Title	Some Results on the Local Weight Distribution of Binary Linear Codes (Algebraic Aspects of Coding Theory and Cryptography)
Author(s)	Fujiwara, Toru; Yasunaga, Kenji
Citation	数理解析研究所講究録 (2005), 1420: 150-162
Issue Date	2005-04
URL	http://hdl.handle.net/2433/47172
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録 1420 巻 2005 年 150-162

Some Results on the Local Weight Distribution of Binary Linear Codes

大阪大学・大学院情報科学研究科 藤原 融 (Toru Fujiwara)
Graduate School of Information Science and Technology
Osaka University
大阪大学・大学院情報科学研究科 安永 憲司 (Kenji Yasunaga)
Graduate School of Information Science and Technology
Osaka University

1 Introduction

In a binary linear code, a zero neighbor is a codeword whose Voronoi region shares a facet with that of the all-zero codeword [2]. The local weight distribution [1, 12] (or local distance profile [2, 3, 4, 5, 11]) of a binary linear code is defined as the weight distribution of zero neighbors in the code. Knowledge of the local weight distribution of a code is valuable 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 AWGN channel than the usual union bound [5].

Formulas for local weight distribution are only known for certain classes of codes, Hamming codes and second-order Reed-Muller codes. Although an efficient method to examine zero neighborship of codeword is presented in [2], the computation for obtaining the local weight distribution is a very time-consuming task. As Agrell noted in [2], the automorphism group of the code can help reduce the complexity. Using the group of cyclic permutations, Mohri et al. devised a computation algorithm for cyclic codes. We call the algorithm the MHM algorithm in this resume. By using the algorithm, they obtained the local weight distributions of the $(63, k)$ binary primitive BCH codes with $k \leq 45$ [3, 4]. They also obtained the distirubutions for $k = 51, 57$ by their another algorithm [3]. 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. The invariance property for the cyclic 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 resume, we give some of our results on the local weight distributions, presented in [11, 12, 13, 14]. Frist, we give 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 the zero neighborship for every representative codewords. This idea was introduced in [7] for computing the weight distributions of extended binary primitive BCH codes. 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 the zero neighborship.

We apply the proposed algorithm to extended binary primitive BCH codes and obtain the local weight distributions of the $(128, k)$ extended binary primitive BCH code for $k \leq 50$ [11],[12]. The complexity of the proposed algorithm is about $1/64$ as much as that of the MHM algorithm for the codes of length 128. Furthermore, we obtain the local weight distributions of the third-order Reed-Muller code of length 128.

However, for cyclic codes, the complexity is not reduced. Then, the local weight distributions of the $(127, k)$ primitive BCH codes for $k \geq 36$ were not obtained although those of the corresponding $(128, k)$ extended primitive BCH codes are obtaiend for $k \leq 50$. A method for obtaining the local weight distribution of a code from that of its extended code should be considered.

For this purpose, relations between local weight distributions of a binary linear code and its extended code are derived. A concrete relation is presented for the case that the extended code is transitive invariant and contains only codewords with weight multiple of four. Extended binary primitive BCH codes and Reed-Muller codes are transitive invariant codes.

A relation between local weight distributions of a binary linear code and its even weight subcode is also given. By using the relations, the local weight distributions of the $(127, k)$ binary primitive BCH codes for $36 \leq k \leq 50$, the punctured third-order Reed-Muller code of length 127, and their even weight subcodes are obtained from the local weight distributions of the $(128, k)$ primitive BCH codes for $36 \leq k \leq 50$ and 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 iff*

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

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

The 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 $(L_0, L_1, \ldots, L_n)$.*

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 $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$.*

Let $A_w(C)$ be the number of codewords with weight w in C. For the local weight distribution and the (global) weight distribution $(A_0(C), A_1(C), \ldots, A_n(C))$, we have the following lemma [1, 6].

Lemma 2 *Let d be the minimum distance of C. Then, we have that*

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

When the weight distribution is known, only $L_w(C)$ with $2d \leq w \leq n - k + 1$ need to be computed to obtain the local weight distribution. Generally, the complexity for computing the local weight distribution is larger than that for computing the weight distribution. Therefore, Lemma 2 is useful for obtaining local weight distributions. Moreover, when all the weights w in a code is confined in $w < 2d$ and $w > n - k + 1$, the local weight distribution can be obtained from the weight distribution straightforwardly. For example, the local weight distribution of the (n, k) primitive BCH code of length 63 for $k \leq 18$, of length 127 for $k \leq 29$, and of length 255 for $k \leq 45$ can be obtained from their weight distributions.

The MHM algorithm proposed in [4] uses the following invariance property under cyclic permutations, as well as Lemma 1.

Theorem 1 *[4] 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_{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_{ex}. Then, for any cyclic permuted codeword $\boldsymbol{v}'$ of $\boldsymbol{v}$, $\mathrm{ex}(\boldsymbol{v}')$ is a zero neighbor in C_{ex} if $\mathrm{ex}(\boldsymbol{v})$ is a zero neighbor in C_{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

$$\pi[D] = \{\pi\boldsymbol{v} : \boldsymbol{v} \in D\}. \tag{3}$$

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

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.*

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, $C/C' = \{\boldsymbol{v} + C' : \boldsymbol{v} \in C\}$. Then,

$$|C/C'| = 2^{k-k'}, \quad \text{and} \quad C = \bigcup_{D \in C/C'} D. \tag{4}$$

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*

$$\begin{aligned} LS_w(D) \;=\; & \{\boldsymbol{v} \in D : \mathrm{Supp}(\boldsymbol{v}') \not\subseteq \mathrm{Supp}(\boldsymbol{v}) \text{ for any } \boldsymbol{v}' \in C \setminus \{\boldsymbol{0}, \boldsymbol{v}\}, \\ & \text{and the Hamming weight of } \boldsymbol{v} \text{ is } w\}, \end{aligned} \tag{5}$$

with $0 \le w \le n$.

Then, from (4), the local weight distribution of C is given as the sum of the local weight subdistributions for the cosets in C/C', that is,

$$L_w = \sum_{D \in C/C'} |LS_w(D)|. \tag{6}$$

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 $\pi \in \mathrm{Aut}(C)$ such that $\pi[D_1] = D_2$.*

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',*

$$\{\pi : \pi[D] \in C/C' \text{ for any } D \in C/C'\} = \mathrm{Aut}(C) \cap \mathrm{Aut}(C').$$

$\mathrm{Aut}(C) \cap \mathrm{Aut}(C')$ (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 $\boldsymbol{v}_1 \in D_1$.*

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

On the basis of 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.

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 $O(n^2k)$.

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 $O(n^2k)$. Hence, the time complexity of Step (3) of the proposed algorithm is $O(n^2k \cdot e2^{k'})$. The time complexity of Step (2), partitioning the set of cosets, is $O(n(k-k')2^{k-k'}|\Pi|)$ [11].

Therefore, the time complexity of the entire algorithm is $O(n^2k \cdot e2^{k'} + n(k-k')2^{k-k'}|\Pi|)$. When k' is chosen as $k' > k/2$, then $2^{k'} > 2^{k-k'}$, and the complexity of partitioning the set of cosets is much smaller than of computing the local weight subdistributions for cosets.

The space complexity of the entire algorithm is $O((k-k')2^{k-k'})$ [11], since the space complexity for computing the local weight subdistributions is much smaller than that for partitioning the set of cosets.

4.3 Selection of the subcode

We should choose the subcode to make the number of permutations in Π large.

If there are several subcodes 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 the set of cosets is relatively small.

4.4 A method for checking the zero neighborship 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 $C(\boldsymbol{v}) = \{\boldsymbol{u} \mid \boldsymbol{u} \in C \setminus \{0\}, \ \mathrm{Supp}(\boldsymbol{u}) \subsetneq \mathrm{Supp}(\boldsymbol{v})\}$. Checking whether a codeword $\boldsymbol{v}$ is a zero neighbor or not is examining whether the dimension of $C(\boldsymbol{v})$, denoted by $\dim(C(\boldsymbol{v}))$, is zero or not [2]. For $\boldsymbol{v} \in C$ and i with $1 \leq i \leq n$, let $\mathrm{S}(\boldsymbol{v}, i) = \{(u_1, u_2, \ldots, u_n) \in C \mid u_j = v_j \text{ with } 1 \leq j \leq i\}$ and $\mathrm{C}(\boldsymbol{v}, i) = \{\boldsymbol{u} \mid \boldsymbol{u} \in \mathrm{S}(\boldsymbol{v}, i), \mathrm{Supp}(\boldsymbol{u}) \cap \{1, \ldots, i\} \subsetneq \mathrm{Supp}(\boldsymbol{v}) \cap \{1, \ldots, i\}\}$. A typical implimentation 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 complexity 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 makes $\mathrm{S}(\boldsymbol{v}, i)$ larger [9].

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 technique when i_0 is chosen as i $(0 \leq i \leq n)$: $((100 - i_0)/100)^2 + (1 - ((100 - i_0)/100)^2)/2^{k_s}$, 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 $i = 32, 46$, 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, since the codewords that need to be checked are generated coset by coset, the subset of $\mathrm{S}(\boldsymbol{v}, i)$ in a coset is used instead of $\mathrm{S}(\boldsymbol{v}, i)$. 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 Relations of Local Weight Distributions

5.1 General relation

Consider a binary linear code C of length n, its extended code C_{ex}, and its even weight subcode C_{even}. For a codeword $\boldsymbol{v} \in C$, let $\mathrm{wt}(\boldsymbol{v})$ be the Hamming weight of $\boldsymbol{v}$ and $\boldsymbol{v}^{(\mathrm{ex})}$ be the corresponding codeword in C_{ex}, that is, $\boldsymbol{v}^{(\mathrm{ex})}$ is obtained from $\boldsymbol{v}$ by adding the over-all parity bit.

If $\boldsymbol{v} \in C$ can be represented as $\boldsymbol{v} = \boldsymbol{v}_1 + \boldsymbol{v}_2$, where $\boldsymbol{v}_1, \boldsymbol{v}_2 \in C$ and $\mathrm{Supp}(\boldsymbol{v}_1) \cap \mathrm{Supp}(\boldsymbol{v}_2) = \emptyset$, $\boldsymbol{v}$ is said to be decomposable. From Lemma 1, $\boldsymbol{v}$ is not a zero neighbor if and only if $\boldsymbol{v}$ is decomposable. To consider a relation between C and C_{ex} with respect to zero neighborship, for even weight codewords, we refine the notion, decomposable, and introduce only-odd-decomposable and even-decomposable.

Definition 5 *Let $\boldsymbol{v} \in C$ be a decomposable codeword with even* $\mathrm{wt}(\boldsymbol{v})$. *That is, $\boldsymbol{v}$ is not a zero neighbor in C. $\boldsymbol{v}$ is said to be only-odd-decomposable, if all the decomposition of*

Table 1: Zero neighborship of $\boldsymbol{v}$ in a linear block code, $\boldsymbol{v}_{\rm ex}$ in its extended code, and $\boldsymbol{v}$ in its even weight subcode.

$\boldsymbol{v}$ in C			$\boldsymbol{v}^{(\rm ex)}$ in $C_{\rm ex}$		$\boldsymbol{v}$ in $C_{\rm even}$	
Zero neighborship	Weight	Decomposability	Zero neighborship	Theorem 5	Zero neighborship	Theorem 8
Yes	Odd	Not decomposable	Yes	1	N/A	N/A
	Even				Yes	1
No	Odd	Decomposable	No	2-(a)	N/A	N/A
	Even	Only-odd-decomposable	Yes	2-(b)	Yes	2
	Even	Even-decomposable	No		No	

$\boldsymbol{v}$ is of the form $\boldsymbol{v}_1 + \boldsymbol{v}_2$ with the odd weight codewords $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Otherwise, $\boldsymbol{v}$ is said to be even-decomposable.

When $\boldsymbol{v}$ is even-decomposable, there is a decomposition of $\boldsymbol{v}$, $\boldsymbol{v}_1 + \boldsymbol{v}_2$ such that both $\mathrm{wt}(\boldsymbol{v}_1)$ and $\mathrm{wt}(\boldsymbol{v}_2)$ are even. The relation between C and $C_{\rm ex}$ with respect to zero neighborship is given in the following theorem, which is also summarized in Table 1.

Theorem 5 *1. For a zero neighbor $\boldsymbol{v}$ in C, $\boldsymbol{v}^{(\rm ex)}$ is a zero neighbor in $C_{\rm ex}$.*

2. For a codeword $\boldsymbol{v}$ which is not a zero neighbor in C, the following a) and b) hold.

(a) When $\mathrm{wt}(\boldsymbol{v})$ is odd, $\boldsymbol{v}^{(\rm ex)}$ is not a zero neighbor in $C_{\rm ex}$.

(b) When $\mathrm{wt}(\boldsymbol{v})$ is even, $\boldsymbol{v}^{(\rm ex)}$ is a zero neighbor in $C_{\rm ex}$ if and only if $\boldsymbol{v}$ is only-odd-decomposable in C.

For the local weight distributions of a code and its extended code, we have the following theorem, which is a direct consequence of Theorem 5.

Theorem 6 *If there is no only-odd-decomposable codeword in C,*

$$L_{2i}(C_{\rm ex}) \;=\; L_{2i-1}(C) + L_{2i}(C), \quad 0 \le i \le n/2. \tag{7}$$

From Theorem 6, the local weight distributions of $C_{\rm ex}$ are obtained from that of C. Next, we give a useful sufficient condition under which no only-odd-decomposable codeword exists.

Theorem 7 *If all the weights of codewords in $C_{\rm ex}$ are multiples of four, no only-odd-decomposable codeword exists in C.*

For example, all the weights of codewords in the $(128, k)$ extended primitive BCH code with $k \leq 57$ are multiples of four. The parameters of Reed-Muller codes with which all the weights of codewords are multiples of four are given by Corollary 13 of Chapter 15 in [15]. The third-order Reed-Muller code of length 128, 256, and 512 are true for the case.

Although the local weight distribution of $C_{\rm ex}$ for these codes can be obtained from that of C by using Theorem 6, what we need is a method for obtaining the local weight distribution of C from that of $C_{\rm ex}$. We need to know the number of zero neighbors with parity bit one. In the next section, we will show a method to obtain the number of zero neighbors with parity bit one for a class of transitive invariant codes.

A similar relation as that between C and $C_{\rm ex}$ holds between C and $C_{\rm even}$. This relation is given in Theorem 8.

Theorem 8 *1. For an even weight zero neighbor $\boldsymbol{v}$ in C, $\boldsymbol{v}$ is a zero neighbor in $C_{\rm even}$.*

2. For an even weight codeword $\boldsymbol{v}$ which is not a zero neighbor in C, $\boldsymbol{v}$ is a zero neighbor if and only if $\boldsymbol{v}$ is only-odd-decomposable in C.

From Theorem 8, we have Theorem 9.

Theorem 9 *If there is no only-odd-decomposable codeword in C,*

$$L_{2i}(C_{\rm even}) = L_{2i}(C), \qquad 0 \leq i \leq n/2. \tag{8}$$

5.2 Relation for transitive invariant extended codes

A Transitive invariant code is the code which is invariant under a transitive group of permutations. A group of permutations is said to be transitive if for any two symbols in a codeword there exists a permutation that interchanges them [16]. The extended primitive BCH codes and Reed-Muller codes are transitive invariant codes. For a transitive invariant $C_{\rm ex}$, a relation between the (global) weight distributions of C and $C_{\rm ex}$ is presented in Theorem 8.15 in [16]. A similar relation holds for local weight distribution. The following lemma can be proved in a similar way as the proof of Theorem 8.15.

Lemma 5 *If $C_{\rm ex}$ is a transitive invariant code of length $n+1$, the number of zero neighbors with parity bit one is $\frac{w}{n+1} L_w(C_{\rm ex})$.*

It is clear that there are $\frac{n+1-w}{n+1} L_w(C_{\rm ex})$ zero neighbors with weight w whose parity bit is zero from this lemma. The following theorem is obtained from Theorem 5 and Lemma 5.

Theorem 10 *If $C_{\rm ex}$ is a transitive invariant code of length n+1,*

$$L_i(C) = \frac{i+1}{n+1} L_{i+1}(C_{\rm ex}), \quad \text{for odd } i, \tag{9}$$

$$L_i(C) \leq \frac{n+1-i}{n+1} L_i(C_{\rm ex}), \quad \text{for even } i. \tag{10}$$

If there is no only-odd-decomposable codeword in a transitive invariant code $C_{\rm ex}$, the equality of (10) holds. That is, in this case, we have that

$$L_i(C) = \frac{n+1-i}{n+1} L_i(C_{\rm ex}), \quad \text{for even } i. \tag{11}$$

Therefore, for a transitive invariant code $C_{\rm ex}$ having no only-odd-decomposable codeword in C, the local weight distributions of C can be obtained from that of $C_{\rm ex}$ by using (9) and (11) in Theorem 10. After computing the local weight distribution of C, that of $C_{\rm even}$ can be obtained by using Theorem 9.

6 Obtained Local Weight Distributions

As discussed in the previous section, the local weight distributions of the $(127, k)$ primitive BCH codes for $k \leq 57$, the punctured third-order Reed-Muller codes of length 127, 255, and 511, and their even weight subcodes are obtained from those of the corresponding extended codes by using Theorems 9 and 10. The obtained local weight distributions are presented in Table 2. Since the local weight distribution for the $(128, 57)$ extended primitive BCH code and the third-order Reed-Muller code of length 256 are unknown, only the local weight distributions of the $(127, k)$ primitive BCH codes for $k = 36, 43, 50$, the punctured third-order Reed-Muller code of length 127 are given in the tables.

7 Conclusion

In this resume, some of our results on the local weight distribution are given. An algorithm is shown for computing the local weight distribution of a code 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. Some relations between local weight distributions of a binary linear code, its extended code, and its even weight subcode are presented. The local weight distributions of the $(127, k)$ primitive BCH codes for $k = 36, 43, 50$, the punctured third-order Reed-Muller code of length 127, and their even weight subcodes are presented.

If the local weight distribution of the $(128, 57)$ extended primitive BCH code and the third-order Reed-Muller code of length 256 are obtained, we can obtain the local weight distributions of the $(127, 57)$ primitive BCH code, the punctured third-order Reed-Muller code of length 255, and their even weight subcodes.

Table 2: The local weight distributions of the $(127, k)$ primitive BCH codes for $k = 36, 43$, and 50 and the punctured third-order Reed-Muller code of length 127 .

(127, 36) BCH code

w	L_w
31	2,667
32	8,001
35	4,572
36	11,684
39	640,080
40	1,408,176
43	12,220,956
44	23,330,916
47	132,560,568
48	220,934,280
51	823,921,644
52	1,204,193,172
55	3,157,059,472
56	4,059,076,464
59	7,022,797,740
60	7,959,170,772
63	9,742,066,368
64	9,742,066,368
67	7,959,170,772
68	7,022,797,740
71	4,059,071,892
72	3,157,055,916
75	1,204,193,172
76	823,921,644
79	217,627,200
80	130,576,320
83	23,330,916
84	12,220,956
87	1,408,176
88	640,080

(127, 43) BCH code

w	L_w
31	31,115
32	93,345
35	2,478,024
36	6,332,728
39	82,356,960
40	181,185,312
43	1,554,145,736
44	2,967,005,496
47	16,837,453,752
48	28,062,422,920
51	106,485,735,720
52	155,632,998,360
55	400,716,792,672
56	515,207,304,864
59	905,612,814,120
60	1,026,361,189,336
63	1,238,334,929,472
64	1,238,334,929,472
67	1,026,345,592,720
68	905,599,052,400
71	515,097,101,376
72	400,631,078,848
75	155,191,535,184
76	106,183,681,968
79	26,980,367,680
80	16,188,220,608
83	1,617,588,840
84	847,308,440

(127, 50) BCH code

w	L_w
27	40,894
28	146,050
31	4,853,051
32	14,559,153
35	310,454,802
36	793,384,494
39	10,538,703,840
40	23,185,148,448
43	199,123,183,160
44	380,144,258,760
47	2,154,195,406,104
48	3,590,325,676,840
51	13,633,106,229,288
52	19,925,309,104,344
55	51,285,782,220,204
56	65,938,862,854,548
59	115,927,157,830,260
60	131,384,112,207,628
63	158,486,906,385,472
64	158,486,906,385,472
67	131,258,388,369,668
68	115,816,225,032,060
71	64,917,266,933,304
72	50,491,207,614,792
75	15,345,182,164,032
76	10,499,335,164,864

(127, 64) punc. RM code

w	L_w
15	11811
16	82677
23	13889736
24	60188856
27	684345088
28	2444089600
31	77893639488
32	233680918464
35	5097898213632
36	13027962101504
39	172489249981440
40	379476349959168
43	3259718804643840
44	6223099536138240
47	35130035853803520
48	58550059756339200
51	218602288622075904
52	319495652601495552
55	766899891905495040
56	986014146735636480
59	1306771964441395200
60	1481008226366914560
63	258664522171023360
64	258664522171023360

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, and M. Morii, "On computing the local distance profile of binary cyclic codes," *Proc. of ISITA2002*, pp.415–418, Oct. 2002.

[4] 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.

[5] G.D. Forney, Jr., "Geometrically uniform codes," *IEEE Trans. Inform. Theory*, vol.37, no.5, pp.1241–1260, Sept, 1991.

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

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

[8] 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.

[9] 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.

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

[11] 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.

[12] K. Yasunaga and T. Fujiwara, "The local weight distributions of the $(128, 50)$ extended binary primitive BCH code and $(128, 64)$ Reed-Muller code," *IEICE Technical Report*, IT2004-19, Jul. 2004.

[13] K. Yasunaga and T. Fujiwara, "An algorithm for computing the local weight distribution of binary linear codes closed under a group of permutations," *Proc. of ISITA2004*, pp.846–851, Oct. 2004.

[14] K. Yasunaga and T. Fujiwara, "Relations among the Local Weight Distributions of a Linear Block Code, Its Extended Code and Its Even Weight Subcode," *SITA2004*, pp.559–562, Dec. 2004.

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

[16] W.W. Peterson and E.J. Weldon, Jr., *Error-correcting codes, 2nd Edition*, MIT Press, 1972.