Some New Results for Optimal Ternary Linear Codes

Iliya Bouyukliev and Juriaan Simonis

***Abstract*—Let $d_3(n, k)$ be the maximum possible minimum Hamming distance of a ternary $[n, k, d]$-code for given values of n and k. We describe a package for code extension and use this to prove some new exact values of $d_3(n, k)$. Moreover, we classify the ternary $[n, k, d_3(n, k)]$-codes for some values of n and k.**

***Index Terms*—Code extension, optimal codes, ternary linear codes.**

I. Introduction

Let $\mathbb{F}_q^n$ be the n-dimensional standard vector space over the finite field $\mathbb{F}_q$. The (Hamming) distance between two vectors of $\mathbb{F}_q^n$ is defined to be the number of coordinates in which they differ. A q-ary linear $[n, k, d]$-code is a k-dimensional linear subspace of $\mathbb{F}_q^n$ with minimum distance d.

A central problem in coding theory is to find $d_q(n, k)$. This is the largest value of d for which a q-ary $[n, k, d]$-code exists. A code with parameters $[n, k, d_q(n, k)]$ is called (distance)-optimal. Another important problem is to classify all optimal codes with given parameters. Of course, this is feasible only if the number of inequivalent codes is relatively small.

In this correspondence, we investigate ternary linear codes. The systematic research of ternary optimal codes has been initiated by Hill and Newton in [17]. Now there is complete information on $d_3(n)$ for $k \leq 5$ and good progress for $k = 6$.

There are some interesting classification examples of ternary codes meeting the Griesmer bound. A classification of optimal codes of small lengths was given by van Eupen and Lisonek in [12]. They used a geometrical approach for the classification of codes of dimensions 2 and 3. In dimensions 4 and 5, their main tool was a computational method based on finite group action.

In this correspondence, we classify optimal ternary codes for 20 different parameter sets and find some new exact values of $d_3(n, k)$. Our main approach is an algorithm for the extension of codes using their residual codes. Extra information on the codes, such as its dual distance and weight set, is very important for this algorithm to work effectively.

The algorithm EXTENSION has been used successfully in the binary case in [5] and [6]. Using EXTENSION and David Jaffe's program language SPLIT, the authors have finished the problem of finding the exact values of $d_2(n, 8)$, and have classified many binary codes with lengths up to 128 and dimension up to 7. We have generalized this algorithm for the nonbinary case and have realized it in the program package Q-EXTENSION.

In Section II, we describe the package Q-EXTENSION and give some information on its complexity. Section III reports some nonexistence and existence results derived from this package. Moreover, it contains classifications of some strongly optimal codes. The correspondence ends with tables containing classification results for codes that are not strongly optimal.

Manuscript received August 9, 2000; revised March 7, 2001. This work was supported in part by the Bulgarian National Science Foundation under Contract MM-901/1999.

I. Bouyukliev is with the Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, 500 Veliko Tarnovo, Bulgaria (e-mail: stefka_illya@yahoo.com).

J. Simonis is with the Department of Mediamatics, Faculty of Information Technology and Systems, Delft University of Technology, 2600 GA Delft, The Netherlands (e-mail: J.Simonis@its.tudelft.nl).

Communicated by S. Litsyn. Associate Editor for Coding Theory.

Publisher Item Identifier S 0018-9448(02)01936-3.

II. The Software Tools

There are two main approaches for the classification of codes. The first is theoretical and it is based on algebraic-combinatorial or geometrical arguments. This method has been used for the classification of specific codes or families of codes with specific properties like the generalized MacDonald codes, the Reed–Muller codes, the Hamming codes, and the Golay codes. In the nonbinary case, minihypers have been used to classify projective codes meeting the Griesmer bound. For more information the reader is referred to [21] and [13]. In almost all other examples, the authors have used a computer in some steps of their proofs.

There are many computer packages for linear codes, for instance, GUAVA, SPLIT, MAGMA, QLC, etc. They provide a construction of codes with a given structure, computations of some code parameters, etc., by means of the method of linear programming for finding bounds on the code parameters. Our goal was to design a universal method for the classification of codes. We are not interested in the structure of the codes but only in their parameters. We use restrictions on the possible nonzero weights and the dual distance of the code and its subcodes.

For the classification and the construction of codes, we use the extension of codes with smaller parameters. There are two kind of extensions. The first is the extension up to length, which is the construction of an $[n, k, d]$-code $\mathcal{C}$ on the basis of an $[n-w, k-1, d']$-code (a residual code of $\mathcal{C}$), or on the basis of an $[n-i, k, d']$-code. The second is the extension up to dimension which is the extension of an $[n, k, d]$-code to an $[n+i, k+i, d]$- or $[n+i+1, k+i, d]$-code. If G is a generator matrix for an $[n, k, d]$-code, we extend it to

$$\left(\begin{array}{c|c} * & I_i \\ \hline G & 0 \end{array}\right) \quad \text{or} \quad \left(\begin{array}{c|c} * & \mathbf{1}\, I_i \\ \hline G & 0 \end{array}\right)$$

where I_i is the identity matrix and $\mathbf{1}$ is the all-one column vector of length i.

Our algorithm for the second kind of extension is similar to the algorithm for the first kind. So we describe the first one. This algorithm has two main parts. The first is finding the solutions, and the second is finding the inequivalent solutions amongst them. In the binary case, finding the solutions is fast enough and it takes up a small part of the computation time. For the second part, we only have to deal with permutation equivalence.

The generalization of the binary version of EXTENSION to Q-EXTENSION was a nontrivial problem. The algorithm consists of many subalgorithms like computing the weight distribution of a code, the rank of a matrix, orbits of codewords which generate the code under the action of a permutation group, and so on. So it is difficult to determine the complexity of EXTENSION, and even more so of Q-EXTENSION. In the next paragraphs we shall only compare the complexity of some parts in the binary and in the nonbinary case. The search of solutions in the nonbinary case increases exponentially with the field size. Moreover, for binary codes we can use a bit presentation of the vectors, which is not the case for nonbinary codes. In the binary case, two codes are equivalent if and only if one can be obtained from the other by permuting the coordinates. In the nonbinary case, we have in addition the multiplication of the elements in a given position by a nonzero element of $\mathbb{F}_q$ and application of a field automorphism to the elements in all coordinate positions.

So Q-EXTENSION contains some new ideas that made it possible to decrease the computation time a few hundred times with respect to its basic algorithm.

A. The Construction of Codes

Let $\mathcal{C}_x$ be an $[n, k, d]$-code and let $\mathcal{C}_0$ be its residual code $\mathrm{Res}(\mathcal{C}, w)$ with respect to a codeword of weight w with $w < d + \lceil w/q \rceil$. In this subsection, we describe how to construct all $[n, k, d]$-codes $\mathcal{C}$ with the set of nonzero weights $W = \{w_1, \ldots, w_s\}$ if we know the residual code $\mathcal{C}_0$. The problem of obtaining the codes $\mathcal{C}$ coincides with that of constructing their generator matrices.

Let G_0 be a generator matrix of the $[n', k-1, d']$-code $\mathcal{C}_0$. Then $\mathcal{C}$ has a generator matrix of the following form:

$$G = \left(\begin{array}{c|c} 1\ 1\ \cdots\ 1 & 0\ \cdots\ 0\ 0 \\ \hline \cdots\cdots\cdots & \\ \cdots\cdots\cdots & G_0 \\ \cdots\cdots\cdots & \end{array}\right).$$

This matrix has two parts, a fixed right-hand one, with rows $u_1 = 0$, $u_2, \ldots, u_k$, and a free left-hand one, G_w (from position 1 to $w = n - n'$). The first row of G_w is the all-one vector.

We shall describe the algorithm for the binary case. It is easy (by means of exhaustive search) to find how many ones the second, the third, up to the kth row, can contain. Suppose that the second row contains x_2 ones. Without loss of generality, we can put these in the first x_2 coordinates of the second row of G_w. So the columns of the constructed part of matrix G_w are lexicographically ordered. This will be the case in each step. In each stage, the coordinate index set $(1, 2, \ldots, w)$ will be partitioned into intervals corresponding to constant columns.

Suppose that we know the first $t-1$ rows of G_w. Let us denote them by $v_1, v_2, \ldots, v_{t-1}$. A *solution* for the row i $(i \geq t)$ is any vector v (lexicographically ordered in the intervals) for which the matrix with rows

$$(v_1, u_1), (v_2, u_2), \ldots, (v_{t-1}, u_{t-1}), (v, u_i)$$

generates an $[n, t, d]$-code with its nonzero weights in the set W. To each solution, we can associate a vector M whose length is the number of the intervals and such that its coordinates show us the number of ones in each interval. When $i = t$, we call the solution *real*, otherwise, we call it *possible*. We use a back-track algorithm to find all real solutions.

Estimate of the Complexity of the Algorithm: A basis operation is a check whether a vector is a solution or not. The question is how many operations we should perform to obtain all successors of one possible solution. Suppose that up to the rth step each interval was divided into two subintervals. Obviously, if any interval was not divided into subintervals, we have only one possibility for it. So we consider the worst case. If we have l intervals with lengths $s_1, s_2, \ldots, s_l$, the complexity in an exhaustive search is $S := (s_1 + 1)(s_2 + 1)\cdots(s_l + 1)$ operations. Let $M := (m_{1,2}, m_{3,4}, \ldots, m_{l-1,l})$ be a possible solution in the previous step. To find the next solutions following from this solution, the complexity will be

$$S^* := (\min\{s_1, m_{1,2}\} + 1)(\min\{s_3, m_{3,4}\} + 1)\cdots (\min\{s_{l-1}, m_{l-1,l}\} + 1).$$

This idea can be easily extended to the nonbinary case. The complexity in the q-ary case is about S^{*q-1}.

The following remarks also hold for codes over arbitrary fields $\mathbb{F}_q$.

Remark 1: How many proportional columns can any interval have? Without loss of generality, we can take the fixed part of the generator matrix of the code in systematic form. In the rth step, we obtain an $[n-k+r, r, d]$-code. The matrix G can contain i proportional columns only if an $[n-k+r-i, r-1, d]$-code exists. So we can obtain an upper bound for the number of the proportional columns in each interval. Then the complexity in the binary case will be about

$$S^{**} = (\min\{s_1, m_{1,2}, i\} + 1)(\min\{s_3, m_{3,4}, i\} + 1)\cdots (\min\{s_{l-1}, m_{l-1,l}, i\} + 1).$$

If the investigated code is projective, we have $i = 1$ in the last step. In some cases, the possible number of proportional columns depends on the minimum distance of the dual code. For example, if we construct a code with dual distance four, there cannot be proportional columns in the $(k-1)$th step.

Remark 2: Is it necessary to know the whole vector v to see that it is not a solution? For any $[n, k, d]$-code, two codewords have at least d different coordinates and so they have at most $n - d$ equal coordinates. This is important for optimal codes of large length and small dimension.

Remark 3: What type of generator matrix of the residual code will be more suitable? The number of the solutions strongly depends on the form of the matrix G_0. If we use a matrix G_0 so that the first t rows generate a subcode of dimension t with minimum effective length among the t-dimensional subcodes of the residual code then the number of the solutions will be smaller. We make a hierarchy of subcodes such that any of them have a minimum effective length. An open question is how to find an optimal solution.

Remark 4: The dual distance of the residual code must be equal to or greater than the dual distance of the code to be constructed.

Remark 5: Restriction on the possible weights. To find the solutions, we can use additional information on the codes. We can use all the traditional methods for the restriction of weights. This is an important part of Jaffe's language SPLIT in the binary case [18].

B. Code Equivalence

In our algorithm, we use two types of equivalences depending on the dimension of the code corresponding to the real solution. Let y_1 and y_2 be two real solutions for the tth row with corresponding codes $\mathcal{C}_{y_1}$ and $\mathcal{C}_{y_2}$. These codes are generated from the first t rows of G. When $t < k$, we say that y_1 and y_2 are *equivalent up to extension* if $\sigma_1\sigma_2(C_{y_1}) = C_{y_2}$ for some permutations σ_1 of the first w coordinates and $\sigma_2 \in \mathrm{Aut}\,(C_0)$. If $t = k$, we have the usual equivalence between linear codes.

Let L denote the set of all linear codes. We will use the notation from [22].

Definition 6: An invariant over a set E is defined to be a mapping $L \to E$ such that any two equivalent codes take the same values.

The use of invariants helps us to determine in an easy way whether two codes are inequivalent. Any invariant is a global property of the code. We need to define a local property, which will be a specific characterization of any coordinate position.

Let $\mathcal{C}$ be a code of length n. The automorphism group $\mathrm{Aut}\,(\mathcal{C})$ of $\mathcal{C}$ acts on the set of coordinates $I_n = \{1, 2, \ldots, n\}$ as a group of permutations and divides it into disjoint orbits so that two coordinates i, j are in the same orbit if and only if there exists a permutation σ from this group for which $\sigma(i) = j$.

Definition 7: A *signature* S over a set F is a map from the set of coordinates of $\mathcal{C}$ into F such that $S(i) = S(j)$ if i and j are in the same orbit of $\mathrm{Aut}\,(\mathcal{C})$.

One of the main problems is how to divide the code coordinates into disjoint orbits without knowing its automorphism group, only using different signatures. As signatures, we use different kind of weight enumerators of the reduced codes. (Reduction is defined a few lines below.) We consider codes with relatively small dimensions and it is easy to compute their weight enumerators.

Any signature S defines an equivalence relation in the coordinate index set of a code $\mathcal{C}$ and hence a partition $\{I_1(S), \ldots, I_t(S)\}$. This partition will match with any other obtained from a code $\mathcal{C}'$ equivalent to $\mathcal{C}$. Any set I_j consists of one or more orbits of $\mathcal{C}$. If we have two different partitions $\{I_1(S_1), \ldots, I_{t_1}(S_1)\}$ and $\{I_1(S_2), \ldots, I_{t_2}(S_2)\}$ we can take the meet partition.

The signatures help us to easily obtain an isomorphism between two equivalent codes. If the two codes have trivial automorphism groups we can partition all coordinates into different orbits using an appropriate signature (this is the support splitting algorithm in [22]).

We use the following invariants and signatures in our algorithm.

1) If there are intervals of length more than 1 in the real solution y (there are equal coordinates), we take one representative from each coordinate. We call this the reduced code $\mathcal{C}_{y_s}$. However, we need a vector z to show us how many times any coordinate was taken. This vector is a signature, and if we order it lexicographically, is an invariant for the code. With this reduction, we usually destroy the optimality of the corresponding code.
2) The weight enumerator of $\mathcal{C}_{y_s}$ is another invariant of the code.
3) For each coordinate we can find the weight enumerator of the code without this coordinate. These polynomials are signatures for the code, and lexicographically ordered they present an invariant.

We use a subset M of $\mathcal{C}$ which generates the code as a vector space and which is stable under the action of $\mathrm{Aut}\,(\mathcal{C})$. We preferably take M as the set of the codewords of minimum weight. Our algorithm uses an algorithm for matrix equivalence, and we extend it with invariants and signatures.

Remark 8: The procedure for equivalence of matrices contains a source code of Kapralov. We also use some procedures from the program package QLC [2].

III. Results

We have two kinds of results. The first one consists of bounds for the function $d_3(n, k)$, and the second one of classification results.

A. New Bounds for $d_3(n, k)$

Bounds for $d_3(n, k)$ for $1 \leq k \leq n \leq 243$ have been published in Brouwer's tables in [7].[1]

We have proved the nonexistence of several codes with given parameters and in this way we have found new upper bounds for the function $d_3(n, k)$. We also have constructed a ternary code with parameters $[23, 7, 12]$. The existence of this code proves that $d_3(23, 7) = 12$.

Theorem 9: No ternary code with parameters $[35, 6, 21]$ exists.

Proof: Van Eupen and Lisonek [12] have proved that there exist exactly 236 inequivalent ternary $[14, 5, 7]$-codes. Since there are no ternary $[13, 5, 7]$- and $[7, 5, 3]$-codes, a hypothetical $[35, 6, 21]$-code cannot contain vectors of weights 22 and 28. Using Q-EXTENSION we infer that none of these codes can be extended to a $[35, 6, 21]$-code without codewords of weights 22 and 28. □

[1] An updated version can be found on his regularly updated web page at http://www.win.tue.nl/math/dw/voorlincod.html.

Theorem 10: No ternary code with parameters $[37, 6, 22]$ exists.

Proof: Van Eupen and Lisonek [12] have proved that there exist exactly four inequivalent ternary $[15, 5, 8]$-codes. Using Q-EXTENSION, we see that no one of them can be extended to a $[37, 6, 22]$-code. □

Theorem 11: No ternary code with parameters $[28, 6, 16]$ exists.

Proof: Since no ternary $[26, 5, 16]$-code exists, it follows that any putative $[28, 6, 16]$-code has dual distance at least 3. After the extension of the $[6, 4, 2; 3]$-codes with dual distance ≥ 3 we obtain exactly five codes with parameters $[15, 5, 6]$ and with dual distance ≥ 3. None of these can be extended to a $[26, 5, 16; 3]$-code. □

Recently, an independent proof of this result was given by Hamada *et al.* [14].

Theorem 12: No code with parameters $[48, 6, 30; 3]$ exists.

Proof: Since no ternary $[45, 4, 30]$-code exists, it follows that any putative $[48, 6, 30]$-code has dual distance at least 4. Since no ternary codes with parameters

$$[17, 5, 10],\ [14, 5, 8],\ [13, 5, 7],\ [8, 5, 4],\ [7, 5, 3],\ \text{or}\ [5, 5, 2]$$

exist, it follows that any putative $[48, 6, 30]$-code does not contain codewords of weights 31, 34, 35, 40, 41, or 43. Van Eupen and Lisonek have proved that there exist exactly seven inequivalent ternary $[18, 5, 10]$-codes. All of them have dual distance 4. But no one of them can be extended to $[48, 6, 30]$-code. □

Recently, an independent proof of this result was given by Hill and Jones [16].

The next corollary follows from the above theorems and Brouwer's table [7].

Corollary 13: $d_3(35, 6) = 20$, $d_3(37, 6) = 21$, $d_3(28, 6) = 15$, and $d_3(48, 6) = 29$.

Theorem 14: There exists a unique ternary $[23, 7, 12]$ code.

Proof: The standard residual code argument tells us that any putative $[23, 7, 12]$-code does not contain codewords of weights 13, 14, 16, or 17. There exist at most 354 $[11, 6, 4]$-codes and a unique $[11, 6, 5]$-code. Using the restriction on the weights and these codes of length 11, we obtain a unique ternary $[23, 7, 12]$-code. A generator matrix of this code is

$$\begin{pmatrix} 11111111111100000000000 \\ 11112222000011110000000 \\ 20001220220120011100000 \\ 21200200212001201201000 \\ 22000011012012102200100 \\ 02122212102012202100010 \\ 01020121122002200200001 \end{pmatrix}$$

and its weight enumerator is

$$1 + 472z^{12} + 1020z^{15} + 668z^{18} + 26z^{21}$$ □

B. Strongly Optimal Codes

If an $[n + 1, k + 1, d]$-code exists, we can obtain an $[n, k, d]$-code by shortening it in one position. Similarly, if an $[n + 1, k, d + 1]$-code exists, we can obtain an $[n, k, d]$-code by puncturing it. So, following [8], we define a linear $[n, k, d]$-code to be *strongly optimal* if no $[n + 1, k + 1, d]$-codes or $[n + 1, k, d + 1]$-codes exist.

In the next theorems we classify some ternary strongly optimal codes.

Theorem 15: There exists a unique ternary $[29, 5, 18]$-code.

Proof: Using Q-EXTENSION, it is easy to verify that there exist exactly eight ternary $[11, 4, 6]$-codes. After the extension of these

TABLE I
BOUNDS FOR $d_3(n, k)$ FOR $k = 4, 5, 6$ AND CLASSIFICATION OF SOME OPTIMAL CODES

$q=3$	$k=4$		$k=5$		$k=6$	
n	d	number	d	number	d	number
6	2		2		1	
7	3	4 [12]	2		2	
8	4	3 [12]	3	3 [12]	2	
9	5	1 [9]	4	1 [12]	3	3 [12]
10	6	1 [9]	5	1 [12]	4	1 [12]
11	6	**8**	6	1 Dual Golay	5	1 Golay
12	6	**361**	6	**9**	6	1 Extended Golay
13	7	72 [12]	6		6	**10**
14	8	14 [12]	7	236 [12]	6	
15	9	3 [12]	8	4 [12]	7	**22**
16	9	**320**	9	1 [12]	7	
17	10	18 [12]	9		8	
18	11	2 [12]	10	7 [12]	9	
19	12	1 [15]	11	2 [12]	9	
20	12	**85**	12	2 [15]	10	
21	12		12		11	
22	13		12		12	
23	14	**165**	13		12	
24	15	**17**	14	**444**	13	
25	16	1 [13]	15	**23**	14	**2**
26	17	1 [13]	15		15	**2**
27	18	1 [13]	16	**11**	15	
28	18	**11**	17	**3**	**15**	
29	18		18	**1**	16	
30	19	8 [13]	18		17	
31	20	2 [13]	18		18	
32	21	1 [13]	19		18	
33	21	**74**	20		18-19	
34	22	3 [13]	21	**442**	19-20	
35	23	1 [13]	21		**20**	
36	24	1 [13]	22		20	
37	24	**17**	23	**2**	**21**	
38	25	1 [13]	24	**1**	22	
39	26	1 [13]	24		23	
40	27	1 [13]	24		24	
41	27		25		24	
42	27		26		25	
43	27		27		26	
44	28		27		27	
45	29	**243**	28		27	
46	30	**27**	29		27-28	
47	30		30	**2**	28	
48	31		30		**29**	

TABLE II
CLASSIFICATION OF SOME TERNARY CODES OF DIMENSION AT LEAST 7

$[n, k, d]$	number
$[13, 7, 5]$	6
$[23, 7, 12]$	1
$[27, 7, 15]$	2
$[27, 8, 14]$	1
$[28, 8, 15]$	1

codes, we obtain a unique code of parameters $[29, 5, 18]$. Hence this code is unique. It was constructed by van Eupen [10]. □

Remark 16: This is the second example of a unique strongly optimal code with $d < q^{k-1}$ which is not projective. The first example was given by Hill and van Eupen in [11].

Theorem 17: There exist exactly 442 ternary codes with parameters $[34, 5, 21]$.

Proof: Van Eupen and Lisonek [12] have classified the ternary $[13, 4, 7]$-codes. After extension of these codes we obtain exactly 442 self-orthogonal inequivalent $[34, 5, 21]$-codes. Recently, it has been shown by Jones in his Ph.D. dissertation [19] that all ternary $[34, 5, 21]$-codes are divisible and hence are self-orthogonal. □

Theorem 18: There exists a unique ternary $[38, 5, 24]$-code.

TABLE III
SCHEME OF THE EXTENSIONS

$[3,3,1]$	→	$[5,3,2]$	→	$[11,4,6]$				
$[3,3,1]$	→	$[6,3,2]$						
			→	$[12,4,6]$				
$[3,3,1]$	→	$[6,3,3]$						
$[3,3,1]$	→	$[8,3,4]$						
			→	$[20,4,12]$				
$[3,3,1]$	→	$[8,3,5]$						
$[3,3,1]$	→	$[9,3,5]$		$[23,4,14]$				
			→					
$[3,3,1]$	→	$[9,3,6]$		$[24,4,15]$				
$[2,2,1]$	→	$[4,2,2]$						
			→	$[10,3,6]$	→	$[28,4,18]$		
$[2,2,1]$	→	$[4,2,3]$						
$[2,2,1]$	→	$[5,2,3]$	→	$[12,3,7]$				
					→	$[33,4,21]$		
$[2,2,1]$	→	$[4,2,3]$	→	$[12,3,8]$				
$[2,2,1]$	→	$[5,2,3]$	→	$[13,3,8]$				
					→	$[37,4,24]$		
$[2,2,1]$	→	$[4,2,3]$	→	$[13,3,9]$				
						$[45,4,29]$		
$[2,2,1]$	→	$[6,2,4]$	→	$[16,3,10]$	→			
						$[46,4,30]$		
$[4,4,1]$	→	$[6,4,2]$	→	$[15,5,6]$				
$[3,3,1]$	→	$[5,3,2]$	→	$[10,4,5]$		$[24,5,14]$		
					→			
		$[4,4,1]$	→	$[10,4,6]$		$[25,5,15]$		
						$[27,5,16]$		
$[3,3,1]$	→	$[5,3,2]$	→	$[11,4,6]$	→	$[28,5,17]$		
						$[29,5,18]$		
$[3,3,1]$	→	$[6,3,3]$	→	$[13,4,7]$	→	$[34,5,21]$		
$[3,3,1]$	→	$[6,3,3]$	→	$[14,4,8]$	→	$[37,5,23]$		
						$[38,5,24]$		
$[3,3,1]$	→	$[7,3,4]$	→	$[17,4,10]$	→	$[47,5,30]$		
$[4,4,1]$	→	$[5,4,1]$						
			→	$[7,5,2]$	→	$[13,6,6]$		
$[4,4,1]$	→	$[5,4,2]$						
$[4,4,1]$	→	$[5,4,1]$						
			→	$[8,5,3]$	→	$[15,6,7]$		
$[4,4,1]$	→	$[5,4,2]$						
$[4,4,1]$	→	$[6,4,2]$	→	$[11,5,5]$				
					→	$[25,6,14]$		
		$[5,5,1]$	→	$[11,5,6]$		$[26,6,15]$		
$[5,5,1]$	→	$[6,5,1]$						
			→	$[8,6,2]$	→	$[13,7,5]$		
$[5,5,1]$	→	$[6,5,2]$						
$[5,5,1]$	→	$[7,5,2]$	→	$[12,6,5]$				
					→	$[27,7,15]$		
		$[6,6,1]$	→	$[12,6,6]$				
$[5,5,1]$	→	$[6,5,1]$						
			→	$[8,6,2]$	→	$[13,7,5]$	→	$[27,8,14]$
$[5,5,1]$	→	$[6,5,2]$						$[28,8,15]$

Proof: Van Eupen and Lisonek [12] have classified the ternary $[14, 4, 8]$-codes. Using Q-EXTENSION, we obtain a unique $[38, 5, 24]$-code. This code was constructed in [1]. □

Theorem 19: There exist exactly two inequivalent ternary $[47, 5, 30]$-codes.

Proof: Van Eupen and Lisonek [12] have classified the ternary $[17, 4, 10]$-codes. After extension of these codes, we obtain exactly two inequivalent $[47, 5, 30]$-codes. One of these codes was constructed by van Eupen [9], and the other by Boukliev [3]. □

Theorem 20: There exists a unique ternary code with parameters $[28, 8, 15]$.

Proof: There exist exactly six ternary $[13, 7, 5]$-codes. Extension leads to a unique $[28, 8, 15]$-ternary code. □

C. Other Results

In this subsection, we give some classification results of optimal codes which are not strongly optimal. We describe these results briefly in tabular form. We summarize all known results for functions

$d_3(n, 4)$, $d_3(n, 5)$, and $d_3(n, 6)$ for $n \leq 48$ in Table I. The number of all inequivalent $[n \leq 48, k = 4, 5, 6, d_3(n, k)]$ codes is given when this is known. The new results are in bold type. In Table II, we give the number of all inequivalent $[n, k, d_3(n, k)]$ codes for $k = 7$ and 8, and $n = 13, 23, 27, 28$. In Table III, we present a scheme for the hierarchy of the extensions which we use to produce our classification results given in Tables I and II.

Remark 21: Codes with the same number of words for each nonzero weight (BWD codes) were studied by Langevin and Zanotti [20] (see also [4]). Here, we classify ternary $[20, 4, 12]$-codes with weight distribution $A_0 = 1$, $A_{12} = A_{15} = 40$. There are exactly four inequivalent codes with these parameters and weight distribution. Two of them are not cyclic. It follows that there are BWD codes which cannot be constructed via a subgroup of the multiplicative group $\mathbb{F}_{q^k}^*$.

REFERENCES

[1] G. Bogdanova and I. Boukliev, "New linear codes of dimension 5 over GF(3)," in *Proc. 4th Int. Workshop ACCT-IV*, Novgorod, Russia, Sept. 11–17, 1994, pp. 41–43.

[2] G. Bogdanova, Pl. Christov, and S. Kapralov, "The new version of QLC—A computer program for linear codes studying," in *Proc. Int. Workshop OCRT'95*, Sozopol, Bulgaria, 1995, pp. 11–14.

[3] I. Boukliev, "New ternary linear codes," in *Proc. Int. Symp. Information Theory*, Whistler, BC, Canada, 1995, p. 135.

[4] ——, "Cyclotomic description of some optimal codes," in *Proc. Int. Workshop ACCT*, Pskov, Russia, 1998, pp. 52–56.

[5] I. Bouyukliev and D. Jaffe, "Optimal binary linear codes of dimension at most seven," *Discr Math*, vol. 226, pp. 51–70, 2001.

[6] I. Bouyukliev, D. Jaffe, and V. Vavrek, "The smallest length of eight-dimensional binary linear codes with prescribed minimum distance," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1539–1544, July 2000.

[7] A. E. Brouwer, "Bounds on the size of linear codes," in *Handbook of Coding Theory*, V. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: Elsevier, 1998. ISBN:0-444-50 088-X [Online] Available: http://www.win.tue. nl/math/dw/voorlincod.html.

[8] S. M. Dodunekov and J. Simonis, "Constructions of optimal linear codes," in *Numbers, Information and Complexity*. Norwell, MA: Kluwer, 2000, pp. 245–263.

[9] M. van Eupen, "Some new results for ternary linear codes of dimension 5 and 6," *IEEE Trans. Inform. Theory*, vol. 41, pp. 2048–2051, Nov. 1995.

[10] ——, "Five new optimal ternary linear codes," *IEEE Trans. Inform. Theory*, vol. 40, p. 193, Jan. 1994.

[11] M. van Eupen and R. Hill, "An optimal ternary [69, 5, 45] code and related codes," *Des., Codes, Cryptogr.*, vol. 4, pp. 271–282, 1994.

[12] M. van Eupen and P. Lisonek, "Classification of some optimal ternary linear codes of small length," *Des., Codes, Cryptogr.*, vol. 10, pp. 63–84, 1997.

[13] N. Hamada, "A survey of recent work on characterization of minihypers in $PG(t, q)$ and nonbinary linear codes meeting the Griesmer bound," *J. Combin. Inform. Syst. Sci.*, vol. 18, pp. 161–191, 1993.

[14] N. Hamada, T. Helleseth, H. M. Martinsen, and Ø. Ytrehus, "There is no ternary $[28, 6, 16]$ code," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1550–1554, July 2000.

[15] N. Hamada, T. Helleseth, and Ø. Ytrehus, "There are exactly two nonequivalent [20, 5, 12; 3]-codes," *Ars Comb.*, vol. 35, pp. 3–14, 1993.

[16] R. Hill and C. Jones, "The nonexistence of ternary [47, 6, 29] codes," Information from Brouwer's database [7].

[17] R. Hill and D. E. Newton, "Optimal ternary linear codes," *Des., Codes, Cryptogr.*, vol. 2, pp. 137–157, 1992.

[18] D. B. Jaffe. (Version 0.4, 1997, Apr.) Binary linear codes: New results on nonexistence. Dept. Math. Statist., Univ. Nebraska, Lincoln. [Online]. Available: http://www.math.unl.edu/djaffe

[19] C. M. Jones, "Optimal ternary linear codes," Ph.D. dissertation, Univ. Salford, Salford, U.K., 2000.

[20] P. Langevin and J. P. Zanotti, "Linear codes with balanced weight distribution," *Appl. Algebra Eng. Comm. Comput.*, vol. 6, pp. 299–307, 1995.

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

[22] N. Sendrier, "Finding the permutation between equivalent codes: The support splitting algorithm," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1193–1203, July 2000.

Optimal Covering Polynomial Sets Correcting Three Errors for Binary Cyclic Codes

Wonjin Sung and John T. Coffey, *Member, IEEE*

***Abstract*—The covering polynomial method is a generalization of error-trapping decoding and is a simple and effective way to decode cyclic codes. For cyclic codes of rate $R < 2/\tau$, covering polynomials of a single term suffice to correct up to τ errors, and minimal sets of covering polynomials are known for various such codes. In this correspondence, the case of $\tau = 3$ and of binary cyclic codes of rate $R \geq 2/3$ is investigated. Specifically, a closed-form specification is given for minimal covering polynomial sets for codes of rate $2/3 \leq R < 11/15$ for all sufficiently large code length n; the resulting number of covering polynomials is, if $R = 2/3 + \rho$ with $\rho > 0$, equal to $n\rho + 2\sqrt{n\rho} + (1/2)\log_\phi(n/\rho) + O(1)$, where $\phi = (1 + \sqrt{5})/2$. For all codes correcting up to three errors, the number of covering polynomials is at least $n\rho + 2\sqrt{n\rho} + O(\log n)$; covering polynomial sets achieving this bound (and thus within $O(\log n)$ of the minimum) are presented in closed-form specifications for rates in the range $11/15 \leq R < 3/4$.**

***Index Terms*—Covering polynomial, cyclic codes, error-trapping decoding, Kasami decoding.**

I. INTRODUCTION

Error-trapping decoding [7], [9], [12] is one of the simplest schemes to decode cyclic codes of rate $R < 1/\tau$, where τ is the maximum number of errors to be corrected. It identifies and corrects any correctable error pattern of an $[n, k]$ cyclic code if there is a set of k (cyclically) consecutive error-free symbols in the error pattern.

Several variations of error-trapping decoding have been proposed to extend its usefulness to codes of rate $R \geq 1/\tau$, e.g., permutation decoding [8], [1], [3], [17], systematic coset search [11], [6], and a similar approach using two complementary information sets [15]. The covering polynomial method suggested by Kasami [5] modifies error-trapping decoding by guessing the error patterns in n information sets, consisting of all sets of k (cyclically) consecutive symbols. These guesses can be represented by a set of polynomials, called *covering polynomials*, and the decoder complexity is determined by the number of required covering polynomials. Detailed descriptions of the algorithm can be also found in many textbooks [9], [7], [10]. We will assume that the reader is familiar with the covering polynomial algorithm, and refer to these textbooks for details.

The covering polynomial method can be applied to codes of rate $R < 2/\tau$ by using a set of polynomials with a single term (called

Manuscript received January 20, 2000; revised August 29, 2001. The material in this correspondence was presented in part at the IEEE International Symposium on Information Theory, Trondheim, Norway, 1994.

W. Sung is with the Department of Electronic Engineering, Sogang University, Seoul, 121-742, Korea (e-mail: wsung@sogang.ac.kr).

J. T. Coffey is with the Wireless Networking Business Unit, Texas Instruments, Santa Rosa, CA 95401 USA (e-mail: coffey@ti.com).

Communicated by R. M. Roth, Associate Editor for Coding Theory.

Publisher Item Identifier S 0018-9448(02)01943-0.