An Algorithm for Computing the Local Weight Distribution of Binary Linear Codes Closed under a Group of Permutations

Kenji YASUNAGA and Toru FUJIWARA

Osaka University, Osaka, Japan

ISITA2004, Parma, Italy

Outline

- Local Weight Distribution
- Algorithm for Cyclic Codes
- Proposed Algorithm
- New Obtained Local Weight Distributions
- Summary

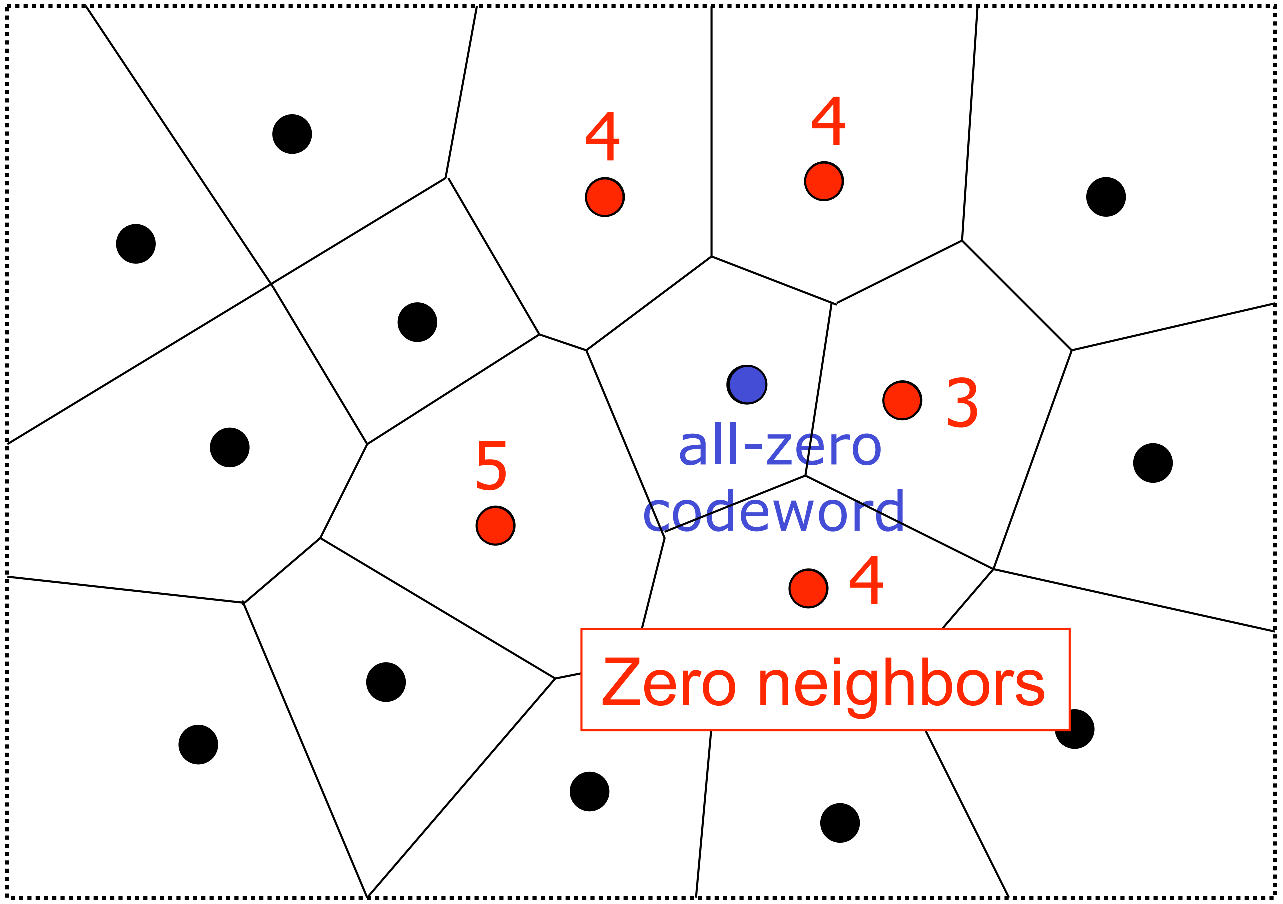
Local Weight Distribution (LWD)

[illegible]

Zero neighbors are codewords whose Voronoi regions share a facet with that of the all-zero codeword.

The LWD is the weight distribution of zero neighbors.

Code C on $\mathbf{R}^n$



The LWD of C

weight	*Number of codewords*
3	1
4	3
5	1

What for We Obtain LWD?

- The LWD is valuable for the error performance analysis of codes.
  - Usually, the error probability for soft decision decoding on AWGN channel is given by the union bound (upper bound).
    - To compute the union bound, the weight distribution is used.
    - We can obtain the tighter upper bound than the usual union bound when using the LWD instead of the weight distribution.

Known Results for Obtaining LWD

- Hamming codes and second-order Reed-Muller codes
  - The formulas for the LWDs are known[6].
- The other codes
  - Checking every codewords whether it is a zero neighbor or not.

⇨ Only computable for small codes.

An algorithm with reduced complexity for cyclic codes is proposed by Mohri and Morii[4].

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

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

Algorithm for Cyclic Codes [4] (1/2)

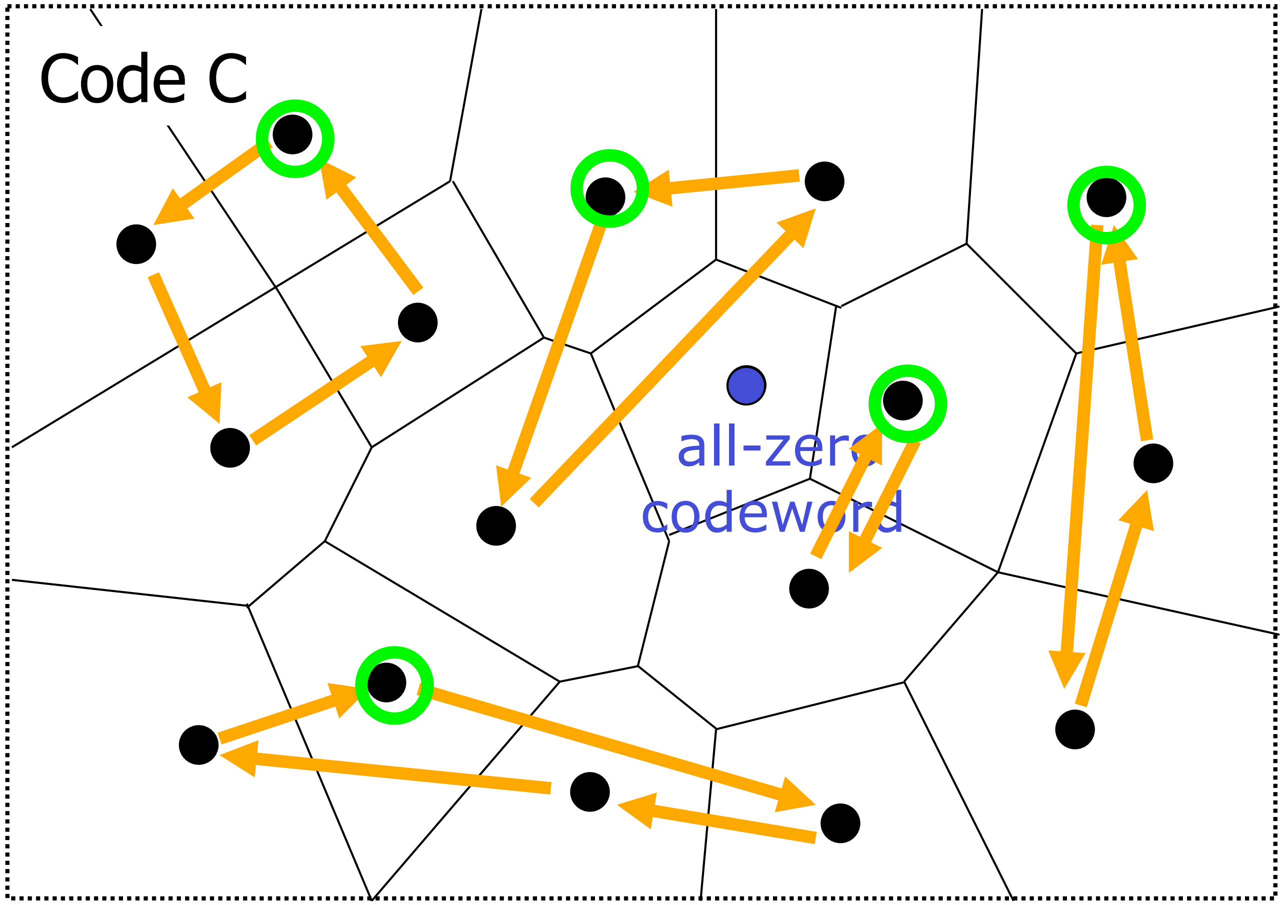
- Invariance property :

$$\boxed{\begin{array}{c}\text{A codeword } \boldsymbol{v} \text{ is a zero neighbor.}\\ \Updownarrow \\ \text{Any cyclic permutation of } \boldsymbol{v} \text{ is a zero neighbor.}\end{array}}$$

- The number of codewords that need to be checked can be reduced to use this invariance property.

Algorithm for Cyclic Codes [4] (2/2)

- Partition all codewords into sets of codewords with cyclic permutations.



Only each of these representative codewords needs to be checked whether it is a zero neighbor or not.

The number of codewords that need to be checked can be reduced.

Proposed Algorithm

- Motivation
- Our idea
- Complexity
- Codes for which the algorithm works effectively

Motivation

- The invariance property can be generalized to that for a group of permutations [2].

A codeword v is a zero neighbor.

$\Updownarrow$

Any permutation* of v is a zero neighbor.

* Under this permutation the code must be closed.

We consider using this generalized invariance property.

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

Advantages/Disadvantages to Use the Invariance Property

- Advantage
  - The number of codewords that need to be checked can be smaller.
    - The extended primitive BCH codes are closed under the affine group of permutations.
      The number of permutations in the affine group is n(n-1), which is larger than that in cyclic permutations n.
- Disadvantage
  - For such groups of permutations, no efficient way is known to generate the representatives.

Our Idea

- To overcome the disadvantage,

$\Downarrow$

We consider using cosets of the subcodes and partitioning all cosets into sets of cosets.

- This idea is used in [7] for computing the weight distribution of extended primitive BCH codes.
- In this paper, we show the idea can be applied to LWD.

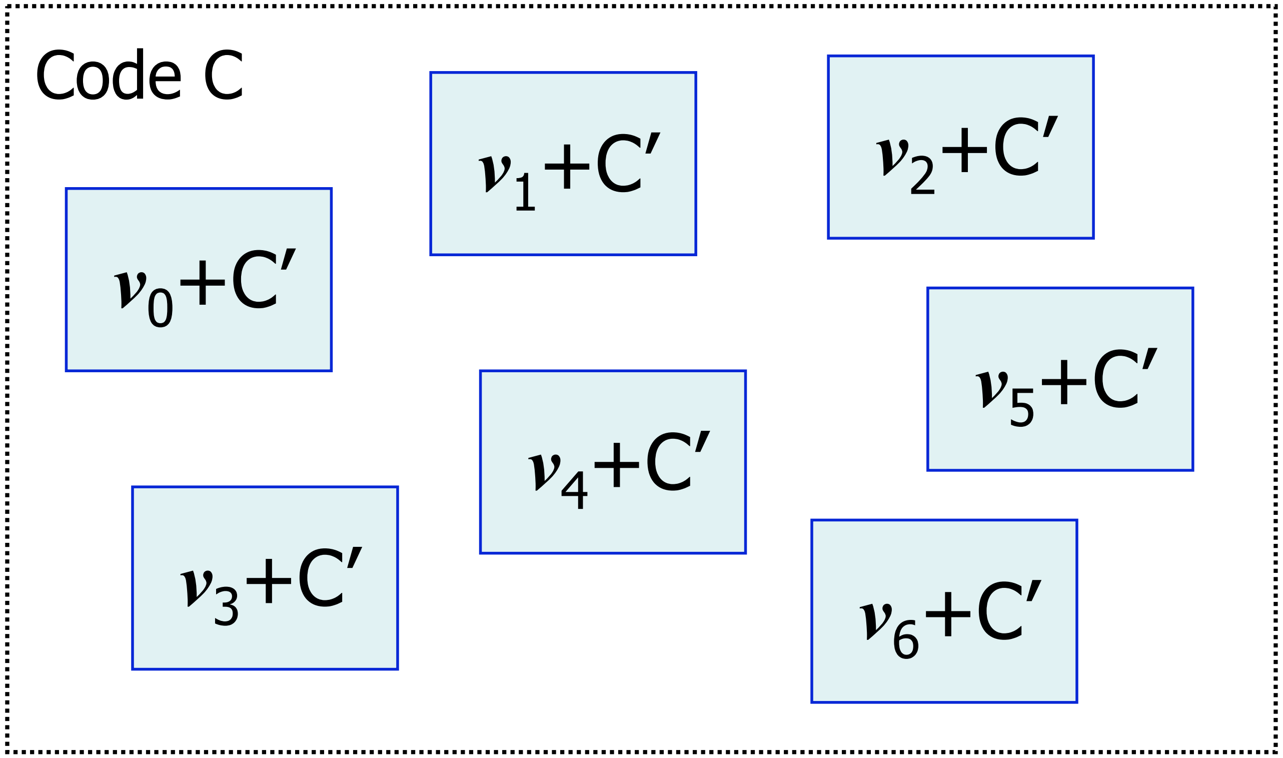
[7] **T.Fujiwara and T.Kasami**, "The weight distribution of (256,k) extended binary primitive BCH codes with k ≦ 63 and k ≧ 207," *IEICE Technical Report*, IT97-46, Sept. 1997.

Cosets of a Subcode C′

- C as a set of cosets of a subcode C′

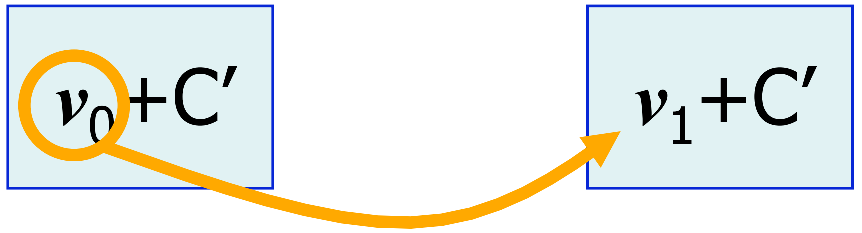
v_i+C′ : coset

v_i : coset leader



Property

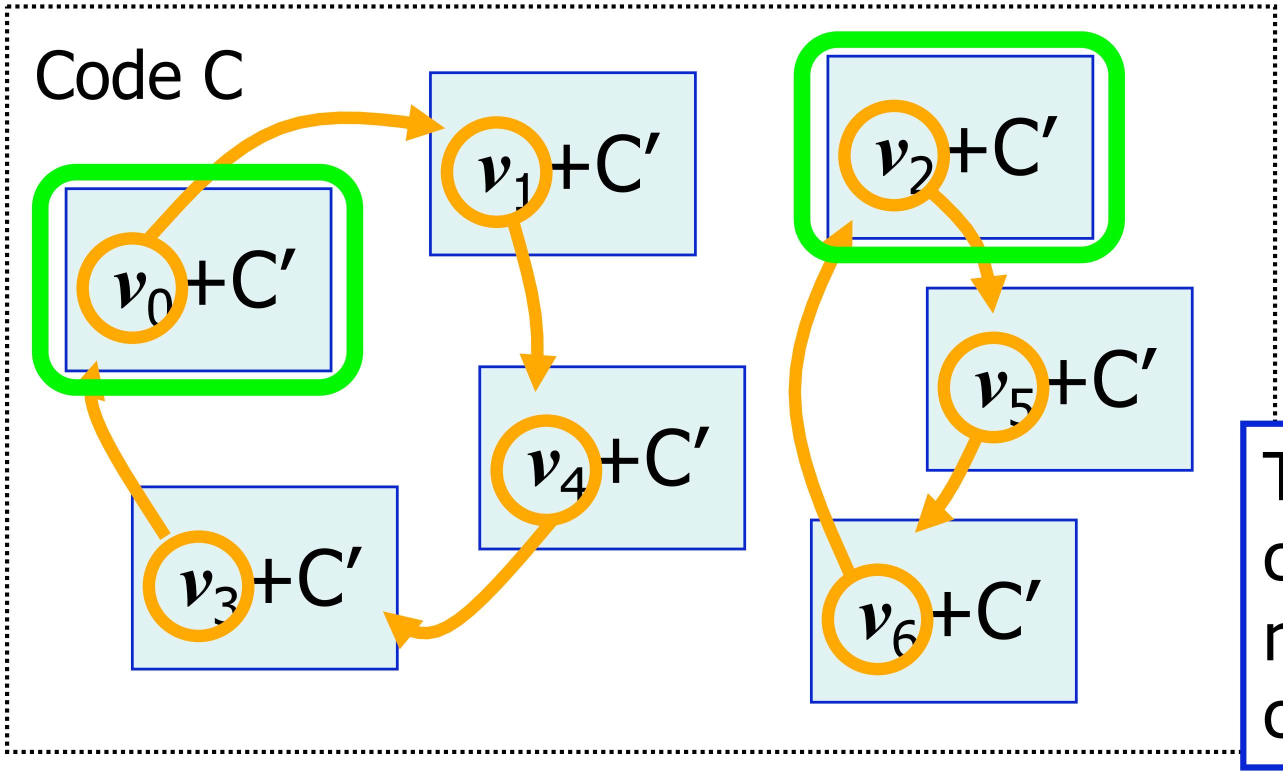
- If a coset leader is permuted into a vector in another coset,



the weight distributions of zero neighbors in these cosets are the same.

Partitioning the Cosets

- Partition all cosets into sets of cosets with a group of permutations.
  - We need to examine the permutations of each coset leaders.



The distributions of only these representative cosets need to be obtained.

The number of codewords that need to be checked can be reduced.

Complexity of the Proposed Algorithm

C : an (n, k) code

C′ : an (n, k′) subcode of C

P : a group of permutations

e : the number of representative cosets (obtained after partitioning cosets)

(a) Partitioning cosets

- $O(\ n(k-k')2^{k-k'}|P|\ )$

(b) Computing the distributions

- $O(\ n^2k \cdot e2^{k'}\ )$
  - Complexity of checking one codeword is $O(n^2k)$.
  - The number of codewords need to be checked is $e2^{k'}$.

⇨ Complexity varies with the selection of C′.

Selection of C′

- We should select C′ to minimize
  (a) partitioning cosets $O(\ n(k-k')2^{k-k'}|P|\ )$

  $$+$$

  (b) computing the distributions $O(\ n^2k \cdot e2^{k'}\ )$.
- Actually, we should care about the space complexity of part (a) (proportional to $2^{k-k'}$).

$$\Downarrow$$

Select C′ to minimize the number of codewords that need to be checked (= $e2^{k}$), as long as part (a) is computable and the complexity of (a) is relatively small.

Codes for which the algorithm works effectively

- The proposed algorithm can be applied to the codes that are closed under a group of permutations.

Code	Automorphism group
Cyclic codes	Cyclic permutations
Extended primitive BCH codes	Affine group
Reed-Muller codes	General linear group

New Obtained LWD

Code	(128,50) extended primitive BCH code	(128,64) Reed-Muller code
Subcode	(128,29) extended primitive BCH code	(128,29) Reed-Muller code
Complexity (CPU time)* (a) Partition cosets (b) Compute the distributions	(a) 3 hours (b) 440 hours	(a) Ignorable ** (b) 13 hours

* CPU: Opteron Processor 1.6GHz

** An efficient method is known.

New Obtained LWD

(128,50) extended primitive BCH code

weight	*Number of zero neighbors*
28	186,994
32	19,412,204
36	113,839,296
40	33,723,852,288
44	579,267,441,920
48	5,744,521,082,944
52	33,558,415,333,632
56	117,224,645,074,752
60	247,311,270,037,888
64	316,973,812,770,944
68	247,074,613,401,728
72	115,408,474,548,096
76	25,844,517,328,896

(128,64) Reed-Muller code

weight	*Number of zero neighbors*
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

Summary

- We proposed an algorithm for computing the LWD of codes closed under a group of permutations.
- We obtained some new LWDs with using the proposed algorithm.

Algorithm for Cyclic Codes

- Outline
  1. Generate (i) the cyclic representatives and (ii) the number of codewords that can be cyclic-permuted into each representative.
  2. Check each of representatives whether it is a zero neighbor or not.
- Complexity
  - About 1/n as much as that of brute force method.
    - The method for obtaining (i) and (ii) without complex computation is known.
    - The number of codewords that need to be checked is reduced by about 1/n.

Outline of the Proposed Algorithm

1. Select a subcode C′.
2. Partition all cosets of C′ into sets of cosets with a group of permutations, and obtain (i) the representative cosets and (ii) the number of cosets in each sets of cosets.
3. Compute the weight distributions of zero neighbors in each representative cosets.
4. Sum up all the distributions.

Comparison of the Complexity

- The algorithm for cyclic codes
  - About 1/n as that for the brute force method.
    - The number of permutations in cyclic permutations is n.
- The proposed algorithm
  - At most 1/n(n-1) as that for the brute force method.
    - The number of permutations in the affine group is n(n-1).

The complexity of the proposed algorithm is about 1/64 as much as that of the algorithm for cyclic codes in the case of n=128.

How tighter upper bound we obtain using LWD?

- We can obtain the tighter upper bound to use the LWD instead of the weight distribution.
- However, the bounds becomes only a little tighter. They are almost the same.
- Because the union bound depends mainly on the numbers of small weight codewords, and the numbers of small weight codewords in LWD are the same as the weight distribution.

Future works

- To reduce the complexity more.
  - Using the invariance property for codewords in each coset.
- To obtain the LWD of a code with using the LWD of its extended one.
  - Some relations hold for the LWD between an extended code and the original one (see section 6 of the paper).