Determination of the Local Weight Distribution of Binary Linear Block Codes

Kenji Yasunaga and Toru Fujiwara, *Member, IEEE*

***Abstract*— Some methods to determine the local weight distribution of binary linear codes are presented. Two approaches are studied: A computational approach and a theoretical approach. For the computational approach, an algorithm for computing the local weight distribution of codes using the automorphism group of the codes is devised. In this algorithm, a code is considered the set of cosets of a subcode, and the set of cosets is partitioned into equivalence classes. Thus, only the weight distributions of zero neighbors for each representative coset of equivalence classes are computed. For the theoretical approach, relations between the local weight distribution of a code, its extended code, and its even weight subcode are studied. As a result, the local weight distributions of some of the extended primitive BCH codes, Reed-Muller codes, primitive BCH codes, punctured Reed-Muller codes, and even weight subcodes of primitive BCH codes and punctured Reed-Muller codes are determined.**

***Index Terms*— Local weight distribution, binary linear code, automorphism group, zero neighbor, coset, primitive BCH code, Reed-Muller code.**

I. 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 [1]. The studies of zero neighbors in a linear code are crucial for the performance analysis of the code under maximum likelihood (ML) decoding. The weight distribution of zero neighbors, called *local weight distribution* [2] (or *local distance profile* [1], [9]), is also important for ML performance of the code. For example, the local weight distribution could give a tighter upper bound on error probability for soft decision decoding over an AWGN channel than the usual union bound [9]. Zero neighbor appears in an optimal hard decision decoding algorithms, so called *gradient-like decoding* [4], [12]. The number of zero neighbors in a code determines the complexity of gradient-like decoding of the code. In the context of cryptography, Massey showed that the access structure of a secret sharing scheme determined by a linear code is characterized by zero neighbors in the dual code [15].

Agrell showed an efficient method to examine zero neighborship of a codeword in a binary linear code and computed the local weight distributions by examining all the codewords for some codes [1]. In [3], Ashikhmin and Barg studied zero neighbors (called *minimal vectors* in [3]) for certain classes of codes, and derived formulas for the local weight distribution of Hamming codes and second-order Reed-Muller codes. Partial results for the local weight distributions of Reed-Muller codes are given in [6]. In [2] and [3], asymptotic analyses for long codes and random codes are given. Mohri et al. proposed the computational algorithms for cyclic codes [16], [17]. The number of codewords to be examined is reduced in their work. The basic idea for the reduction was suggested by Agrell [1]. Using the algorithms, they determined the local weight distributions of all the primitive BCH codes of length 63.

In this paper, some methods to determine the local weight distribution of binary linear block codes are studied. Two approaches are studied: A computational approach and a theoretical approach. The basic idea of the computational approach is the one suggested by Agrell [1], which is also used in the algorithms in [16] and [17]. The proposed computational algorithm is for codes that are closed under a group of permutations. The proposed algorithm is also based on that of computing the (global) weight distribution for primitive BCH codes in [10]. In this algorithm, a code is considered a set of cosets of a subcode, and the set of cosets are partitioned into equivalence classes with an invariance property. Only the weight distributions for each representative coset are computed. Thereby the computational complexity is reduced. In this paper, we show that this idea can be applied to local weight distribution. The local weight distributions for some of extended primitive BCH codes and Reed-Muller codes are obtained using this computational approach. As for the theoretical approach, relations between the local weight distributions of a code, its extended code, and its even weight subcode are studied. We show that, for a code that the extended code is a transitive invariant code and contains no codewords with weight multiples of four, the local weight distribution is determined from that of the corresponding extended code. As a result, the local weight distributions for some of primitive BCH codes, punctured Reed-Muller codes, and their even weight subcodes are obtained.

The outline of this paper is as follows. In Section II, definitions and some properties for local weight distribution are given. In Section III, an algorithm for computing the local weight distribution is proposed. The algorithm uses the automorphism group of a code and performs effectively for extended primitive BCH codes and Reed-Muller codes. In Section IV, two methods for improving the algorithm proposed in Section III are presented. The first method uses the code tree structure of a code. The second uses the automorphism group of a code. In Section V, a theoretical approach to determine

Preliminary versions of this paper were presented at the 2004 International Symposium on Information Theory and Its Applications (ISITA2004), Parma, Italy, and the 2005 IEEE International Symposium on Information Theory (ISIT2005), Adelaide, Australia.

The authors are with Graduate School of Information Science and Technology, Osaka University, Suita, 565-0871, Japan.

the local weight distribution is presented. Relations between the local weight distributions of a code, its extended code, and its even weight subcode are given. In Section VI, the local weight distributions that are obtained using the methods described in Sections III-V are presented. For primitive BCH codes, the local weight distributions of the $(127, k)$ codes for $k \leq 50$, their extended codes, and their even weight subcodes are obtained. For Reed-Muller codes, the local weight distributions of the third-order Reed-Muller code of length 128, its punctured code, and the even weight subcode of the punctured code are obtained.

II. Local Weight Distribution

In this section, the definition, some properties, and applications for the local weight distribution of binary linear block codes are presented.

A. Definitions

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 $\mathbf{R}^n$. A zero neighbor of C is defined as follows [1]:

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 Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbf{R}^n$.

A zero neighbor is also called a minimal codeword in [3]. The following lemma is useful to check whether a given codeword is a zero neighbor or not [1].

Lemma 1: $\boldsymbol{v} \in C$ is a zero neighbor if and only if there is not a $\boldsymbol{v}' \in C \setminus \{\mathbf{0}\}$ 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 $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$.

The local weight distribution is defined as follows:

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

B. Some Properties

For the local weight distribution, we have the following lemma [2], [3].

Lemma 2: Let $A_w(C)$ be the number of codewords with weight w in C and d be the minimum distance of C.

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

When the global weight distribution $(A_0(C),\ A_1(C), \ldots,$ $A_n(C))$ is known, only $L_w(C)$ with $2d \leq w \leq n - k + 1$ needs 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 global weight distribution. Therefore, Lemma 2 is useful for obtaining local weight distributions. Moreover, when all the weights w in a code are confined in $w < 2d$ and $w > n - k + 1$, the local weight distribution can be obtained from the global 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 global weight distributions.

In general, the complexity for computing the local weight distribution, as well as that for the global weight distribution, is very large. Agrell noted in [1] that the automorphism group of codes helps reduce the complexity. Using the automorphism group of cyclic codes, i.e. cyclic permutations, Mohri et al. obtained the local weight distributions of the $(63, k)$ primitive BCH codes for $k \leq 45$ [16], [17]. The algorithm uses the following invariance property for cyclic permutations.

Theorem 1: Let C be a binary cyclic code. A codeword $\boldsymbol{v} \in C$ is a zero neighbor if and only if any cyclic permuted codeword of $\boldsymbol{v}$ 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 otherwise, (2) all codewords in S are not zero neighbors.

In their algorithm, the representative codeword of cyclic permutations (a representative codeword of S in Corollary 1) and the number of the equivalent codewords (the size of S) are generated efficiently. The complexity is about $1/n$ that of the brute force method. The local weight distributions of the $(63, k)$ primitive BCH codes with $k = 51, 57$ are obtained by using another algorithm [16]. The latter algorithm generates the representative codewords once or more, although the former algorithm generates the representative codewords only once.

The following corollary implies that the algorithms in [16] and [17] can be applied to extended cyclic codes straightforwardly.

Corollary 2: Let C and C_{ex} be a binary cyclic code and its extended code, respectively. For $\boldsymbol{v} \in C$, let $\boldsymbol{v}^{(\text{ex})}$ be the corresponding extended codeword in C_{ex}, that is, $\boldsymbol{v}^{(\text{ex})}$ is obtained from $\boldsymbol{v}$ by adding the over-all parity bit. For any cyclic permuted codeword $\sigma^i\boldsymbol{v}$ of $\boldsymbol{v}$, $(\sigma^i\boldsymbol{v})^{(\text{ex})}$ is a zero neighbor in C_{ex} if and only if $\boldsymbol{v}^{(\text{ex})}$ is a zero neighbor in C_{ex}.

Proof: (If part) Suppose that $(\sigma^i\boldsymbol{v})^{(\text{ex})}$ is not a zero neighbor in C_{ex}. There exists $\boldsymbol{u} \in C$ such that $\mathrm{Supp}((\sigma^i\boldsymbol{u})^{(\text{ex})}) \subsetneq \mathrm{Supp}((\sigma^i\boldsymbol{v})^{(\text{ex})})$. Then $\mathrm{Supp}(\boldsymbol{u}^{(\text{ex})}) \subsetneq \mathrm{Supp}(\boldsymbol{v}^{(\text{ex})})$, and this contradicts the fact that $\boldsymbol{v}^{(\text{ex})}$ is a zero neighbor in C_{ex}. (Only if part) Suppose that $\boldsymbol{v}^{(\text{ex})}$ is not a zero neighbor in C_{ex}. There exists $\boldsymbol{u} \in C$ such that $\mathrm{Supp}(\boldsymbol{u}^{(\text{ex})}) \subsetneq \mathrm{Supp}(\boldsymbol{v}^{(\text{ex})})$. Hence, $\mathrm{Supp}((\sigma^i\boldsymbol{u})^{(\text{ex})}) \subsetneq \mathrm{Supp}((\sigma^i\boldsymbol{v})^{(\text{ex})})$, and this contradicts the fact that $(\sigma^i \boldsymbol{v})^{(\mathrm{ex})}$ is a zero neighbor in C_{ex}. $\square$

From Corollaries 1 and 2, the zero neighborships of codewords in $S' = \{\boldsymbol{v}^{(\mathrm{ex})}, (\sigma\boldsymbol{v})^{(\mathrm{ex})}, (\sigma^2\boldsymbol{v})^{(\mathrm{ex})}, \ldots, (\sigma^{p(\sigma,\boldsymbol{v})-1}\boldsymbol{v})^{(\mathrm{ex})}\}$ are the same. To compute the local weight distribution of an extended cyclic code C_{ex}, we only have to check zero neighborship for the representative extended codewords of cyclic permutations. Thus, we can compute the local weight distribution of an extended cyclic code in the same way as that in the algorithms in [16] and [17] for representative codewords with respect to the cyclic group of permutations. However, extended primitive BCH codes are closed under the affine group of permutations, which are larger than the cyclic group of permutations. Using a larger group of permutations, the complexity for computing the local weight distribution may be reduced. This is a basic observation for the computational approach described in Section III.

C. Applications

For BPSK transmission, a codeword $\boldsymbol{v} \in C$ is transmitted as $s(\boldsymbol{v})$ (the mapping s is defined in II-A). Assuming AWGN interference, the received sequence when $s(\boldsymbol{v})$ is transmitted is

$$\boldsymbol{r} = s(\boldsymbol{v}) + \boldsymbol{n},$$

where $\boldsymbol{r}$ is the n-dimensional vector and $\boldsymbol{n}$ is an n-dimensional vector whose elements are independent Gaussian random variables with zero mean and variance $N_0/2$. Since C is a linear code, we assume that the all-zero codeword $\mathbf{0}$ is transmitted. The word error probability of soft-decision decoding (ML decoding) is given as

$$P_e = P\left[\bigcup_{\boldsymbol{v}\in C\setminus\{\mathbf{0}\}} \mathcal{E}_{\mathbf{0}\to\boldsymbol{v}}\right] \tag{3}$$

$$\leq \sum_{\boldsymbol{v}\in C\setminus\{\mathbf{0}\}} P[\mathcal{E}_{\mathbf{0}\to\boldsymbol{v}}], \tag{4}$$

where $\mathcal{E}_{\mathbf{0}\to\boldsymbol{v}}$ denotes the pairwise error event. This is the event that, when the all-zero codeword $\mathbf{0}$ is transmitted, ML decoder metric (the Euclidean distance) between the received vector $\boldsymbol{r}$ and $s(\boldsymbol{v})$ is smaller than that between $\boldsymbol{r}$ and $s(\mathbf{0})$, i.e., $\mathcal{E}_{\mathbf{0}\to\boldsymbol{v}} = \{\boldsymbol{r} : d_E(\boldsymbol{r}, s(\boldsymbol{v})) \leq d_E(\boldsymbol{r}, s(\mathbf{0}))\}$. (4) is a union upper bound of P_e. Then the union bound of P_e using the weight distribution of C is obtained [7] as

$$P_e \leq \sum_{\boldsymbol{v}\in C\setminus\{\mathbf{0}\}} Q\left(\sqrt{\mathrm{wt}(\boldsymbol{v})\frac{2E_b}{N_0}}\right) \tag{5}$$

$$= \sum_{i=1}^{n} A_i(C)\, Q\left(\sqrt{i\frac{2E_b}{N_0}}\right), \tag{6}$$

where $\mathrm{wt}(\boldsymbol{v})$ denotes the Hamming weight of $\boldsymbol{v}$ and $Q(x)$ is the *complementary error function*; $Q(x) = \int_x^\infty (2\pi)^{-1/2}\exp(-z^2/2)dz$.

Using the set $Z(C)$ of zero neighbors in C, (3) and (4) can be rewritten by

$$P_e = P\left[\bigcup_{\boldsymbol{v}\in Z(C)} \mathcal{E}_{\mathbf{0}\to\boldsymbol{v}}\right] \tag{7}$$

$$\leq \sum_{\boldsymbol{v}\in Z(C)} P[\mathcal{E}_{\mathbf{0}\to\boldsymbol{v}}]. \tag{8}$$

Inequality (8) is called a minimal union bound [8]. A minimal union bound using the local weight distribution of C is obtained in the same way as (6) [2]:

$$P_e \leq \sum_{i=1}^{n} L_i(C)\, Q\left(\sqrt{i\frac{2E_b}{N_0}}\right). \tag{9}$$

The right-hand side of (9) is strictly smaller than that of (6). Agrell pointed out in [1] that other bounds, related to the union bound, such as Berlekamp's tangential union bound [5], may be improved in a similar fashion.

Zero neighbor appears in an optimal hard decision decoding algorithms [12]. The number of zero neighbors in a code determines the complexity of the decoding. This decoding method is so called *gradient-like decoding* [4]. See [4] for details.

Zero neighbors in a linear code have a link to secret-sharing schemes using error-correcting codes. Massey showed that the set of zero neighbors in the dual code completely specifies the access structure of the secret-sharing scheme [15].

III. Computational Approach to Determine Local Weight Distribution

In this section, a method for computing the local weight distribution using the automorphism group of the code is presented. In [16] and [17], the complexity for computing the local weight distribution is reduced by using an invariance property for cyclic permutations. This invariance property for cyclic permutations can be generalized to an invariance property for any group of permutations. Using the invariance property for the larger group of permutations, we may reduce the number of representative codewords. However, it is not easy to obtain the representative codewords and the number of the equivalent codewords.

In order to use the generalized invariance property, the invariance property is applied to the set of cosets of a subcode rather than the set of codewords. This application reduces the complexity of finding the representatives, which is much smaller than the complexity of checking whether every representative is a zero neighbor or not. This idea is used in [10] for computing the global weight distribution of extended binary primitive BCH codes. In the following, we show that this idea can be applicable for computing local weight distribution.

A. Invariance Property

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

$$\pi[D] = \{\pi\boldsymbol{v} : \boldsymbol{v} \in D\}. \tag{10}$$

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

$$\mathrm{Aut}(C) = \{\pi : \pi[C] = C\}. \tag{11}$$

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

Theorem 2 (Invariance property): For $\pi \in \mathrm{Aut}(C)$ and $\boldsymbol{v} \in C$, $\pi\boldsymbol{v}$ is a zero neighbor if and only 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 $\mathrm{Supp}(\boldsymbol{v}) \supsetneq \mathrm{Supp}(\boldsymbol{v}'')$, contradicting the fact that $\boldsymbol{v}$ is a zero neighbor, from Lemma 1. $\square$

This theorem derives the following corollary.

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; otherwise, (2) all codewords in S are not zero neighbors.

In order to use this generalized invariance property, we apply the invariance property to the set of cosets of a subcode rather than the set of codewords.

B. Local Weight Subdistribution for Cosets of Subcode

For a binary (n, k) linear code C and its linear subcode C' with dimension k', let C/C' denote the set of cosets of C' in C, that is, $C/C' = \{\boldsymbol{v} + C' : \boldsymbol{v} \in C \setminus C'\}$. Then

$$|C/C'| = 2^{k-k'}, \quad \text{and} \quad C = \bigcup_{D \in C/C'} D. \tag{12}$$

Definition 3 (Local weight subdistribution for cosets): The local weight subdistribution for a coset $D \in C/C'$ (with respect to C) is the weight distribution of zero neighbors of C in D. The local weight subdistribution for D is $(|Z_0(D)|, |Z_1(D)|, \ldots, |Z_n(D)|)$, where

$$\begin{aligned} Z_w(D) \;=\; & \{\boldsymbol{v} \in D : \mathrm{Supp}(\boldsymbol{v}') \not\subseteq \mathrm{Supp}(\boldsymbol{v}) \text{ for any} \\ & \boldsymbol{v}' \in C \setminus \{\boldsymbol{0}, \boldsymbol{v}\}, \text{ and } \mathrm{wt}(\boldsymbol{v}) \text{ is } w\}, \end{aligned} \tag{13}$$

with $0 \le w \le n$.

Then, from (12), the local weight distribution of C is given as the sum of the local weight subdistributions for the cosets in C/C', that is,

$$L_w = \sum_{D \in C/C'} |Z_w(D)|. \tag{14}$$

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

Theorem 3 (Invariance property for cosets): 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 $\pi[D_1] = D_2$.

Proof: For any codewords $\boldsymbol{v} \in D_1$, from Theorem 2, $\pi\boldsymbol{v} \in D_2$ 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. $\square$

This theorem is 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',

$$\{\pi : \pi[D] \in C/C' \text{ for any } D \in C/C'\} = \mathrm{Aut}(C) \cap \mathrm{Aut}(C'). \tag{15}$$

Proof: Let $\pi \in \mathrm{Aut}(C) \cap \mathrm{Aut}(C')$. For a coset $\boldsymbol{v}_1 + C' \in C/C'$, suppose that $\pi\boldsymbol{v}_1 \in \boldsymbol{v}_2 + C'$. For any codeword $\boldsymbol{v}_1 + \boldsymbol{u}_1 \in \boldsymbol{v}_1 + C'$,

$$\begin{aligned} \pi(\boldsymbol{v}_1 + \boldsymbol{u}_1) &= \pi\boldsymbol{v}_1 + \pi\boldsymbol{u}_1 \\ &= \boldsymbol{v}_2 + \boldsymbol{u}_2 + \pi\boldsymbol{u}_1, \quad \boldsymbol{u}_2 \in C', \\ &= \boldsymbol{v}_2 + (\boldsymbol{u}_2 + \pi\boldsymbol{u}_1) \;\in\; \boldsymbol{v}_2 + C'. \end{aligned} \tag{16}$$

Thus, $\pi[\boldsymbol{v}_1 + C'] = \boldsymbol{v}_2 + C' \in C/C'$. Then $\{\pi : \pi[D] \in C/C' \text{ for any } D \in C/C'\} \supseteq \mathrm{Aut}(C) \cap \mathrm{Aut}(C')$.

Let $\pi \in \{\rho : \rho[D] \in C/C' \text{ for any } D \in C/C'\}$. For any codeword $\boldsymbol{v} \in C$, $\boldsymbol{v}$ must be in either coset in C/C', and then $\pi\boldsymbol{v} \in C$. Thus, $\pi \in \mathrm{Aut}(C)$. C' itself is one of cosets in C/C'. For any codeword $\boldsymbol{u} \in C'$, $\pi\boldsymbol{u} \in C'$ because $\pi[C'] = C'$. Thus, $\pi \in \mathrm{Aut}(C')$. Then $\{\pi : \pi[D] \in C/C' \text{ for any } D \in C/C'\} \subseteq \mathrm{Aut}(C) \cap \mathrm{Aut}(C')$. $\square$

$\mathrm{Aut}(C) \cap \mathrm{Aut}(C')$ (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 $\Pi \subseteq \mathrm{Aut}(C) \cap \mathrm{Aut}(C')$. 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.

Proof: Let $D_1, D_2, D_3 \in C/C'$.

(Reflexive: $D_1 \sim_\Pi D_1$) Since the identity permutation π_0 is in Π, $D_1 \sim_\Pi D_1$.

(Symmetric: $D_1 \sim_\Pi D_2 \to D_2 \sim_\Pi D_1$) Suppose that $D_1 \sim_\Pi D_2$ and $\pi[D_1] = D_2$ for $\pi \in \Pi$. Since Π forms a group, there exists $\rho \in \Pi$ such that $\rho[\pi[D_1]] = D_1$. Then $\rho[D_2] = D_1$, and $D_2 \sim_\Pi D_1$.

(Transitive: $D_1 \sim_\Pi D_2, D_2 \sim_\Pi D_3 \to D_1 \sim_\Pi D_3$) Suppose that $D_1 \sim_\Pi D_2$ and $D_2 \sim_\Pi D_3$. There exists $\pi, \rho \in \Pi$ such that $\pi[D_1] = D_2$, $\rho[D_2] = D_3$. Then $D_3 = \rho[D_2] = \rho\pi[D_1]$. Since $\rho\pi \in \Pi$, $D_1 \sim_\Pi D_3$. $\square$

When the set of cosets are partitioned into 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 $\Pi \subseteq \mathrm{Aut}(C) \cap \mathrm{Aut}(C')$. 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

$$\begin{aligned} \pi(\boldsymbol{v}_1 + \boldsymbol{v}) &= \pi \boldsymbol{v}_1 + \pi \boldsymbol{v} \\ &= \boldsymbol{v}_2 + \pi \boldsymbol{v}. \end{aligned} \quad (17)$$

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

From Theorem 4, in order 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 cosets 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.

C. Outline of the Proposed Algorithm

On the basis of the method for partitioning the set of cosets described in the previous section, we can 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.

D. Partitioning Cosets into Equivalence Classes

Our implementation of Step 2) of the algorithm is based on Theorem 4 and Lemma 4. Let H' be a parity check matrix of C' with

$$H' = \begin{pmatrix} H_0 \\ H \end{pmatrix}, \quad (18)$$

where H is a parity check matrix of C. H_0 is an $n \times (k - k')$ matrix. In order to partition cosets efficiently, we use the following condition:

$$\pi[\boldsymbol{v} + C'] = \boldsymbol{v}' + C' \quad \text{iff} \quad \pi \boldsymbol{v} H_0^T = \boldsymbol{v}' H_0^T, \quad (19)$$

where H_0^T represents the transpose of H_0.

Using a table with size $2^{k-k'}$, we need to compute the syndromes of length $k - k'$ for all the permuted coset leaders to partition these cosets into equivalence classes. The computational complexity of partitioning cosets into equivalence classes is $O(n(k-k')2^{k-k'}|\Pi|)$. If Π forms a group, the actual complexity would be much small. Suppose that $\pi[\boldsymbol{v} + C'] = \boldsymbol{v}' + C'$. After we found the equivalence cosets of $\boldsymbol{v} + C'$, including $\boldsymbol{v}' + C'$, we need not to find the equivalence cosets for $\boldsymbol{v}' + C'$ because the equivalence cosets of $\boldsymbol{v}' + C'$ are equal to that of $\boldsymbol{v} + C'$ when Π forms a group. Then the complexity for partitioning cosets into equivalence classes is $O(n(k-k')e|\Pi| + 2^{k-k'})$ where e is the number of equivalence classes in C/C'. The complexity $O(2^{k-k'})$ is for computing syndromes and the bookkeeping operations for the $2^{k-k'}$ coset leaders. Since e seems to be much smaller than $2^{k-k'}$, although we cannot know e before running a coset partitioning algorithm, the actual complexity for partitioning cosets into equivalence classes would be much small when Π forms a group.

E. Complexity

Here, we analyze the computational complexity of the algorithm. Let C be an (n, k) linear code and C' be an (n, k') linear subcode of C.

An efficient method for checking whether a codeword is a zero neighbor or not is presented in [1]. This method is used to check zero neighborship of codewords in [16] and [17]. We also use this method to check zero neighborship.

1) Time complexity: The time complexity of checking one codeword of the method in [1] is $O(n^2 k)$. Since the number of codewords in each coset is $2^{k'}$, the total number of codewords to be checked for zero neighborship is $e2^{k'}$, where e is the number of the equivalence classes. Hence, the time complexity of the proposed algorithm in Step 3 is $O(n^2 k \cdot e 2^{k'})$. The time complexity of partitioning cosets into equivalence classes in Step 2 is $O(n(k-k')2^{k-k'}|\Pi|)$, as described in Section III-D.

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.

2) Space complexity: The space complexity of checking a zero neighborship is very small, because we need space to store only a generator matrix of C in the method in [1], which is $O(nk)$. On the other hand, the space complexity of partitioning cosets into equivalence classes is much larger. We need space to store the entries proportional to $2^{k-k'}$, which is $O((k-k')2^{k-k'})$. We need $O(n(n-k'))$ space to store the parity check matrices of C and C'. The space complexity of the entire algorithm is $O(n^2 + (k-k')2^{k-k'})$.

F. Selection of the Subcode

In order to reduce the number of codewords that need to be checked for zero neighborship, we should choose the subcode C' for which the number of permutations in $\Pi \subseteq \mathrm{Aut}(C) \cap \mathrm{Aut}(C')$ is larger. However, the complexity of partitioning cosets into equivalence classes may become larger.

Therefore, 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 cosets into equivalence classes is relatively small.

G. Target Codes for the Computational Approach

The proposed algorithm can be applied to codes that are closed under a group of permutations and whose subcodes are also closed under the same group of permutations. The algorithm is suitable for extended primitive BCH codes and Reed-Muller codes. Extended primitive BCH codes are closed under the affine group and Reed-Muller codes are closed under the general affine group [14].

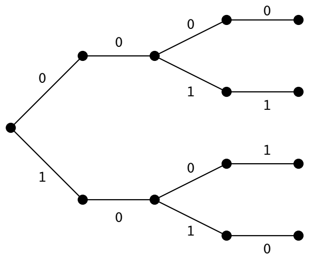

Fig. 1. The code tree of the code $\{0000, 0011, 1001, 1010\}$.

IV. Improvements of the Computational Approach

In this section, some improvements of the proposed algorithm for computing the local weight distribution are shown.

A. Code Tree Structure

We consider reducing the complexity of checking zero neighborship in a coset of C' by using the code tree structure of the coset. For simplicity, we consider C' itself as the coset. For $\boldsymbol{v} \in C'$, let

$$C(\boldsymbol{v}) = \{\boldsymbol{u} \mid \boldsymbol{u} \in C,\ \mathrm{Supp}(\boldsymbol{u}) \subseteq \mathrm{Supp}(\boldsymbol{v})\}. \tag{20}$$

A codeword $\boldsymbol{v}$ is a zero neighbor if and only if $C(\boldsymbol{v}) = \{\boldsymbol{0}, \boldsymbol{v}\}$. Thus, checking the zero neighborship of $\boldsymbol{v}$ is examining whether the dimension of $C(\boldsymbol{v})$, denoted by $\dim(C(\boldsymbol{v}))$, is one or not. For $\boldsymbol{v} \in C'$ and i with $1 \le i \le n$, let

$$\begin{aligned} C(\boldsymbol{v}, i) \quad = \quad & \{\boldsymbol{u} \mid \boldsymbol{u} \in C, \mathrm{Supp}(\boldsymbol{u}) \cap \{1, \ldots, i\} \\ & \subseteq \mathrm{Supp}(\boldsymbol{v}) \cap \{1, \ldots, i\}\}. \end{aligned} \tag{21}$$

Therefore, $C(\boldsymbol{v}, n) = C(\boldsymbol{v})$. A typical implementation to construct $C(\boldsymbol{v})$ is as follows: Construct $C(\boldsymbol{v}, 1)$ from C, and $C(\boldsymbol{v}, 2)$ from $C(\boldsymbol{v}, 1)$, and $C(\boldsymbol{v}, 3)$ from $C(\boldsymbol{v}, 2)$, and so on. This procedure can be done by using the generator matrix of C [1].

A code tree of a binary (n, k) code is an edge-labeled tree with depth n. Either 0 or 1 is labeled on each edge. For the code tree of a code C, the sequence of edge labels along each path from the root to a leaf is a codeword of C. There are 2^k leaves on the tree. For example, the code tree of $C = \{0000, 0011, 1001, 1010\}$ is shown in Fig. 1.

Now, we consider reducing the complexity for computing $C(\boldsymbol{v})$ for $\boldsymbol{v} \in C'$. For i with $1 \le i \le n$, let

$$C'^{\mathrm{f}}_i = \{(u_1, u_2, \ldots, u_n) \in C' \mid u_j = 0 \text{ with } 1 \le j \le i\}. \tag{22}$$

C'^{f}_i is the future subcode of C' at time i. For $\boldsymbol{v} \in C'$, $\boldsymbol{v} + C'^{\mathrm{f}}_i$ shares the same path to depth i in the code tree. This means, if we construct $C(\boldsymbol{v}, i)$ once, we do not need to construct $C(\boldsymbol{u}, i)$ for other $\boldsymbol{u} \in \boldsymbol{v} + C'^{\mathrm{f}}_i$ later, because $C(\boldsymbol{v}, i) = C(\boldsymbol{u}, i)$. We can save the computational complexity of constructing $C(\boldsymbol{u}, i)$ from C for each $\boldsymbol{u} \in \boldsymbol{v} + C'^{\mathrm{f}}_i$. However, to compute $C(\boldsymbol{u}, i)$ for all $\boldsymbol{u} \in C'$ along with the code tree is space-consuming. Therefore, we take the following method for checking zero neighborship of them.

- Choose an integer i with $1 \le i \le n$.
- For each coset $\boldsymbol{v} + C'^{\mathrm{f}}_i \in C'/C'^{\mathrm{f}}_i$, construct $C(\boldsymbol{v}, i)$ from C.
  - For each $\boldsymbol{u} \in \boldsymbol{v} + C'^{\mathrm{f}}_i$, construct $C(\boldsymbol{u})$ from $C(\boldsymbol{v}, i)$ and investigate $\dim(C(\boldsymbol{u}))$.

We can construct the generator matrix of C'^{f}_i by row operations of the generator matrix of C' (see Fig. 2). In Fig. 2, the dimension of C'^{f}_i is k'^{f}_i. We should choose i properly in order to make C'^{f}_i large and the complexity of examining the dimension of $C(\boldsymbol{u})$ from $C(\boldsymbol{v}, i)$ for each $\boldsymbol{u} \in C'^{\mathrm{f}}_i$ small; that is, make k'^{f}_i large and i large. The $k'^{\mathrm{f}}_i \times i$ zero matrix in Fig. 2 varies depending on the code tree structure of C'. For extended binary primitive BCH codes, permuting the symbol positions of codewords properly makes the $k'^{\mathrm{f}}_i \times i$ matrix larger [13]. To choose i properly, we should estimate the effect by using the above technique.

Estimating precisely how the computational complexity is reduced is not easy. We will estimate the effect roughly. When $\dim(C(\boldsymbol{u})) = 1$, $\dim(C(\boldsymbol{u}))$ is found to be one before constructing $C(\boldsymbol{u})$, since $C(\boldsymbol{u}, j)$ for $i \le j \le n$ may be equal to $C(\boldsymbol{u})$ for certain i with $i < n$. Let i_{end} be the average position i at which $\dim(C(\boldsymbol{v}, i))$ is found to be one or not for $\boldsymbol{v} \in C$. We observe that the number of zero neighbors is much more than that of non-zero-neighbors. For example, the rate of the number of zero neighbors to the number of all codewords is $0.9994\cdots$ for the $(128, 43)$ primitive BCH code. For any $\boldsymbol{v} \in C$ and $1 \le i \le i_{\mathrm{end}}$, assume:

$$\dim(C(\boldsymbol{v}, i)) = \frac{i_{\mathrm{end}} - i}{i_{\mathrm{end}}}(k - 1) + 1. \tag{23}$$

This equation means that $\dim(C(\boldsymbol{v}, i))$ decreases linearly with i and is equal to k (or 1) when $i = 0$ (or $i = i_{\mathrm{end}}$). The complexity of computing $C(\boldsymbol{v}, i+1)$ from $C(\boldsymbol{v}, i)$ is proportional to $\dim(C(\boldsymbol{v}, i))$. Thus, the complexity is given as $a \cdot \dim(C(\boldsymbol{v}, i))$ where a is a nonzero constant.

Consider the case i_0 is chosen as i for using the technique described in this section. Let U_1 be the complexity for computing $C(\boldsymbol{v})$, which is equal to the complexity for checking zero neighborship without the technique, U_2 be the complexity for computing $C(\boldsymbol{v}, i_{end})$ from $C(\boldsymbol{v}, i_0)$, and U_3 be the average complexity for computing $C(\boldsymbol{v}, i_0)$. Then

$$U_1 \quad = \quad \frac{a(\dim(C) - 1)\, i_{\mathrm{end}}}{2} = \frac{a(k-1) i_{\mathrm{end}}}{2}, \tag{24}$$

$$\begin{aligned} U_2 \quad &= \quad \frac{a(\dim(C, i_0) - 1)(i_{\mathrm{end}} - i_0)}{2} \\ &= \quad \frac{a(i_{\mathrm{end}} - i_0)^2 (k-1)}{2\, i_{\mathrm{end}}}, \end{aligned} \tag{25}$$

$$U_3 \quad = \quad U_1 - U_2. \tag{26}$$

Let R_{i_0} be the relative complexity of checking zero neighborship with the technique and without the technique. Then

$$\begin{aligned} R_{i_0} \quad &= \quad \frac{U_3 + U_2 \times 2^{k'^{\mathrm{f}}_{i_0}}}{U_1 \times 2^{k'^{\mathrm{f}}_{i_0}}} \\ &= \quad \left(1 - \frac{U_2}{U_1}\right) \frac{1}{2^{k'^{\mathrm{f}}_{i_0}}} + \frac{U_2}{U_1}, \end{aligned} \tag{27}$$

where

$$\frac{U_2}{U_1} \quad = \quad \left(\frac{i_{\mathrm{end}} - i_0}{i_{\mathrm{end}}}\right)^2. \tag{28}$$

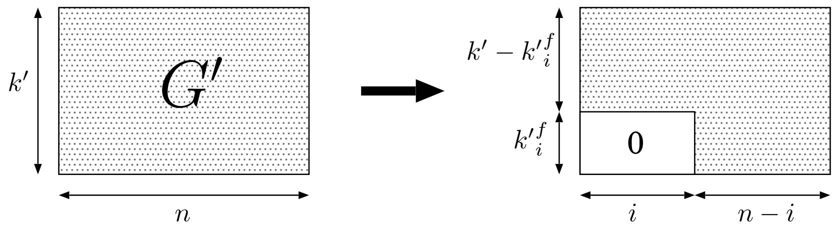

Fig. 2. A way of constructing C'^{f}_i from the generator matrix G'.

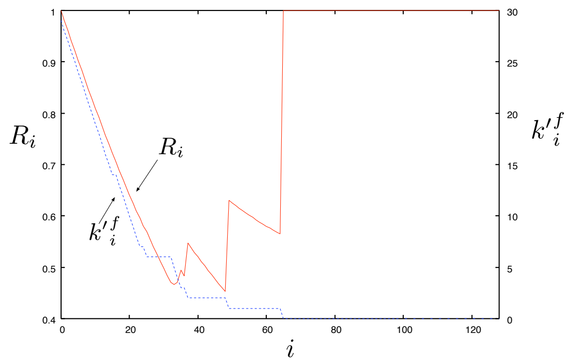

Fig. 3. Relative complexity R_i with $i_{\mathrm{end}} = 100$ and the dimension k'^f_i of C^f_i for the $(128, 50)$ extended BCH code using the $(128, 29)$ code as a subcode.

We estimated R_{i_0} for the case of the $(128, 50)$ extended BCH code. In this case, the $(128, 29)$ code is chosen as the subcode C' and the number of representative cosets is 258. To determine i_{end}, we use $2^{15} \times 258$ codewords by choosing 2^{15} codewords randomly from each of the 258 representative cosets. For every codeword $\boldsymbol{v}$ in such codewords, we examined the position in which $\dim(C(\boldsymbol{v}))$ is found to be one or not. Then the average was 100, that is, $i_{\mathrm{end}} = 100$. Since $k'^{\mathrm{f}}_{i_0}$ depends on i_0, we investigated $k'^{\mathrm{f}}_{i_0}$ and computed R_{i_0} for every i_0 $(1 \leq i_0 \leq n)$ (see Fig. 3). In this investigation, we use the permutation technique for making $k'^{\mathrm{f}}_{i_0}$ and i_0 larger proposed in [10] for extended BCH codes. From Fig. 3, the complexity of checking zero neighborship would reduced by $1/2$ for $i_0 = 33$ and 48. $k'^{\mathrm{f}}_{i_0} = 5, 2$ for $i_0 = 33, 48$, respectively. Actually, for the $(128, 50)$ extended BCH code and the $(128, 29)$ extended BCH subcode, the complexity is reduced by about 1/2 when we choose $i_0 = 48$.

If the dimension of the subcode is small, k'^{f}_i may become small and the effect of using the code tree structure is small. We should choose the subcode by considering the effect of using the code tree structure.

B. Invariance Property in Cosets

In the proposed algorithm, the invariance property for zero neighborship is applied to the set of cosets of a subcode rather than the set of codewords. This reduces the complexity of finding the representatives. However, we do not use the invariance property completely. That is, the invariance property is not used for codewords in cosets. In computing the local weight subdistribution for a coset, we can apply the invariance property to codewords in the coset. An invariance property in a coset is given in the following theorem.

Theorem 5: For a coset $\boldsymbol{v} + C' \in C/C'$, $\pi \in \{\rho : \rho\boldsymbol{v} \in \boldsymbol{v} + C'\}$, and $\boldsymbol{u} \in \boldsymbol{v} + C'$, $\pi\boldsymbol{u}$ is a zero neighbor in C if and only if $\boldsymbol{u}$ is a zero neighbor in C.

No efficient way is known for generating the representative codewords in a coset as in a code. Therefore, we use a similar method: Just as we applied the invariance property to the set of cosets in a code rather than the set of codewords in the code, we apply the invariance property to the set of cosets in a coset rather than the set of codewords in the coset. Thus, we consider a coset $\boldsymbol{v} + C' \in C/C'$ the set of cosets of C'', where C'' is a subcode of C'.

For a coset $\boldsymbol{v} + C' \in C/C'$, let $(\boldsymbol{v} + C')/C''$ denote the set of all cosets of C'' in $\boldsymbol{v} + C'$, that is, $(\boldsymbol{v} + C')/C'' = \{\boldsymbol{v} + \boldsymbol{u} + C'' : \boldsymbol{u} \in C' \setminus C''\}$. Then

$$|(\boldsymbol{v} + C')/C''| = 2^{k'-k''} \quad \text{and} \quad \boldsymbol{v} + C' = \bigcup_{E \in (\boldsymbol{v}+C')/C''} E,$$

where k' and k'' are the dimensions of C' and C''. We also call the weight distribution of zero neighbors in $E \in (\boldsymbol{v} + C')/C''$ the local weight subdistribution for E. The following theorem gives an invariance property for cosets in $(\boldsymbol{v} + C')/C''$.

Theorem 6: For $E_1, E_2 \in (\boldsymbol{v} + C')/C''$, the local weight subdistribution for E_1 and that for E_2 are the same if there exists $\pi \in \{\rho : \rho\boldsymbol{v} \in \boldsymbol{v} + C', \rho \in \mathrm{Aut}(C) \cap \mathrm{Aut}(C')\}$ such that $\pi[E_1] = E_2$.

We consider partitioning $(\boldsymbol{v} + C')/C''$ into equivalence classes. Permutations which are used to partition cosets into equivalence classes are presented in the following lemma.

Lemma 5: For a coset $\boldsymbol{v} + C' \in C/C'$,

$$\begin{aligned} &\{\pi : \pi[E] \in (\boldsymbol{v} + C')/C'' \text{ for any } E \in (\boldsymbol{v} + C')/C''\} \\ &\quad = \{\rho : \rho\boldsymbol{v} \in \boldsymbol{v} + C',\ \rho \in \mathrm{Aut}(C) \cap \mathrm{Aut}(C') \cap \mathrm{Aut}(C'')\}. \end{aligned} \tag{29}$$

Proof: Let $\pi \in \{\rho : \rho\boldsymbol{v} \in \boldsymbol{v} + C',\ \rho \in \mathrm{Aut}(C) \cap \mathrm{Aut}(C') \cap \mathrm{Aut}(C'')\}$. For a coset $\boldsymbol{v} + \boldsymbol{v}_1 + C'' \in (\boldsymbol{v} + C')/C''$, suppose that $\pi\boldsymbol{v} = \boldsymbol{v} + \boldsymbol{v}_2, \boldsymbol{v}_2 \in C'$ and $\pi\boldsymbol{v}_1 = \boldsymbol{v}_3 \in C'$. For any codeword $\boldsymbol{v} + \boldsymbol{v}_1 + \boldsymbol{u}_1 \in \boldsymbol{v} + \boldsymbol{v}_1 + C''$, $\boldsymbol{u}_1 \in C''$,

$$\begin{aligned} \pi(\boldsymbol{v} + \boldsymbol{v}_1 + \boldsymbol{u}_1) &= \pi\boldsymbol{v} + \pi\boldsymbol{v}_1 + \pi\boldsymbol{u}_1 \\ &= \boldsymbol{v} + \boldsymbol{v}_2 + \boldsymbol{v}_3 + \boldsymbol{u}_2,\ \pi\boldsymbol{u}_1 = \boldsymbol{u}_2 \in C'' \\ &= \boldsymbol{v} + (\boldsymbol{v}_2 + \boldsymbol{v}_3) + \boldsymbol{u}_2 \\ &\in \boldsymbol{v} + (\boldsymbol{v}_2 + \boldsymbol{v}_3) + C''. \end{aligned} \tag{30}$$

Thus, $\pi[\boldsymbol{v}+\boldsymbol{v}_1+C''] = \boldsymbol{v}+(\boldsymbol{v}_2+\boldsymbol{v}_3)+C'' \in (\boldsymbol{v}+C')/C''$. Therefore, $\{\pi : \pi[E] \in (\boldsymbol{v}+C')/C''$ for any $E \in (\boldsymbol{v}+C')/C''\} \supseteq \{\ \rho : \rho\boldsymbol{v} \in \boldsymbol{v}+C',\ \rho \in \mathrm{Aut}(C) \cap \mathrm{Aut}(C') \cap \mathrm{Aut}(C'')\}$.

Let $\pi \in \{\rho : \rho[E] \in (\boldsymbol{v}+C')/C''$ for any $E \in (\boldsymbol{v}+C')/C''\}$. For any codeword $\boldsymbol{v}+\boldsymbol{v}_1 \in \boldsymbol{v}+C'$, $\boldsymbol{v}+\boldsymbol{v}_1$ must be in either coset in $(\boldsymbol{v}+C')/C''$, thus, $\pi(\boldsymbol{v}+\boldsymbol{v}_1) \in \boldsymbol{v}+C''$ and $\pi \in \mathrm{Aut}(C)$. For $\boldsymbol{v}+\boldsymbol{v}_1+C'' \in (\boldsymbol{v}+C')/C''$, let $\boldsymbol{v}+\boldsymbol{v}_1+\boldsymbol{u}_1, \boldsymbol{v}+\boldsymbol{v}_1+\boldsymbol{u}_2 \in \boldsymbol{v}+\boldsymbol{v}_1+C''$. $\pi(\boldsymbol{v}+\boldsymbol{v}_1+\boldsymbol{u}_1) = \pi\boldsymbol{v}+\pi\boldsymbol{v}_1+\pi\boldsymbol{u}_1$ and $\pi(\boldsymbol{v}+\boldsymbol{v}_1+\boldsymbol{u}_2) = \pi\boldsymbol{v}+\pi\boldsymbol{v}_1+\pi\boldsymbol{u}_2$ must be in the same coset of $\boldsymbol{v}+\boldsymbol{v}_2+C''$. Hence, $\pi \in \mathrm{Aut}(C')$ and $\pi \in \mathrm{Aut}(C'')$. Therefore, $\{\pi : \pi[E] \in (\boldsymbol{v}+C')/C''$ for any $E \in (\boldsymbol{v}+C')/C''\} \subseteq \{\rho : \rho\boldsymbol{v} \in \boldsymbol{v}+C',\ \rho \in \mathrm{Aut}(C) \cap \mathrm{Aut}(C') \cap \mathrm{Aut}(C'')\}$. $\square$

In order to partition cosets into equivalence classes, we should use permutations presented in Lemma 5. Although $\mathrm{Aut}(C)$, $\mathrm{Aut}(C')$, and $\mathrm{Aut}(C'')$ are known, we should obtain permutations π that satisfy $\pi\boldsymbol{v} \in \boldsymbol{v}+C'$. However, finding such permutations is difficult in general. We show that we have a clue as to finding the permutations for a coset of a Reed-Muller code.

Let $\mathrm{RM}(r,m)$ denote the r-th order Reed-Muller code of length 2^m. For instance, we consider the case of the $(256, 93)$ third-order Reed-Muller code, denoted by $\mathrm{RM}(3,8)$. The 32 equivalence classes of $\mathrm{RM}(3,8)/\mathrm{RM}(2,8)$ are presented in [11]. We choose $\mathrm{RM}(1,8)$ as a subcode of $\mathrm{RM}(2,8)$. Then the general affine group [14] is a subgroup of $\mathrm{Aut}(\mathrm{RM}(3,8)) \cap \mathrm{Aut}(\mathrm{RM}(2,8)) \cap \mathrm{Aut}(\mathrm{RM}(1,8))$. For each coset in $\mathrm{RM}(3,8)/\mathrm{RM}(2,8)$, the estimated time for computing the local weight subdistribution is about 54 days using the proposed algorithm in Section III and its improvement in Section IV-A. The total estimated time is about 1700 days. To compute the local weight distribution of $\mathrm{RM}(3,8)$ in practical time, we should find the permutations π that satisfy $\pi\boldsymbol{v} \in \boldsymbol{v}+\mathrm{RM}(2,8)$ for each 32 representative cosets $\boldsymbol{v}+\mathrm{RM}(2,8)$ in $\mathrm{RM}(3,8)/\mathrm{RM}(2,8)$. For instance, one of the representative cosets is $x_1x_2x_3 + \mathrm{RM}(2,8)$ (we use a Boolean polynomial representation for Reed-Muller codewords [14]). For this coset, the permutations that does not permute x_1, x_2, x_3 but permute the other variables $x_4, x_5, \ldots, x_8$ are candidate permutations that satisfy $\pi(x_1x_2x_3) \in x_1x_2x_3 + RM(2,8)$. Such permutations do exist in the general affine group. If we could find more than 50 permutations π that satisfy $\pi\boldsymbol{v} \in \boldsymbol{v}+\mathrm{RM}(2,8)$ for each 32 representative cosets, the local weight distribution of $\mathrm{RM}(3,8)$ may be computable.

V. Theoretical Approach to Determine Local Weight Distribution

In this section, we consider relations between the local weight distributions of a binary linear code, its extended code, and its even weight subcode.

A. General Relation

Consider a 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 $\boldsymbol{v}^{(\mathrm{ex})}$ be the corresponding extended codeword in C_{ex}. We define a *decomposable* codeword (see Fig. 4).

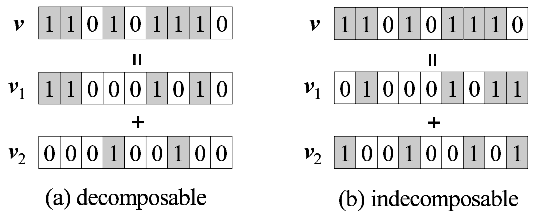

Fig. 4. Examples of a decomposable codeword and an indecomposable codeword.

Definition 5 (Decomposable codeword): $\boldsymbol{v} \in C$ is called *decomposable* if $\boldsymbol{v}$ can be represented as $\boldsymbol{v} = \boldsymbol{v}_1 + \boldsymbol{v}_2$ where $\boldsymbol{v}_1, \boldsymbol{v}_2 \in C$ and $\mathrm{Supp}(\boldsymbol{v}_1) \cap \mathrm{Supp}(\boldsymbol{v}_2) = \emptyset$.

From Lemma 1, $\boldsymbol{v}$ is not a zero neighbor if and only if $\boldsymbol{v}$ is decomposable. For even weight codewords, we introduce an *only-odd-decomposable* codeword and an *even-decomposable* codeword.

Definition 6: 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 decompositions of $\boldsymbol{v}$ are of the form $\boldsymbol{v}_1+\boldsymbol{v}_2$ with the odd weight codewords $\boldsymbol{v}_1, \boldsymbol{v}_2 \in C$. 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. Then $\boldsymbol{v}^{(\mathrm{ex})}$ is decomposable into $\boldsymbol{v}_1^{(\mathrm{ex})} + \boldsymbol{v}_2^{(\mathrm{ex})}$. On the other hand, for an only-odd decomposable codeword $\boldsymbol{v} = \boldsymbol{v}_1+\boldsymbol{v}_2$, $\boldsymbol{v}^{(\mathrm{ex})}$ is not decomposable into $\boldsymbol{v}_1^{(\mathrm{ex})} + \boldsymbol{v}_2^{(\mathrm{ex})}$ for any decompositions.

The relation between C and C_{ex} with respect to zero neighborship is given in the following theorem, which is also summarized in Table I.

Theorem 7: 1) For a zero neighbor $\boldsymbol{v}$ in C, $\boldsymbol{v}^{(\mathrm{ex})}$ is a zero neighbor in C_{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, $\boldsymbol{v}^{(\mathrm{ex})}$ is not a zero neighbor in C_{ex}.
   b) When $\mathrm{wt}(\boldsymbol{v})$ is even, $\boldsymbol{v}^{(\mathrm{ex})}$ is a zero neighbor in C_{ex} if and only if $\boldsymbol{v}$ is only-odd-decomposable in C.

Proof: 1) Suppose that $\boldsymbol{v}^{(\mathrm{ex})}$ is not a zero neighbor in C_{ex}. Then $\boldsymbol{v}^{(\mathrm{ex})}$ is decomposable into $\boldsymbol{v}_1^{(\mathrm{ex})}+\boldsymbol{v}_2^{(\mathrm{ex})}$. Hence, $\boldsymbol{v}$ is decomposable into $\boldsymbol{v}_1+\boldsymbol{v}_2$, contradicting the indecomposability of $\boldsymbol{v}$.

2) Suppose that $\boldsymbol{v}$ is decomposed into $\boldsymbol{v} = \boldsymbol{v}_1+\boldsymbol{v}_2$. a) Since $\mathrm{wt}(\boldsymbol{v})$ is odd, the sum of the parity bits in $\boldsymbol{v}_1^{(\mathrm{ex})}$ and $\boldsymbol{v}_2^{(\mathrm{ex})}$ is one. Also, the parity bit in $\boldsymbol{v}^{(\mathrm{ex})}$ is one. Then $\boldsymbol{v}^{(\mathrm{ex})}$ is decomposable into $\boldsymbol{v}_1^{(\mathrm{ex})} + \boldsymbol{v}_2^{(\mathrm{ex})}$, and $\boldsymbol{v}^{(\mathrm{ex})}$ is not a zero neighbor in C_{ex}. b) Since $\mathrm{wt}(\boldsymbol{v})$ is even, the parity bit in $\boldsymbol{v}^{(\mathrm{ex})}$ is zero. (If part) Suppose that $\boldsymbol{v}^{(\mathrm{ex})}$ is not a zero neighbor in C_{ex}. Then there exists a decomposition $\boldsymbol{v}^{(\mathrm{ex})} = \boldsymbol{v}_1^{(\mathrm{ex})}+\boldsymbol{v}_2^{(\mathrm{ex})}$. Because the parity bit in $\boldsymbol{v}^{(\mathrm{ex})}$ is zero, the parity bits in $\boldsymbol{v}_1^{(\mathrm{ex})}$ and $\boldsymbol{v}_2^{(\mathrm{ex})}$ must be zero. Thus, $\boldsymbol{v}$ is even-decomposable into $\boldsymbol{v}_1+\boldsymbol{v}_2$, contradicting the assumption that $\boldsymbol{v}$ is only-odd-decomposable. (Only if part) Suppose that $\boldsymbol{v}$ is even-decomposable. Then there is a decomposition such that the

TABLE I

ZERO NEIGHBORSHIP OF $\boldsymbol{v}$ IN A LINEAR BLOCK CODE, $\boldsymbol{v}^{(\mathrm{ex})}$ IN ITS EXTENDED CODE, AND $\boldsymbol{v}$ IN ITS EVEN WEIGHT SUBCODE.

$\boldsymbol{v}$ in C			$\boldsymbol{v}^{(\mathrm{ex})}$ in C_{ex}		$\boldsymbol{v}$ in C_{even}	
Zero neighborship	Weight	Decomposability	Zero neighborship	Theorem 7	Zero neighborship	Theorem 10
Yes	Odd	Not decomposable	Yes	1)	N/A	N/A
	Even				Yes	1)
No	Odd	Decomposable	No	2a)	N/A	N/A
	Even	Only-odd-decomposable	Yes	2b)	Yes	2)
	Even	Even-decomposable	No		No	

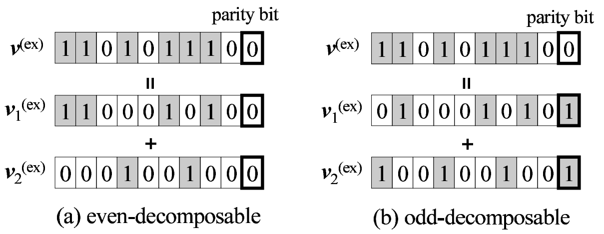

Fig. 5. Examples of an even-decomposable codeword and an odd-decomposable codeword mentioned in the proof of Theorem 7-2b).

parity bits in both $\boldsymbol{v}_1^{(\mathrm{ex})}$ and $\boldsymbol{v}_2^{(\mathrm{ex})}$ are zero. For such the decomposition, $\boldsymbol{v}^{(\mathrm{ex})}$ is decomposable into $\boldsymbol{v}_1^{(\mathrm{ex})} + \boldsymbol{v}_2^{(\mathrm{ex})}$, and $\boldsymbol{v}^{(\mathrm{ex})}$ is not a zero neighbor in C_{ex}. (see Fig. 5). □

From 2b) of Theorem 7, there may be codewords that are not zero neighbors in C although their extended codewords are zero neighbors in C_{ex}. Such codewords are the only-odd decomposable codewords. For investigating relations of local weight distributions between a code and its extended code, only-odd decomposable codewords are important.

The following theorem is a direct consequence of Theorem 7.

Theorem 8: For a code C of length n,

$$L_{2i}(C_{\mathrm{ex}}) = L_{2i-1}(C) + L_{2i}(C) + N_{2i}(C), \ 0 \le i \le n/2, \tag{31}$$

where $N_j(C)$ is the number of only-odd decomposable codewords with weight j in C.

From Theorem 8, if no only-odd decomposable codeword exists in C, then the local weight distributions of C_{ex} are obtained from that of C. Next, we give a useful sufficient condition under which no only-odd-decomposable codeword exists.

Theorem 9: If all the weights of codewords in C_{ex} are multiples of four, no only-odd-decomposable codeword exists in C.

Proof: If $\boldsymbol{v} \in C$ is an only-odd-decomposable codeword and is decomposed into $\boldsymbol{v}_1 + \boldsymbol{v}_2$, the weights of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ can be represented as $\mathrm{wt}(\boldsymbol{v}_1) = 4i - 1$ and $\mathrm{wt}(\boldsymbol{v}_2) = 4j - 1$ where i and j are integers. Then $\mathrm{wt}(\boldsymbol{v}) = \mathrm{wt}(\boldsymbol{v}_1 + \boldsymbol{v}_2) = \mathrm{wt}(\boldsymbol{v}_1) + \mathrm{wt}(\boldsymbol{v}_2) = (4i-1) + (4j-1) = 4i + 4j - 2$, contradicting the fact that $\mathrm{wt}(\boldsymbol{v})$ is a multiple of four. □

For example, all the weights of codewords in the $(128, k)$ extended primitive BCH code with $k \le 57$ are multiples of four. The parameters of the Reed-Muller codes with which all the weights of codewords are multiples of four are given by Corollary 13 of Chapter 15 in [14]. From the corollary, the third-order Reed-Muller codes of length $n \ge 128$ have only codewords whose weights are multiples of four.

Although the local weight distribution of C_{ex} for these codes can be obtained from that of C by using Theorem 8, in order to obtain the local weight distribution of C from that of C_{ex}, we need to know the number of zero neighbors with parity bit one. In Section V-B, 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 to that between C and C_{ex} holds between C and C_{even}. This relation is given in Theorem 10 without proof (see Table I).

Theorem 10: 1) For an even weight zero neighbor $\boldsymbol{v}$ in C, $\boldsymbol{v}$ is a zero neighbor in C_{even}.

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

From Theorem 10, we derive Theorem 11.

Theorem 11: For a code C of length n,

$$L_{2i}(C_{\mathrm{even}}) = L_{2i}(C) + N_{2i}(C), \quad 0 \le i \le n/2. \tag{32}$$

B. Relation for Transitive Invariant Extended Codes

A transitive invariant code is a 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 [18]. The extended primitive BCH codes and Reed-Muller codes are transitive invariant codes. For a transitive invariant C_{ex}, a relation between the global weight distributions of C and C_{ex} is presented in Theorem 8.15 in [18]. A similar relation holds for local weight distribution.

Lemma 6: If C_{ex} is a transitive invariant code of length $n+1$, the number of zero neighbors with parity bit one is $\frac{w}{n+1} L_w(C_{\mathrm{ex}})$.

Proof: This lemma can be proved in a similar way as the proof of Theorem 8.15. Arrange all zero neighbors with weight w in a column. Next, interchange the j-th column and the last column, which is the parity bit column, for all these codewords with the permutation. All the resulting codewords have weight w and must be the same as the original set of codewords. Thus, the number of ones in the j-th column and that in the last column are the same. Denote this number l_w, which is the same as the number of zero neighbors of weight w with parity bit one. Then the number of total ones in the original set of codewords is $(n+1)\, l_w$, or $L_w(C_{\rm ex})$ times the weight w. Thus, $(n+1)\, l_w = wL_w(C_{\rm ex})$, and $l_w = \frac{w}{n+1} L_w(C_{\rm ex})$. □

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 7 and Lemma 6.

Theorem 12: If $C_{\rm ex}$ is a transitive invariant code of length $n+1$,

$$L_w(C) = \begin{cases} \dfrac{w+1}{n+1} L_{w+1}(C_{\rm ex}), & \text{for odd } w, \\ \dfrac{n+1-w}{n+1} L_w(C_{\rm ex}) - N_w(C), & \text{for even } w. \end{cases} \tag{33}$$

If there is no only-odd-decomposable codeword in a transitive invariant code C, then we have:

$$L_w(C) = \frac{n+1-w}{n+1} L_w(C_{\rm ex}), \quad \text{for even } w. \tag{34}$$

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 (33) and (34) in Theorem 12. After computing the local weight distribution of C, that of $C_{\rm even}$ can be obtained by using Theorem 11.

VI. Obtained Local Weight Distributions

Using the algorithm described in Sections III and IV, we compute the local weight distributions of extended primitive BCH codes and Reed-Muller codes.

The local weight distributions of the $(128, k)$ extended primitive BCH codes for $k \le 50$ are obtained and shown in Table II. It took about 440 hours (CPU time) to compute the distribution of the $(128, 50)$ code with a 1.6 GHz Opteron processor. In this case, the $(128, 29)$ code is used as the subcode, and it took only one minute to partition cosets into equivalence classes.

We also apply the proposed algorithm to the third-order Reed-Muller code of length 128. We use the second-order Reed-Muller code as the subcode. The representative codewords of cosets for this case are presented in [11]. A method to obtain the number of equivalent cosets to the representative cosets are presented in [19]. Thus, the process of obtaining the representative cosets and the number of equivalent cosets are different from that for extended primitive BCH codes. Note that the computing time for this process is vanishingly small.

The local weight distributions of the $(127, k)$ primitive BCH codes for $k \le 50$ and the punctured third-order Reed-Muller code of length 127 are obtained from those of the corresponding extended codes by using Theorems 11 as shown in Table IV. If we could obtain the local weight distributions of the $(128, 57)$ extended primitive BCH code and the third-order Reed-Muller codes of length 256 and 512, the local weigh distributions of the $(127, 57)$ primitive BCH code and the punctured third-order Reed-Muller codes of length 256 and 512 could be determined by using Theorems 11 and 12.

The local weight distributions of even weight subcodes of the $(127, k)$ primitive BCH codes for $k = 36, 43, 50$ and the punctured third-order Reed-Muller code of length 127 are obtained from those of the corresponding original codes shown in Table IV by using Theorem 11. Note that $N_i(C)$ in Theorem 11 is equal to zero for all i in these cases.

VII. Conclusions

In this paper, some methods to determine the local weight distribution of binary linear block codes have been studied.

For the computational approach, an algorithm for computing the local weight distribution using the automorphism group of a code has proposed. In this algorithm, a code is considered a set of cosets of a linear subcode. The set of cosets are partitioned into equivalence classes with the invariance property for zero neighborship under a group of permutations. The local weight distribution is obtained by computing the local weight subdistributions for each representative coset. The algorithm can be applied to codes closed under a group of permutations. Extended primitive BCH codes are closed under the affine group and Reed-Muller codes are closed under the general affine group. The algorithm is applied to these codes, and we determined the local weight distributions for some of these codes.

For the theoretical approach, relations between the local weight distribution of a code, its extended code, and its even weight subcode were studied. Only-odd decomposable codewords are key codewords when we intend to determine the local weight distribution of an extended code from that of the original code, or vice versa. A sufficient condition is derived under which no only-odd decomposable codeword exists. The local weight distributions for some of primitive BCH codes, and punctured Reed-Muller codes, and their even weight subcodes are determined from those of extended primitive BCH codes and Reed-Muller codes.

References

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

[2] 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.

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

[4] A. Barg, "Complexity issues in coding theory," in V. Pless and W.C. Huffman, Eds. *Handbook of Coding Theory*, North-Holland, vol. 1, pp. 649–754, 1998.

[5] E.R. Berlekamp, "The technology of error-correcting codes," *Proc. IEEE*, vol. 68, no. 5, pp. 564-593, May 1980.

[6] Y. Borissov, N. Manev, and S. Nikova, "On the non-minimal codewords in binary Reed-Muller codes," Discrete Applied Mathematics, vol. 128, issue 1, pp. 65-74, May 2003.

[7] G.C. Clark, Jr. and J.B. Cain, *Error-Correction Coding for Digital Communications*, New York: Plenum, 1981.

[8] D. Divsalar and E. Biglieri, "Upper bounds to error probabilities of coded systems beyond the cutoff rate," *IEEE Trans. Inform. Theory*, vol. 51, no. 12, pp. 2011–2018, Dec. 2003.

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

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

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

TABLE II

LOCAL WEIGHT DISTRIBUTIONS OF THE $(128, k)$ EXTENDED PRIMITIVE BCH CODES.

$k=36$		$k=43$		$k=50$	
w	L_w	w	L_w	w	L_w
32	10 688	32	124 460	28	186 944
36	16 256	36	8 810 752	32	19 412 204
40	2 048 256	40	263 542 272	36	113 839 296
44	35 551 872	44	4 521 151 232	40	33 723 852 288
48	353 494 848	48	44 899 876 672	44	579 267 441 920
52	2 028 114 816	52	262 118 734 080	48	5 744 521 082 944
56	7 216 135 936	56	915 924 097 536	52	33 558 415 333 632
60	14 981 968 512	60	1 931 974 003 456	56	117 224 645 074 752
64	19 484 132 736	64	2 476 669 858 944	60	247 311 270 037 888
68	14 981 968 512	68	1 931 944 645 120	64	316 973 812 770 944
72	7 216 127 808	72	915 728 180 224	68	247 074 613 401 728
76	2 028 114 816	76	261 375 217 152	72	115 408 474 548 096
80	348 203 520	80	43 168 588 288	76	25 844 517 328 896
84	35 551 872	84	2 464 897 280		
88	2 048 256				

TABLE III

THE LOCAL WEIGHT DISTRIBUTIONS OF THE $(128, 64)$ THIRD-ORDER REED-MULLER CODE.

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	551 965 599 940 608
44	9 482 818 340 782 080
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

TABLE IV

THE LOCAL WEIGHT DISTRIBUTIONS OF THE $(127, k)$ PRIMITIVE BCH CODES.

$k=36$		$k=43$		$k=50$	
w	L_w	w	L_w	w	L_w
31	2 667	31	31 115	27	40 894
32	8 001	32	93 345	28	146 050
35	4 572	35	2 478 024	31	4 853 051
36	11 684	36	6 332 728	32	14 559 153
39	640 080	39	82 356 960	35	310 454 802
40	1 408 176	40	181 185 312	36	793 384 494
43	12 220 956	43	1 554 145 736	39	10 538 703 840
44	23 330 916	44	2 967 005 496	40	23 185 148 448
47	132 560 568	47	16 837 453 752	43	199 123 183 160
48	220 934 280	48	28 062 422 920	44	380 144 258 760
51	823 921 644	51	106 485 735 720	47	2 154 195 406 104
52	1 204 193 172	52	155 632 998 360	48	3 590 325 676 840
55	3 157 059 472	55	400 716 792 672	51	13 633 106 229 288
56	4 059 076 464	56	515 207 304 864	52	19 925 309 104 344
59	7 022 797 740	59	905 612 814 120	55	51 285 782 220 204
60	7 959 170 772	60	1 026 361 189 336	56	65 938 862 854 548
63	9 742 066 368	63	1 238 334 929 472	59	115 927 157 830 260
64	9 742 066 368	64	1 238 334 929 472	60	131 384 112 207 628
67	7 959 170 772	67	1 026 345 592 720	63	158 486 906 385 472
68	7 022 797 740	68	905 599 052 400	64	158 486 906 385 472
71	4 059 071 892	71	515 097 101 376	67	131 258 388 369 668
72	3 157 055 916	72	400 631 078 848	68	115 816 225 032 060
75	1 204 193 172	75	155 191 535 184	71	64 917 266 933 304
76	823 921 644	76	106 183 681 968	72	50 491 207 614 792
79	217 627 200	79	26 980 367 680	75	15 345 182 164 032
80	130 576 320	80	16 188 220 608	76	10 499 335 164 864
83	23 330 916	83	1 617 588 840		
84	12 220 956	84	847 308 440		
87	1 408 176				
88	640 080				

TABLE V

THE LOCAL WEIGHT DISTRIBUTION OF THE $(127, 64)$ PUNCTURED THIRD-ORDER REED-MULLER CODE.

w	L_w
15	11 811
16	82 677
23	13 889 736
24	60 188 856
27	684 345 088
28	2 444 089 600
31	77 893 639 488
32	233 680 918 464
35	5 097 898 213 632
36	13 027 962 101 504
39	172 489 249 981 440
40	379 476 349 959 168
43	3 259 718 804 643 840
44	6 223 099 536 138 240
47	35 130 035 853 803 520
48	58 550 059 756 339 200
51	218 602 288 622 075 904
52	319 495 652 601 495 552
55	766 899 891 905 495 040
56	986 014 146 735 636 480
59	1 306 771 964 441 395 200
60	1 481 008 226 366 914 560
63	258 664 522 171 023 360
64	258 664 522 171 023 360

[12] T.-Y. Hwang, "Decoding linear block codes for minimizing word error rate," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 733–737, Nov. 1979.

[13] 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.

[14] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1977.

[15] J. Massey, "Minimal codewords and secret sharing," in *Proc. 6th Joint Swedish-Russian Workshop of Information Theory*, pp. 246-249, 1993.

[16] M. Mohri, and M. Morii, "On computing the local distance profile of binary cyclic codes," in *Proc. the 2002 International Symposium on Information Theory and its Applications (ISITA2002)*, pp. 415–418, Oct. 2002.

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

[18] W.W. Peterson and E.J. Weldon, Jr., *Error-Correcting Codes, 2nd Edition*, MIT Press, 1972.

[19] 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.

Kenji Yasunaga received the B.E. degree in information and computer sciences in 2003 and M.Sc. degree in information science and technology in 2005, both from Osaka University, Osaka, Japan. Currently, he is working towards the Ph.D. degree at the same university.

His research itersets include coding theory.

Toru Fujiwara (S'83–M'86) was born in Wakayama, Japan, on June 18, 1958. He received the B.E., M.E., and Ph.D. degrees in information and computer sciences from Osaka University, Toyonaka, Osaka, Japan, in 1981, 1983, and 1986, respectively.

In 1986, he jointed the faculty of Osaka University. During 1989–1990, he was on leave as a Post Doctoral Fellow in the Department of Electrical Engineering, University of Hawaii, Honolulu. From 1992 to 1997, he was an Associate Professor at the Department of Information and Computer Sciences, Osaka University. From 1997 to 2003, he was a Professor in the Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University. He is now with the Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University. From 1998 to 2000, he was simultaneously a Professor in the Graduate School of Information Science and Technology, Nara, Japan, and at Osaka University. His current research interests include coding theory and cryptography.

Dr. Fujiwara is the chair person of IEEE Information Theory Society Japan Chapter. He is a Member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan, the Information Processing Society of Japan (IPSJ), and the Association for Computing Machinery (ACM).