社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
IT2005-13(2005-05)

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

Hideki YAGI*† Toshiyasu MATSUSHIMA* Shigeichi HIRASAWA*

*Department of Industrial and Management Systems Engineering, Waseda University
3-4-1 Ohkubo Shinjuku-ku, Tokyo, 169-8555 Japan
†Media Network Center, Waseda University.
E-mail: yagi@hirasa.mgmt.waseda.ac.jp

Abstract The reliability-based heuristic search methods for maximum likelihood decoding (MLD) generate test error patterns (or, equivalently, candidate codewords) according to their heuristic values. Test error patterns are stored in lists and this makes the space complexity crucially large for MLD of long block codes. Then some studies have proposed methods for reducing the list size of test error patterns in these MLD algorithms including the well-known A* decoding algorithm proposed by Han et al. 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 almost negligible.

Keywords maximum likelihood decoding, binary block codes, priority-first search, most reliable basis, reliability

1 Introduction

Maximum likelihood decoding (MLD) of block codes minimizes the probability of decoding error when we assume that all codewords have equal probability to be transmitted. Since the complexity of searching the most likely (ML) codeword among all codewords is significantly large, many researchers have devoted to develop efficient algorithms for MLD. One of the most efficient MLD algorithms is the reliability-based decoding algorithm that uses the column permuted generator matrix in non-increasing order of reliability.

In this paper, we consider the priority-first search-type MLD algorithms where candidate codewords are generated in increasing value of a heuristic (or evaluation) function. As known to the authors, G. Battail and J. Fang first proposed a priority-first search MLD algorithm where a simple heuristic function is employed [1] (we will call this algorithm the BF decoding algorithm). One of the most well-known priority-first search MLD algorithms is the A* decoding algorithm proposed by Y. S. Han et al. [2]. Recently, A. Valembois and M. Fossorier have indicated that a certain generalization makes the BF decoding algorithm equivalent to the A* decoding algorithm [6, 7]. Subsequently, Valembois et al. [6] and the present authors [8] have proposed methods for drastically reducing the space complexity of the generalized BF (GBF) decoding algorithm employing some class of heuristic functions. Furthermore, the method in [10] can theoretically reduce the space complexity of the A* decoding algorithm in which the search is guided by more sophisticated heuristic functions than that considered in [6, 8]. Although the space complexity of priority-first search MLD algorithms has been reduced, their time complexity hardly decreases since the numbers of generating candidate codewords in priority-first search MLD algorithms have not been reduced. The dominant complexity per one iteration of the decoding algorithm may be $O(kn)$ binary operations for generating a candidate codeword where n and k represent the code length and the number of information symbols, respectively.

In this paper, based on the method in [10], we propose a new method for reducing the time complexity for 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 [10], its space complexity is theoretically upper bounded by $\frac{n}{k}$ times that for the method in [10]. Consequently, we show by computer simulations that the space complexity for the proposed decoding algorithm is still reduced compared with the A* decoding algorithm. We also devise a new method for reducing the number of real number operations in the A* decoding algorithm.

2 Reliability-based MLD Algorithm

Let $\mathcal{C}$ 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 $\mathcal{C}$ by G and the weight profile of $\mathcal{C}$ by $W(\mathcal{C})$. We assume any codewords $\boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \{0,1\}^n$ of $\mathcal{C}$ 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 $z_j = 0$ if $\theta_j \geq 0$ and $z_j = 1$ otherwise. 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 $\tilde{G} = [I_k|\tilde{P}]$ where the leftmost k positions are the **most reliable and linearly independent** (MRI) [2, 5, 6, 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 $\tilde{G}$. 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., $u_j = \tilde{z}_j, 1 \leq \forall j \leq k$. 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[1]. This procedure is repeated until a sufficient condition for the ML codeword is satisfied.

Definition 1 For a location set $J \subseteq [1,k]$, the test error pattern (TEP) $\boldsymbol{t}(J) = \big(t_1(J), t_2(J), \ldots, t_k(J)\big)$ has element one in J. Such J is called the support of $\boldsymbol{t}(J)$. 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

$$L(\boldsymbol{v}) = \sum_{j|v_j \neq \tilde{z}_j} |\tilde{\theta}_j|. \tag{1}$$

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

3 Priority-first Search Method of Test Error Patterns

3.1 The A* Decoding Algorithm

The A* decoding algorithm [2] searches the ML codeword through trellis diagram or binary tree of the code. Valembois and Fossorier have indicated that the GBF decoding algorithm and the A* decoding algorithm are equivalent when both algorithms adopt the same heuristic function. Both algorithms are called the **priority-first search MLD algorithms** where TEPs are generated in increasing order of heuristic values. In this section, we state the A* decoding algorithm from the Valembois' perspective [6].

We first describe heuristic functions of TEPs. For a given $\tilde{\boldsymbol{c}}_{\mathrm{ref}} \in \tilde{C}$ and $\boldsymbol{t} = (t_1, t_2, \ldots, t_k)$, we define

$$T(\boldsymbol{t}, \tilde{\boldsymbol{c}}_{\mathrm{ref}}) = \Big\{ \boldsymbol{v} = (\boldsymbol{u} \oplus \boldsymbol{t}) \circ (v_{k+1}, v_{k+2}, \ldots, v_n) \Big| \text{ and } d_H(\boldsymbol{v}, \tilde{\boldsymbol{c}}_{\mathrm{ref}}) \in W(\tilde{C}) \Big\}, \tag{2}$$

where the symbol $\circ$ and $d_H(\cdot,\cdot)$ denote the concatenation of two sequences and the Hamming distance, respectively. Then the heuristic function considered in [2] is defined as

$$f(\boldsymbol{t}, \tilde{\boldsymbol{c}}_{\mathrm{ref}}) = \sum_{j|\, t_j = 1} |\tilde{\theta}_j| + \min_{\boldsymbol{v} \in T(\boldsymbol{t}, \tilde{\boldsymbol{c}}_{\mathrm{ref}})} \Big\{ \sum_{\tilde{z}_j \neq v_j} |\tilde{\theta}_j| \Big\}. \tag{3}$$

Such $\tilde{\boldsymbol{c}}_{\mathrm{ref}}$ is called the **referenced codeword** [3, 6] or the seed [2, 5].

The A* decoding algorithm performs the priority-first search with not only the function $f(\cdot)$ but any heuristic functions $F(\cdot)$ satisfying the following condition:

(C1) $\quad F\big(\boldsymbol{t}(J)\big) \leq F\big(\boldsymbol{t}(J \cup j)\big) \quad$ for $j \notin J$.

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

$$F\big(\boldsymbol{t}(J)\big) \leq L\big(\tilde{\boldsymbol{c}}(J)\big), \tag{4}$$

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

Hereafter, we mention how to perform the priority-first search of TEPs using a heuristic function satisfying (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 $M^{(j)} = \{\boldsymbol{t}(j)\}$ 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 A* decoding algorithm which is equivalent to the GBF decoding algorithm suggested by Valembois et al. [6, 7].

[The A* decoding algorithm]

S1) Set $\tilde{\boldsymbol{c}}_{\emptyset} := \boldsymbol{u}\tilde{G}$, $\tilde{\boldsymbol{c}}_{\mathrm{best}} := \tilde{\boldsymbol{c}}_{\emptyset}$ 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 topmost TEPs in non-empty lists $M^{(j)}$. If $F\big(\boldsymbol{t}(J)\big) \geq \underline{L}$, then output $\tilde{\boldsymbol{c}}_{\mathrm{best}}$ 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 $L\big(\tilde{\boldsymbol{c}}(J)\big) < \underline{L}$, then set $\underline{L} := L\big(\tilde{\boldsymbol{c}}(J)\big)$ and $\tilde{\boldsymbol{c}}_{\mathrm{best}} := \tilde{\boldsymbol{c}}(J)$.

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 $\tilde{\boldsymbol{c}}_{\mathrm{best}}$ and halt the algorithm. Otherwise, go to S2). □

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

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 A* decoding algorithm is maintained [6].

In the original A* decoding algorithm, the only one list of TEPs is used. If we combine the k lists into the united list and order TEPs in increasing heuristic values in it, then the above algorithm becomes identical to the original A* decoding algorithm although the behaviors of the two algorithms are seemingly different [6, 7].

3.2 A Reduced-List Method of the A* Decoding Algorithm

In [10], the present authors have proposed a technique for reducing the list size of TEPs in the A* decoding algorithm. We call this method the **reduced-list A* decoding algorithm**. We here briefly review it.

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

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

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

The function $f(\cdot)$ has the following property.

Proposition 1 ([10]) For a given $\tilde{\boldsymbol{c}}_{\rm ref} \in \tilde{\mathcal{C}}$, let $\boldsymbol{t}_{\rm ref}$ be the TEP of $\tilde{\boldsymbol{c}}_{\rm ref}$. Assuming that $S^{(1)}$ be the support of $\boldsymbol{t}_{\rm ref}$. Then the heuristic function $f(\cdot, \tilde{\boldsymbol{c}}_{\rm ref})$ satisfies the condition (C2). □

In the following, we consider heuristic functions satisfying both[2](C1) and (C2).

The strategy of the reduced-list A* decoding algorithm is like *lazy evaluation* where any TEPs are not generated as long as possible. This approach is similar to improved techniques in [6, 8] where other heuristic functions are considered. Hereafter, for convenience, we assume $S^{(0)} = \{i_1, i_2, \ldots, i_s\}$ 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, we may as well construct the initial lists as

$$M^{(j)} = \begin{cases} \{\boldsymbol{t}(j)\}, & \text{if } j \in \{i_s, i'_p\}; \\ \emptyset, & \text{otherwise,} \end{cases} \qquad (5)$$

by the condition (C2).

At S2) of the A* 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. This is guaranteed by (C2), since $F\big(\boldsymbol{t}(J \cup j)\big) \geq F\big(\boldsymbol{t}(J \cup i_s)\big)$ for $\forall j \in S^{(0)}$ and $F\big(\boldsymbol{t}(J \cup j)\big) \geq F\big(\boldsymbol{t}(J \cup i'_p)\big)$ for $\forall j \in S^{(1)}$. Following this modification, after $\boldsymbol{t}(J \cup i_q)$ is chosen as the best pattern at S2), $\boldsymbol{t}(J \cup i_{q-1})$ is inserted into the list $M^{(i_{q-1})}$ if it exists.

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

[The reduced-list A* decoding algorithm]

P1) Set $\tilde{\boldsymbol{c}}_{\emptyset} := \boldsymbol{u}\tilde{G}$, $\tilde{\boldsymbol{c}}_{\rm best} := \tilde{\boldsymbol{c}}_{\emptyset}$ and $\underline{L} := L(\tilde{\boldsymbol{c}}_{\emptyset})$. Construct the initial lists of TEPs by eq. (5).

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 $\boldsymbol{t}(J^{\rm a} \cup i_{q-1})$ exists where $J^{\rm a} = J \setminus \mu(J)$, then insert it into the list $M^{(i_{q-1})}$.

b) If $\mu(J) = i'_q$ (i.e., $\mu(J) \in S^{(1)}$) and $\boldsymbol{t}(J^{\rm a} \cup i'_{q-1})$ 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), while we need to store at most $k - \mu(J)$ TEPs at S4) of the A* decoding algorithm.

We here mention the time complexity of the original and the reduced-list A* 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, 3, 4, 9]. These steps are carried out only once in a decoding procedure. Contrary to the above steps, encoding $\boldsymbol{t}(J)$ by $\tilde{\boldsymbol{c}}(J) = \tilde{\boldsymbol{c}}_{\emptyset} \oplus \boldsymbol{t}(J)\tilde{G}$ and computing its discrepancy are carried out iteratively, where each encoding requires $O(kn)$ binary operations by conventional encoding method [4, 5, 9]. Therefore, both generating candidate codewords and their discrepancy computations dominate the whole decoding complexity [2, 5, 6] and the time complexity per one iteration is $O(kn)$.

4 Proposed Decoding Algorithm

In this section, we propose a method for reducing the time complexity for the original and the reduced-list A* decoding algorithms.

4.1 Fast Method for Generating Candidate Codewords

In this section, we state a fast method for generating candidate codewords. We first derive a method for the original A* decoding algorithm.

For $j \in [1,k]$, we denote 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 $\tilde{\boldsymbol{b}}(J) \in \{0,1\}^{n-k}$ express the right $n-k$ symbols of candidate

[2]Even if we use heuristic functions considered in [1, 3, 4, 6], the same argument described here holds.

codeword $\tilde{\boldsymbol{c}}(J) = \tilde{\boldsymbol{c}}_{\emptyset} \oplus \boldsymbol{t}(J)\tilde{G}$. i.e, $\tilde{\boldsymbol{c}}(J) = (\boldsymbol{u} \oplus \boldsymbol{t}(J)) \circ \tilde{\boldsymbol{b}}(J)$. We arrange a new list Γ as the list of $(n-k)$-dimensional vectors. We store parity check symbols of already generated candidate codewords in the list Γ.

For extended patterns $\boldsymbol{t}(J \cup j), j > \mu(J)$, of the best pattern $\boldsymbol{t}(J)$, consider generating the candidate codeword $\tilde{\boldsymbol{c}}(J \cup j) = \tilde{\boldsymbol{c}}_{\emptyset} \oplus \boldsymbol{t}(J \cup j)\tilde{G}$. By the linearity, the $\tilde{\boldsymbol{c}}(J \cup j)$ is rewritten as

$$\tilde{\boldsymbol{c}}(J \cup j) = \big(\boldsymbol{u} \oplus \boldsymbol{t}(J \cup j)\big)\tilde{G} \tag{6}$$

$$= \big(\boldsymbol{u} \oplus \boldsymbol{t}(J)\big)\tilde{G} \oplus \tilde{\boldsymbol{g}}_j, \tag{7}$$

where the $\tilde{\boldsymbol{g}}_j$ 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 right $n-k$ symbols of it is $\tilde{\boldsymbol{b}}(J) \oplus \tilde{\boldsymbol{p}}_j$ by eq. (7), we have the following proposition.

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

$$\tilde{\boldsymbol{c}}(J \cup j) = \big(\boldsymbol{u} \oplus \boldsymbol{t}(J \cup j)\big) \circ \big(\tilde{\boldsymbol{b}}(J) \oplus \tilde{\boldsymbol{p}}_j\big). \tag{8}$$

□

Proposition 2 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 list Γ.

At the step S4) of the A* decoding algorithm, we insert $k - \mu(J)$ extended patterns of the best pattern $\boldsymbol{t}(J)$. For any TEPs $\boldsymbol{t}(J \cup j)$, we allocate a pointer connected to the parity check symbols $\tilde{\boldsymbol{b}}(J)$ in Γ. Then we modify the A* decoding algorithm as follows:

- If there are any extended patterns $\boldsymbol{t}(J \cup j)$ of the best pattern $\boldsymbol{t}(J)$, then we store $\tilde{\boldsymbol{b}}(J)$ at the end of the list Γ. After a new extended pattern $\boldsymbol{t}(J \cup j)$ is generated at S4), we store it into the list $M^{(j)}$ and connect its pointer to $\tilde{\boldsymbol{b}}(J)$.
- If a TEP $\boldsymbol{t}(J \cup j)$ is selected as the best pattern, then 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 new candidate codeword $\tilde{\boldsymbol{c}}(J \cup j)$ by eq. (8) at S3).

These modifications reduce the time complexity for generating candidate codewords from $O(kn)$ binary operations to $O(n)$ ones in the A* decoding algorithm. Remark that if there are no TEPs whose pointers connect to $\tilde{\boldsymbol{b}}(J)$ in lists, we need not maintain such $\tilde{\boldsymbol{b}}(J)$ in Γ.

We can similarly devise a fast method for generating candidate codewords in the reduced-list A* decoding algorithm. Consider an adjacent pattern $\boldsymbol{t}(J^{\mathrm{a}} \cup j), J^{\mathrm{a}} = J \setminus \mu(J)$, and generating the candidate codeword $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j)$.

Proposition 3 We can generate the candidate codeword $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j)$ by

$$\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup j) = \big(\boldsymbol{u} \oplus \boldsymbol{t}(J^{\mathrm{a}} \cup j)\big) \circ \big(\tilde{\boldsymbol{b}}(J) \oplus \tilde{\boldsymbol{p}}_j \oplus \tilde{\boldsymbol{p}}_{\mu(J)}\big). \tag{9}$$

□

Proposition 3 implies that we can also 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 list Γ.

We modify the reduced-list A* 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 list $M^{(j)}$ and we connect its pointer to $\tilde{\boldsymbol{b}}(J)$ in the list Γ 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 list $M^{(j)}$ and we connect its pointer to $\tilde{\boldsymbol{b}}(J)$ in the list Γ at P4-c).
- If $\boldsymbol{t}(J \cup j), j \in \{i_s, i'_p\}$, is selected as the best pattern, 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. (9) at P3).
- If $\boldsymbol{t}(J^{\mathrm{a}} \cup i_q), q \neq s$, is selected as the best pattern, we calculate the parity check symbols of $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup i_q)$ by

  $$\tilde{\boldsymbol{b}}(J^{\mathrm{a}} \cup i_q) = \tilde{\boldsymbol{b}}(J) \oplus \tilde{\boldsymbol{p}}_{i_{q+1}} \oplus \tilde{\boldsymbol{p}}_{i_q}. \tag{10}$$

  Then we generate the candidate codeword $\tilde{\boldsymbol{c}}(J^{\mathrm{a}} \cup i_q)$ by eq. (9) at P3). Similar arguments hold if $\boldsymbol{t}(J^{\mathrm{a}} \cup i'_q), q \neq p$, is selected as the best pattern. □

We note that if the Hamming weight of a generated TEP is one, we need not to store any candidate codewords in the list Γ since we can encode it by adding one row of $\tilde{G}$ to $\tilde{\boldsymbol{c}}_{\emptyset}$. We call the decoding algorithm with these modifications the **proposed decoding algorithm A**.

If $S^{(0)}$ is fixed during the decoding procedure, two rows of $\tilde{P}$ in eq. (10) are also fixed for $q = 1, 2, \ldots, s-1$. In this case, if we store $\tilde{\boldsymbol{q}}_{i_q} = \tilde{\boldsymbol{p}}_{i_q} \oplus \tilde{\boldsymbol{p}}_{i_{q+1}}, q = 1, 2, \ldots, s-1$, we can calculate eq. (10) by at most n binary operations.

On the time and space complexity of the proposed decoding algorithm A, we show the following theorems.

Theorem 1 The time complexity for generating candidate codewords in the proposed decoding algorithm A is at most $2n-k$ binary operations if a heuristic function satisfies conditions (C1) and (C2). □

Theorem 2 The space complexity of the proposed decoding algorithm A is upper bounded by $\frac{n}{k}$ times of that for the reduced-list A* decoding algorithm.

(**Proof**) Consider the worst case that there is one to one correspondence between TEPs in the lists and a candidate codewords stored in Γ. Since it is enough to store $n-k$ parity check symbols of candidate codewords in Γ, the increased complexity from the reduced-list A* decoding algorithm is at most $(n-k) \times M(\tilde{\boldsymbol{r}})$ where the $M(\tilde{\boldsymbol{r}})$ represents the maximum list size of TEPs. Note that the space complexity for the reduced-list A* decoding algorithm is $k \times M(\tilde{\boldsymbol{r}})$ and thus the space complexity for the proposed decoding algorithm A is at most $\frac{n}{k}$ times that for the reduced-list A* decoding algorithm.

□

4.2 Method for Reducing the Number of Real Number Operations

When an extended pattern of the selected best pattern $\boldsymbol{t}(J)$ is inserted into a list at P4), sorting is needed to keep the list in increasing value of $F(\cdot)$ and this complexity may be large[3]. In the following, if a heuristic function $F(\cdot)$ satisfies a certain condition, we show that the time complexity for sorting can be reduced.

Definition (C3) For $J, J' \subseteq [1,k]$, assume that $j_1, j_2 \notin J \cup J'$ and $1 \leq j_1 < j_2 \leq k$. If $j_1, j_2 \in S^{(\alpha)}$ with $\alpha \in \{0,1\}$, then a heuristic function $F(\cdot)$ satisfies

$$\textbf{(C3)} \quad F\big(\boldsymbol{t}(J \cup j_2)\big) \leq F\big(\boldsymbol{t}(J' \cup j_2)\big), \Rightarrow F\big(\boldsymbol{t}(J \cup j_1)\big) \leq F\big(\boldsymbol{t}(J' \cup j_1)\big).$$

We will call this condition the **condition (C3)**. □

Assume that a function $F(\cdot)$ satisfies the condition (C3) as well as (C1) and (C2). When a TEP $\boldsymbol{t}(J \cup i_s)$ is inserted into $M^{(i_s)}$ at P4-c), we still need sorting to keep the list in increasing value of $F(\cdot)$. However, when the best pattern $\boldsymbol{t}(J \cup i_q)$ such that $i_q \in S^{(0)}$ and $i_q \notin J$ is selected, it is enough to store its adjacent pattern $\boldsymbol{t}(J \cup i_{q-1})$ at the end of $M^{(i_{q-1})}$ without sorting at P4-a). It is guaranteed from the condition (C3) that $F(\boldsymbol{t}(J' \cup i_{q-1})) \leq F(\boldsymbol{t}(J \cup i_{q-1}))$ where $\boldsymbol{t}(J' \cup i_{q-1})$ represent any TEPs already stored in the list $M^{(i_{q-1})}$ since $i_{q-1} < i_q$ and TEPs $\boldsymbol{t}(J' \cup i_q)$ and $\boldsymbol{t}(J \cup i_q)$ satisfy

$$F\big(\boldsymbol{t}(J' \cup i_q)\big) \leq F\big(\boldsymbol{t}(J \cup i_q)\big). \tag{11}$$

Note that such $\boldsymbol{t}(J' \cup i_q)$ is selected as the best pattern in a previous iteration.

If the reduced-list A* decoding algorithm employs a function satisfying (C3), the step P4) can be modified as follows where steps P'4-a) and P'4-b) correspond to the modifications:

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 store it at the end of $M^{(i_{q-1})}$.

b) If $\mu(J) = i'_q$ (i.e., $\mu(J) \in S^{(1)}$) and the adjacent pattern $\boldsymbol{t}(J^{\mathrm{a}} \cup i'_{q-1})$ exists, then store it at the end of $M^{(i'_{q-1})}$.

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 $M^{(\mu(J))}$.

For the function $f(\cdot)$, we show the following proposition which is the counterpart to Proposition 1.

Proposition 4 Assume that $S^{(1)}$ be the support of $\boldsymbol{t}_{\mathrm{ref}}$. For a given $\boldsymbol{t}_{\mathrm{ref}}$, the heuristic function $f(\cdot, \boldsymbol{t}_{\mathrm{ref}})$ satisfies the condition (C3).

(Proof) If $j_1, j_2 \notin J$ and $j_1, j_2 \in S^{(\alpha)}$ with $\alpha \in \{0,1\}$, we can easily show

$$f\big(\boldsymbol{t}(J \cup j_2), \tilde{\boldsymbol{c}}_{\mathrm{ref}}\big) - f\big(\boldsymbol{t}(J \cup j_1), \tilde{\boldsymbol{c}}_{\mathrm{ref}}\big) = f\big(\boldsymbol{t}(J' \cup j_2), \tilde{\boldsymbol{c}}_{\mathrm{ref}}\big) - f\big(\boldsymbol{t}(J' \cup j_1), \tilde{\boldsymbol{c}}_{\mathrm{ref}}\big), \tag{12}$$

since $d_H(\boldsymbol{t}(J \cup j_1), \boldsymbol{t}_{\mathrm{ref}}) = d_H(\boldsymbol{t}(J \cup j_2), \boldsymbol{t}_{\mathrm{ref}})$ and $d_H(\boldsymbol{t}(J' \cup j_1), \boldsymbol{t}_{\mathrm{ref}}) = d_H(\boldsymbol{t}(J' \cup j_2), \boldsymbol{t}_{\mathrm{ref}})$. By transposing eq. (12), we have the proposition. □

If the function $f(\cdot)$ is employed by the reduced-list A* decoding algorithm and the proposed decoding algorithm A, P'4) instead of P4) can be used[4]. This saves the time complexity for sorting. Note again that even when the A* decoding algorithm employs heuristic functions satisfying the condition (C3), it is necessary to sort the adjacent pattern of the best pattern selected at S4). This fact indicates the use of the condition (C3) is tightly related to our reduced-list method. We call this decoding algorithm the **proposed decoding algorithm B**.

Theorem 3 The number of real number operations for the proposed decoding algorithm B is no more than that for the original A* decoding algorithm. □

5 Simulation Results

In this section, we evaluate the effectiveness of the proposed decoding algorithm A by computer simulations.

5.1 Conditions of Simulations

Since the time complexity of the proposed decoding algorithm A is theoretically and quantitatively reduced so we here evaluate its space complexity. For the (63, 30, 13) BCH code and the (104, 52, 20) quadratic residue (QR) code, we perform MLD by three algorithms: the A* decoding algorithm [2] (denoted by "A* " in tables), the proposed decoding algorithm A (denoted by "Proposed") and the reduced-list A* decoding algorithm [10] (denoted by "RL A* "). We compare the maximum list size in each algorithm. To fairly evaluate the maximum list size, we define the normalized list size[5] by k as $M(\boldsymbol{r}) + \frac{n-k}{k}\Gamma(\boldsymbol{r})$ where we denote the maximum list size of TEPs by $M(\boldsymbol{r})$ and the maximum size of the list Γ by $\Gamma(\boldsymbol{r})$. At each signal to noise ratio (SNR) E_{b}/S_0 [dB], all decoding algorithms are carried out 10,000 times.

We use the function $f(\cdot)$ as the heuristic function in all decoding algorithms. We set the referenced codeword as $\tilde{\boldsymbol{c}}_{\mathrm{ref}} = \tilde{\boldsymbol{c}}_{\mathrm{best}}$ for the calculation of eq. (3). When a temporally best codeword $\tilde{\boldsymbol{c}}_{\mathrm{best}}$ is obtained, the referenced codeword is updated (for details, see [10]).

5.2 Results and Discussion

We show the results for the (63, 30, 13) BCH code and the (104, 52, 20) QR code in Tables 1 and 2, respectively. In Tables, we denote the average value of the normalized list size by "Ave" and the maximum value of it by "Max" among 10,000 decoding. The values in round brackets

[3] Since the list size at each iteration step is reduced compared to that of the A* decoding algorithm, the time complexity for sorting is also reduced.

[4] We can show that heuristic functions considered in [1, 4, 3, 6] also satisfy the condition (C3).

[5] Note that the normalized list size expresses the maximum number of k dementional vectors stored in memory since $kM(\boldsymbol{r}) + (n-k)\Gamma(\boldsymbol{r})$ is the number of bits in memory.

Table 1: Results of decoding for the (63,30,13) BCH code

E_b/N_0 [dB]		A*	Proposed	RL A*
5.0	Ave	1.19 (1.000)	0.311 (0.262)	0.176 (0.148)
	Max	$2.07 \cdot 10^3$ (1.000)	$8.76 \cdot 10^2$ (0.424)	$4.37 \cdot 10^2$ (0.212)
4.0	Ave	11.4 (1.000)	4.07 (0.359)	2.19 (0.193)
	Max	$9.58 \cdot 10^3$ (1.000)	$5.86 \cdot 10^3$ (0.612)	$3.27 \cdot 10^3$ (0.342)
3.0	Ave	96.6 (1.000)	46.4 (0.481)	24.4 (0.253)
	Max	$1.43 \cdot 10^4$ (1.000)	$9.34 \cdot 10^3$ (0.653)	$5.10 \cdot 10^3$ (0.356)
2.0	Ave	$5.37 \cdot 10^2$ (1.000)	$2.96 \cdot 10^2$ (0.552)	$1.56 \cdot 10^2$ (0.291)
	Max	$7.05 \cdot 10^4$ (1.000)	$5.09 \cdot 10^4$ (0.722)	$2.80 \cdot 10^4$ (0.397)

Table 2: Results of decoding for the $(104, 52, 20)$ QR code

E_b/N_0 [dB]		A*	Proposed	RL A*
6.5	Ave	$6.50 \cdot 10^{-2}$ (1.000)	$9.70 \cdot 10^{-3}$ (0.149)	$6.20 \cdot 10^{-3}$ (0.095)
	Max	87 (1.000)	26 (0.299)	17 (0.195)
5.5	Ave	0.995 (1.000)	0.184 (0.185)	0.116 (0.116)
	Max	$1.23 \cdot 10^3$ (1.000)	$1.49 \cdot 10^2$ (0.122)	93 (0.076)
4.5	Ave	21.1 (1.000)	6.15 (0.292)	3.27 (0.155)
	Max	$5.56 \cdot 10^4$ (1.000)	$2.07 \cdot 10^4$ (0.373)	$1.05 \cdot 10^4$ (0.188)
3.5	Ave	$1.34 \cdot 10^3$ (1.000)	$6.12 \cdot 10^2$ (0.455)	$3.37 \cdot 10^2$ (0.251)
	Max	$3.67 \cdot 10^6$ (1.000)	$2.26 \cdot 10^6$ (0.616)	$1.36 \cdot 10^6$ (0.371)

indicate the ratio to the maximum list size of the A* decoding algorithm.

By Table 1, the average value of maximum list size in the proposed decoding algorithm is about 1/4 of that in the A* decoding algorithm at 5.0 [dB]. Although the ratio to the A* decoding algorithm becomes high as the SNR decreases, all values of the proposed decoding algorithm are less than those of the A* decoding algorithm. These results show that there is no increase of space complexity in the proposed decoding algorithm although the time complexity is drastically reduced. By Table 2, average values and maximum values in the proposed decoding algorithm are much less than those in the A* decoding algorithm. Theses results indicate the proposed method also works well for a longer code.

6 Conclusion and Future Works

In this paper, we propose a new method reducing the time complexity for generating candidate codewords in the A* decoding algorithm. We also devise a method for reducing the number of real number operations in the A* decoding algorithm. Although the proposed decoding algorithm A requires some additional space complexity for storing some already generated candidate codewords, the complexity is upper bounded by $\frac{n}{k}$ times that for the reduced-list A* decoding algorithm [10].

As future works, evaluations employing other heuristic functions such as the function in [4] are needed. We also need to develop a method for heuristic search MLD algorithm with powerful heuristic functions such as in [5].

Acknowledgment

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 valuable comments.

References

[1] G. Battail and J. Fang, "Décodage pondéré optimal descodes linéaires en blocs," *Annales des télé-communications*, vol.41, nos.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. Inform. 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. Inform. 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. Inform. 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] H. Yagi, T. Matsushima, and S. Hirasawa, "A method for reducing space complexity of reliability-based heuristic search maximum likelihood decoding algorithms", *Proc. of the 26th Symposium on Inform. Theory and its Applications (SITA2003)*, pp.185–188, Hyogo, Japan, Dec. 2003.

[9] 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.

[10] H. Yagi, T. Matsushima, and S. Hirasawa, "A heuristic search algorithm with the reduced list of test error patterns for maximum likelihood decoding algorithms", *Proc. of the 27th Symposium on Inform. Theory and its Applications (SITA2004)*, pp.571–574, Gifu, Japan, Dec. 2004.