$d(x_2, c_2) \le R-1$. As all pairs in the encoding scheme cover F_2^3 with covering radius 1, $d(x_1, \{c_1', c_1''\}) = 1$. Also in this case, $d(x, \{c', c''\}) = d(x_1, \{c_1', c_1''\}) + d(x_2, c_2) \le 1 + (R-1) = R$. Hence the covering radius of C' is at most R. In fact, it is easily seen that it is exactly R. □

The aforementioned encoding scheme can be applied to the first q coordinates of a code $C \subseteq F_4^q F_2^b$.

Theorem 2: Let $C \subseteq F_4^q F_2^b$ be a code of covering radius R that has n codewords. Then $K(b+3q, R) \le 2^q n$.

Proof: Immediate by the the proof of Theorem 1 and the previous discussion. □

The result can further be extended to other mixed codes.

The following 60 codewords cover $F_4^1 F_2^7$ with covering radius 1:

00000011	10000000	20000101	30001110
00001001	10001110	20001110	30010010
00001110	10010010	20011000	30010100
00010111	10010111	20011011	30100001
00100111	10011101	20100010	30101011
00110000	10100100	20110101	30101101
00111100	10101000	20111010	30110110
01000100	10110011	21000010	30111001
01001101	10111111	21010001	31000111
01010001	11001011	21010110	31001000
01011010	11010001	21101001	31010001
01100011	11011100	21101100	31011111
01101010	11100101	21101111	31100000
01110100	11100110	21110010	31110111
01111011	11111000	21111101	31111110.

Theorem 1 now gives $K(10, 1) \le 120$.

Simulated annealing [3] has been used in the search of the code. This method has earlier been adopted to construct good source codes, error-correcting codes and spherical codes [4].

The attempts to cover $F_4^1 F_2^7$ with 59 codewords left one word in the space uncovered. It can still be mentioned that all straight attacks to cover F_2^{10} with 120 codewords have been unsuccessful.

Acknowledgment

The author would like to thank the referees for some helpful comments.

References

[1] G. J. M. van Wee, "Improved sphere bounds on the covering radius of codes," *IEEE Trans. Inform. Theory*, vol. 34, no. 2, pp. 237–245, Mar. 1988.

[2] H. Hämäläinen and S. Rankinen, "Upper bounds for football pool problems and mixed covering codes," *J. Comb. Theory A*, to appear.

[3] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, May 1983.

[4] A. A. El Gamal, L. A. Hemachandra, I. Shperling, and V. K. Wei, "Using simulated annealing to design good codes," *IEEE Trans. Inform. Theory*, vol. IT-33, no. 1, pp. 116–123, Jan. 1987.

On the Problem of Finding Zero-Concurring Codewords

Alexander Vardy and Yair Be'ery

***Abstract*—Zero-concurring codewords disclose a certain structure of the code that may be employed for efficient soft-decision decoding and for designing dc-free codes. New methods for constructing sets of zero-concurring codewords are presented for several families of codes. For the general case an algorithmic solution of the problem is offered. A table of results obtained using the proposed techniques is supplied for all the primitive narrow-sense binary BCH codes of length up to 127.**

***Index Terms*—Zero-concurring codewords, soft-decision decoding, dc-free codes, BCH codes.**

I. Introduction

The concept of zero-concurring codewords was first defined by Be'ery and Snyders [1].[1] We propose a slightly more general definition.

Definition 1: Let C be a block code of length n over a finite field GF(q). A set of j independent codewords of C is called zero-concurring if in all the n positions at most one of the j codewords has a nonzero entry.

The idea of using some specific (symmetric) structure of the code in order to simplify its decoding may be found in a few works, for example [1]–[7]. The decoding algorithms employed in each case seem to be very different in their nature. It appears, however, that zero-concurring codewords disclose a certain inherent structure of the code, which can be exploited for its efficient soft decoding. Thus, Be'ery and Snyders [1] demonstrated that a considerable computational gain may be obtained by employing zero-concurring codewords in soft decision maximum likelihood decoding of low and mid-rate binary linear codes. Conway and Sloane [3] and Forney [4] provide similar results. The computational gain increases at least exponentially with the number J of zero-concurring codewords found in the code. An even greater computational gain may be achieved by introducing the so-called λ-concurring codewords (see [1], [6]). However, the problem of search for such codewords is out of the scope of this correspondence and will be dealt with elsewhere [18].

Yet another application of zero-concurring codewords for designing dc-free and run length constrained error-correcting codes was recently presented by Deng and Herro in [8]. The authors introduce a class of dc-free codes derived by finding a set of zero-concurring codewords in a binary linear block code. They also propose a construction method applicable to primitive BCH codes of composite block length n and designed distance d, provided that d is a factor of n. This paper significantly extends the results of Deng and Herro by exhibiting zero-con-

Manuscript received March 8, 1989; revised January 16, 1990. This work was presented in part at the 25th Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL 1987.

The authors are with the Department of Electronic Communications, Control and Computer Systems, Tel-Aviv University, Tel-Aviv 69978, Israel.

IEEE Log Number 9038864.

[1] The term "zero-concurring" stems from the word concurring, encountered in the literature on majority logic decoding [13]. Some of the subcodes of the Golay code discussed by Conway and Sloane [3] and the coset codes of Forney [4] are spanned by sets of zero-concurring codewords, though neither Conway and Sloane nor Forney provide a specific definition.

curring codewords in *all* the primitive BCH codes of length up to 127, several nonprimitive BCH codes and other binary codes. It should be pointed out that the problems of constructing a zero-concurring subset for efficient soft decoding and constructing a zero-concurring subset for the design of dc-free codes are slightly different. In the former case the main objective is to find a zero-concurring set with maximum possible cardinality while in the latter case we need a zero-concurring set covering all the length of the code but not necessarily of maximal cardinality. Nevertheless, the results derived in the sequel have immediate applications for both problems and possibly for other problems in coding theory.

Given a linear block code $C(n,k,d)$ of length n, dimension k, and minimum Hamming distance d, let J denote the cardinality of a maximal set of zero-concurring codewords of C. Evidently [1],

$$J \le \left[\frac{n}{d}\right] \tag{1}$$

where $[x]$ denotes the largest integer less than or equal to x. Although this bound is not always tight (see Section IV), it will be shown in the correspondence that the number of zero-concurring codewords in many codes meets the bound. In the next section we derive methods for constructing bound reaching zero-concurring sets in several frequently encountered families of codes. Various combinatorial algorithms for the search of zero-concurring codewords are presented in Section III. The efficiency of the proposed algorithms is demonstrated by the results obtained that are summarized in Table I of Section IV.

II. Constructions

Be'ery and Snyders [1] consider several methods for constructing zero-concurring codewords.

1) Let RM(r,m) be the binary Reed–Muller code of order r and length $n=2^m$. A set of $J=n/d=2^r$ zero-concurring codewords of RM may be obtained by performing elementary row operations on the $2^r \times 2^m$ matrix consisting of all the codewords $a_j(x)$, such that

$$a_j(x) = (1+x)^{2^{m-r}-1+j\cdot 2^{m-r}}, \quad \text{for } j=0,1,\cdots,2^r-1 \tag{2}$$

where $a(x)$ stands for

$$\sum_{i=1}^{n} a_i x^{i-1} \quad \text{and } (a_1,a_2,\cdots,a_n) \in \text{RM}(r,m).$$

2) Let $C(n,k)$ be a binary cyclic code and let $\delta \ge 2$ be a divisor of n, such that $x^\delta+1$ is a divisor of the parity check polynomial of C. Then the set

$$\{x^j a_\delta(x): j=0,1,\cdots,\delta-1\} \tag{3}$$

where

$$a_\delta(x) = 1+x^\delta+x^{2\delta}+\cdots+x^{n-\delta} = \frac{x^n+1}{x^\delta+1}$$

is a zero-concurring subset of C.

New constructions are presented in the following.

A. Lower Bounds

A simple constructive lower bound on the number of zero-concurring codewords in a linear code may be formulated as follows.

Theorem 1.1: Any linear code $C(n,k,d)$ contains a set of $[n/(n-k+1)]$ zero-concurring codewords.

Proof: Let $\boldsymbol{G}=[g_{ij}]$, $1 \le i \le n$, $1 \le j \le k$, be a generator matrix of C. Evidently, $\boldsymbol{G}$ may be always brought into the form of a band matrix, such that $g_{ij} \ne 0$ only if $j \le i \le j+n-k$. Now let $\boldsymbol{g}_j$, $1 \le j \le k$, denote the rows of $\boldsymbol{G}$. Then the set

$$\left\{\boldsymbol{g}_{1+j(n-k+1)}: j=0,1,\cdots,\left[\frac{n}{n-k+1}\right]-1\right\} \tag{4}$$

is a zero-concurring subset of C. □

For high-rate codes the above construction provides large, though not bound reaching, sets of zero-concurring codewords. However, if the code is maximum distance separable (MDS), i.e., such that $d=n-k+1$, the lower bound coincides with the upper bound of (1).

If the minimum distance of the code is known, the lower bound of Theorem 1.1 may be slightly improved. As shown in Section III, the set of all the codewords of C that are nonconcurring to a specific codeword of Hamming weight d may be regarded as a linear code $C'(n',k',d')$ where $n'=n-d$ and $k' \ge k-d$. Hence, applying Theorem 1.1 to C' we get

$$J \ge 1+\left[\frac{n'}{n'-k'+1}\right] \ge 1+\left[\frac{n-d}{n-k+1}\right].$$

Now let d_1 be an upper bound on the minimum distance of a code of length $(n-d)$ and dimension $(k-d)$. Then clearly $d' \le d_1$ and using a similar argument, we obtain

$$J \ge 2+\left[\frac{n-d-d_1}{n-k+1}\right]$$

provided that $k'>0$. Proceeding in this manner we arrive at the following lower bound on J.

Theorem 1.2: Denote by $D_q(n;k)$ the largest minimum distance of a linear code of length n and dimension k over GF(q). Let $d_0=d$ and let $d_i=D_q(n-d_0-d_1-\cdots-d_{i-1};k-d_0-d_1-\cdots-d_{i-1})$ for $i=1,2,\cdots,\nu+1$, where ν is the largest integer such that $k-d_0-d_1-\cdots-d_\nu>0$. Then $J \ge \nu+2$.

Theorem 1.2 implies that various lower bounds on J may be derived using the known upper bounds on the minimum distance of linear codes.

B. Cyclic Codes with Degenerate Subcodes

A cyclic code which consists of several repetitions of a code of smaller block length is said to be degenerate. In the following we derive a simple procedure for identifying a degenerate cyclic subcode within a given cyclic code, which leads to a powerful construction method.

Lemma 2: A linear cyclic code C of length n with parity check polynomial $h(x)$ contains a nontrivial degenerate cyclic subcode if and only if $\gcd(x^\delta-1,h(x)) \ne 1$ for some $\delta<n$.

Proof: Suppose $\gcd(x^\delta-1,h(x)) \ne 1$. Let C_δ be the cyclic code with parity check polynomial $h_\delta(x)=\gcd(x^\delta-1,h(x))$. Since $h_\delta(x)$ divides $h(x)$, C_δ is a cyclic subcode of C. As $h_\delta(x)$ also divides $x^\delta-1$, every codewords of C_δ is of the form

$$f(x)\frac{x^n-1}{h_\delta(x)} = f(x)\frac{x^\delta-1}{h_\delta(x)}(1+x^\delta+x^{2\delta}+\cdots+x^{n-\delta}), \tag{5}$$

where $\deg f(x) < \deg h_\delta(x)$. Therefore C_δ is degenerate. Conversely, if C_δ is a nontrivial degenerate cyclic subcode of C with parity check polynomial $h_\delta(x)$, then $h_\delta(x) | h(x)$ and $h_\delta(x) | x^\delta-1$ for some $\delta<n$. Therefore, $\gcd(x^\delta-1,h(x)) \ne 1$. □

The condition of Lemma 2 may be also formulated in terms of the zeros of the cyclic code C. Obviously $\gcd(x^\delta-1,h(x)) \ne 1$ iff $(x^\delta-1)$ and $h(x)$ have common roots. This is only possible if δ

is a divisor of n and therefore we may define $m = n/\delta$, where m is an integer. Now let α be a primitive nth root of unity. Then the roots of $(x^\delta - 1)$ are $1, \alpha^m, \cdots, \alpha^{n-m}$. Thus we have proved the following corollary.

Corollary 2.1: A linear cyclic code C of length n contains a nontrivial degenerate cyclic subcode if and only if for some m, a factor of n, at least one of the elements of the set $\{1, \alpha^m, \cdots, \alpha^{n-m}\}$ is not a zero of C.

Consider the following special cases of Lemma 2. If $\gcd(x^\delta - 1, h(x)) = h(x)$, i.e., $h(x) | x^\delta - 1$, then the whole code is degenerate (see [9, p. 224]). On the other hand if $\gcd(x^\delta - 1, h(x)) = x^\delta - 1$, i.e. $x^\delta - 1 | h(x)$, then the degenerate code C_δ is isomorphic to $\mathrm{GF}(q)^\delta$, where $\mathrm{GF}(q)$ is the ground field of C. Indeed, the generator polynomial of C_δ is $1 + x^\delta + x^{2\delta} + \cdots + x^{n-\delta}$ and therefore C_δ consists of all the vectors of length n and of the form $|\boldsymbol{u}|\boldsymbol{u}|\cdots|\boldsymbol{u}|$, where $\boldsymbol{u} \in \mathrm{GF}(q)^\delta$. Choosing δ unit vectors from $\mathrm{GF}(q)^\delta$ we obtain a binary zero-concurring subset of C with δ elements. For $q = 2$ this reduces to (3). Hence, Lemma 2 may be viewed as a generalization of the second construction method of Be'ery and Snyders [1]. Also, if C is a BCH code with zeros at $\alpha, \alpha^2, \cdots, \alpha^{2t}$ and their cyclotomic conjugates, then [8] for any $n/\delta = m \geq 2t + 1$ none of the elements in the set $\{1, \alpha^m, \cdots, \alpha^{n-m}\}$ is a zero of C and therefore $\gcd(x^\delta - 1, h(x)) = x^\delta - 1$. Thus Corollary 2.1 may be regarded as a generalization of Theorem 2 of [8].

In general C_δ is a much smaller code than C. It is therefore easier to find a maximal set of zero-concurring codewords in C_δ rather than in C. In many cases this set is also a maximal zero-concurring subset of the original code C. Thus, for example, for the (65,40,10) binary BCH code with a generator polynomial given by $g(x) = 1 + x + x^2 + x^4 + x^5 + x^9 + x^{10} + x^{15} + x^{16} + x^{20} + x^{21} + x^{23} + x^{24} + x^{25}$ we have $h_\delta(x) = \gcd(x^{13} - 1, h(x)) = 1 + x + x^2 + \cdots + x^{12}$. Hence C_δ is the (65,12,10) degenerate cyclic code given by the Kronecker product of the (13,12,2) binary parity code and the (5,1,5) repetition code. A maximal set of $J = 6$ zero-concurring codewords of C_δ is readily found to be

$$\{x^j(1 + x + x^{13} + x^{14} + x^{26} + x^{27} + x^{39} + x^{40} + x^{52} + x^{53}): \quad j = 0,2,4,6,8,10\},$$

which is a maximal zero-concurring subset of the (65,40,10) BCH code as well. Bound reaching zero-concurring subsets of many other BCH codes (for instance entries 2, 3, 10, 12, 13, 17–19, 39 and 44 of Table I) may be obtained using this technique.

C. Primitive BCH Codes with Distance One less than a Power of 2

This construction is due to E. R. Berlekamp.

Theorem 3: Any primitive binary BCH code of length $n = 2^m - 1$ and minimum distance $d = 2^i - 1$ contains a set of $[n/d]$ zero-concurring codewords, each of weight d or $d + 1$.

Proof: For any i, the $2^m - 1$ nonzero m-dimensional binary vectors can be partitioned into $[(2^m - 1)/(2^i - 1)]$ disjoint sets, such that each set is either the 2^i vectors in an i-dimensional affine subspace or the $2^i - 1$ nonzero vectors in an i-dimensional subspace of $\mathrm{GF}(2)^m$ (the existence of such partition is proved in the Appendix). It is known [10, p. 363] that codewords of weight 2^i, $i = m - r$, in the binary Reed–Muller code of order r and length $n = 2^m$ are given by

$$c(x) = \sum_{\alpha^i \in A} x^i$$

where A is an i-dimensional (affine) subspace of $\mathrm{GF}(2^m)$ and $c_\alpha = 1$ iff $\alpha^\alpha \in A$. The singly punctured Reed–Muller codes $\mathrm{RM}^*(r,m)$ have one parity check (at location ∞) removed. Hence, the $[(2^m - 1)/(2^i - 1)]$ *disjoint* sets indicated in the foregoing partition correspond to $[n/d]$ zero-concurring codewords in $\mathrm{RM}^*(r,m)$, each of weight $d = 2^i - 1$ or $d + 1 = 2^i$. All primitive binary BCH codes whose designed distances are one less than a power of 2 are supercodes of singly punctured Reed–Muller codes of the same minimum distance, which proves the theorem. □

This construction was explicitly used to obtain bound reaching sets of zero-concurring codewords in entries 21, 23, and 27 of Table I.

D. Direct Product and Concatenated Codes

Let A and B be respectively (n_1, k_1, d_1) and (n_2, k_2, d_2) linear codes over $\mathrm{GF}(q)$. The direct product code $C = A \otimes B$ is the $(n_1 n_2, k_1 k_2, d_1 d_2)$ code whose codewords consist of all $n_1 \times n_2$ arrays in which the columns belong to A and the rows to B.

Theorem 4a: Let J_1 and J_2 denote the number of elements in a maximal zero-concurring subset of A and B, respectively. Then C contains a set of $J_1 J_2$ zero-concurring codewords.

Proof: The generator matrix of C is given by $\boldsymbol{G} = \boldsymbol{G}_A \otimes \boldsymbol{G}_B$, where $\boldsymbol{G}_A$ and $\boldsymbol{G}_B$ are the generator matrices of respectively A and B, and $\otimes$ stands for the Kronecker product over $\mathrm{GF}(q)$. Choose $\boldsymbol{G}_A$ and $\boldsymbol{G}_B$, such that the top J_1 and J_2 rows of $\boldsymbol{G}_A$ and $\boldsymbol{G}_B$, respectively, are zero-concurring. The Kronecker product of these rows produces the following set of $J_1 J_2$ zero-concurring codewords of C

$$\{g_l: jk_1 + 1 < l \leq jk_2 + J_2, \qquad \text{for } j \in \{0,1,\cdots,J_1 - 1\}\} \quad (6)$$

where $\boldsymbol{g}_l$ denotes the lth row of $\boldsymbol{G}$. □

Now let $C = A * B$ be an $(Nn, Kk, d^* \geq Dd)$ concatenated code over $\mathrm{GF}(q)$, where the outer code A is an (N, K, D) code over $\mathrm{GF}(q^k)$ and the inner code B is an (n, k, d) code over $\mathrm{GF}(q)$.

Theorem 4b: Let J_1 and J_2 denote the number of elements in a maximal zero-concurring subset of A and B, respectively, and let A also contain a subset of J_3 zero-concurring codewords, such that all of its elements belong to $\mathrm{GF}(q)^N$. Then there exists a zero-concurring subset of C with $\max\{J_1, J_2 J_3\}$ elements.

Proof: The generator matrix of C is again given by $\boldsymbol{G} = \boldsymbol{G}_A \times \boldsymbol{G}_B$. Note that $\alpha \cdot \boldsymbol{G}_B$, where $\alpha \in \mathrm{GF}(q^k)$, is understood here as a product of a $k \times k$ matrix representation of α over $\mathrm{GF}(q)$ and a $k \times n$ generator matrix of B. Now let $\boldsymbol{G}_B$ be such that its top J_2 rows are zero-concurring. Taking $\boldsymbol{G}_A$ either such that its top J_1 rows are zero-concurring, or such that its top J_3 rows are zero-concurring and belong to $\mathrm{GF}(q)^N$, proves the theorem. □

The foregoing theorem may be generalized as follows. Let $\Omega = \{\alpha_1, \alpha_2, \cdots, \alpha_k\}$ be a basis of $\mathrm{GF}(q^k)$ over $\mathrm{GF}(q)$. When the inner encoder is presented at its input with a q-ary k-tuple corresponding to one of the elements of Ω, it will produce at its output a scalar multiple of one of the rows of $\boldsymbol{G}_B$. Now let $\boldsymbol{G}_B$ have J_2 zero-concurring rows and let F be some fixed subset of Ω of cardinality J_2; then the inner encoder may be chosen such that all the elements of F are mapped onto zero-concurring n-tuples. Let us call two nonbinary codewords zero-concurring in general sense (z.c.g.) if they coincide only in positions containing zeros. Evidently, if the inner encoder is presented with z.c.g. vectors, such that all their entries belong to F, it will produce zero-concurring strings at its output. Thus we have proved the following.

Theorem 4c: Let J_4 denote the number of codewords in a maximal z.c.g. subset of A, such that all its elements belong to

F^N. Then there exists a zero-concurring subset of C with $\max\{J_1, J_4\}$ elements.

The proposition of Theorem 4b is seen to be a special case of Theorem 4c if we construct a set of $J_4 = J_2 J_3$ z.c.g. vectors in A by taking the J_3 zero-concurring codewords that belong to $\mathrm{GF}(q)^N$ and all the multiples thereof by elements of F.

It would be interesting to investigate the case of generalized concatenated codes (GCC). For the definition of such codes and notation see for instance [9, p. 590]. Obviously, if GCC contains a usual concatenated code then Theorems 4b and 4c may be applied. If this is not the case other constructions should be used. As a simple example consider two outer codes: A_1 a code over $\mathrm{GF}(2^{k-1})$ and a binary code A_2. Thus the inner code B_0 is the union of 2^{k-1} cosets of the code consisting of two words, say $(00\cdots0)$ and $(11\cdots1)$. Now let A_1 and A_2 both contain J_1 zero-concurring codewords of equal Hamming weights and let these codewords be such that their nonzero coordinates coincide. Then the number of zero-concurring codewords in GCC is at least $2J_1$.

It is noteworthy that if a Reed–Solomon code is used as the outer code in the concatenation the set of $J_2 J_3$ zero-concurring codewords of C would often be bound reaching. Consider, for example, the (120,44,20) binary code obtained by concatenating the (15,11,5) RS code over $\mathrm{GF}(2^4)$ with the (8,4,4) extended binary Hamming code. In this case we have $J_2 = 2$ and $J_3 = 3$. Thus, a bound reaching set of $J_2 J_3 = 6$ zero-concurring codewords of the concatenated code may be constructed using Theorem 4b. As the Reed–Solomon codes are MDS, a maximal set $[N/D]$ zero-concurring codewords of the RS code is given by (4). Hence, another direct consequence of Theorem 4b is that a $(2N, K, 2D)$ Justesen code obtained from an (N, K, D) RS code contains a bound reaching set of $J_1 = [2N/2D]$ zero-concurring codewords. In general, since many cyclic codes are representable as either direct product or concatenated codes, the foregoing results suggest that an efficient method of search for zero-concurring codewords in a long cyclic code may be based on its factorization.

E. Constructing Zero-concurring Codewords from Concurring Codewords

A set of independent codewords of C, a code over $\mathrm{GF}(q)$, is called *concurring* if in some l coordinates $i_1, i_2, \cdots, i_l$ all the codewords are identical (up to multiplication by a constant) and in all the other coordinates at most one of the codewords has a nonzero entry. Concurring codewords were extensively studied in the context of majority logic decoding. Blahut [13] provides an upper bound on the number of concurring codewords in a linear code of length n and minimum distance d

$$J^* \le \left[\frac{2n}{d}\right] - 1 \tag{7}$$

Let S be a set of J^* concurring codewords of C. For any pair of codewords $\boldsymbol{c}_{j_1}, \boldsymbol{c}_{j_2} \in S$ there exists a linear combination $\boldsymbol{c}_j = \alpha \boldsymbol{c}_{j_1} + \beta \boldsymbol{c}_{j_2}$, where $\alpha, \beta \in \mathrm{GF}(q)$, such that $\boldsymbol{c}_j$ contains zeros in coordinates $i_1, i_2, \cdots, i_l$. Thus if vectors of S are taken together in pairs, each vector of S occurring in at most one pair, then the resulting set $\{\boldsymbol{c}_j\colon j = 1, 2, \cdots, [J^*/2]\}$ is a zero-concurring subset of C. If J^* is odd, the remaining codeword (i.e., the one with no pair) may be appended to the foregoing zero-concurring subset. Hence, for any linear code

$$J \ge \left[\frac{J^* + 1}{2}\right]. \tag{8}$$

Comparing (7), (8), and (1), a linear code that contains a bound reaching set of concurring codewords also contains a bound reaching zero-concurring subset.

It is known that the generalized Reed–Muller (GRM) codes are in many cases majority logic decodable. The dual of a GRM code over $\mathrm{GF}(q)$ is also a GRM code over $\mathrm{GF}(q)$. Thus, the GRM codes that attain the upper bound of (7) contain $[n/d]$ zero-concurring codewords. It may be shown [6] that the binary (24,12,8) Golay code and the ternary (12,6,6) Golay code have $[2n/d] - 1$ concurring codewords. Hence, using the foregoing construction, they also contain bound reaching sets of zero-concurring codewords.

III. Algorithms

Each of the constructions derived in the previous section applies solely to a certain specific family of codes. Apparently in the case of a general linear code one has to adopt a computer search by some efficient algorithms. In fact, the table of maximal zero-concurring subsets of several binary linear codes, presented in [1], was compiled with the help of a "brute force" search. In such a brute force search all possible sets of distinct codewords are checked for being zero-concurring. Therefore, if checking whether two given codewords are zero-concurring is regarded as a single operation, then the number of operations required by the brute force search is of the order of

$$2 \cdot \binom{q^k}{2} + 3 \cdot \binom{q^k}{3} + \cdots + \min\{J+1, [n/d]\} \cdot \binom{q^k}{\min\{J+1, [n/d]\}},$$

which obviously makes such search probibitive even for moderately large codes. In this section we derive several general algorithms for the search of zero-concurring codewords, with computational complexity substantially lower than that of the brute force search. The efficiency of these algorithms is manifested by the fact that we were able to obtain subsets with $[n/d]$ or $[n/d] - 1$ zero-concurring codewords for 44 codes in Table I in a typical running time of a few minutes for a code of length 127.

The formulation of our algorithms demands a certain elaboration of the well-known technique of shortening (or taking the *cross-section* of) linear codes. As before, $\boldsymbol{G}$ is a generator matrix of a linear code $C(n, k, d)$.

Definition 2: For each codeword $\boldsymbol{c} \in C$, the cross-section weight of $\boldsymbol{c}$, $w_{cs}(\boldsymbol{c})$, is the rank of the submatrix $\boldsymbol{\Lambda}_c$ of $\boldsymbol{G}$ consisting of the columns that correspond to the nonzero coordinates of $\boldsymbol{c}$.

The foregoing definition is unambiguous, in the sense that the cross-section weight of a codeword does not depend on the choice of $\boldsymbol{G}$, since the rank of a matrix is invariant under elementary row operations. Furthermore, it is easily verified that if C is binary the cross-section weight generates a norm in the linear space C in a way similar to the Hamming weight, which can also be viewed as a norm in this space.

Lemma 5: The set C' of all the codewords of C that are nonconcurring to a specific codeword $\boldsymbol{c} \in C$ is a linear code $C'(n', k', d')$ obtained by means of shortening the original code C by all the nonzero coordinates of $\boldsymbol{c}$, where

$$k' = k - w_{cs}(\boldsymbol{c}).$$

Proof: Let $L = \{i_1, i_2, \cdots, i_l\}$ be the set of nonzero coordinates of $\boldsymbol{c}$. As the rank of $\boldsymbol{\Lambda}_c$ is by definition $w_{cs}(\boldsymbol{c})$, there exists a nonsingular $k \times k$ matrix $\boldsymbol{E}$, such that the top $w_{cs}(\boldsymbol{c})$ rows of $\boldsymbol{E}\boldsymbol{\Lambda}_c$ are linearly independent and all the other rows are zero. Since $\boldsymbol{E}$ is nonsingular, $\boldsymbol{EG}$ is also a generator matrix of C. Let $\boldsymbol{G}'$ be a matrix obtained by deleting the top $w_{cs}(\boldsymbol{c})$ rows for $\boldsymbol{EG}$. Evidently $\boldsymbol{G}'$ is a row submatrix of $\boldsymbol{EG}$, such that the columns corresponding to the nonzero coordinates of $\boldsymbol{c}$ contain only zero entries. Therefore $\boldsymbol{G}'$ generates a shortened subcode of C, given

by

$$C' = \{\boldsymbol{c} = (c_1, c_2, \cdots, c_n) : \boldsymbol{c} \in C \quad \text{and} \quad c_i = 0 \text{ for all } i \in L\},$$

which is the required set of all the codewords of C that are nonconcurring to $\boldsymbol{c}$. The dimension of C' is by construction $k' = \operatorname{rank}(\boldsymbol{G}') = k - w_{cs}(\boldsymbol{c})$. □

Alternatively, the dimension of C' may be expressed as $k' = k - w_H(\boldsymbol{c}) + a$, where $w_H(\boldsymbol{c})$ is the Hamming weight of $\boldsymbol{c}$, and a is the dimension of the code that consists of all the codewords of the dual code of C whose nonzero coordinates are confined to $\{i_1, i_2, \cdots, i_t\}$. Given $\boldsymbol{G}$ and $\boldsymbol{c}$, a generator matrix of C' may be obtained by performing elementary row operations on $\boldsymbol{G}$. We refer to this as *performing cross-section on $\boldsymbol{G}$ by $\boldsymbol{c}$*. Now let Z be a given zero-concurring subset of C. We say that Z is *complete* if it is not contained in any larger zero-concurring subset of C. It follows from Lemma 5 that Z is complete if and only if $w_{cs}(\sigma_z) = k$, where σ_z is the sum of all the elements of Z. This implies the following recursive algorithm.

Algorithm A

1) Select a nonzero codeword $\boldsymbol{c} \in C$. Let C' be the subcode of C, resulting from performing cross-section on $\boldsymbol{G}$ by $\boldsymbol{c}$.
2) If $k' = 0$, set $Z_{\boldsymbol{c}} \leftarrow \{\boldsymbol{c}\}$; otherwise use *Algorithm A* to find a maximal zero-concurring subset of C', say $Z'_{\boldsymbol{c}}$, and set $Z_{\boldsymbol{c}} \leftarrow Z'_{\boldsymbol{c}} \cup \{\boldsymbol{c}\}$.
3) Repeat Steps 1) and 2) for all the nonzero codewords of C, unless at some execution of Step 2) a bound reaching set of zero-concurring codewords is found.

A set $Z_{\boldsymbol{c}}^*$, such that for all $\boldsymbol{c} \in (C - \{\boldsymbol{0}\})$: $|Z_{\boldsymbol{c}}^*| \geq |Z_{\boldsymbol{c}}|$, where $|\cdot|$ with respect to a set denotes its cardinality, is a maximal zero-concurring subset of the code C.

The main idea of the foregoing algorithm is to consider only complete zero-concurring subsets of C instead of all the possible subsets of C as in the brute force approach. This results in a substantial computational gain. Let M and M' be the number of codewords in C and C', respectively. It follows from Lemma 5 that

$$M' = M/q^{w_{cs}(\boldsymbol{c})} \leq M/q^{d^{cs}},$$

where d^{cs} is the minimum cross-section weight of C. Thus, $M/q^{d^{cs}}$ is an upper bound on M'; yet it does not provide sufficient insight. It will be shown in the sequel (see Lemma 6) that $d^{cs} \leq d$, with strict equality for about half of the linear codes. Therefore, for the purpose of estimating the computational complexity of Algorithm A, we approximate the average value of M' by

$$M' \approx M/q^d.$$

This estimate has been experimentally verified for many codes, yielding in all cases an upper bound on the actual average value of M'. We assume that performing cross-section requires, on the average, about $nk/2$ elementary row operations. Then substituting q^k for M gives the following expression for the computational complexity of Algorithm A in terms of the total number of elementary row operations required by the algorithm

$$N_1 = \frac{nk}{2} \sum_{j=0}^{J-1} q^{(j+1)(k-jd/2)}$$

$$= \frac{nk}{2}\left(q^k + q^{2k-d} + q^{3k-3d} + q^{4k-6d} + q^{5k-10d} + \cdots\right).$$

The complexity of Algorithm A may be considerably reduced if we remove the necessity to repeat Steps 1) and 2) of the algorithm for all the nonzero codewords of C. This calls for some selection criterion that could be used to distinguish between the elements of a maximal zero-concurring subset of the code and all other codewords of C. Let $Z_{\boldsymbol{c}}$ be one of the $q^k - 1$ zero-concurring subsets of C generated by Algorithm A, and let $Z_{\boldsymbol{c}}^*$ be a maximal zero-concurring subset of C, eventually resulting from the algorithm. We define the *average cross-section weight* of a set $Z_{\boldsymbol{c}}$ as

$$W_{cs}(Z_{\boldsymbol{c}}) = \frac{1}{|Z_{\boldsymbol{c}}|} \cdot \sum_{\boldsymbol{c}_j \in Z_{\boldsymbol{c}}} w_{cs}^j(\boldsymbol{c}_j),$$

where $w_{cs}^j(\boldsymbol{c}_j)$ is the cross-section weight of a codeword $\boldsymbol{c}_j \in Z_{\boldsymbol{c}}$ in the code obtained at stage j of recursion in Algorithm A. By Lemma 5,

$$\sum_{\boldsymbol{c}_j \in Z_{\boldsymbol{c}}} w_{cs}^j(\boldsymbol{c}_j) = \sum_{\boldsymbol{c}_j \in Z_{\boldsymbol{c}}^*} w_{cs}^j(\boldsymbol{c}_j) = k,$$

and since $|Z_{\boldsymbol{c}}^*| \geq |Z_{\boldsymbol{c}}|$, we have $W_{cs}(Z_{\boldsymbol{c}}^*) \leq W_{cs}(Z_{\boldsymbol{c}})$. Consequently, if one of the $q^k - 1$ sets $Z_{\boldsymbol{c}}$ could be chosen, such that its average cross-section weight would be the smallest—this set would be a maximal zero-concurring subset of C. Hence, we suggest the minimum cross-section weight selection criterion to be employed at Step 1) of the following algorithm.

Algorithm B

1) Select a codeword $\boldsymbol{v} \in C$, such that for all $\boldsymbol{c} \in (C - \{\boldsymbol{0}\})$: $w_{cs}(\boldsymbol{v}) \leq w_{cs}(\boldsymbol{c})$, and obtain C' by performing cross-section on $\boldsymbol{G}$ by $\boldsymbol{v}$.
2) If $k' = 0$ set $Z_{\boldsymbol{v}} \leftarrow \{\boldsymbol{v}\}$; otherwise use *Algorithm B* to find a zero-concurring subset of C', say $Z'_{\boldsymbol{v}}$, and set $Z_{\boldsymbol{v}} \leftarrow Z'_{\boldsymbol{v}} \cup \{\boldsymbol{v}\}$.

It is conjectured that the set $Z_{\boldsymbol{v}}$ is a maximal zero-concurring subset of C. In fact, no example to the contrary has been found. If a minimum cross-section weight codeword of C is selected with the help of an exhaustive search through the code then the number of elementary row operations required by Algorithm B is given by

$$N_2 = \frac{1}{2}\left(nkq^k + n_1(k - d^{cs})q^{k-d^{cs}} + n_2(k - d^{cs} - d_1^{cs})q^{k-d^{cs}-d_1^{cs}} + \cdots\right) \approx \frac{nk}{2} \cdot q^k, \quad (9)$$

where d_j^{cs} and n_j are the minimum cross-section weight and the block length of the code obtained at stage j of recursion in Algorithm B. This may be cut down further if instead of an exhaustive search a more efficient method is employed to find a minimum cross-section weight codeword of the code. Let $\boldsymbol{G}_s$ be a systematic generator matrix of C and let $\boldsymbol{c} = \boldsymbol{s}\boldsymbol{G}_s$, where $\boldsymbol{s} \in \mathrm{GF}(q)^k$ is a source vector. Then evidently $w_{cs}(c) \geq w_H(s)$. Hence, we may consider source vectors in order of nondecreasing Hamming weight and terminate on $w_H(s) = d^{cs}$. Note that d^{cs} need not be *a priori* known. The worst case complexity of an algorithm utilizing this property of a systematic generator matrix is given by

$$N = \frac{nk}{2} \sum_{i=1}^{d^{cs}-1} \binom{k}{i} (q-1)^i \quad (10)$$

where N stands for the number of elementary row operations required. Obviously (10) is always lower than the complexity of exhaustive search given by (9). In addition, for a given k this is a rapidly decreasing function of d^{cs} thus making the algorithm efficiently applicable to long high-rate codes where d^{cs} is small. If this is not the case but d^{cs} is *a priori* known, the algorithm may be terminated on $w_{cs}(\boldsymbol{c}) = d^{cs}$. The problem is that d^{cs} is seldom known and, hence, the complexity of Algorithm B remains essentially exponential. A suboptimal polynomial time algorithm may be derived with the help of the following result.

Lemma 6: Let $C^{\perp}(n, n-k, d^{\perp})$ be the dual code of $C(n, k, d)$. Then the minimum cross-section weight of C is bounded by: $\min\{d^{\perp} - 1, d\} \leq d^{cs} \leq d$.

Proof: Let $\boldsymbol{u} \in C$ be a codeword of Hamming weight d. Then $w_{cs}(\boldsymbol{u}) = \operatorname{rank}(\boldsymbol{\Lambda}_{\boldsymbol{u}}) \leq d$, as $\boldsymbol{\Lambda}_{\boldsymbol{u}}$ contains only d columns. Since $\boldsymbol{G}$ is a parity check matrix of $C^{\perp}$, every $d^{\perp} - 1$ column of $\boldsymbol{G}$ is linearly independent. Therefore, if $d \geq d^{\perp} - 1$ we have for all $\boldsymbol{c} \in (C - \{\boldsymbol{0}\})$: $w_{cs}(\boldsymbol{c}) \geq d^{\perp} - 1$. If $d < d^{\perp}$, codewords $\boldsymbol{c} \in C$

exist, such that $w_H(c) \le d^{\perp} - 1$. For these codewords Λ_c contains $w_H(c)$ linearly independent columns, so that $w_{cs}(c) = \operatorname{rank}(\Lambda_c) = w_H(c) \ge d$. □

It follows from the foregoing lemma and its proof that whenever $d < d^{\perp}$, $d^{cs} = d$ and a minimum Hamming weight codeword of the code is also a minimum cross-section weight codeword. This justifies, to some extent, substitution of the minimum cross-section weight selection criterion in Algorithm B by the minimum Hamming weight selection criterion. We shall refer to the modified algorithm as *Algorithm C*. Algorithm C is suboptimal, in the sense that it does not necessarily provide a maximal set of zero-concurring codewords. Consider, for example, a binary code given by the following generator matrix,

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

It can be readily verified that the maximal zero-concurring subset of the code is

$$\begin{Bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{Bmatrix}.$$

Yet, applying Algorithm C gives

$$\begin{Bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{Bmatrix}.$$

Apparently many other such codes exist. However, in all but one of the 45 codes in Table 1 zero-concurring subsets with $[n/d]$ or $[n/d]-1$ elements have been found with the help of Algorithm C. The complexity of Algorithm C is given by

$$N_3 = \sum_j N_{\text{search}}(n_j, k_j, d_j),$$

where j is upper bounded by $[n/d]$; n_j, k_j and d_j are the length, the dimension and the minimum distance of the code obtained at stage j or recursion in Algorithm C. $N_{\text{search}}(n_j, k_j, d_j)$ is the computational complexity of the algorithm employed for the search of a minimum Hamming weight codeword in the code. As the minimum Hamming distance of many codes is known we may use any kind of search and terminate on $w_H(c) = d$. Alternatively, since Algorithm C makes no claim on optimality, one may employ a suboptimal polynomial time algorithm to find minimum weight codewords (such an algorithm was proposed by Harari [14]). In this case the computational complexity of Algorithm C would be polynomial.

A different approach to reducing the complexity of proposed algorithms may be based on employing the automorphism group of the code. Let π be a permutation on the set $\{1, 2, \cdots, n\}$ sending each i into $\pi(i)$ and each $c = (c_1, c_2, \cdots, c_n) \in C$ into $c^{\pi} = (c_{\pi(1)}, c_{\pi(2)}, \cdots, c_{\pi(n)})$. We denote by C_c' and $C_{c^{\pi}}'$ the set of all the codewords of C that are nonconcurring to, respectively, c and c^{π}. Evidently, if $\pi \in \operatorname{Aut}(C)$, where $\operatorname{Aut}(C)$ is the automorphism group of C, the codes C_c' and $C_{c^{\pi}}'$ are equivalent and, hence, contain the same number of zero-concurring codewords. Now let the set Ω_c be defined by $\Omega_c = \{a \in C: a = c^{\pi} \text{ for some } \pi \in \operatorname{Aut}(C)\}$; then $\Omega_{c_1} \cap \Omega_{c_2} = \phi$ if and only if $c_2 \notin \Omega_{c_1}$, and $\Omega_{c_1} = \Omega_{c_2}$, otherwise. Thus, any linear code C may be partitioned into t disjoint cosets $\Omega_{c_1}, \Omega_{c_2}, \cdots, \Omega_{c_t}$, where the codewords $c_1, c_2, \cdots, c_t$ are automorphic representatives of C. It follows that when searching for a minimum cross-section weight codeword (Algorithm B) or for a maximal set of zero-concurring codewords (Algorithm A) it is sufficient to consider the set of automorphic representatives of C instead of the entire code. This implies computational gain of approximately q^k/t at each stage of recursion, provided that the automorphism group of the code and its automorphic representatives can be efficiently found. A polynomial time algorithm for computing automorphism groups of linear codes appears in [15]. The problem of finding automorphic representatives is, to the best of our knowledge, still open. However, in the special case of cyclic permutations the situation is different. A way to find cycle representatives of a minimal cyclic code is described in [9] and [16]. If the code is not minimal one may employ other techniques [17].

TABLE I
ZERO-CONCURRING CODEWORDS IN BCH CODES

Code	Upper Bound	Number of Zero-Concurring Codewords Found
Primitive Narrow-Sense BCH Codes		
1) BCH [7,4,3]	2	2
2) BCH [15,11,3]	5	5
3) BCH [15,7,5]	3	3
4) BCH [15,5,7]	2	2
5) BCH [31,26,3]	10	10
6) BCH [31,21,5]	6	6
7) BCH [31,16,7]	4	4
8) BCH [31,11,11]	2	2
9) BCH [31,6,15]	2	2
10) BCH [63,57,3]	21	21
11) BCH [63,51,5]	12	12
12) BCH [63,45,7]	9	9
13) BCH [63,39,9]	7	7
14) BCH [63,36,11]	5	5
15) BCH [63,30,13]	4	4
16) BCH [63,24,15]	4	4
17) BCH [63,18,21]	3	3
18) BCH [63,16,23]	2	2
19) BCH [63,10,27]	2	2
20) BCH [63,7,31]	2	2
21) BCH [127,120,3]	42	42
22) BCH [127,113,5]*	25	24
23) BCH [127,106,7]	18	18
24) BCH [127,99,9]*	14	12
25) BCH [127,92,11]*	11	10
26) BCH [127,85,13]*	9	8
27) BCH [127,78,15]	8	8
28) BCH [127,71,19]*	6	5
29) BCH [127,64,21]*	6	5
30) BCH [127,57,23]*	5	4
31) BCH [127,50,27]	4	4
32) BCH [127,43,31]	4	4
33) BCH [127,36,31]	4	4
34) BCH [127,29,43]	2	2
35) BCH [127,22,47]	2	2
36) BCH [127,15,55]	2	2
37) BCH [127,8,63]	2	2
Nonprimitive Narrow-Sense BCH Codes		
38) BCH [17,9,5]	3	3
39) BCH [21,12,5]	3	3
40) BCH [23,12,7]	3	3
41) BCH [33,22,6]	5	5
42) BCH [41,20,10]	3	3
43) BCH [47,24,11]	4	4
44) BCH [65,53,5]	13	13
45) BCH [65,40,10]	6	6

IV. RESULTS

All the algorithms derived in the previous section are general, in the sense that they apply to any linear code. We arbitrarily chose the family of primitive binary BCH codes as a "benchmark" for the proposed algorithms. Some results obtained for these codes and also for several nonprimitive BCH codes are listed in Table I. Entries 1–4, 7–9, 19, 20, 37–40 were obtained with the help of Algorithm A. Entries 5, 6, 11, 17, 18, 22, 41 follow by applying Algorithm B. Entries 10, 12, 13, 41 and 44 were derived by using Lemma 2. Entries 21, 23, and 27 are due specifically to the construction of Theorem 3. All other

entries in the table were obtained by means of Algorithm C. The (127,99,9) BCH code is the only code in the table for which Algorithm C provides less than $[n/d]-1$ zero-concurring codewords. The (63,18,21) BCH code is peculiar in that the maximal set of zero-concurring codewords, namely $\{x^j(1+x^3+x^6+\cdots+x^{60})$: $j=0,1,2\}$, is unique (it is invariant under the automorphism group of the code; any other zero-concurring set contains at most 2 codewords).

The upper bound is everywhere $[n/d]$, with two exceptions where this bound may be tightened. Short proofs of the tighter bound follow.

1) Consider the (41,20,10) binary BCH code with generator polynomial $g(x)=1+x^2+x^3+x^5+x^6+x^7+x^9+x^{12}+x^{14}+x^{15}+x^{16}+x^{18}+x^{19}+x^{21}$. Assume that there exists a subset of $[n/d]=4$ zero-concurring codewords. Since $(x+1)$ is a factor of $g(x)$, the sum of all the 4 codewords must have weight 40. It follows that the augmented code generated by $g(x)/x+1$ has to contain a codeword of weight 1. Yet, by the BCH bound the minimum distance of the augmented code is at least 9, which contradicts the initial assumption. Consequently, the (41,20,10) BCH code contains no more than 3 zero-concurring codewords.
2) Consider the (21,12,5) BCH code.[2] It is precisely the (24,12,8) Golay code punctured in three coordinates (cf. [11]). If the punctured code contained two codewords of weight 5 that where nonconcurring, then this would correspond to two Golay codewords of weight 8 that concurred in the 3 punctured coordinates. However, it is well known that the (24,12,8) Golay code contains only codewords of weights divisible by 4 and no codewords of weight 10. Hence, it does not contain any pair of codewords of weight 8 that intersect in three coordinates. Therefore, any set of zero-concurring codewords of the (21,12,5) code contains, at most, one codeword of weight 5. From this it follows that there can be no more than 3 codewords in such a set.

The remaining seven codes for which the upper and lower bounds do not agree are marked with an asterisk.

A nonnegative integer λ is said to be the *contraction index* of a linear (n,k) code C, if a maximal set of pairwise linearly independent columns of a generator matrix of C contains exactly $k+\lambda$ elements. Given this notation, finding a maximal zero-concurring subset of C is equivalent to finding the highest dimension subcode of C with contraction index zero. Thus the problem discussed herein may be viewed as a special case of a more general problem of finding large subcodes with small contraction index in a given code. Algorithms employing such subcodes for efficient soft-decision decoding appear in [6], while bounds on the dimension of codes and subcodes with prescribed contraction index are presented in [18].

Acknowledgment

The authors are greatly indebted to Elwyn R. Berlekamp for the contribution of Theorem 3 and valuable comments. They are also grateful to G. David Forney, Jr. for preprints of his papers, to Jakov Snyders for helpful discussions and to the referees for many constructive remarks. Alexander Vardy wishes to thank Hagit Itzkowitz for her invaluable help.

Appendix

This Appendix shows the existence of partition used in the proof of Theorem 3 [12]. For each fixed j, $0 \le j < i$, we describe a construction which, by induction, covers $m = j+i, j+2i, j+3i \cdots$. In the initial case $m = i+j$ and $2^m-1=(2^j-1)2^i+(2^i-1)$. Hence, there is only one subspace. Evidently any m-dimensional binary space can be partitioned into an i-dimensional space and its cosets, which are affine subspaces. The induction must show how to extend an m-dimensional partition to an $(m+i)$-dimensional partition. In general, for $m=j+li, 2^m-1 \equiv 2^j-1 \bmod(2^i-1)$, so that any partition has to include exactly 2^j-1 affine subspaces and $(2^{li}-1)/(2^i-1)-(2^j-1)$ subspaces. We obtain 2^j-1 $(m+i)$-dimensional affine subspaces by appending a string of i zeroes to all the vectors in an m-dimensional affine subspace. Now let z be a string of $m-i-j$ zeroes. We assume that an m-dimensional partition contains a set of the form

$$\begin{array}{ll} s_1 & z \\ s_2 & z \\ & \cdots \\ s_{2^j-1} & z \end{array}$$

where $\{s_1, s_2, \cdots, s_{2^j-1}, \mathbf{0}\}$ constitutes an i-dimensional subspace. We then include in an $(m+i)$-dimensional partition one set of the form

$$\begin{array}{lll} & & \leftarrow i \rightarrow \\ s_1 & z & 0 \cdots 0 \\ s_2 & z & 0 \cdots 0 \\ & \cdots & \\ s_{2^j-1} & z & 0 \cdots 0 \end{array}$$

and 2^{i+j} sets of the form

$$\begin{array}{lll} (\alpha^k s_1) & z & \boldsymbol{\nu}_1 \\ (\alpha^k s_2) & z & \boldsymbol{\nu}_2 \\ & \cdots & \\ (\alpha^k s_{2^j-1}) & z & \boldsymbol{\nu}_{2^j-1} \end{array} \quad , \quad \text{for } k=0,1,\cdots,2^{i+j}-2,\infty$$

where α is a primitive element of GF(2^{i+j}) and α^∞ stands for $\mathbf{0}$. The i-tuples $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \cdots, \boldsymbol{\nu}_{2^i-1}$ are the 2^i-1 nonzero binary i-tuples in some fixed order, such that $\boldsymbol{\nu}_{i_1}+\boldsymbol{\nu}_{i_2}=\boldsymbol{\nu}_{i_3}$ whenever $s_{i_1}+s_{i_2}=s_{i_3}$. It is easily verified that each of the above sets together with $\mathbf{0}$ is a subspace of GF$(2)^{m+i}$. Finally, if $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_{2^j-1}\}$ is a subspace set in the m-dimensional partition and if it is ordered lexicographically, i.e., so that $\boldsymbol{b}_1 < \boldsymbol{b}_2 < \cdots < \boldsymbol{b}_{2^j-1}$, then we include in our partition for $m+i$ dimensions all 2^i sets of the form

$$\begin{array}{ll} b_1 & (\alpha^k \boldsymbol{\nu}_1) \\ b_2 & (\alpha^k \boldsymbol{\nu}_2) \\ & \cdots \\ b_{2^j-1} & (\alpha^k \boldsymbol{\nu}_{2^j-1}) \end{array} \quad , \quad \text{for } k=0,1,\cdots,2^i-2,\infty$$

where $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \cdots, \boldsymbol{\nu}_{2^i-1}$ are the nonzero elements of GF(2^i), in lexicographic order, and α is a primitive element of GF(2^i). Altogether we have

$$(2^j-1)+1+2^{i+j}+2^i \cdot 2^j \cdot \left(\frac{2^{il}-1}{2^i-1}-1\right)=\left[\frac{2^{m+i}-1}{2^i-1}\right]$$

disjoint sets. This completes the construction.

[2] This proof is due to E. R. Berlekamp.

References

[1] Y. Be'ery and J. Snyders, "Optical soft decision block decoders based on fast Hadamard transform," *IEEE Trans. Inform. Theory*, vol. IT-32, no. 3, pp. 355–364, May 1986.
[2] E. R. Berlekamp, "The construction of fast, high-rate, soft-decision block decoders," *IEEE Trans. Inform. Theory*, vol. IT-29, no. 3, pp. 372–377, May 1983.

[3] J. H. Conway and N. J. A. Sloane, "Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice," *IEEE Trans. Inform. Theory*, vol. IT-32, no. 1, pp. 41–50, Jan. 1986.
[4] G. D. Forney, Jr., "Coset codes II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. IT-34, no. 5, pp. 1152–1187, Sept. 1988.
[5] A. D. Abbaszadeh and C. K. Rushforth, "VLSI implementation of a maximum likelihood decoder for the Golay (24,12) code," *IEEE J. Select. Areas Comm.*, vol. SAC-6, pp. 558–565, 1988.
[6] J. Snyders and Y. Be'ery, "Maximum likelihood soft decoding of binary blocks codes and decoders for the Golay codes," *IEEE Trans. Inform. Theory*, vol. 35, no. 5, pp. 963–975, Sept. 1989.
[7] Y. Be'ery and J. Snyders, "A recursive Hadamard Transform optimal soft-decision decoding algorithm", *SIAM J. Algebraic and Discrete Methods*, vol. 8, pp. 778–789, 1987.
[8] R. H. Deng and M. A. Herro, "DC-free coset codes," *IEEE Trans. Inform. Theory*, vol. 34, no. 4, pp. 786–792, July 1988.
[9] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.
[10] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.
[11] ___, "Coding theory and the Mathieu groups," *Inform. Contr.*, vol. 18, pp. 40–64, 1971.
[12] ___, personal communication.
[13] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.
[14] S. Harari, "A polynomial time algorithm for finding minimum weight codewords in a linear code," *IEEE Int. Symp. Inform. Theory*, Brighton, England, June 1985.
[15] J. S. Leon, "Computing automorphism groups of error-correcting codes," *IEEE Trans. Inform. Theory*, vol. IT-28, no. 3, pp. 496–511, May 1982.
[16] F. J. MacWilliams and J. Seery, "The weight distributions of some minimal cyclic codes," *IEEE Trans. Inform. Theory*, vol. IT-27, no. 6, p. 796, Nov. 1981.
[17] S. E. Tavares, P. E. Allard, and S. S. Shiva, "On the decomposition of cyclic codes into cyclic classes," *Inform. Contr.*, vol. 18, pp. 342–354, 1971.
[18] A. Vardy, J. Snyders, and Y. Be'ery, "Bounds on the dimension of codes and subcodes with prescribed contraction index," *Linear Algebra Appl.*, to appear, 1990.

More on the Minimum Distance of Cyclic Codes

P. J. N. de Rooij and J. H. van Lint

***Abstract*—It was recently shown that the so-called Jensen bound is generally weaker than the product method and the shifting method introduced by van Lint and Wilson. We show that the minimum distance of the two cyclic codes of length 65 for which it is known that the product method does not produce the desired result can be proved using Jensen's method with some adaptations.**

***Index Terms*—Minimum distance, 2-D-cyclic code, concatenated code, shifting.**

I. Introduction

In 1986 a new method for calculating the minimum distance of cyclic codes was developed by J. H. van Lint and R. M. Wilson [4]. Their paper contained two related methods: a matrix-product method and a method called "*shifting*." Previous bounds, such as the BCH bound, the Hartmann–Tzeng bound and the method developed by Roos are all special cases of this method. It turned out that the minimum distance of all cyclic codes of length < 63 (all codes in that paper are binary codes) with two exceptions can be determined using this method. The number of cyclic codes of length 63 is exceedingly large, and it is still not clear how many of them can be handled by this method. For the codes of length 65, it was shown by M. H. M. Smid [7] that again all but two of these codes can be handled by the product method. The first purpose of this correspondence is to determine the minimum distance of these two exceptional codes (for which presently only computer searches have established the minimum distance).

In 1985 J. M. Jensen [3] developed another method for calculating the minimum distance of cyclic codes based on the idea of Berlekamp and Justesen of representing these codes as two-dimensional cyclic codes. Jensen's method was recently analyzed by the first author in his master's thesis [6] with the rather disappointing result that the method is usually weaker than shifting. (However, the amount of computation required for shifting is often quite large.) The second purpose of this correspondence is to show that, with some extra tricks, Jensen's method is strong enough to handle the two cyclic codes of length 65 that could not be done by the product method. Clearly, it is not of great importance to consider two isolated examples of length 65, but the method of this correspondence can be used in many other situations e.g., for Blokh–Zyablov codes [2]. Hence, explaining the methods that we use in our examples in Section III is our main goal.

In the following, we shall use terminology, notation, and results from the paper by Jensen on the structure of cyclic codes. We assume that the reader is familiar with that paper and also with the product method. In Section II we only briefly review what we shall need in the sequel.

II. Definitions

Let G be an Abelian group of order nN that is the direct product of two cyclic subgroups G_x and G_y of order n resp. N, that is, $G = G_x \times G_y$ contains (w.l.o.g.) the elements $\{x^i y^j | 0 \le i < n \wedge 0 \le j < N\}$, $(x^n = y^N = 1)$. Furthermore let q be a prime power and $\gcd(nN, q) = 1$.

Definition 2.1: The *group algebra* $\boldsymbol{F}_q G$ is the ring (with unity) consisting of all (formal) polynomials

$$c(x, y) = \sum_{i=0}^{n-1} \sum_{j=0}^{N-1} c_{ij} x^i y^j, \qquad \text{where } c_{ij} \in \boldsymbol{F}_q.$$

Definition 2.2: A 2-D *cyclic code* of size n by N over the alphabet $\boldsymbol{F}_q$ is an ideal in $\boldsymbol{F}_q G$.

We represent a codeword by the corresponding polynomial or by the $n \times N$ matrix $\|c_{ij}\|$. We shall not distinguish between these notations.

If $\gcd(n, N) = 1$, then the Chinese remainder theorem shows that every element $x^i y^j$ is a power of $Z = xy$; so, Z is a generator of G. Thus G is cyclic.

Using this, the following can be derived (cf. [1]).

Theorem 2.3: If $\gcd(n, N) = 1$ and G and q are as above, then a 2-D cyclic code $\mathscr{C}$ in $\boldsymbol{F}_q G$ is cyclic.

The converse is true as well.

Theorem 2.4: A cyclic code of length nN, with $\gcd(n, N) = 1$ is 2-D cyclic.

In the following, $\mathscr{A}_s$ will denote a minimal cyclic code and we shall use the symbol θ_s for the idempotent of this code. It is well

Manuscript received December 18, 1989.
P. J. N. de Rooij is with PTT Research Neher Lab, P.O. Box 421, 2260AK, Leidschendam, The Netherlands.
J. H. van Lint is with the Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600MB, Eindhoven, The Netherlands.
IEEE Log Number 9039291.

0018-9448/91/0100-0187$01.00 © 1991 IEEE