LETTER *Special Section on Information Theory and Its Applications*

Fast Algorithm for Generating Candidate Codewords in Reliability-Based Maximum Likelihood Decoding

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

SUMMARY We consider the reliability-based heuristic search methods for maximum likelihood decoding, which generate test error patterns (or, equivalently, candidate codewords) according to their heuristic values. Some studies have proposed methods for reducing the space complexity of these algorithms, which is crucially large for long block codes at medium to low signal to noise ratios of the channel. In this paper, we propose a new method for reducing the time complexity of generating candidate codewords by storing some already generated candidate codewords. Simulation results show that the increase of memory size is small.

key words: *maximum likelihood decoding, binary block codes, priority-first search, most reliable basis, reliability*

1. Introduction

One of the most efficient maximum likelihood decoding (MLD) algorithms for linear block codes is the reliability-based decoding algorithm that uses the column-permuted generator matrix in non-increasing order of reliability. The object of the reliability-based MLD algorithms are medium to low (but not extremely low) SNRs[*]. The reliability-based MLD algorithms are divided into two types due to the generation rule of candidate codewords. The former type of them generates candidate codewords according to a predetermined generation rule [3], [4], [8], [10]. The latter one is called **priority-first search-type MLD algorithms**, where candidate codewords are generated in increasing value of the heuristic function [1], [2], [6], [7], [9], [12].

In this paper, we consider the priority-first search-type MLD algorithms. The generalized Battail-Fang (GBF) decoding algorithm indicated by A. Valembois and M. Fossorier [6], [7] includes a wide variety of this type of algorithms such as the original Battail-Fang decoding algorithm [1] or the well-known A* decoding algorithm proposed by Y.S. Han et al. [2]. Subsequently, Valembois et al. [6] and the present authors [9], [12] have proposed methods for greatly reducing the space complexity of the GBF decoding algorithm employing some class of heuristic functions. Although the space complexity of priority-first search MLD algorithms has been reduced by these methods, their time complexity remains quite large at medium to low signal to noise ratios (SNRs) of the channel. One of the dominant complexity per one iteration of the decoding algorithm is $O(kn)$ binary operations for generating a candidate codeword where n and k represent the code length and the number of information symbols, respectively.

This paper focuses on issues of the time complexity of the priority-first search-type MLD algorithm. Based on the method in [12], we propose a new method for reducing the time complexity of generating candidate codewords by storing some already generated candidate codewords. The time complexity for it is reduced from $O(kn)$ binary operations to $O(n)$ ones[**]. Although the new method requires more space complexity than that for the method in [12], its space complexity is $\frac{n}{k}$ times that for the method in [12]. Since the method in [12] is a reduced-list version of the GBF decoding algorithm and the A* decoding algorithm, the proposed method can reduce the time complexity of all of these decoding algorithms. We show by computer simulations that the space complexity for the fast GBF decoding algorithm is still reduced compared with the GBF decoding algorithm.

This paper is organized as follows. In Sect. 2, we briefly review the reliability-based MLD algorithm. In Sect. 3, we describe the conventional GBF decoding algorithms. In Sect. 4, we propose a new method for reducing the time complexity of the GBF decoding algorithms. In Sect. 5, we show some simulation results and we state the conclusion in Sect. 6.

2. Reliability-Based MLD Algorithm

Let C be a binary linear (n, k, d) block code of the code length n, the number of information symbols k and the minimum distance d. We denote a generator matrix of C by G. We assume any codewords $\boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \{0, 1\}^n$ of C are transmitted over the additive white Gaussian noise (AWGN) channel. The receiver maps a received sequence $\boldsymbol{r} = (r_1, r_2, \ldots, r_n) \in \mathcal{R}^n$ into a reliability sequence $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$, $\theta_j = \ln \frac{P(r_j|c_j=0)}{P(r_j|c_j=1)}$, where $P(r_j|c_j)$ represents the likelihood of the symbol c_j. Furthermore, a hard-decision received sequence $\boldsymbol{z} = (z_1, z_2, \ldots, z_n) \in \{0, 1\}^n$ is obtained by setting

Manuscript received January 24, 2006.
Manuscript revised April 17, 2006.
Final manuscript received June 9, 2006.

[†]The author is with Media Network Center, Waseda University, Tokyo, 169-8005 Japan.
[††]The authors are with the Department of Industrial and Management Systems Engineering, School of Science and Engineering, Waseda University, Tokyo, 169-8555 Japan.
a) E-mail: yagi@hirasa.mgmt.waseda.ac.jp

DOI: 10.1093/ietfec/e89–a.10.2676

[*]At extremely low SNRs, other type of MLD algorithms such as trellis-based MLD algorithms may work well.

[**]For other type of the reliability-based MLD algorithm, the same complexity has been achieved [8], [10].

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

for $1 \le j \le n$. The soft-decision decoder estimates the transmitted codeword from $\boldsymbol{\theta}$ and $\boldsymbol{z}$.

In reliability-based decoding algorithms, we utilize the column-permuted systematic generator matrix $\tilde{G} = \left[I_k|\tilde{P}\right]$ where the leftmost k positions are the **most reliable and linearly independent** (MRI) [2], [5]–[7] in non-increasing value of reliability. Let $\tilde{\boldsymbol{\theta}} = (\tilde{\theta}_1, \tilde{\theta}_2, \ldots, \tilde{\theta}_n)$ and $\tilde{\boldsymbol{z}} = (\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_n)$ be permuted sequences of $\boldsymbol{\theta}$ and $\boldsymbol{z}$, respectively, in the same ordering of columns of $\tilde{G}$. This ordering satisfies $|\tilde{\theta}_{j_1}| \ge |\tilde{\theta}_{j_2}|$ for $1 \le j_1 < j_2 \le k$ and for $k+1 \le j_1 < j_2 \le n$. Let $\tilde{C}$ be the code whose codewords are generated by $\tilde{G}$. Define $\boldsymbol{u} = (u_1, u_2, \ldots, u_k) \in \{0,1\}^k$ as the leftmost k symbols of $\tilde{\boldsymbol{z}}$, i.e., $u_j = \tilde{z}_j, 1 \le \forall j \le k$. The decoder first encodes $\boldsymbol{u}$ by $\tilde{G}$ to obtain the initial codeword $\tilde{\boldsymbol{c}}_\emptyset (= \boldsymbol{u}\tilde{G})$. Afterwards, k-dimensional vectors, called **test error patterns** $\boldsymbol{t} \in \{0,1\}^k$, are iteratively generated and encoded by $\tilde{G}$. Then, $\tilde{\boldsymbol{c}} = \tilde{\boldsymbol{c}}_\emptyset \oplus \boldsymbol{t}\tilde{G}$ is a candidate codeword[†]. This procedure is repeated until a sufficient condition for the most likely (ML) codeword is satisfied.

Definition 1: For a location set $J \subseteq [1,k]$, the test error pattern (TEP) $\boldsymbol{t}(J) = \left(t_1(J), t_2(J), \ldots, t_k(J)\right)$ has element one in J. Define that $\mu(J)$ be the rightmost position in J, i.e., $\mu(J) = \max J$. For $j > \mu(J)$, the TEP $\boldsymbol{t}(J \cup \{j\})$ (or simply $\boldsymbol{t}(J \cup j)$) is called an **extended pattern** of $\boldsymbol{t}(J)$. Define $J^{\mathrm{a}} = J \setminus \mu(J)$. For $j > \mu(J)$, the TEP $\boldsymbol{t}(J^{\mathrm{a}} \cup j)$ is called an **adjacent pattern** of $\boldsymbol{t}(J)$ in j.

□

For a binary vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n) \in \{0,1\}^n$, we define the **correlation discrepancy** [6], [7] of $\boldsymbol{v}$ as

$$L(\boldsymbol{v}) = \sum_{j | v_j \ne \tilde{z}_j} |\tilde{\theta}_j|. \tag{2}$$

It is well-known that $\tilde{\boldsymbol{c}}_{\text{best}}$ is the ML codeword if and only if $L(\tilde{\boldsymbol{c}}_{\text{best}}) = \min_{\tilde{\boldsymbol{c}} \in \tilde{C}} \left\{L(\tilde{\boldsymbol{c}})\right\}$.

3. Priority-First Search Method of Codewords

3.1 The GBF Decoding Algorithm

The priority-first search-type MLD algorithm searches the ML codeword where TEPs are generated in increasing order of heuristic values. Valembois and Fossorier have indicated that the GBF decoding algorithm [6], [7] and the well-known A* decoding algorithm [2] are equivalent when both algorithms adopt the same heuristic function. In this section, we state the GBF decoding algorithm.

We first mention heuristic functions of TEPs. The GBF decoding algorithm performs the priority-first search with any heuristic functions $F(\cdot)$ satisfying the following condition:

$$\textbf{(C1)} \qquad F\left(\boldsymbol{t}(J)\right) \le F\left(\boldsymbol{t}(J \cup j)\right) \qquad \text{for } j \notin J.$$

It is guaranteed that the GBF decoding algorithm always finds the ML codeword if

$$F\left(\boldsymbol{t}(J)\right) \le L\left(\tilde{\boldsymbol{c}}(J)\right), \tag{3}$$

where $\tilde{\boldsymbol{c}}(J) = \tilde{\boldsymbol{c}}_\emptyset \oplus \boldsymbol{t}(J)\tilde{G}$.

Hereafter, we describe how to perform the priority-first search of TEPs using a heuristic function satisfying the condition (C1). Let $M^{(1)}, M^{(2)}, \ldots, M^{(k)}$ be k lists of TEPs. A TEP $\boldsymbol{t}(J)$ is supposed to be in $M^{(\mu(J))}$ where $\mu(J) = \max J$. In a list $M^{(j)}, \forall j \in [1,k]$, TEPs are ordered in increasing order of heuristic values.

By the condition (C1), the TEP with the minimum heuristic value in $M^{(j)}, j \in [1,k]$, is $\boldsymbol{t}(j)$ whose Hamming weight is one. Therefore, we just need to set the initial lists as $M^{(j)} = \left\{\boldsymbol{t}(j)\right\}$ for $j \in [1,k]$. Then, the algorithm searches the TEP with the minimum heuristic value (we will call this pattern the **best pattern**) among all TEPs not being encoded.

We here describe the GBF decoding algorithm which is equivalent to the A* decoding algorithm [2].

[The GBF decoding algorithm]

S1) Set $\tilde{\boldsymbol{c}}_\emptyset := \boldsymbol{u}\tilde{G}$, $\tilde{\boldsymbol{c}}_{\text{best}} := \tilde{\boldsymbol{c}}_\emptyset$ and $\underline{L} := L(\tilde{\boldsymbol{c}}_\emptyset)$. Construct the initial lists of TEPs.

S2) Choose the best pattern $\boldsymbol{t}(J) \in M^{(\mu(J))}$ among the top-most TEPs in non-empty lists $M^{(j)}$. If $F(\boldsymbol{t}(J)) \ge \underline{L}$, then output $\tilde{\boldsymbol{c}}_{\text{best}}$ and halt the algorithm.

S3) Generate the next candidate codeword by $\tilde{\boldsymbol{c}}(J) := \tilde{\boldsymbol{c}}_\emptyset \oplus \boldsymbol{t}(J)\tilde{G}$. If $L(\tilde{\boldsymbol{c}}(J)) < \underline{L}$, then set $\underline{L} := L(\tilde{\boldsymbol{c}}(J))$ and $\tilde{\boldsymbol{c}}_{\text{best}} := \tilde{\boldsymbol{c}}(J)$.

S4) For all lists $M^{(j)}$ such that $j > \mu(J)$, insert extended patterns $\boldsymbol{t}(J \cup j)$ at the position such that the list remains increasing order of heuristic values. Delete $\boldsymbol{t}(J)$ from $M^{(\mu(J))}$.

S5) If $M^{(j)} = \emptyset$ for all $j \in [1,k]$, then output $\tilde{\boldsymbol{c}}_{\text{best}}$ and halt the algorithm. Otherwise, go to S2). □

In S4), we need to sort extended patterns so that the list $M^{(j)}$ remains increasing order of heuristic values. By sorting, the priority-first search of the GBF decoding algorithm is maintained [6].

3.2 A Reduced-List Method of the GBF Decoding Algorithm

In [12], the present authors have proposed a technique for reducing the list size of TEPs in the GBF decoding algorithm. We call this method the **reduced-list GBF decoding algorithm**. Hereafter, we briefly review it.

We define the following condition (C2) for a heuristic function $F(\cdot)$.

Definition 2: Let $S^{(0)}$ be a certain subset of $[1,k]$ and $S^{(1)}$ be the complement of $S^{(0)}$. For $J \subseteq [1,k]$, assume $j_1, j_2 \notin J$

[†]The symbol $\oplus$ represents Exclusive OR operation.

and $j_1 < j_2$. If $j_1, j_2 \in S^{(\alpha)}$ with $\alpha \in \{0, 1\}$, then a function $F(\cdot)$ satisfies

$$\textbf{(C2)} \qquad F\Big(\boldsymbol{t}(J \cup j_1)\Big) \geq F\Big(\boldsymbol{t}(J \cup j_2)\Big).$$

We will call this condition the **condition (C2)**[†]. □

Hereafter, we consider heuristic functions satisfying both (C1) and (C2). It has been shown in [12] that most of heuristic functions such as in [1]–[4], [6] satisfy the conditions (C1) and (C2).

The strategy of the reduced-list GBF decoding algorithm is like *lazy evaluation* where any TEPs are not generated as long as possible. Hereafter, for convenience, we denote $S^{(0)} = \{i_1, i_2, \ldots, i_s\}$ and $S^{(1)} = \{i'_1, i'_2, \ldots, i'_p\}$.

By the condition (C1), the best pattern in a list $M^{(j)}$, $j \in [1, k]$, is $\boldsymbol{t}(j)$ whose Hamming weight is one. Therefore, it is enough to construct the initial lists as

$$M^{(j)} = \begin{cases} \big\{\boldsymbol{t}(j)\big\}, & \text{if } j \in \{i_s, i'_p\}; \\ \emptyset, & \text{otherwise,} \end{cases} \tag{4}$$

by the condition (C2).

At S2) of the GBF decoding algorithm, if $\boldsymbol{t}(J)$ is chosen as the best pattern, $k - \mu(J)$ extended patterns of $\boldsymbol{t}(J)$ will be stored at S4). However, it is enough to store only its extended patterns $\boldsymbol{t}(J \cup i_s)$ and $\boldsymbol{t}(J \cup i'_p)$ in the list $M^{(i_s)}$ and $M^{(i'_p)}$, respectively. Following this modification, after $\boldsymbol{t}(J \cup i_q), i_q \in S^{(0)}$, is chosen as the best pattern at S2), $\boldsymbol{t}(J \cup i_{q-1}), i_{q-1} \in S^{(0)}$, is inserted into the list $M^{(i_{q-1})}$ if it exists. A similar procedure is carried out if the best pattern at S2) is $\boldsymbol{t}(J \cup i'_q), i'_q \in S^{(1)}$.

We describe a decoding algorithm employing the above method where the steps P1) and P4) correspond to the modifications.

[The reduced-list GBF decoding algorithm]

P1) Set $\tilde{\boldsymbol{c}}_\emptyset := \boldsymbol{u}\tilde{G}$, $\tilde{\boldsymbol{c}}_{\text{best}} := \tilde{\boldsymbol{c}}_\emptyset$ and $\underline{L} := L(\tilde{\boldsymbol{c}}_\emptyset)$. Construct the initial lists of TEPs by Eq. (4).

P2) This step is the same as S2).

P3) This step is the same as S3).

P4) a) If $\mu(J) = i_q$ (i.e., $\mu(J) \in S^{(0)}$) and the adjacent pattern $\boldsymbol{t}(J^{\text{a}} \cup i_{q-1})$ exists where $J^{\text{a}} = J \backslash \mu(J)$, then insert it into the list $M^{(i_{q-1})}$.

b) If $\mu(J) = i'_q$ (i.e., $\mu(J) \in S^{(1)}$) and $\boldsymbol{t}(J^{\text{a}} \cup i'_{q-1})$ exists, then insert it into $M^{(i'_{q-1})}$.

c) If $\mu(J) < i_s$, then insert $\boldsymbol{t}(J \cup i_s)$ into $M^{(i_s)}$. If $\mu(J) < i'_p$, then insert $\boldsymbol{t}(J \cup i'_p)$ into $M^{(i'_p)}$. Delete $\boldsymbol{t}(J)$ from $M^{(\mu(J))}$.

P5) This step is the same as S5). □

Note that we need to store at most three TEPs at P4) and this leads to a significant reduction of the list size of TEPs.

3.3 Decoding Complexity

We here mention the time complexity of the original and the reduced-list GBF decoding algorithms. Permuting $\boldsymbol{\theta}$ in the non-increasing order of reliability costs $O(n \log n)$ comparisons and constructing $\tilde{G}$ costs $O(n \times \kappa^2)$ binary operations where $\kappa = \min\{k, n-k\}$ [2]–[4], [10]. These steps are carried out only once in a decoding procedure. Contrary to the above steps, encoding $\boldsymbol{t}(J)$ by $\tilde{G}$ and computing its discrepancy are carried out iteratively, where each encoding requires $O(kn)$ binary operations by conventional encoding method [4], [5], [10]. Therefore, both generating candidate codewords and their discrepancy computations dominate the whole decoding complexity [2], [5], [6]. The time complexity per one iteration is $O(kn)$.

We state encoding methods in detail. We can consider the following two strategies.

(i) For a TEP $\boldsymbol{t}(J)$, let $l = |J|$. The number of binary operations for generating the candidate codeword $\tilde{\boldsymbol{c}}(J)$ by calculating $\tilde{\boldsymbol{c}}(J) = \tilde{\boldsymbol{c}}_\emptyset \oplus \boldsymbol{t}(J)\tilde{G}$ is $l(n-k+1)+n$ where the first term is required for the calculation of $\boldsymbol{t}(J)\tilde{G}$ and the second term is required for adding two n-dimensional vectors. This complexity depends on the Hamming weight l of $\boldsymbol{t}(J)$.

(ii) We can modify the algorithm to reduce the number of binary operations for generating candidate codewords. For a TEP $\boldsymbol{t}(J)$, let $\tilde{\boldsymbol{w}}(J) = \boldsymbol{t}(J)\tilde{G}$ and we have $\tilde{\boldsymbol{c}}(J) = \tilde{\boldsymbol{c}}_\emptyset \oplus \tilde{\boldsymbol{w}}(J)$. Then we can process the decoding algorithm by iteratively generating $\tilde{\boldsymbol{w}}(J)$ by TEPs $\boldsymbol{t}(J)$ instead of candidate codewords $\tilde{\boldsymbol{c}}(J)$ [3], [4], [6], [7], [9]. In this case, the number of binary operations for generating a codeword $\tilde{\boldsymbol{w}}(J)$ such that $\tilde{\boldsymbol{w}}(J) = \boldsymbol{t}(J)\tilde{G}$ is $l(n-k)$. This complexity also depends on the Hamming weight l of the TEP $\boldsymbol{t}(J)$.

It has been shown that the value l is fairly small when the SNR of the channel is high, however, it becomes greater as the SNR decreases. Therefore, time complexity for generating candidate codewords becomes larger as the SNR of the channel tends to be low.

4. A Method for Reducing the Time Complexity of the GBF Decoding Algorithm

4.1 Fast Method for Generating Candidate Codewords

In this section, we propose a fast method for generating candidate codewords.

In the reduced-list GBF decoding algorithm, the Hamming distance between a newly generated TEP and the best pattern in some decoding stage is not so large. Considering this fact, the key ideas of the proposed method are 1) candidate codewords generated previously are stored in memory and 2) a candidate codeword $\tilde{\boldsymbol{c}}(J)$ is constructed by adding one or two rows of $\tilde{G}$.

For $j \in [1, k]$, we denote the j-th row of $\tilde{P}$ by $\tilde{\boldsymbol{p}}_j$ where $\tilde{P}$ is the right $k \times (n-k)$ submatrix of $\tilde{G}$. Let $\tilde{\boldsymbol{b}}(J) \in \{0, 1\}^{n-k}$

[†]It is not necessary to fix the set $S^{(0)}$ in a decoding procedure. See [12], for detail.

express the right $n-k$ symbols of candidate codeword $\tilde{\boldsymbol{c}}(J) = \tilde{\boldsymbol{c}}_\emptyset \oplus \boldsymbol{t}(J)\tilde{G}$. i.e, $\tilde{\boldsymbol{c}}(J) = (\boldsymbol{u} \oplus \boldsymbol{t}(J)) \circ \tilde{\boldsymbol{b}}(J)$ where the symbol $\circ$ denotes concatenation of two sequences. We arrange the lists $M^{(j)}, j = 1, 2, \ldots, k$, as lists of n-dimensional vectors. We store newly generated TEPs in the left k bits of the list $M^{(j)}, j = 1, 2, \ldots, k$, and parity check symbols of already generated candidate codewords in the right $n-k$ bits of the list $M^{(j)}, j = 1, 2, \ldots, k$.

We first consider generating the candidate codeword $\tilde{\boldsymbol{c}}(J \cup j)$ by extended patterns $\boldsymbol{t}(J \cup j), j > \mu(J)$, of the best pattern $\boldsymbol{t}(J)$. We show the following proposition.

Proposition 1: We can generate the candidate codeword $\tilde{\boldsymbol{c}}(J \cup j)$ by

$$\tilde{\boldsymbol{c}}(J \cup j) = \Big(\boldsymbol{u} \oplus \boldsymbol{t}(J \cup j)\Big) \circ \Big(\tilde{\boldsymbol{b}}(J) \oplus \tilde{\boldsymbol{p}}_j\Big). \tag{5}$$

(**Proof**) By the linearity, $\tilde{\boldsymbol{c}}(J \cup j)$ is rewritten as

$$\begin{aligned}\tilde{\boldsymbol{c}}(J \cup j) &= \Big(\boldsymbol{u} \oplus \boldsymbol{t}(J \cup j)\Big)\tilde{G} \quad (6)\\ &= \Big(\boldsymbol{u} \oplus \boldsymbol{t}(J)\Big)\tilde{G} \oplus \tilde{\boldsymbol{g}}_j, \quad (7)\end{aligned}$$

where $\tilde{\boldsymbol{g}}_j$ denotes the j-th row of $\tilde{G}$. To derive Eq. (7), we use $\boldsymbol{t}(J \cup j)\tilde{G} = \boldsymbol{t}(J)\tilde{G} \oplus \tilde{\boldsymbol{g}}_j$. Since the left k symbols of $\tilde{\boldsymbol{c}}(J \cup j)$ is $\boldsymbol{u} \oplus \boldsymbol{t}(J \cup j)$ by Eq. (6) and the systematic property of $\tilde{G}$, and the right $n-k$ symbols of it is $\tilde{\boldsymbol{b}}(J) \oplus \tilde{\boldsymbol{p}}_j$ by Eq. (7). This proves the proposition. □

Note that by the condition (C1), at the iteration of generating $\tilde{\boldsymbol{c}}(J \cup j)$, we have already generated $\tilde{\boldsymbol{b}}(J)$ (or, equivalently, $\tilde{\boldsymbol{c}}(J)$) since $F(\boldsymbol{t}(J)) \leq F(\boldsymbol{t}(J \cup j))$. Proposition 1 implies that we can obtain the candidate codeword $\tilde{\boldsymbol{c}}(J \cup j)$ by at most n binary operations if we store $\tilde{\boldsymbol{b}}(J)$ in the right $n-k$ bits in the list $M^{(j)}$.

We can similarly devise a fast method for generating candidate codewords $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j)$ by an adjacent pattern $\boldsymbol{t}(J^{\mathrm{a}} \cup j), J^{\mathrm{a}} = J \setminus \mu(J)$.

Proposition 2: We can generate the candidate codeword $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j)$ by

$$\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j) = \Big(\boldsymbol{u} \oplus \boldsymbol{t}(J^{\mathrm{a}} \cup j)\Big) \circ \Big(\tilde{\boldsymbol{b}}(J) \oplus \tilde{\boldsymbol{p}}_j \oplus \tilde{\boldsymbol{p}}_{\mu(J)}\Big). \tag{8}$$

(**Proof**) By the linearity, $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j)$ is rewritten as

$$\begin{aligned}\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j) &= \Big(\boldsymbol{u} \oplus \boldsymbol{t}(J^{\mathrm{a}} \cup j)\Big)\tilde{G} \quad (9)\\ &= \Big(\boldsymbol{u} \oplus \boldsymbol{t}(J^{\mathrm{a}})\Big)\tilde{G} \oplus \tilde{\boldsymbol{g}}_j, \quad (10)\end{aligned}$$

where $\tilde{\boldsymbol{g}}_j$ denotes the j-th row of $\tilde{G}$. Since $J^{\mathrm{a}} = J \setminus \mu(J)$, we have $\boldsymbol{t}(J^{\mathrm{a}}) = \boldsymbol{t}(J) \oplus \boldsymbol{t}(\mu(J))$. Therefore, the first term of the r.h.s. of Eq., (10) is rewritten as

$$\Big(\boldsymbol{u} \oplus \boldsymbol{t}(J^{\mathrm{a}})\Big)\tilde{G} = \Big(\boldsymbol{u} \oplus \boldsymbol{t}(J)\Big)\tilde{G} \oplus \tilde{\boldsymbol{g}}_{\mu(J)}. \tag{11}$$

Hence from Eqs. (10) and (11), we have

$$\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j) = \Big(\boldsymbol{u} \oplus \boldsymbol{t}(J)\Big)\tilde{G} \oplus \tilde{\boldsymbol{g}}_j \oplus \tilde{\boldsymbol{g}}_{\mu(J)}. \tag{12}$$

Since the left k symbols of $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j)$ is $\boldsymbol{u} \oplus \boldsymbol{t}(J^{\mathrm{a}} \cup j)$ by Eq. (9) and the right $n-k$ symbols of it is $\tilde{\boldsymbol{b}}(J) \oplus \tilde{\boldsymbol{p}}_j \oplus \tilde{\boldsymbol{p}}_{\mu(J)}$ by Eq. (12). This proves the proposition. □

Note that by the condition (C2), at the iteration of generating $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j)$, we have already generated $\tilde{\boldsymbol{b}}(J)$ since $F(\boldsymbol{t}(J)) \leq F(\boldsymbol{t}(J^{\mathrm{a}} \cup j))$. Proposition 2 implies that we can obtain the candidate codeword $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j)$ by at most $k + 2(n-k) = 2n - k$ binary operations if we store $\tilde{\boldsymbol{b}}(J)$ in the right $n-k$ bits in the list $M^{(j)}$.

We modify the reduced-list GBF decoding algorithm as follows, where we store $\tilde{\boldsymbol{b}}(J)$ into Γ if any TEPs are generated from the selected best pattern $\boldsymbol{t}(J)$:

- After a new adjacent pattern $\boldsymbol{t}(J^{\mathrm{a}} \cup j)$ is generated, we store $\boldsymbol{t}(J^{\mathrm{a}} \cup j)$ in the left k bits of the list $M^{(j)}$ and store $\tilde{\boldsymbol{b}}(J)$ in the right $n-k$ bits of it at P4-a) or P4-b).
- After a new extended pattern $\boldsymbol{t}(J \cup j)$ is generated, we store the $\boldsymbol{t}(J \cup j)$ in the left k bits of the list $M^{(j)}$ and store $\tilde{\boldsymbol{b}}(J)$ in the right $n-k$ bits of it at P4-c).
- If $\boldsymbol{t}(J \cup j), j \in \{i_s, i'_p\}$, is selected as the best pattern at P2), we calculate the parity check symbols of $\tilde{\boldsymbol{c}}(J \cup j)$ by $\tilde{\boldsymbol{b}}(J \cup j) = \tilde{\boldsymbol{b}}(J) \oplus \tilde{\boldsymbol{p}}_j$. Then we generate the candidate codeword $\tilde{\boldsymbol{c}}(J \cup j)$ by Eq. (5) at P3).
- If $\boldsymbol{t}(J^{\mathrm{a}} \cup i_q), q \neq s$, is selected as the best pattern at P2), we calculate the parity check symbols of $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup i_q)$ by

$$\tilde{\boldsymbol{b}}(J^{\mathrm{a}} \cup i_q) = \tilde{\boldsymbol{b}}(J) \oplus \tilde{\boldsymbol{p}}_{i_{q+1}} \oplus \tilde{\boldsymbol{p}}_{i_q}. \tag{13}$$

  Then we generate the candidate codeword $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup i_q)$ by Eq. (8) at P3). Similar arguments hold if $\boldsymbol{t}(J^{\mathrm{a}} \cup i'_q), q \neq p$, is selected as the best pattern at P2).

For each TEP $\boldsymbol{t}(J)$ in the right k bits of the list $M^{(\mu(J))}$, the parity symbols in the right $n-k$ bits in it is used when we generate a new candidate codeword $\tilde{\boldsymbol{c}}(J)$. These modifications reduce the time complexity for generating candidate codewords from $O(kn)$ binary operations to $O(n)$ ones in the reduced-list GBF decoding algorithm.

We describe an improved GBF decoding algorithm adopting these modifications.

[The Fast GBF Decoding Algorithm]

P'1) This step is the same as P1).
P'2) This step is the same as P2).
P'3) a) If $\mu(J) \in \{i_s, i'_p\}$ for the selected best pattern $\boldsymbol{t}(J)$, generate the next candidate codeword $\tilde{\boldsymbol{c}}(J)$ by Eq. (5). Otherwise, generate the next candidate codeword $\tilde{\boldsymbol{c}}(J)$ by Eq. (8).
b) If $L(\tilde{\boldsymbol{c}}(J)) < \underline{L}$, then set $\underline{L} := L(\tilde{\boldsymbol{c}}(J))$ and $\tilde{\boldsymbol{c}}_{\mathrm{best}} := \tilde{\boldsymbol{c}}(J)$.
P'4) a) If $\mu(J) = i_q$ (i.e., $\mu(J) \in S^{(0)}$) and the adjacent pattern $\boldsymbol{t}(J^{\mathrm{a}} \cup i_{q-1})$ exists, then insert it into the left k bits of the list $M^{(i_{q-1})}$. Insert $\tilde{\boldsymbol{b}}(J)$ into the right $n-k$ bits of the list.
b) If $\mu(J) = i'_q$ (i.e., $\mu(J) \in S^{(1)}$) and $\boldsymbol{t}(J^{\mathrm{a}} \cup i'_{q-1})$ exists, then insert it into $M^{(i'_{q-1})}$. Insert $\tilde{\boldsymbol{b}}(J)$ into the right $n-k$ bits of the list.
c) If $\mu(J) < i_s$, then insert $\boldsymbol{t}(J \cup i_s)$ into $M^{(i_s)}$. If $\mu(J) < i'_p$, then insert $\boldsymbol{t}(J \cup i'_p)$ into $M^{(i'_p)}$. Insert $\tilde{\boldsymbol{b}}(J)$ into the right $n-k$ bits of the list. Delete $\boldsymbol{t}(J)$ from $M^{(\mu(J))}$.

P'5) This step is the same as P5). □

We call the decoding algorithm with these modifications the **fast GBF decoding algorithm**.

4.2 Performance Analysis

On the time complexity of the fast GBF decoding algorithm, we show the following theorems.

Theorem 1: The time complexity for generating candidate codewords in the fast GBF decoding algorithm is at most $2n-k$ binary operations if a heuristic function satisfies conditions (C1) and (C2). If we take the second encoding strategy in Sect. 3.3, this complexity is only $2(n-k)$ operations. □

The number of binary operations for encoding adjacent patterns at P'4-a) and P'4-b) is larger than that for extended patterns at P'4-c), however, this complexity can be reduced in some cases. The parity check symbols of the candidate codeword $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup i_q)$, $i_q \in S^{(0)}$, is calculated by using Eq. (13). If the set $S^{(0)}$ is fixed during a decoding procedure, two rows of $\tilde{P}$ in Eq. (13) are also fixed for $q = 1, 2, \ldots, s-1$. In this case, letting

$$\tilde{\boldsymbol{q}}_{i_q}^{(0)} = \tilde{\boldsymbol{p}}_{i_q} \oplus \tilde{\boldsymbol{p}}_{i_{q+1}}, \quad \text{for } q = 1, 2, \ldots, s-1, \tag{14}$$

and if we store these $\tilde{\boldsymbol{q}}_{i_q}^{(0)}$ in memory, we can calculate Eq. (8) by at most n binary operations. Similarly, letting

$$\tilde{\boldsymbol{q}}_{i'_q}^{(1)} = \tilde{\boldsymbol{p}}_{i'_q} \oplus \tilde{\boldsymbol{p}}_{i'_{q+1}}, \quad \text{for } q = 1, 2, \ldots, p-1, \tag{15}$$

and if we store these $\tilde{\boldsymbol{q}}_{i'_q}^{(1)}$ in memory, we can calculate Eq. (8) by at most n binary operations. The space complexity for storing these $k-1$ vectors is $O((k-1)(n-k))$ binary arrays and this fixed value is less than the arrays for $\tilde{G}$. This method can be applicable to the decoding algorithm with heuristic functions in [1]–[4], [6].

Theorem 2: If the set $S^{(0)}$ is fixed in a decoding procedure, the time complexity for it is n binary operations with increasing the space complexity of $O((k-1)(n-k))$ binary arrays. If we take the second encoding strategy stated in Sect. 3.3, this complexity is only $n-k$ binary operations. □

We here summarize the effectiveness of the fast GBF decoding algorithm in time complexity compared with the original and the reduced-list GBF decoding algorithms. The time complexity for generating candidate codewords in the conventional GBF decoding algorithms depends on $l = |J|$, which is the Hamming weight of a TEP $\boldsymbol{t}(J)$. The value l tends to increase as the SNR decreases. On the other hand, the time complexity for generating candidate codewords in the fast GBF decoding algorithm is independent of l. Therefore, the effectiveness of the fast GBF decoding algorithm in time complexity becomes larger as the SNR of the channel tends to decrease.

The fast GBF decoding algorithm increases the total list size of the reduced-list GBF decoding algorithm to reduce the time complexity. On the total list size of the fast GBF decoding algorithm, we show the following theorem.

Theorem 3: The total list size of the fast GBF decoding algorithm is exactly $\frac{n}{k}$ times of that for the reduced-list GBF decoding algorithm.

(**Proof**) The number of parity check symbols of $(n-k)$-dimensional vectors in the lists is the same as that of TEPs in the lists. The increased space complexity of the fast GBF decoding algorithm from the reduced-list GBF decoding algorithm is at most $(n-k) \times M(\tilde{\boldsymbol{r}})$ binary arrays where $M(\tilde{\boldsymbol{r}})$ represents the maximum list size of TEPs in decoding of $\tilde{\boldsymbol{r}}$. Therefore, the number of binary arrays for the total list in the fast GBF decoding algorithm is $n \times M(\tilde{\boldsymbol{r}})$ since that for the maximum list of TEPs is $k \times M(\tilde{\boldsymbol{r}})$. On the other hand, the number of binary arrays for the total list in the reduced-list GBF decoding algorithm is $k \times M(\tilde{\boldsymbol{r}})$ and thus the space complexity for the fast GBF decoding algorithm is exactly as $\frac{n}{k}$ times as that for the reduced-list GBF decoding algorithm. □

5. Simulation Results

In this section, we evaluate the effectiveness of the fast GBF decoding algorithm by computer simulations.

5.1 Conditions of Simulations

For the (63, 30, 13) BCH code and the (104, 52, 20) quadratic residue (QR) code, we perform MLD by three algorithms: the GBF decoding algorithm [2], [6] (denoted by "GBF" in tables), the proposed fast GBF decoding algorithm (denoted by "Fast GBF") and the reduced-list GBF decoding algorithm [12] (denoted by "RL GBF"). We compare the following two items:

- **(i) [The time complexity]** the number of binary operations for generating candidate codewords
- **(ii) [The space complexity]** the maximum list size in each decoding algorithm.

At each SNR E_{b}/S_0 [dB], all decoding algorithms are carried out 10,000 times.

We assume the second strategy for generating candidate codeword in Sect. 3.3: for a TEP $\boldsymbol{t}(J)$, the number of binary operations is $l(n-k)$ where $l = |J|$ in the GBF (also, the reduced-list GBF) decoding algorithm and it is $n-k$ in the fast GBF decoding algorithm by storing consecutive two rows of $\tilde{G}$.

In simulations, we employ two heuristic functions for each decoding algorithm: (a) a heuristic function proposed by Han et al. [2], which is considered as a standard one of the priority-first search-type decoding algorithms, and (b) a heuristic function proposed by Fossorier and Lin [4], which is shown to be more effective than that of [2]. For both heuristic functions, we fix the set $S^{(0)}$ as $S^{(0)} = [1, k]$ in a decoding procedure.

Table 1 The number of binary operations for generating candidate codewords for the (63, 30, 13) BCH code with a heuristic function of Han et al. [2].

E_b/N_0	GBF	Fast GBF	Ratio
5.0	$7.28 \cdot 10^1$	$3.79 \cdot 10^1$	0.521
4.5	$3.53 \cdot 10^2$	$1.53 \cdot 10^2$	0.434
4.0	$1.10 \cdot 10^3$	$4.75 \cdot 10^2$	0.431
3.5	$3.90 \cdot 10^3$	$1.53 \cdot 10^3$	0.392
3.0	$1.33 \cdot 10^4$	$4.62 \cdot 10^3$	0.347
2.5	$3.65 \cdot 10^4$	$1.16 \cdot 10^4$	0.319
2.0	$8.58 \cdot 10^4$	$2.58 \cdot 10^4$	0.300

Table 2 The number of binary operations for generating candidate codewords for the (104, 52, 20) QR code with a heuristic function of Han et al. [2].

E_b/N_0	GBF	Fast GBF	Ratio
6.0	$1.85 \cdot 10^1$	$1.13 \cdot 10^1$	0.615
5.5	$1.03 \cdot 10^2$	$5.71 \cdot 10^1$	0.552
5.0	$8.25 \cdot 10^2$	$3.60 \cdot 10^2$	0.437
4.5	$7.26 \cdot 10^3$	$2.52 \cdot 10^3$	0.347
4.0	$6.46 \cdot 10^4$	$1.80 \cdot 10^4$	0.279
3.5	$5.44 \cdot 10^5$	$1.24 \cdot 10^5$	0.227
3.0	$2.54 \cdot 10^6$	$5.41 \cdot 10^5$	0.213

Table 3 The number of binary operations for generating candidate codewords for the (63, 30, 13) BCH code with a heuristic function of Fossorier and Lin [4].

E_b/N_0	GBF	Fast GBF	Ratio
5.0	$4.99 \cdot 10^1$	$2.74 \cdot 10^1$	0.549
4.5	$2.65 \cdot 10^2$	$1.16 \cdot 10^2$	0.439
4.0	$8.70 \cdot 10^2$	$3.78 \cdot 10^2$	0.435
3.5	$3.28 \cdot 10^3$	$1.28 \cdot 10^3$	0.389
3.0	$1.19 \cdot 10^4$	$4.07 \cdot 10^3$	0.343
2.5	$3.39 \cdot 10^4$	$1.07 \cdot 10^4$	0.315
2.0	$8.20 \cdot 10^4$	$2.44 \cdot 10^4$	0.298

Table 4 The number of binary operations for generating candidate codewords for the (104, 52, 20) QR code with a heuristic function of Fossorier and Lin [4].

E_b/N_0	GBF	Fast GBF	Ratio
6.0	4.98	4.60	0.925
5.5	$2.93 \cdot 10^1$	$2.27 \cdot 10^1$	0.775
5.0	$2.72 \cdot 10^2$	$1.38 \cdot 10^2$	0.507
4.5	$3.60 \cdot 10^3$	$1.17 \cdot 10^3$	0.326
4.0	$4.20 \cdot 10^4$	$1.07 \cdot 10^4$	0.254
3.5	$4.35 \cdot 10^5$	$9.25 \cdot 10^4$	0.212
3.0	$2.29 \cdot 10^6$	$4.67 \cdot 10^5$	0.203

5.2 Results about Time Complexity

We show the results about the number of binary operations for generating candidate codewords. We show the results for the (63, 30, 13) BCH code and the (104, 52, 20) QR code with a heuristic function of [2] in Tables 1 and 2, respectively. The results for both codes with a heuristic function of [4] are in Tables 3 and 4, respectively. In tables, we show the average values among 10,000 decoding procedures. The values in the column "Ratio" indicate the ratio of the number of binary operations of the fast GBF decoding algorithm to that of the conventional (the original and the reduced-list GBF) decoding algorithms.

By Table 1 for the (63, 30, 13) code, the ratio of the number of binary operations in the fast GBF decoding algorithm to that in the GBF decoding algorithm is about 1/2 at 5.0 [dB]. The ratio to the GBF decoding algorithm tends to be small as the SNR decreases and it reaches less than 1/3 at 2.0 [dB]. These results demonstrate that the effectiveness of the fast GBF decoding algorithm becomes high as the SNR decreases. By Table 2 for the (104, 52, 20) QR code, although the ratio is about 3/5 at high SNRs, the speed for the ratio to decrease is faster than that for the (63, 30, 13) BCH code. Theses results demonstrate the proposed method works well for a longer code.

From Table 3, for the (63, 30, 13) code with a heuristic function of [4], we can see that the ratios of the fast GBF decoding algorithm to the original GBF decoding algorithm are similar to those in Table 1, although the numbers of binary operations are less than those in Table 1. On the other hand, from Table 4 for the (104, 52, 20) code, the ratios are small at high SNRs. This reason is that the heuristic function of [4] is more effective at high SNRs, so the Hamming weight of TEPs, l, is very small. In this case, there are almost no differences between both algorithms. At medium to low SNRs, however, the reduction rates of the fast GBF decoding algorithm to the GBF decoding algorithm show the similar behavior to those in Table 2.

We see that the reduction rate of the number of binary operations from the simulation results is not so large as that from the theoretical analysis. This is due to the effectiveness of the conventional reliability-based decoding algorithm that finds out the ML codeword before generating TEPs with a large Hamming weight. However, we can say that it is very significant that the order of the time complexity is theoretically reduced from $O(kn)$ to $O(n)$ and the maximum number of binary operation is bounded. At this point, the fast GBF decoding algorithm has the high efficiency.

Is is known that maximum likelihood decoding of block codes at extremely low SNRs is impractically difficult. In our simulations, we have observed that the numbers of binary operations at 0.0 [dB] for the (63, 30, 13) BCH code are $6.94 \cdot 10^5$ in the GBF decoding algorithm and $1.82 \cdot 10^5$ in the fast GBF decoding algorithm. As for longer codes with $n > 100$ such as the (104, 52, 20) QR code, we still cannot carry out simulations at extremely low SNRs.

5.3 Results about Space Complexity

We show the results about the total list size for the (63, 30, 13) BCH code and (104, 52, 20) QR code. The results with the heuristic function of [2] are in Tables 5 and 6. The results for each code with the heuristic function of [4] are in Tables 7 and 8, respectively. In tables, we denote the average value of the normalized list size by "Ave" and the maximum

Table 5 The total list size for the $(63, 30, 13)$ BCH code with a heuristic function of Han et al. [2].

E_b/N_0 [dB]		GBF	Proposed	RL GBF
5.0	Ave	1.46 (1.000)	$4.03 \cdot 10^{-1}$ (0.276)	$1.92 \cdot 10^{-1}$ (0.131)
	Max	$2.07 \cdot 10^{3}$ (1.000)	$8.63 \cdot 10^{2}$ (0.417)	$4.11 \cdot 10^{2}$ (0.198)
4.0	Ave	$1.40 \cdot 10^{1}$ (1.000)	4.79 (0.342)	2.28 (0.163)
	Max	$9.59 \cdot 10^{3}$ (1.000)	$5.06 \cdot 10^{3}$ (0.528)	$2.41 \cdot 10^{3}$ (0.251)
3.0	Ave	$1.13 \cdot 10^{2}$ (1.000)	$4.85 \cdot 10^{1}$ (0.429)	$2.31 \cdot 10^{1}$ (0.204)
	Max	$1.58 \cdot 10^{4}$ (1.000)	$9.66 \cdot 10^{3}$ (0.611)	$4.60 \cdot 10^{3}$ (0.291)
2.0	Ave	$5.85 \cdot 10^{2}$ (1.000)	$2.86 \cdot 10^{2}$ (0.488)	$1.36 \cdot 10^{2}$ (0.233)
	Max	$7.05 \cdot 10^{4}$ (1.000)	$4.33 \cdot 10^{4}$ (0.614)	$2.06 \cdot 10^{4}$ (0.291)

Table 6 The total list size for the $(104, 52, 20)$ QR code with a heuristic function of Han et al. [2].

E_b/N_0 [dB]		GBF	Proposed	RL GBF
6.5	Ave	$6.50 \cdot 10^{-2}$ (1.000)	$9.20 \cdot 10^{-3}$ (0.142)	$4.60 \cdot 10^{-3}$ (0.071)
	Max	$8.70 \cdot 10^{1}$ (1.000)	$2.00 \cdot 10^{1}$ (0.230)	$1.00 \cdot 10^{1}$ (0.115)
5.5	Ave	1.59 (1.000)	$2.74 \cdot 10^{-1}$ (0.172)	$1.37 \cdot 10^{-1}$ (0.086)
	Max	$1.67 \cdot 10^{3}$ (1.000)	$5.34 \cdot 10^{2}$ (0.320)	$2.67 \cdot 10^{2}$ (0.160)
4.5	Ave	$4.44 \cdot 10^{1}$ (1.000)	$1.13 \cdot 10^{1}$ (0.255)	5.66 (0.127)
	Max	$7.26 \cdot 10^{4}$ (1.000)	$2.16 \cdot 10^{4}$ (0.298)	$1.08 \cdot 10^{4}$ (0.149)
3.5	Ave	$1.88 \cdot 10^{3}$ (1.000)	$7.04 \cdot 10^{2}$ (0.374)	$3.52 \cdot 10^{2}$ (0.187)
	Max	$3.68 \cdot 10^{6}$ (1.000)	$1.78 \cdot 10^{6}$ (0.483)	$8.88 \cdot 10^{5}$ (0.242)

Table 7 The total list size for the $(63, 30, 13)$ BCH code with a heuristic function of Fossorier and Lin [4].

E_b/N_0 [dB]		GBF	Proposed	RL GBF
5.0	Ave	1.20 (1.000)	$3.05 \cdot 10^{-1}$ (0.254)	$1.45 \cdot 10^{-1}$ (0.121)
	Max	$2.07 \cdot 10^{3}$ (1.000)	$8.63 \cdot 10^{2}$ (0.417)	$4.11 \cdot 10^{2}$ (0.198)
4.0	Ave	$1.16 \cdot 10^{1}$ (1.000)	3.89 (0.335)	1.85 (0.160)
	Max	$9.59 \cdot 10^{3}$ (1.000)	$5.06 \cdot 10^{3}$ (0.528)	$2.41 \cdot 10^{3}$ (0.251)
3.0	Ave	$9.97 \cdot 10^{1}$ (1.000)	$4.33 \cdot 10^{1}$ (0.434)	$2.06 \cdot 10^{1}$ (0.207)
	Max	$1.52 \cdot 10^{4}$ (1.000)	$9.32 \cdot 10^{3}$ (0.613)	$4.44 \cdot 10^{3}$ (0.292)
2.0	Ave	$5.52 \cdot 10^{2}$ (1.000)	$2.73 \cdot 10^{2}$ (0.495)	$1.30 \cdot 10^{2}$ (0.235)
	Max	$7.05 \cdot 10^{4}$ (1.000)	$4.33 \cdot 10^{4}$ (0.614)	$2.06 \cdot 10^{4}$ (0.291)

Table 8 The total list size for the $(104, 52, 20)$ QR code with a heuristic function of Fossorier and Lin [4].

E_b/N_0 [dB]		GBF	Proposed	RL GBF
6.5	Ave	$6.50 \cdot 10^{-2}$ (1.000)	$9.20 \cdot 10^{-3}$ (0.142)	$4.60 \cdot 10^{-3}$ (0.071)
	Max	$8.70 \cdot 10^{1}$ (1.000)	$2.00 \cdot 10^{1}$ (0.230)	$1.00 \cdot 10^{1}$ (0.115)
5.5	Ave	$9.99 \cdot 10^{-1}$ (1.000)	$1.55 \cdot 10^{-1}$ (0.155)	$7.75 \cdot 10^{-2}$ (0.078)
	Max	$1.23 \cdot 10^{3}$ (1.000)	$1.12 \cdot 10^{2}$ (0.091)	$5.60 \cdot 10^{1}$ (0.046)
4.5	Ave	$2.16 \cdot 10^{1}$ (1.000)	5.90 (0.273)	2.95 (0.137)
	Max	$5.67 \cdot 10^{4}$ (1.000)	$2.08 \cdot 10^{4}$ (0.367)	$1.04 \cdot 10^{4}$ (0.183)
3.5	Ave	$1.38 \cdot 10^{3}$ (1.000)	$5.54 \cdot 10^{2}$ (0.401)	$2.77 \cdot 10^{2}$ (0.201)
	Max	$3.68 \cdot 10^{6}$ (1.000)	$1.78 \cdot 10^{6}$ (0.483)	$8.88 \cdot 10^{5}$ (0.242)

value of it by “Max” among 10,000 decoding. The values in round brackets indicate the ratio to the maximum list size of the GBF decoding algorithm.

By Table 5 for the (63, 30, 13) code, the average value of maximum list size in the fast GBF decoding algorithm is about 1/4 of that in the GBF decoding algorithm at 5.0 [dB]. Although the ratio to the GBF decoding algorithm becomes high as the SNR decreases, all values of the fast GBF decoding algorithm are less than those of the GBF decoding algorithm. These results show that there is no increase of space complexity in the fast GBF decoding algorithm from the GBF decoding algorithm although the time complexity is greatly reduced. By Table 6 for the (104, 52, 20) code, the average value of maximum list size in the fast GBF decoding algorithm is about 1/8 of that in the GBF decoding algorithm at 6.5 [dB]. Although the ratio to the GBF decoding algorithm also increases as the SNR decreases, the total list size of the fast GBF decoding algorithm is still less than half of that of the GBF decoding algorithm. Theses results indicate the proposed method works well for a longer code, too.

From Tables 7 and 8, we can see that the reduction rates of the fast GBF decoding algorithm to the GBF decoding algorithm with a heuristic function of [4] are almost the same as those in Tables 5 and 6 at each SNR. In terms of the space complexity, the effectiveness of the fast GBF decoding algorithm is almost the same even for a different heuristic function.

6. Conclusion and Future Works

In this paper, we have proposed a new method for reducing the time complexity of generating candidate codewords in the GBF decoding algorithm. Although the fast GBF decoding algorithm requires some additional space complexity for storing some already generated candidate codewords, the space complexity for lists is $\frac{n}{k}$ times that for the reduced-list

GBF decoding algorithm [12]. We showed by simulation results that the total list size of the fast GBF decoding algorithm is still smaller than that of the GBF decoding algorithm, even though the time complexity is significantly reduced.

As future works, we need to develop an efficient method for heuristic search MLD algorithm with powerful heuristic functions such as in [5].

Acknowledgments

This work is supported by Waseda University Grant for Special Research Project No. 2005A-952 and 2006B-293. The authors would like to thank anonymous reviewers for their valuable comments. H. Yagi wishes to thank Dr. M. Kobayashi at Shonan Institute of Technology and Mr. T. Ishida and Mr. G. Hosoya at Waseda University for their suggestions.

References

[1] G. Battail and J. Fang, "Décodage pondéré optimal descodes linéaires en blocs," Annales des télé-communications, vol.41, no.11-12, pp.580–604, Nov.-Dec. 1986.

[2] Y.S. Han, C.R.P. Hartmann, and C.C. Chan, "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.

[3] 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.

[4] 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.

[5] 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.

[6] A. Valembois and M. Fossorier, "An improved method to compute lists of binary vectors that optimize a given weight function with application to soft decision decoding," IEEE Commun. Lett., vol.5, no.3, pp.456–458, Nov. 2001.

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

[8] Y. Wu and C.N. Hadjicostis, "Soft-decision decoding of linear block codes using efficient G-space encodings," Proc. 2001 IEEE Global Telecom. Conf., vol.2, pp.921–925, San Antonio, USA, Nov. 2001.

[9] H. Yagi, T. Matsushima, and S. Hirasawa, "A method for reducing space complexity of reliability-based heuristic search maximum likelihood decoding algorithms," Proc. 26th Symposium on Inform. Theory and its Applications, (SITA2003), pp.185–188, Hyogo, Japan, Dec. 2003.

[10] H. Yagi, M. Kobayashi, T. Matsushima, 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.2461–2472, Oct. 2003.

[11] H. Yagi, T. Matsushima, and S. Hirasawa, "Fast algorithm for generating candidate codewords in reliability-based maximum likelihood decoding," IEICE Technical Report, IT2005-13, May 2005.

[12] H. Yagi, T. Matsushima, and S. Hirasawa, "A heuristic search algorithm with the reduced list of test error patterns for maximum likelihood decoding algorithms," IEICE Trans. Fundamentals, vol.E88-A, no.10, pp.2721–2733, Oct. 2005.