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

信学技報
TECHNICAL REPORT OF IEICE.

2元(128,50)拡大原始BCH符号と(128,64)リード・マラー符号の局所重み分布

安永　憲司†　　藤原　　融†

† 大阪大学 大学院情報科学研究科 マルチメディア工学専攻
〒 565-0871 吹田市山田丘 1–5
E-mail: †{k-yasunaga,fujiwara}@ist.osaka-u.ac.jp

あらまし　2元 $(128, 50)$ 拡大原始 BCH 符号と $(128, 64)$ リード・マラー符号の局所重み分布の計算を行った．局所重み分布の記号位置置換に対する不変性を利用することで計算量を削減し，これまでに 2元 $(128, k)$ 拡大原始 BCH 符号 $k \leqq 43$ の局所重み分布を求めていた．今回さらに符号のトレリス構造を利用することで計算量を削減し，2元 $(128, 50)$ 拡大原始 BCH 符号の局所重み分布を求めた．また，$(128, 64)$ リード・マラー符号の局所重み分布も求めた．

キーワード　局所重み分布，拡大原始 BCH 符号，リード・マラー符号，トレリス構造

The local weight distributions of the (128,50) extended binary primitive BCH code and the (128,64) Reed-Muller code

Kenji YASUNAGA† and Toru FUJIWARA†

† Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka, Suita-shi, 565–0871 Japan
E-mail: †{k-yasunaga,fujiwara}@ist.osaka-u.ac.jp

Abstract　The local weight distribution of a block code gives some aspects of its error performance. We obtained the local weight distributions of the $(128, k)$ extended binary primitive BCH codes for $k \leqq 43$, using the invariance property to reduce the computational complexity. In this paper, the trellis structure of a code is used to reduce the complexity more, and the local weight distributions of the $(128, 50)$ extended binary primitive BCH code and the $(128, 64)$ Reed-Muller code are computed.

Key words　local weight distribution, extended primitive BCH code, Reed-Muller code, trellis structure

1. Introduction

For a binary linear code, the local weight distribution [1] (or local distance profile [2], [3]) is defined as the weight distribution of the zero neighbors in the code, where a zero neighbor is a codeword whose Voronoi region shares a facet with that of the all-zero codeword [2]. Knowing the local weight distribution of a code is useful for the error performance analysis of the code. For example, the local weight distribution gives a tighter upper bound on error probability for soft decision decoding over an AWGN (additive white gaussian noise) channel than the usual union bound.

For binary Hamming codes, the formula for the local weight distribution is known [4]. For binary second order Reed-Muller codes, the relation between the local weight distribution and the weight distribution is known [4]. For other codes, theoretically, we can compute the local weight distribution by examining every codeword whether it is a zero neighbor or not. However, it is infeasible to compute the distribution except for the codes with small dimensions.

Recently, an algorithm for computing the local weight distribution of binary cyclic codes has been proposed by Mohri, Honda, and Morii [3], and the local weight distribution of the binary $(63, k)$ primitive BCH codes with $k \leqq 45$ are computed. We call the algorithm the MHM algorithm in this paper. The MHM algorithm reduces the computational complexity by using the following invariance property [2]: Any cyclic permutation of a codeword is a zero neighbor if and only if the codeword is a zero neighbor. It generates the representative codeword and the number of the equivalent

codewords, which are able to be cyclic-permuted into the representative one, and checks whether each of the representatives is a zero neighbor or not. The key subalgorithm in the algorithm is the generation of the representative codewords.

The MHM algorithm can also be applied straightforwardly to compute the local weight distribution of extended cyclic codes (see Corollary 2). Extended primitive BCH codes are closed under the affine group of permutations, which is larger than the cyclic group of permutations. It is known that the invariance property for the cyclic group of permutations can be generalized to that for any group of permutations [2]. Using the invariance property for the larger group of permutations, we may reduce the number of the representative codewords. However, it is not easy to obtain the representative codewords and the number of the equivalent codewords.

In this paper, we propose an algorithm for computing the local weight distribution of binary linear codes which are closed under any group of permutations. To use the invariance property, we will apply the invariance property to the set of **cosets** of a subcode rather than the set of codewords. The representative **cosets** and the number of equivalent **cosets** are computed in the proposed algorithm. This reduces the complexity of finding the representatives, which is much smaller than that for checking whether every representative is a zero neighbor or not. This idea is used in [5] for computing the weight distributions of extended binary primitive BCH codes. In this paper, we show that the idea can also be applied for the local weight distribution. To reduce the complexity more, we use the trellis structure of the code, in checking whether a codeword is a zero neighbor or not.

We apply the proposed algorithm to extended binary primitive BCH codes, which are closed under the affine permutations, and obtain the local weight distributions of $(128, k)$ extended binary primitive BCH code for $k \leqq 50$ (the distributions of $k \leqq 43$ were computed and presented in [6]). The complexity of the proposed algorithm is about 1/64 as much as that of the MHM algorithm for the codes of length 128 (refer to Table 2). Furthermore, we obtain the local weight distributions of the third-order Reed-Muller code of length 128.

2. Local Weight Distribution

Let C be a binary (n, k) linear code. Define a mapping s from $\{0, 1\}$ to $\mathbf{R}$ as $s(0) = -1$ and $s(1) = 1$. The mapping s is naturally extended to one from $\{0, 1\}^n$ to $\mathbf{R}^n$. Zero neighbor is defined as follows:

Definition 1. [Zero neighbor] For $\boldsymbol{v} \in C$, define $\boldsymbol{m}_0 \in \mathbf{R}^n$ as $\boldsymbol{m}_0 = \frac{1}{2}(s(\mathbf{0}) + s(\boldsymbol{v}))$, where $\mathbf{0} = (0, 0, \ldots, 0)$. The codeword $\boldsymbol{v}$ is a zero neighbor if and only if

$$d_E(\boldsymbol{m}_0, s(\boldsymbol{v})) = d_E(\boldsymbol{m}_0, s(\mathbf{0})) < d_E(\boldsymbol{m}_0, s(\boldsymbol{v}')), \quad \text{for any } \boldsymbol{v}' \in C \setminus \{\mathbf{0}, \boldsymbol{v}\}, \tag{1}$$

where $d_E(\boldsymbol{x}, \boldsymbol{y})$ is the squared Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbf{R}^n$.

The following lemma is useful to check whether a given codeword is a zero neighbor or not.

Lemma 1. [2] $\boldsymbol{v} \in C$ is a zero neighbor if and only if there does not exist $\boldsymbol{v}' \in C \setminus \{\mathbf{0}\}$ such that $\mathrm{Supp}(\boldsymbol{v}') \subsetneq \mathrm{Supp}(\boldsymbol{v})$. Note that $\mathrm{Supp}(\boldsymbol{v})$ is the set of support of $\boldsymbol{v}$, which is the set of positions of nonzero elements in $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$, that is,

$$\mathrm{Supp}(\boldsymbol{v}) = \{i : v_i \neq 0,\ 1 \leqq i \leqq n\}. \tag{2}$$

The local weight distribution is defined as follows:

Definition 2. [Local weight distribution] Let L_w be the number of zero neighbors with weight w in C. The local weight distribution of C is defined as the $(n + 1)$-tuple $(L_0, L_1, \ldots, L_n)$.

On the local weight distribution, the following lemma holds.

Lemma 2. [4] Let A_w be the number of codewords with weight w in C, and d be the minimum distance of C.

$$L_w = \begin{cases} A_w, & w < 2d, \\ 0, & w > n - k + 1. \end{cases} \tag{3}$$

To obtain the local weight distribution, if the weight distribution is known, only L_w with $2d \leqq w \leqq n - k + 1$ are need to be obtained.

These lemmas are useful to compute the local weight distribution of a code. The MHM algorithm proposed in [3] uses the following invariance property under cyclic permutations, as well as the lemmas.

Theorem 1. [3] Let C be a binary cyclic code. Any cyclic permuted codeword of $\boldsymbol{v}$ is a zero neighbor if a codeword $\boldsymbol{v} \in C$ is a zero neighbor.

Corollary 1. Let C be a binary cyclic code, and $\sigma^i \boldsymbol{v}$ be an i times cyclic permuted codeword of $\boldsymbol{v} \in C$. Consider a set $S = \{\boldsymbol{v}, \sigma\boldsymbol{v}, \sigma^2\boldsymbol{v}, \ldots, \sigma^{p(\sigma,\boldsymbol{v})-1}\boldsymbol{v}\}$, where $p(\sigma, \boldsymbol{v})$ is the period of σ, which is the minimum i such that $\sigma^i \boldsymbol{v} = \boldsymbol{v}$. Then, (1) if $\boldsymbol{v}$ is a zero neighbor, all codewords in the set S are zero neighbors; and (2) otherwise, all codewords in S are not zero neighbors.

In the MHM algorithm, all codewords are partitioned into the sets described in Corollary 1. Only one representative codeword in each sets is generated and checked whether it is

a zero neighbor or not.

We can easily modify the MHM algorithm to compute the local weight distribution of the extended cyclic code using the following corollary:

Corollary 2. Let C and $C_{\rm ex}$ be a binary cyclic code and its extended code, respectively. For $\boldsymbol{v} \in C$, let $\mathrm{ex}(\boldsymbol{v})$ be the corresponding codeword in $C_{\rm ex}$. Then, for any cyclic permuted codeword $\boldsymbol{v}'$ of $\boldsymbol{v} \in C$, $\mathrm{ex}(\boldsymbol{v}')$ is a zero neighbor in $C_{\rm ex}$ if $\mathrm{ex}(\boldsymbol{v})$ is a zero neighbor in $C_{\rm ex}$.

As mentioned in Section 1, the extended primitive BCH codes are closed larger class of permutations, and Theorem 1 can be generalized to any group of permutations of the automorphism group of the code [2]. In the following section, we review the invariance property under a group of permutation, and present a coset partitioning technique to use the property effectively.

3. Coset Partitioning for Computing the Local Weight Distribution of Codes Closed under a Group of Permutations

3.1 An invariance property under permutations

For a permutation π and a set of vectors D, the set of the permuted vectors $\pi[D]$ is defined as

$$\pi[D] = \{\pi\boldsymbol{v} : \boldsymbol{v} \in D\}. \tag{4}$$

The automorphism group of a code C is the set of all permutations by which C is permuted into C, and denoted by $\mathrm{Aut}(C)$, i.e.,

$$\mathrm{Aut}(C) = \{\pi : \pi[C] = C\}. \tag{5}$$

An invariance property under the automorphism group is given in the following theorem.

Theorem 2. For $\pi \in \mathrm{Aut}(C)$ and $\boldsymbol{v} \in C$, $\pi\boldsymbol{v}$ is a zero neighbor if $\boldsymbol{v}$ is a zero neighbor.

(Proof) Suppose that $\boldsymbol{v}$ is a zero neighbor and $\pi\boldsymbol{v}$ is not a zero neighbor. There exists a nonzero codeword $\boldsymbol{v}' \in C$ such that $\mathrm{Supp}(\pi\boldsymbol{v}) \supsetneq \mathrm{Supp}(\boldsymbol{v}')$ from Lemma 1. Since $\mathrm{Aut}(C)$ is a group, there exists $\boldsymbol{v}'' \in C$ such that $\boldsymbol{v}' = \pi\boldsymbol{v}''$. Thus $\mathrm{Supp}(\pi\boldsymbol{v}) \supsetneq \mathrm{Supp}(\pi\boldsymbol{v}'')$, and $\mathrm{Supp}(\boldsymbol{v}) \supsetneq \mathrm{Supp}(\boldsymbol{v}'')$. This contradicts being the zero neighbor of $\boldsymbol{v}$ from Lemma 1. □

This theorem extends Corollary 1 as follows:

Corollary 3. For $\boldsymbol{v} \in C$, consider a set $S = \{\pi\boldsymbol{v} : \forall\pi \in \mathrm{Aut}(C)\}$. Then, (1) if $\boldsymbol{v}$ is a zero neighbor, all codewords in S are zero neighbors; and (2) otherwise, all codewords in S are not zero neighbors.

It is not easy to devise a similar algorithm as the MHM algorithm, since for almost all group of permutations, no efficient way is known for generating the representative codewords and obtaining the number of the equivalent codewords. To use this invariance property, we apply the invariance property to the set of cosets of a subcode rather than the set of codewords.

3.2 Local weight subdistribution for a coset

For a binary (n, k) linear code C and its linear subcode with dimension k', let C/C' denote the set of cosets of C' in C, that is,

$$C/C' = \{\boldsymbol{v} + C' : \boldsymbol{v} \in C\}. \tag{6}$$

Then,

$$|C/C'| = 2^{k-k'}, \quad \text{and} \quad C = \bigcup_{D \in C/C'} D. \tag{7}$$

Definition 3. [Local weight subdistribution for a coset] The local weight subdistribution for a coset $D \in C/C'$ is the weight distribution of zero neighbors of C in D. The local weight subdistribution for D is $(|LS_0(D)|, |LS_1(D)|, \ldots, |LS_n(D)|)$, where

$$\begin{aligned} LS_w(D) = \{&\boldsymbol{v} \in D : \mathrm{Supp}(\boldsymbol{v}') \nsubseteq \mathrm{Supp}(\boldsymbol{v}) \text{ for any} \\ &\boldsymbol{v}' \in C \setminus \{\boldsymbol{0}, \boldsymbol{v}\}, \text{and the Hamming} \\ &\text{weight of } \boldsymbol{v} \text{ is } w\}, \end{aligned} \tag{8}$$

with $0 \leqq w \leqq n$.

Then, from (7), the local weight distribution of C is given as the sum of the local weight subdistributions for the cosets in C/C', that is,

$$L_w = \sum_{D \in C/C'} |LS_w(D)|. \tag{9}$$

The following theorem gives an invariance property under permutations for cosets.

Theorem 3. For $D_1, D_2 \in C/C'$, the local weight subdistribution for D_1 and that for D_2 are the same if there exists a permutation $\pi \in \mathrm{Aut}(C)$ such that $\pi[D_1] = D_2$.

(Proof) For any codewords $\boldsymbol{v} \in D_1$, $\pi\boldsymbol{v} \in D_2$, from Theorem 2, $\pi\boldsymbol{v}$ is a zero neighbor if and only if $\boldsymbol{v}$ is a zero neighbor. Therefore the local weight subdistribution for D_1 and that for D_2 are the same. □

3.3 Partitioning the set of cosets with the same local weight subdistribution

We give a condition for cosets having the same local weight subdistribution. The following lemma gives the set of all permutations by which every coset in C/C' is permuted into one in C/C'.

Lemma 3. For a linear code C and its linear subcode C', the set $\{\pi : \pi[D] \in C/C' \text{ for any } D \in C/C'\}$ is equal to $\mathrm{Aut}(C) \cap \mathrm{Aut}(C')$.

$\mathrm{Aut}(C) \cap \mathrm{Aut}(C')$ (or even $\mathrm{Aut}(C)$) is generally not known. Only subgroups of $\mathrm{Aut}(C) \cap \mathrm{Aut}(C')$ are known. Therefore, we use a subgroup.

Definition 4. Let Π be a subset of $\mathrm{Aut}(C) \cap \mathrm{Aut}(C')$. For $D_1, D_2 \in C/C'$, we denote $D_1 \sim_\Pi D_2$ if and only if there exists $\pi \in \Pi$ such that $\pi[D_1] = D_2$.

Lemma 4. The relation "$\sim_\Pi$" is an equivalence relation on C/C' if Π forms a group.

When the set of cosets are partitioned into the equivalence classes by the relation "$\sim_\Pi$", the local weight subdistributions for cosets which belong to the same equivalence class are the same.

We give a useful theorem for partitioning the set of cosets into equivalence classes by the relation "$\sim_\Pi$."

Theorem 4. Let Π be a subset of $\mathrm{Aut}(C) \cap \mathrm{Aut}(C')$. For $D_1, D_2 \in C/C'$ and $\pi \in \Pi$, we have $D_1 \sim_\Pi D_2$ if $\pi \boldsymbol{v}_1 \in D_2$ for any $\boldsymbol{v}_1 \in D_1$.

(Proof) Let $\pi \boldsymbol{v}_1 = \boldsymbol{v}_2 \in D_2$. Any codeword in D_1 is represented by $\boldsymbol{v}_1 + \boldsymbol{v}$ ($\boldsymbol{v} \in C'$). Then,

$$\begin{aligned} \pi(\boldsymbol{v}_1 + \boldsymbol{v}) &= \pi \boldsymbol{v}_1 + \pi \boldsymbol{v} \\ &= \boldsymbol{v}_2 + \pi \boldsymbol{v}. \end{aligned} \tag{10}$$

Since $\pi \in \mathrm{Aut}(C')$, $\pi \boldsymbol{v}$ is in C'. Thus $\pi[D_1] = D_2$. □

From Theorem 4, to partition the set of cosets into equivalence classes, we only need to check whether the representative codeword of a coset is permuted into another coset. After partitioning into equivalence classes, the local weight subdistribution for only one coset in each equivalence class needs to be computed. Thereby the computational complexity is reduced.

4. An Algorithm for Computing the Local Weight Distribution

4.1 Outline of the algorithm

Based on the method of partitioning the set of cosets described in the previous section, we propose an algorithm to compute the local weight distribution as follows:

(1) Choose a subcode C' and a subgroup Π of permutations of $\mathrm{Aut}(C) \cap \mathrm{Aut}(C')$.

(2) Partition C/C' into equivalence classes with permutations in Π, and obtain the number of codewords in each equivalence class.

(3) Compute the local weight subdistributions for the representative cosets in each equivalence class.

(4) Sum up all the local weight subdistributions for the cosets.

4.2 Computational complexity

Here, we analyze computational complexity of the proposed algorithm for the binary (n, k) linear code C, assuming that the subcode C' with dimension k' is given. Also, to check whether a codeword is a zero neighbor or not, the procedure presented in [2] is used as in the MHM algorithm. The procedure uses Lemma 1, and its computational complexity to check one codeword is $O(n^2 k)$.

Since the number of codewords in each cosets is $2^{k'}$, the total number of codewords to be checked by the procedure is $e2^k$, where e is the number of the equivalence classes. The computational complexity to check one codeword is $O(n^2 k)$. Hence, the time complexity of Step (3) of the proposed algorithm is $O(n^2 k \cdot e 2^{k'})$. The time complexity of Step (2), partitioning into equivalence classes, is $O(n(k-k')2^{k-k'}|\Pi|)$ [6].

Therefore, The time complexity of the entire algorithm is $O(n^2 k \cdot e 2^{k'} + n(k-k')2^{k-k'}|\Pi|)$. When k' is chosen as $k' > k/2$, then $2^{k'} > 2^{k-k'}$, and the complexity of partitioning into equivalence classes is much smaller than of computing the local weight subdistributions for cosets.

The space complexity of the entire algorithm is $O((k-k')2^{k-k'})$ [6], since the space complexity for computing the local weight subdistributions is much smaller than that for partitioning into equivalence cosets.

4.3 Selection of the subcode

We should choose the subcode for which the number of permutations in Π is large.

If there are several such subcode with the same Π, then the subcode with the smaller dimension should be chosen to minimize the number of codewords that need to be checked, as long as the complexity of partitioning into equivalence classes is relatively small.

4.4 A method of checking for zero neighbor using the trellis structure

We consider reducing the complexity of checking whether a codeword is a zero neighbor or not.

For $\boldsymbol{v} \in C$, let $C(\boldsymbol{v}) = \{\boldsymbol{u} \,|\, \boldsymbol{u} \in C \setminus \{\boldsymbol{0}\}, \mathrm{Supp}(\boldsymbol{u}) \subsetneq \mathrm{Supp}(\boldsymbol{v})\}$. Checking whether a codeword $\boldsymbol{v}$ is a zero neighbor or not is examining whether the dimension of $C(\boldsymbol{v})$, denoted $\dim(C(\boldsymbol{v}))$, is zero or not. For $\boldsymbol{v} \in C$ and i with $1 \leqq i \leqq n$, let $\mathrm{S}(\boldsymbol{v}, i) = \{(u_1, u_2, \ldots, u_n) \in C \,|\, u_j = v_j \text{ with } 1 \leqq j \leqq i\}$ and $\mathrm{C}(\boldsymbol{v}, i) = \{\boldsymbol{u} \,|\, \boldsymbol{u} \in \mathrm{S}(\boldsymbol{v}, i), \mathrm{Supp}(\boldsymbol{u}) \cap \{1, \ldots, i\} \subsetneq \mathrm{Supp}(\boldsymbol{v}) \cap \{1, \ldots, i\}\}$. A typical implimentation of the procedure in [2] to check whether $\boldsymbol{v}$ is a zero neighbor or not computes $\mathrm{C}(\boldsymbol{v}, i)$ for $i = 1, 2, \ldots, n$, where $\mathrm{C}(\boldsymbol{v}, n) = \mathrm{C}(\boldsymbol{v})$. For any $\boldsymbol{u}, \boldsymbol{u}' \in \mathrm{S}(\boldsymbol{v}, i)$, $\mathrm{C}(\boldsymbol{u}, i) = \mathrm{C}(\boldsymbol{u}', i)$. That is, if we generate $\mathrm{C}(\boldsymbol{u}, i)$ once, we does not need to generate $\mathrm{C}(\boldsymbol{u}, i)$ for each $\boldsymbol{u} \in \mathrm{S}(\boldsymbol{v}, i)$ later. By this, the computational complex-

ity is reduced. We should choose i properly, say $n/2$ or $n/4$, in order to make $\mathrm{S}(\boldsymbol{v}, i)$ large and $\mathrm{C}(\boldsymbol{u}, i)(\boldsymbol{u} \in \mathrm{S}(\boldsymbol{v}, i))$ small. To obtain $\mathrm{S}(\boldsymbol{v}, i)$, we use the trellis structure of cosets. This method does not make the space complexity much larger, since a codeword in $\mathrm{S}(\boldsymbol{v}, i)$ is generated successively. The advantage of this method depends on the code. For extended binary primitive BCH codes, permuting the symbol positions of codewords properly make $\mathrm{S}(\boldsymbol{v}, i)$ larger [8].

It is not easy to estimate precisely how the computational complexity is reduced. We will estimate the effect roughly in the case of the $(128, 50)$ extended BCH code. In this case, the $(128, 29)$ code is chosen as the subcode and the number of representative cosets is 258. We pick up $2^{15} \times 258$ codewords by choosing 2^{15} codewords randomly from each of the 258 representative coset. For every codeword $\boldsymbol{v}$ in such codewords, we examined a position at which the $\dim(\mathrm{C}(\boldsymbol{v}))$ is found out to be zero or not. Then, the average was 100. Assuming that the complexity of obtaining $\mathrm{C}(\boldsymbol{v}, i+1)$ from $\mathrm{C}(\boldsymbol{v}, i)$ is proportional to the dimension of $\mathrm{C}(\boldsymbol{v}, i)$, we estimated the relative computational complexity to that without using the above techniquie when i_0 is chosen as i $(0 \leqq i \leqq n)$: $((100 - i_0)/100)^2 + (1 - ((100 - i_0)/100)^2)/2^{k_s}$, where k_s is the dimension of $\mathrm{S}(\boldsymbol{v}, i_0)$. It turned out that the complexity would reduced mostly by $1/2$ for $i = 32$ and 47. Therefore, we choose 32 as i for the $(128, 50)$ code. Actually, for the $(128, 50)$ extended BCH code and the $(128, 29)$ extended BCH subcode, the complexity is reduced by about $1/2$.

In the proposed algorithm, the codewords that need to be checked are generated by the cosets, so a part of $\mathrm{S}(\boldsymbol{v}, i)$ is generated in each cosets. If the dimension of the subcode is small, k_s may become small and the effect of using the trellis structure is small. We should choose the subcode considering the effect of using the trellis structure.

5. Computing the Local Weight Distribution of Extended Primitive BCH Codes

Extended primitive BCH codes are closed under the affine permutations. To choose Π as large as possible, we choose an extended primitive BCH code with the smaller dimension as the subcode C'.

We apply the proposed algorithm to $(128, k)$ extended primitive BCH codes. We obtained the local weight distribution of the codes for $k \leqq 50$. The local weight distribution with $k = 50$, which was not obtained, is shown in Table 1. It takes about 440 hours (CPU time) to compute it with 1.6 GHz Opteron processor. Note that for the $(128, 50)$ code, its $(128, 29)$ subcode is chosen as C'. In this case, it took only one minute to partition the cosets into equivalence classes.

We compare the MHM algorithm and the proposed algorithm for extended primitive BCH codes with length 128 in Table 2. In the table, CT is the observed average time to check a codeword for zero neighbor without using the trellis structure. This is estimated by observing the CPU time for 2^{20} codewords. N_{MHM} and N_{new} are the number of codewords that need to be checked in the MHM algorithm and the proposed algorithm, respectively. PT is the time to partition the cosets into the equivalence classes. T_{MHM} (or T_{new}) are the estimated total time to obtain the local weight distributions from CT, PT, and N_{MHM} (or N_{new}). That is, $T_{\mathrm{MHM}} = CT \times N_{\mathrm{MHM}}$ and $T_{\mathrm{new}} = PT + CT \times N_{\mathrm{MHM}}$. k' is the dimension of the chosen subcode, such that PT is smaller than T_{new} and N_{new} is enough small. The table shows that the time complexity for partitioning the cosets into the equivalence classes is actually much smaller than that for checking each codewords for zero neighbor.

In the MHM algorithm, the computational complexity (or the number of codewords that need to be checked) for the extended BCH codes of length n is as much as that for BCH codes, which is about $1/n$ as much as that of the brute force method. The number of the affine permutations in (n, k) extended primitive BCH codes is $n(n-1)$. When the proposed algorithm is applied to the codes, the complexity is at most about $1/n^2$ as much as that of the brute force method. In fact, as shown in Table 2, the complexity compared with the MHM algorithm is about $1/64$ in the case of $n = 128$.

Table 1 The local weight distributions of the $(128, 50)$ extended primitive BCH code.

w	L_w
28	186,944
32	19,412,204
36	113,839,296
40	33,723,852,288
44	579,267,441,920
48	5,744,521,082,944
52	33,558,415,333,632
56	117,224,645,074,752
60	247,311,270,037,888
64	316,973,812,770,944
68	247,074,613,401,728
72	115,408,474,548,096
76	25,844,517,328,896

6. Computing the Local Weight Distribution of Reed-Muller Codes

We also apply the proposed algorithm to the third-order Reed-Muller code of length 128. Reed-Muller codes are closed under the general linear group [9]. We use the second-order Reed-Muller code as the subcode. The representative codewords of cosets are presented in [10]. A method to obtain the number of equivalence cosets to the representative

Table 2 Comparison of the prospective time complexity between the MHM algorithm and the proposed algorithm for extended primitive BCH codes with length 128.

k	CT (sec)	$N_{\rm MHM}$	$T_{\rm MHM}$ (hour) $= CT \times N_{\rm MHM}$	k'	PT (sec)	$N_{\rm new}$	$T_{\rm new}$ (hour)* $= PT + CT \times N_{\rm new}$	$T_{\rm new}/T_{\rm MHM}$
36	1.44×10^{-5}	5.41×10^{8}	2.16×10^{0}	15	86	1.26×10^{7}	7.46×10^{-2}	1/29
43	2.02×10^{-5}	6.93×10^{10}	3.89×10^{2}	22	65	1.08×10^{9}	6.10×10^{0}	1/64
50	2.40×10^{-5}	8.87×10^{12}	5.92×10^{4}	29	66	1.39×10^{11}	9.25×10^{2}	1/64
57	3.02×10^{-5}	1.13×10^{15}	9.53×10^{6}	29	11138	1.77×10^{13}	1.48×10^{5}	1/64
64	3.31×10^{-5}	1.45×10^{17}	1.34×10^{9}	43	65	2.27×10^{15}	2.09×10^{7}	1/64

* This estimate is for computing time without using the technique in Section 4. 4

Table 3 The number of equivalence cosets ($\nu_i(3,7)$ in [7]).

i	$\nu_i(3,7)$
1	1
2	11,811
3	2,314,956
4	45,354,240
5	59,527,440
6	21,165,312
7	1,763,776
8	2,222,357,760
9	238,109,760
10	17,778,862,080
11	444,471,552
12	13,545,799,680

Table 4 The local weight distribution of third-order Reed-Muller code of length 128.

w	L_w
16	94,488
24	74,078,592
28	3,128,434,688
32	311,574,557,952
36	18,125,860,315,136
40	5,519,65,599,940,608
44	948,28,18,340,782,080
48	93,680,095,610,142,720
52	538,097,941,223,571,456
56	1,752,914,038,641,131,520
60	2,787,780,190,808,309,760
64	517,329,044,342,046,720

cosets are presented in [7]. Thus, the process of obtaining the representative cosets and the number of equivalence cosets are different from that for extended primitive BCH codes. Note that the computing time for this process is vanishingly small. Using it, we compute the local weight distribution of the third-order Reed-Muller code of length 128. It took about thirteen hours. The number of equivalent cosets and the local weight distribution is shown in Tables 3 and 4, respectively.

7. Conclusion

In this paper, we have proposed an algorithm for computing the local weight distribution of binary linear codes which are closed under a group of permutations. The algorithm uses an invariance property under the automorphism group. This property is applied to the set of cosets of a subcode. We apply the proposed algorithm to $(128, k)$ extended primitive BCH codes for $k \leqq 50$ and the third-order Reed-Muller code of length 128.

References

[1] E. Agrell, "On the Voronoi Neighbor Ratio for Binary Linear Block Codes," *IEEE Trans. Inform. Theory*, vol.44, no.7, pp.3064–3072, Nov. 1998.

[2] E. Agrell, "Voronoi regions for binary linear block codes," *IEEE Trans. Inform. Theory*, vol.42, no.1, pp.310–316, Jan. 1996.

[3] M. Mohri, Y. Honda, and M. Morii,"A method for computing the local weight distribution of binary cyclic codes," *IEICE Trans. Fundamentals (Japanese Edition)*, vol.J86-A, no.1, pp.60–74, Jan. 2003.

[4] A. Ashikhmin and A. Barg, "Minimal vectors in linear codes," *IEEE Trans. Inform. Theory*, vol.44, no.5, pp.2010–2017, Sept. 1998.

[5] T. Fujiwara and T. Kasami, "The weight distribution of $(256, k)$ extended binary primitive BCH codes with $k \leqq 63$ and $k \geqq 207$," *IEICE Technical Report*, IT97-46, Sept. 1997.

[6] K. Yasunaga and T.Fujiwara, "An algorithm for computing the local distance profile of binary linear codes closed under a group of permutations," *IEICE Technical Report*, IT2003-47, Sept. 2003.

[7] T. Sugita, T. Kasami, and T. Fujiwara, "The weight distribution of the third-order Reed-Muller codes of length 512," *IEEE Trans. Inform. Theory*, vol.42, no.5, pp.1622–1625, Sept. 1996.

[8] T. Kasami, T. Tanaka, T. Fujiwara, and S. Lin, "On complexity of trellis structure of linear block codes," *IEEE Trans. Inform. Theory*, vol.39, no.3, pp.1057–1064, May. 1993.

[9] F.J. MacWilliams and N.J.A. Sloane, *The theory of error-correcting codes*, North-Holland, 1977.

[10] X. Hou, "GL(m,2) acting on R(r,m)/R(r-1,m)," *Discr. Math.*, 149, pp.99–122, 1996.