymbol{c}}_0)$[††]. Then, the function $f(\boldsymbol{t}_i)$ is defined as

$$f(\boldsymbol{t}_i) = \Delta(\boldsymbol{t}_i) + \sum_{j\in\mathcal{A}(\overline{\boldsymbol{c}}_0,\boldsymbol{t}_i)} |\overline{\theta}_j|, \tag{25}$$

where $\Delta(\boldsymbol{t}_i)$ is given by Eq. (23). The second term of r.h.s. of Eq. (25) is non-negative, so the function $f(\boldsymbol{t}_i)$ is a tighter lower bound of $L(\overline{\boldsymbol{c}}_i)$ than $\Delta(\boldsymbol{t}_i)$. The function $f(\boldsymbol{t}_i)$ is the same as the *heuristic function* of the A* decoding algorithm [7], if we set $\overline{\boldsymbol{c}}_{\text{seed}} = \overline{\boldsymbol{c}}_0$.

The main difference between $\Delta(\cdot)$ and $f(\cdot)$ is the second term of r.h.s. of Eq. (25). The second term of Eq. (25) depends only on the Hamming weight of a test error pattern so it can be computed beforehand for each Hamming weight $1, 2, \ldots, d - w_H(\overline{\boldsymbol{e}}_0) - 1$ and be stored in memory. Furthermore, the second term of Eq. (25) for the larger Hamming weight than one can be computed during the computation of the second term for the Hamming weight one.

We consider the two GS decoding algorithms using (i) the evaluation function $\Delta(\cdot)$ (denoted as the algorithm GS(Δ)) and (ii) the evaluation function $f(\cdot)$ (denoted as the algorithm GS(f)). For each original GS decoding algorithm, we consider the following two modifications:

(1) [The algorithm A(Δ) and A(f)]: In algorithms GS(Δ) and GS(f), respectively, each time a test error codeword $\overline{\boldsymbol{w}}_i$ is constructed, Omitting Criterion A is tested if $\overline{\boldsymbol{c}}_i(= \overline{\boldsymbol{c}}_0 \oplus \overline{\boldsymbol{w}}_i)$ cannot be better than $\overline{\boldsymbol{c}}_0$.

(2) [The algorithm B(Δ) and B(f)]: In algorithms GS(Δ) and GS(f), respectively, each time the best candidate codeword $\overline{\boldsymbol{c}}^*$ is obtained, the codeword referenced by Omitting Criterion B is updated. After each test error codeword $\overline{\boldsymbol{w}}_i$ is constructed, Omitting Criterion B is tested if $\overline{\boldsymbol{c}}_i$ cannot be better than $\overline{\boldsymbol{c}}^*$.

Note that the numbers of real number operations for modified algorithms A(Δ) and B(Δ) are no more than that for the algorithm GS(Δ) by Theorem 5. As for algorithms using $f(\cdot)$, the same relation holds.

The results are obtained by decoding 10000 codewords for each SNR and the average values are shown in tables. In tables, we show the results of the following simulations.

(1) In order to evaluate the effectiveness of modified algorithms employing Omitting Criterion A or B, we compare the average number of real number operations for each decoding algorithm[†††]. For computation of $L(\overline{\boldsymbol{c}}_0)$, we count $w_H(\overline{\boldsymbol{e}}_0) - 1$ real number operations. Similarly, for each computation of $\Lambda(\overline{\boldsymbol{w}}_i)$, we count $w_H(\overline{\boldsymbol{w}}_i) - 1$ real number operations. The results are shown in Tables 1 and 2.

(2) In order to evaluate the effectiveness of Omitting Criterion A and B, we compare the average number of computations of Eq. (17) in six decoding algorithms. The results are shown in Tables 3 and 4.

4.2 Results about the Number of Real Number Operations

First we describe results at low to medium SNRs. By Tables 1 and 2, we can see that the numbers of real number operations for algorithms GS(Δ) and GS(f) are almost the same. These results imply that there is almost no difference between the effects of two evaluation functions. The number of real number operations for the algorithm GS(Δ) is the largest and that for the algorithm GS(f) is the second largest. The number of real number operations for the algorithm A(Δ) is the third largest (and the largest among modified algorithms A(Δ), B(Δ), A(f) and B(f)). Even the algorithm A(Δ) reduces the number of real number operations less than 1/3 that for the (63,30,13) and the (127,64,21) codes, compared with the algorithm GS(Δ). The algorithm B(Δ) requires less number of real number operations than that of the algorithm A(Δ) for the (63,30,13) and the (127,64,21) codes. The similar results are obtained for algorithms with the function $f(\cdot)$ for both codes.

Next we describe results at high SNRs. Contrary to the case at low to medium SNRs, the algorithm A(f) (or B(f)) requires more number of real number operations than that of the algorithm A(Δ) (or B(Δ)). The reason is that the number of candidate codewords are relatively small and computations of the second term of Eq. (25) dominate for the whole decoding complexity of A(f) and B(f). Note that the complexity for computing the second term of Eq. (25) is independent of the number of candidate codewords. The numbers of real number operations for the algorithm B(Δ) were the least among six algorithms and the values for B(Δ) were less than 1/3 that for GS(f), which required less real number operations between two conventional algorithms. These results indicate that we should select the evaluation function depending on SNRs if we adopt proposed conditions.

4.3 Results about the Number of Metrics Computations

First we describe results at low SNRs. By Tables 3 and 4, the numbers of metrics computations for algorithms GS(Δ) and

[†] Note that before generating any $\boldsymbol{t}_i$, the initial codeword $\overline{\boldsymbol{c}}_0$ is already obtained.

[††] We define $\mathcal{A}(\overline{\boldsymbol{c}}_0, \boldsymbol{t}_i) = \emptyset$ if $d - w_H(\overline{\boldsymbol{e}}_0) - w_H(\boldsymbol{t}_i) \leq 0$.

[†††] We do not include the number of real number operations for permuting from $\boldsymbol{\theta}$ to $\overline{\boldsymbol{\theta}}$ because it depends on sorting method. For the (63,30,13) and the (127,64,21) codes at each SNR, on average 251 and 582 real number operations are required, respectively, by the quick sort technique.

Table 1 The number of real number operations for the $(63, 30, 13)$ BCH code with the function $\Delta(\cdot)$ and $f(\cdot)$.

E_b/N_0	original	proposed		original	proposed	
[dB]	GS(Δ)	A(Δ)	B(Δ)	GS(f)	A(f)	B(f)
1.00	$6.94 \cdot 10^4$	$2.44 \cdot 10^4$	$2.17 \cdot 10^4$	$6.89 \cdot 10^4$	$2.43 \cdot 10^4$	$2.16 \cdot 10^4$
1.50	$3.97 \cdot 10^4$	$1.39 \cdot 10^4$	$1.21 \cdot 10^4$	$3.91 \cdot 10^4$	$1.38 \cdot 10^4$	$1.20 \cdot 10^4$
2.00	$2.04 \cdot 10^4$	$6.92 \cdot 10^3$	$6.03 \cdot 10^3$	$1.98 \cdot 10^4$	$6.86 \cdot 10^3$	$5.98 \cdot 10^3$
2.50	$9.36 \cdot 10^3$	$3.08 \cdot 10^3$	$2.62 \cdot 10^3$	$8.82 \cdot 10^3$	$3.03 \cdot 10^3$	$2.57 \cdot 10^3$
3.00	$3.88 \cdot 10^3$	$1.21 \cdot 10^3$	$8.93 \cdot 10^2$	$3.44 \cdot 10^3$	$1.17 \cdot 10^3$	$8.61 \cdot 10^2$
3.50	$1.43 \cdot 10^3$	$4.26 \cdot 10^2$	$3.00 \cdot 10^2$	$1.12 \cdot 10^3$	$4.07 \cdot 10^2$	$2.81 \cdot 10^2$
4.00	$5.49 \cdot 10^2$	$1.42 \cdot 10^2$	$9.21 \cdot 10^1$	$3.50 \cdot 10^2$	$1.34 \cdot 10^2$	$8.40 \cdot 10^1$
4.50	$2.26 \cdot 10^2$	$5.71 \cdot 10^1$	$4.02 \cdot 10^1$	$1.21 \cdot 10^2$	$5.73 \cdot 10^1$	$4.04 \cdot 10^1$
5.00	$8.39 \cdot 10^1$	$2.03 \cdot 10^1$	$1.27 \cdot 10^1$	$3.81 \cdot 10^1$	$2.52 \cdot 10^1$	$1.76 \cdot 10^1$
5.50	$3.31 \cdot 10^1$	7.63	5.79	$1.70 \cdot 10^1$	$1.45 \cdot 10^1$	$1.27 \cdot 10^1$

Table 2 The number of real number operations for the $(127, 64, 21)$ BCH code with the function $\Delta(\cdot)$ and $f(\cdot)$.

E_b/N_0	original	proposed		original	proposed	
[dB]	GS(Δ)	A(Δ)	B(Δ)	GS(f)	A(f)	B(f)
2.50	$3.16 \cdot 10^7$	$1.04 \cdot 10^7$	$9.23 \cdot 10^6$	$3.15 \cdot 10^7$	$1.04 \cdot 10^7$	$9.21 \cdot 10^6$
3.00	$8.04 \cdot 10^6$	$2.38 \cdot 10^6$	$1.64 \cdot 10^6$	$7.93 \cdot 10^6$	$2.37 \cdot 10^6$	$1.63 \cdot 10^6$
3.50	$8.80 \cdot 10^5$	$2.69 \cdot 10^5$	$1.55 \cdot 10^5$	$8.13 \cdot 10^5$	$2.63 \cdot 10^5$	$1.49 \cdot 10^5$
4.00	$1.10 \cdot 10^5$	$3.38 \cdot 10^4$	$1.59 \cdot 10^4$	$8.40 \cdot 10^4$	$3.18 \cdot 10^4$	$1.40 \cdot 10^4$
4.50	$1.62 \cdot 10^4$	$4.03 \cdot 10^3$	$1.85 \cdot 10^3$	$8.34 \cdot 10^3$	$3.52 \cdot 10^3$	$1.34 \cdot 10^3$
5.00	$2.99 \cdot 10^3$	$5.83 \cdot 10^2$	$2.89 \cdot 10^2$	$9.58 \cdot 10^2$	$4.89 \cdot 10^2$	$1.95 \cdot 10^2$
5.50	$6.86 \cdot 10^2$	$8.25 \cdot 10^1$	$4.73 \cdot 10^1$	$1.23 \cdot 10^2$	$7.31 \cdot 10^1$	$3.79 \cdot 10^1$
6.00	$1.82 \cdot 10^2$	$1.93 \cdot 10^1$	$1.33 \cdot 10^1$	$3.51 \cdot 10^1$	$2.93 \cdot 10^1$	$2.33 \cdot 10^1$
6.50	$4.94 \cdot 10^1$	5.59	4.80	$2.02 \cdot 10^1$	$1.97 \cdot 10^1$	$1.89 \cdot 10^1$

Table 3 The number of computations of Eq. (17) for the (63,30,13) BCH code with the function $\Delta(\cdot)$ and $f(\cdot)$.

E_b/N_0	original	proposed		original	proposed	
[dB]	GS(Δ)	A(Δ)	B(Δ)	GS(f)	A(f)	B(f)
1.00	$2.68 \cdot 10^3$	$6.66 \cdot 10^2$	$5.39 \cdot 10^2$	$2.66 \cdot 10^3$	$6.65 \cdot 10^2$	$5.38 \cdot 10^2$
1.50	$1.55 \cdot 10^3$	$3.87 \cdot 10^2$	$3.03 \cdot 10^2$	$1.52 \cdot 10^3$	$3.86 \cdot 10^2$	$3.02 \cdot 10^2$
2.00	$8.08 \cdot 10^2$	$1.95 \cdot 10^2$	$1.52 \cdot 10^2$	$7.81 \cdot 10^2$	$1.95 \cdot 10^2$	$1.52 \cdot 10^2$
2.50	$3.77 \cdot 10^2$	$8.89 \cdot 10^1$	$6.60 \cdot 10^1$	$3.52 \cdot 10^2$	$8.89 \cdot 10^1$	$6.60 \cdot 10^1$
3.00	$1.60 \cdot 10^2$	$3.56 \cdot 10^1$	$2.03 \cdot 10^1$	$1.40 \cdot 10^2$	$3.56 \cdot 10^1$	$2.03 \cdot 10^1$
3.50	$6.10 \cdot 10^1$	$1.32 \cdot 10^1$	6.93	$4.64 \cdot 10^1$	$1.32 \cdot 10^1$	6.92
4.00	$2.39 \cdot 10^1$	4.24	1.76	$1.44 \cdot 10^1$	4.24	1.76
4.50	9.91	1.65	$8.00 \cdot 10^{-1}$	4.64	1.65	$7.98 \cdot 10^{-1}$
5.00	3.71	$5.42 \cdot 10^{-1}$	$1.57 \cdot 10^{-1}$	1.15	$5.41 \cdot 10^{-1}$	$1.57 \cdot 10^{-1}$
5.50	1.42	$1.36 \cdot 10^{-1}$	$4.16 \cdot 10^{-2}$	$2.54 \cdot 10^{-1}$	$1.36 \cdot 10^{-1}$	$4.14 \cdot 10^{-2}$

Table 4 The number of computations of Eq. (17) for the (127,64,21) BCH code with the function $\Delta(\cdot)$ and $f(\cdot)$.

E_b/N_0	original	proposed		original	proposed	
[dB]	GS(Δ)	A(Δ)	B(Δ)	GS(f)	A(f)	B(f)
2.50	$6.98 \cdot 10^5$	$1.61 \cdot 10^5$	$1.30 \cdot 10^5$	$6.94 \cdot 10^5$	$1.61 \cdot 10^5$	$1.30 \cdot 10^5$
3.00	$1.78 \cdot 10^5$	$3.47 \cdot 10^4$	$1.55 \cdot 10^4$	$1.76 \cdot 10^5$	$3.47 \cdot 10^4$	$1.55 \cdot 10^4$
3.50	$2.05 \cdot 10^4$	$4.61 \cdot 10^3$	$1.55 \cdot 10^3$	$1.89 \cdot 10^4$	$4.61 \cdot 10^3$	$1.55 \cdot 10^3$
4.00	$2.69 \cdot 10^3$	$6.52 \cdot 10^2$	$1.63 \cdot 10^2$	$2.04 \cdot 10^3$	$6.52 \cdot 10^2$	$1.63 \cdot 10^2$
4.50	$4.15 \cdot 10^2$	$7.97 \cdot 10^1$	$1.87 \cdot 10^1$	$2.10 \cdot 10^2$	$7.97 \cdot 10^1$	$1.87 \cdot 10^1$
5.00	$7.92 \cdot 10^1$	$1.16 \cdot 10^1$	3.21	$2.44 \cdot 10^1$	$1.16 \cdot 10^1$	3.21
5.50	$1.86 \cdot 10^1$	1.37	$3.51 \cdot 10^{-1}$	2.73	1.37	$3.52 \cdot 10^{-1}$
6.00	4.95	$2.58 \cdot 10^{-1}$	$8.10 \cdot 10^{-2}$	$4.14 \cdot 10^{-1}$	$2.58 \cdot 10^{-1}$	$8.10 \cdot 10^{-2}$
6.50	1.30	$3.16 \cdot 10^{-2}$	$8.40 \cdot 10^{-3}$	$4.56 \cdot 10^{-2}$	$3.16 \cdot 10^{-2}$	$8.40 \cdot 10^{-3}$

GS(f) are almost the same. This relationship holds between A(Δ) and A(f) and between B(Δ) and B(f). By Table 3, the algorithm A(Δ) (or A(f)) computes metrics of less than 1/4 test error codewords for the (63,30,13) code compared with the algorithm GS(Δ) (or GS(f)). i.e., Omitting Criterion A holds for more than 3/4 test error codewords. By Table 4, the value which Omitting Criterion A holds is also more than 3/4 for the (127,64,21) code. These results indicate that Omitting Criterion A works well and $\overline{\boldsymbol{c}}_0$ is a good candidate as the initial codeword in the reliability-based decoding algorithms. The values for the algorithm B(Δ) (or B(f)) is less than that for the algorithm A(Δ) (or A(f)) at each SNR for both codes. This implies that Omitting Criterion B is, in general, more effective than Omitting Criterion A. If we use Omitting Criterion B, the number of computing Eq. (17) is less than 1/5 test error codewords for the (63,30,13) and the (127,64,21) codes.

Next we describe results at medium to high SNRs. The numbers of computing Eq. (17) for algorithms A(Δ), A(f), B(Δ) and B(f) decrease as the SNR increases for both codes. The difference between GS(Δ) and GS(f) becomes large at high SNRs. Nevertheless, the numbers of computing Eq. (17) in the algorithm A(Δ) and A(f) are almost the same and similar results hold for algorithm B(Δ) and B(f). These results indicate the effects of proposed conditions are independent of evaluation functions. It is noteworthy that, at high SNR, the numbers of computing Eq. (17) in the algorithm B(Δ) and B(f) are almost negligible for both (63,30,13) and (127,64,21) codes.

5. Concluding Remarks

In this paper, we have derived two sufficient conditions for omitting unnecessary metrics computations of candidate codewords in the reliability-based MLD algorithms. A simple method for testing the proposed conditions is presented. For implementation of this method, we need no real number operations. The results of computer simulations show the effectiveness of the proposed conditions for the (63,30,13) and the (127,64,21) BCH codes. As a result, we can reduce the number of real number operations which is one of the typical measures for evaluating the efficiency of MLD algorithms. The proposed conditions are applicable to any reliability-based MLD algorithms such as one in [6], [7], [12], [14], [17].

As future improvements, a rest of decoding complexity such as for encoding test error patterns should be reduced. A method that quantitatively reduces the number of metrics computations of candidate codewords is also to be developed.

Acknowledgement

The authors wish to thank the anonymous reviewers for their valuable comments. One of the authors, H. Yagi would like to thank Mr. T. Ishida and Mr. G. Hosoya at Waseda University for their supports.

This work is supported by Japan Society for the Promotion of Science under Grants-in-Aid for Scientific Research No. 1556-0338 and No. 1576-0281 and Waseda University Grant for Special Research Project No. 2001A-566.

References

[1] D. Chase, “A new class for decoding block codes with channel measurement information,” IEEE Trans. Inf. Theory, vol.IT-18, no.1, pp.170–182, Jan. 1972.

[2] M.P.C. Fossorier and S. Lin, “Soft-decision decoding of linear block codes based on ordered statistics,” IEEE Trans. Inf. Theory, vol.41, no.5, pp.1379–1396, Sept. 1995.

[3] M.P.C. Fossorier, S. Lin, and J. Snyders, “Reliability-based syndrome decoding of linear block codes,” IEEE Trans. Inf. Theory, vol.44, no.1, pp.388–398, Jan. 1998.

[4] M.P.C. Fossorier and S. Lin, “Reliability-based information set decoding of binary linear block codes,” IEICE Trans. Fundamentals, vol.E82-A, no.10, pp.2034–2042, Oct. 1999.

[5] T. Fujiwara, H. Yamamoto, T. Kasami, and S. Lin, “A trellis-based recursive maximum likelihood decoding algorithm for binary linear block codes,” IEEE Trans. Inf. Theory, vol.44, no.2, pp.714–729, March 1998.

[6] D. Gazelle and J. Snyders, “Reliability-based code-search algorithm for maximum likelihood decoding of block codes,” IEEE Trans. Inf. Theory, vol.43, no.1, pp.239–249, Jan. 1997.

[7] Y.S. Han, C.P.R. Hartman, and C.C. Chen, “Efficient priority-first search maximum-likelihood soft decision decoding of linear block codes,” IEEE Trans. Inf. Theory, vol.39, no.5, pp.1514–1523, Sept. 1993.

[8] T. Kaneko, T. Nishijima, H. Inazumi, and S. Hirasawa, “An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder,” IEEE Trans. Inf. Theory, vol.40, no.2, pp.320–327, March 1994.

[9] T. Kasami, Y. Tang, T. Koumoto, and T. Fujiwara, “Sufficient conditions for ruling-out useless iteration steps in a class of iterative decoding algorithms,” IEICE Trans. Fundamentals, vol.E82-A, no.10, pp.2061–2073, Oct. 1999.

[10] T. Koumoto, T. Kasami, and S. Lin, “A sufficient condition for ruling out some useless test error patterns in iterative decoding algorithms,” IEICE Trans. Fundamentals, vol.E81-A, no.2, pp.321–326, Feb. 1998.

[11] A. Lafourcade and A. Vardy, “Optimal sectionalization of a trellis,” IEEE Trans. Inf. Theory, vol.42, no.3, pp.689–703, May 1996.

[12] T. Okada, M. Kobayashi, and S. Hirasawa, “An efficient heuristic search method for maximum likelihood decoding of linear block codes using dual codes,” IEICE Trans. Fundamentals, vol.E85-A, no.2, pp.485–489, Feb. 2002.

[13] C.C. Shih, C.R. Wulff, C.R.P. Hartmann, and C.K. Mohan, “Efficient heuristic search algorithms for soft-decision decoding of linear block codes,” IEEE Trans. Inf. Theory, vol.44, no.6, pp.3023–3038, Nov. 1998.

[14] A. Valembois and M.P.C. Fossorier, “An improved method to compute lists of binary vectors that optimize a given weight function with application of soft-decision decoding,” IEEE Commun. Lett., vol.5, no.11, pp.456–458, Nov. 2001.

[15] A. Valembois and M.P.C. Fossorier, “A comparison between “most-reliable-basis reprocessing” strategies,” IEICE Trans. Fundamentals, vol.E85-A, no.7, pp.1727–1741, July 2002.

[16] Y. Wu and D.A. Pados, “An adaptive two-stage algorithm for ML and sub-ML decoding of binary linear block codes,” IEEE Trans. Inf. Theory, vol.49, no.1, pp.261–269, Jan. 2003.

[17] H. Yagi, M. Kobayashi, and S. Hirasawa, “Complexity reduction of the Gazelle and Snyders decoding algorithm for maximum likelihood decoding,” IEICE Trans. Fundamentals, vol.E86-A, no.10, pp.2461–2471, Oct. 2003.

[18] H. Yagi, M. Kobayashi, and S. Hirasawa, “An improved method of maximum likelihood decoding algorithms using the most reliable basis based on an order relation among binary vectors,” IEICE Technical Report, IT2003-6, May 2003.

Hideki Yagi was born in Yokohama, Japan, on Oct. 14, 1975. He received the B.E. degree and M.E. degree in Industrial and Management Systems Engineering from Waseda University, Tokyo, Japan, in 2001 and 2003, respectively. He is currently a doctorial student in Industrial and Management Systems Engineering at Graduate School of Waseda University. His research interests are coding and information theory.

Manabu Kobayashi was born in Yokohama, Japan, on Oct. 30, 1971. He received the B.E. degree, M.E. degree and Dr.E. degree in Industrial and Management Systems Engineering form Waseda University, Tokyo, Japan, in 1994, 1996 and 2000, respectively. From 1998 to 2001, he was a research associate in Industrial and Management Systems Engineering at Waseda University. He is currently a full-time lecturer of the Department of Information Science at Shonan Institute of Technology, Kanagawa, Japan. His research interests are coding and information theory and data mining. He is a member of the Society of Information Theory and Its Applications, Information Processing Society of Japan and IEEE.

Toshiyasu Matsushima was born in Tokyo, Japan, on Nov. 26, 1955. He received the B.E. degree, M.E. degree and Dr.E. degree in Industrial and Management Systems Engineering form Waseda University, Tokyo, Japan, in 1978, 1980 and 1991, respectively. From 1980 to 1986, he was with Nippon Electric Corporation, Kanagawa, Japan. From 1986 to 1992, he was a lecturer at Department of Management Information, Yokohama College of Commerce. From 1993, he was an associate professor and since 1996 has been a professor of School of Science and Engineering, Waseda University, Tokyo, Japan. His research interests are information theory and its application, statistics and artificial intelligence. He is a member of the Society of Information Theory and Its Applications, the Japan Society for Quality Control, the Japan Industrial Management Association, the Japan Society for Artificial Intelligence and IEEE.

Shigeichi Hirasawa was born in Kobe, Japan, on Oct. 2, 1938. He received the B.S. degree in mathematics and the B.E. degree in electrical communication engineering from Waseda University, Tokyo, Japan, in 1961 and 1963, respectively, and the Dr.E. degree in electrical communication engineering from Osaka University, Osaka, Japan, in 1975. From 1963 to 1981, he was with the Mitsubishi Electric Corporation, Hyogo, Japan. Since 1981, he has been a professor of School of Science and Engineering, Waseda University, Tokyo, Japan. In 1979, he was a Visiting Scholar in the Computer Science Department at the University of California, Los Angels (CSD, UCLA), CA. He was a Visiting Researcher at the Hungarian Academy of Science, Hungary, in 1985, and at the University of Trieste, Italy, in 1986. In 2002, he was also a Visiting Faculty at CSD, UCLA. From 1987 to 1989, he was the Chairman of Technical Group on Information Theory of IEICE. He received the 1993 Achievement Award, and the 1993 Kobayashi-Memorial Achievement Award from IEICE. In 1996, he was the President of the Society of Information Theory and Its Applications (Soc. of ITA). His research interests are information theory and its applications, and information processing systems. He is an IEEE Fellow, and a member of Soc. of ITA, the Operations Research Society of Japan, the Information Processing Society of Japan, the Japan Industrial Management Association, and Informs.