Determination of Minimum Weight for Binary Cyclic Codes

Dr. Abbas F. AL-Hashemi
Baghdad College for Economic Sciences University

Dr. Wasan S. Awad.
University of Bahrain

Abstract

The aim of this work to solve the problem of determining the minimum weight of binary cyclic codes. Since knowledge of this weight is necessary for the efficiency of the code in detecting error in stored or transferred digital data. This problem was solved by designing an efficient algorithm. This algorithm depends on new results obtained in this works. The algorithm needs only few codewords to calculate its weights. These codewords are chosen by some procedure after deciding their weight exactly. This number of codewords is also used in deciding the algorithm order of complexity by obtaining an indication of its running time.

Key words:
Binary cyclic code, minimum weights, code error detection.

المستخلص:

يهدف هذا البحث الى حل مشكلة تحديد اقل وزن للرمز الدوريالثنائي، اذ ان معرفة أقل وزن لرمز معين ضروري جدا وذلك لمعرفة قابليته على كشف وتصحيح الاخطاء التي تظهر في البيانات المخزونة او المنقولة. تم حل هذه المشكلة بتصميم خوارزمية كفوءة مستندة على بعض النتائج التي تم التوصل اليها في هذا البحث. الخوارزمية المقترحة تحتاج الى اقل عدد ممكن من الكلمات الرمز الدوري لحساب اوزانها والتي يتم اختيارها بطريقة معينة.بعد تحديد عدد هذه الكلمات بصورة مضبوطة، تتم الاستفادة من هذا العدد في تحديد درجة تعقيد الخوارزمية من حيث الوقت. كذلكتم توليد جدول يحتوي على الرموز الدورية الثنائية ذات الاطوال الفردية مع اقل اوزانها التي حددت قيمتها باستخدام الخوارزمية المقترحة.

1. Introduction

A group of vectors of length *k* and (2^k) in numbers, where *k* is the number of information digits, which are given the symbol *(n,k)*represent a linear block code. Each vector is a codeword in the code. An important code is cyclic code where a cyclic shift of any vector gives another vector in the code [14, 8, 7, 2]. The most important property of the code is the minimum weight (or hamming distance *d*), because it decides the error correcting capability of the code [27, 26, 22]. To estimate it attracted many researchers. To decide *d* is not an easy task [25, 24, 22,19]. An obvious method is to find all codewords and calculate the minimum weight. Obviously this is not practical. Another method is to use the parity check matrix. This is still impractical.

An important property of cyclic codes is to divide the code into classes each having the same weight [23]. This simplifies the problem since the number of vectors needs investigation is $[(2^k-1)/n]$. Many researchers gave importance to this approach [12, 9]. But still the number of codewords need investigation is not small. The difficulty in finding of made researchers resort to giving bounds on *d* only. Such as bounds for BCH, HT, Roos and Wilson [20, 13, 11, 8, 4, 3].still another approach is to find *d* for

specific codes. For example quadratic residue codes [6]. Jeffry.S leon [10] gave a probabilistic approach, but the algorithm running time rises exponentially with increase in *d*. Daniel and others[5] gave *d* for BCH codes by finding allocator polynomial satisfying certain condition. Daniel had the following to say

"*We have no systematic way of finding the true minimum distance. Of Couse there exist many bounds for cyclic codes, but these are not necessarily tight bounds. It is a difficult problem to find the true minimum distance of long BCH codes*"

This statement is still valid [26, 24, 23, 18]. Searching the literature it appears that methods for finding *d* for cyclic codes are:

1. Methods requiring exponential time.
2. Methods requiring involved theoretical procedures.
3. Methods which deal with specific code.
4. Methods which do not give exact values for *d*, but only bounds on its value.

Here a method for finding *d for cyclic codes. The algorithm derived is characterized by*,

1. Linear time execution.
2. Apply to any cyclic code.
3. Gives exact value for *d*.
4. The execution requires only a sequence of simple steps.

In section (2.1) some necessary definitions are given. In section (2.2) the main result is given two new theorems needed in the design of the algorithm are given. in section (3) an algorithm for the generation of information vectors is given. In section (4) the analysis of the algorithm is presented.

2. **Determining the minimum weight for Binary cyclic code:-**

2.1 Definitions:-

In what follows the following definitions are needed.

a. Cyclic W-class
   Is a group of binary vector generated from the binary vector

$$a = (a_0, \dots, a_{n-1})$$

   With weight W and length n, by n cyclic shift of a

b. $W(a)$.
   Is the weight of vector a is the number of ones in a.

c. $m_1, \dots, m_w$
   Is the position of ones in vector of weight W.

2.2 Determining the minimum weight (d):-

Finding (d) relies on results related to cyclic codes. In what follows two known lemmas are given. Then two new theorems which make the bases of the proposed algorithm are derived.

Lemma (1) [6,1]

For any real number k' such that $1 \leq k' \leq n$, any W-class for (n,k) cyclic code contains the vectors a, b with

$$W(a_0, \dots, a_{k-1}) = \lfloor k'\, w/n \rfloor$$
$$W(b_0, \dots, b_{k-1}) = \lceil k'\, w/n \rceil$$

Lemma (2) [1]

For R numbers, $L_1, \dots, L_r$, and $L \leq L_1 < \cdots < L_n \leq n$, any cyclic W-class contain the vectors a, b with

$$W(a_0, \dots, a_{L_i-1}) = \lfloor L_i\, w/n \rfloor$$
$$W(b_0, \dots, b_{L_{i-1}-1}) = \lceil L_i\, w/n \rceil$$

For i=1, 2, 3, …, r

Corollary (1) [1]

Any cyclic W- class contain vector a with

$$\boldsymbol{m_i \geq [i\, n/W], W \geq i \geq 1}$$

From above the following two theorems are devised

Theorem (1)

Any systematic cyclic C (n,k) having minimum weight(d)contains a code word C_w with weight W $n \geq w \geq d$, satisfy the condition.

(1) $w(a_0, \dots, a_{k-1}) = \lfloor kw/n \rfloor = wt$

(2) $m_i \geq \lceil {}^{in}/_{W} \rceil, wt \geq i \geq 1$

Proof:-

By encoding an information vector of length *k* which satisfy the above condition using the systematic generator matrix of code *C*. then a systematic code word satisfying the above condition can be obtained.

If code *C* has a W-class. Then the resulting code words belong to this class. This is due to lemma (1) and corollary (1). Thus this code word has weight *W*.

Theorem (2)

The minimum weight (d) for cyclic code *C(n,k)* must satisfy the following condition

$$\boldsymbol{d \geq \left[\frac{n}{k}\right]}$$

Proof:

If *k d*< *n* then the weight of information vector, lemma(1) is [k d/n]=0 i.e. the information vector is [00…0] and the corresponding code word is a zero vector of weight zero which is not equal to (*d*), since d>0.

Thus *k d*≥ *n* and then *d*≥ *[n/k]*.

This means no cyclic code of length *n* and dimension *K* has minimum weight less than [*n/k*]. This in fact a lower bound for minimum weight of (*n,k)* cyclic code. From above theorems the following corollary is deduced.

Corollary (2):-

The cyclic code *C* (*n,k)* has minimum weight equal *w* ilf *w* is the smallest possible value and *w*≥[*n/k*] such that C_w exists.

Proof:-

Obvious for if C_w does not exist then there is no code word of weight *w.*

The above result is used to design an algorithm for computing the minimum weight of cyclic codes starting with

$w = [n/k]$ Which is regarded as an initial value used to begin the trials. Then the codeword corresponding to the information vectors which satisfy the condition (1) and (2) of theorem (1) are generated. If there is any code word of weight *w* then *d* is equal to *w*, else if no. such code word exists, the process is repeated after increasing *w* by one.

Algorithm (1) (Computing (*d*) of binary cyclic codes)

Input: the systematic generator matrix of cyclic code *(n,k)*
Output: *d*
Algorithm:

```
Begin
    w=[n/k] {the lower bound}
    flag =false
repeat {Trials loop}
    wt=[kw/n]
    for i=1 to wt do
        m_i = [in/w]
    Generate the information vectors of weight wt and the nonzero
    Positions in each vectors≥ m_i for all wt≥i≥1
Generate the corresponding codewords by using the systematic generator matrix.
if there is any codeword of weight w then flag= true
  else
    if there is any codeword of weight w+1 then
      begin
        flag=true
        w=w+1
      end
      else
      w=w+1
    until flag
    d=w
end
```

Remark:-

Sometimes in two or more trials, the same set of information vectors is obtained that is when *wt* in the current trial is equal to wt in the previous trial. In this case, there is no need to generate the new set of vectors and its corresponding code words.

It is only needed to examine the set of weights of the previous trial. By this process, the number of needed codewords is greatly reduced in addition to algorithms running time.

3. Generation of information vectors:-

An important step in the implementation of the above algorithm is the generation of information vectors. To do this usually all the binary vectors (2^k) are generated. Then the vectors with weight *wt* are chosen.

From these the vectors satisfying the second condition in theorem (1) are taken. But this needs long time. Thus the following more efficient alterative way is used.

As is well brown the number of binary vectors of weight *wt* is [17,21,23]

C(k,wt)= k!/ (wt/(k-wt)!)

Therefore to generate the binary vectors, the generation of vectors of all combinations of *C(k,wt)* is obtained. Also to deal with binary numbers it is easier to represent the binary vectors using decimal notation. The decimal numbers used are the position of ones in there binary vectors. For example 1011 is represented from the left by 124.

Furthermore only vectors satisfying the second condition of theorem (1) are needed. According by the generation of combination *C (k,wt)* is modified as is shown in the following algorithm

Algorithm (2):- (generation of information vectors)

Input: k, wt, m, where m is an array with wt entries

Output: the set of combinations.

Algorithm:

```
Begin
    for i=1 to wt do
    begin
        x_i=k-wt+I {Last combination}
        y_i= m_i {First combination first development}
        if y_i=x_i then z_i=true
            z_i= false
    end
    while z_1=false do
        begin
            k_1=0
            for i=2 to wt do
                if not z_(i-1) and z_i then
                    begin
                    increment y_(i-1)
                    k_1=i
                    end
            if k_1=0 then increment y_wt
                else
                for i=k1 to wt do
                if (y_(i-1)≥m_i ) then {second development}
                    y_i= y_(k1-1)+(i-k1+1)
                    else y_i=m_i
                for i=1 to wt do
                    if y_i=x_i then z_i=true
                        else z_i=false
        end
end
```

Example:

If *k=5, wt= 3,m1=2 , m2=3, and m3=5*

Then C(k,wt)=C(5,3)= 5!/(3!+2!)=10

Therefore there are 10 binary vectors with *n=5.* Each vector is of weight 3.there are

135, 145, 234, 235, 245, 345, 123, 124,125, 134. In decimal.

In binary they are,

11100	11010	11001	10110	10101
10011				
01110	01101	01011	00111.	

All these vectors are not needed since only vectors with position of the first one≥2, the second one ≥3, and the third one≥5 are needed.

Thus the algorithm gives 345, 245, and 235 only representing 01101, 01011, and 00111 respectively.

Now if the generator matrix rows 2, 3, 5 are added vector 01101 is obtained. The same process may be applied to other rows thus the vectors which are a subnet of the rows of the generator matrix are the only rows needed to generate the necessary vectors whore weights need be investigated. Thus the output of the algorithm is a set of combinations each of which determines a certain set of rows of the systematic generator matrix that will be added together to produce the code words whose weight need be computed .

4. The analysis of algorithm[14]

The purpose is to predict the running time of the algorithm. First the number of trials and code words whose weights need to be computed is determined.

The number of k-tuples (information vectors) of weight *wt* is

C(k,wt)= (k!/wt!(k-wt)!). This number is greatly reduced by applying condition (2) of theorem (1)this number is denoted *C* . These information vectors are used to generate a set of code words whose weight of their check bits of length *n-k.*is determined.

The lower bound on the number of these vectors is *L=[n/k]* and the upper bound *U=1+n+k* (singleton bound). Therefore the number of trials is *T=U-L+1.* In each trial (*n-k*) tuples are needed. Let the number of vectors in the trial *i* be C_i then the maximum number of vectors needed is

$$\sum_{i=1}^{t} C_i$$

If *L<d* the number of trials is to *=d-L* and the number of (*n-k*) tuples needed is

$$\sum_{i=1}^{t_0} C_i$$

If $L=d$ *then* d is determined in the first trial and this is the best case. The number of *(n-k)* tuples C_i needed in trial i is determined by the following lemma.

Lemma (3):

The number of information vectors C_i needed in each trial i to generate the corresponding *(n-k)*-tuples that will be used for the weight computation is *C(n1, wt)-e, where n1=k-m1+1, and*

$$\sum_{j=1}^{wt-1} \sum_{s=m_j+1}^{m_{j-1}+1} [C(n1-s, wt-(j+1)) * ((s-1)-m_j-m_{j-1}-1]$$

Proof:

The number of *k-tuples* in which the first nonzero position $\geq m_1$ and have weight *wt* is *C(n1,wt)* where $n1=k-m_1+1$. By using condition (2) of theorem (1) this number is reduced and the number of k-tuples that are suppressed in the *C(n1,wt) k-tuples is C.* this is true since . If $m_{j+1}-m_j=1$ *for all* $0<j<wt$ *then* $e=0$ *if* $m_{j+1}-m_j>1$ *then for all S, where S is a digit whose occurrence is unwanted in the (j+1) the position of each combination representing a binary k-tuple and* $m_{j+1}-1 \geq S \geq m_j+1$. The combination in which S is in the (j+1) the position is eliminated. The number of such combinations is *C(n1-S, wt –j+1). This set* occurs (S-1) times in all *C(n1,wt)* combinations. Therefor *C(n1-S, wt-j+1)*is multiplied by(*S-1).* The quantity ($m_i-m_{j-1}-1$) represents the number of eliminations that were done for the set of combination containing 5 in (j+1) the position.

Example: k=5, wt=3, $m_1=1$ $m_2=4$, $m_3=5$ the set of all combination *C(5,3)=10*

For j=1 ⟶ $m_2-m_1=3$ *5=2,3*

for *s=2: C(n1-s), wt-(j+1)=C(3,1)=3*

s-1=2-1=1, $m_j-m_{j-1}-1=0$

The number of combination in which 2is the second element is 3and they are *123,124, 125.*

For s=3: *C(2,1)=2* *s-1=2*

*2*2=4 is the number of combination in which the second element is 3 and they are 134, 135, 234, and 235.*

$$\sum_{s=2}^{3} [C(5-s,1) * (s-1)] = 3+4 = 7$$

For j=2 ⟶ $m_3-m_2=5-4=1$ the number of combinations is zero.

Finally, *e=7+0=7 and 10-7=3 is the set of remaining combinations is: 145, 245, and 345.* This set is what is required.

Now the running time of the algorithm is an important factor. Unfortunately it is usually almost impossible to predict this time. Therefore, it is reasonable to attempt to only estimate it. An affection way to characterize the speed of an algorithm is to state how its execution time grows as a function of the size of a problem. Here the problem is the dimension of the generator matrix. This calculated time is usually called the time

complexity usually the *O* –notation is standard to evaluate the running time of an algorithm. This approach allows the proof of mathematical statements about the running time of the algorithm [15, 21, 23, and 26].
The *O*- notation allows ignoring the constant factors because it is independent of the inputs. Thus the time complexity of the algorithm is

$$O(c_i + wt_i)$$

Generating one *(n-k)* tuple that corresponds to information vector of weight *wt* requires adding *wt* rows of the generator matrix. Thus the time complexity of this is $O\big(wt_i(n-k)\big)$. Then the running time of the algorithm for computing (*d*) is

$$O\big(f(n,k)\big) = O\Big(\sum_{i=1}^{T}[C_i * \big(wt_i + wt_i(n-k)\big)]\Big)$$

$$= O\Big(\sum_{i=1}^{T}[C_i * wt_i(n-k))]\Big)$$

5. Examples

The algorithm for calculating (*d*) is used for some BCH codewords whose minimum weights are known [16].

Example (1):
If n=7, k=4, *g(x) =1011, w=2, d=3.*

Trial	w	wt	no.of codewords
1	2	1	1

Number of codewords=1
Number of trial =1
Here the start is with w=2. This is so since wt=1 and m_1=4. This number of vectors with length k is 1. The information vector is (0001) and the corresponding codeword is 0001011, but *w+1=3.* Thus *d=w+1=3.*

Example (2):
If n=15, k=5, g(x)=11101100101, w=3, d=7.

Trial	w	wt	no.of codewords
1	3	1	1
2	4	1	1
3	5	1	1
4	6	2	1

Example (3):

If n=31, k=6, g(x)=1110010001010111011010011, w=6, d=15.

Trial	w	wt	no.of codewords
1	6	1	1
2	7	1	1
3	8	1	1
4	9	1	0
5	10	1	0
6	11	2	3
7	12	2	0
8	13	2	2
9	14	3	0

Example (4):

If n=127, k=113, g(x)=111011101100001, w=2, d=5.

Trial	w	wt	no.of codewords
1	2	1	50
2	3	2	1624
3	4	3	107

6. Conclusion:

An algorithm for determining the minimum weight of binary cyclic code is obtained. It is characterized by its simplicity, since its execution requires the performance of non-complicated steps. The efficiency of the algorithm is indicated by having a linear time running as shown by its time complexity.

The algorithm is based on the idea of obtaining same cyclic codewords with special features presented by theorem (1). Then their weights are calculated. this is repeated until the condition in the lemma (2) are satisfied .the algorithm depends to large extent on the lower bound for ***d*** given by theorem (2), which in many cases is equal to the minimum weight.

The above means that in the algorithm, the least number of codewords weights is required for the calculation of the minimum weight. This is another parameter in its execution indicating its efficiency.

Reference:

[1] Alexander M.Barg and Hyal. Dumer, “on computing the weight spectrum of cyclic codes”, IEEE Trans. On information theory, Vol. 38, No. 4, july 192, pp. 1382-1386.

[2] Arnold M. Michelson and Allen H. Levesque, “Error-control techniques for digital communication”, john Wiley and Sons, NewYork, 1985.

[3]C.R.P Hartmann, “Some results on the minimum distance structure of cyclic code”, IEEE Trans. On Information Theory, Vol. 18,1972, pp. 439-440.

[4] Cornelis Roos, “A new Lower bound for the minimum distance of a cyclic code”, IEEE Trans. On Information Theory, Vol. IT-29, No. 3, May 1983,pp.330-332.

[5]Daniel Augot, Pascal Charpin, and Nicolas Sendrier, “Studying the locator polynomials of minimum weight codewords of BCH code”, IEEE Trans. On Information Theory, Vol. 38, No. 3, May 1992, pp. 960-973.

[6] Don Coppersmith and Gadiel Seroussi, " On them minimum distance of some quadratic residue codes", IEEE Trans. On Information Theory, Vol. IT- 30, No. 2, March 1984, pp.407-411.

[7] Elwyn R. Berlekamp, "Algebraic coding theory" McGrawHill, NewYork, 1986.

[8]Jacobus H. Van Lint and Richard M.Wilson, " On the minimum distance of cyclic codes" IEEE Trans. On Information Theory, Vol. IT- 32, No. 1, January 1986,pp. 23-40.

[9] Jessie Mac Williams and Judith seery, "The weight distributions of some minimal cyclic codes", IEEE Trans. On Information Theory, Vol. lT 27, No. 6, November 1981, pp. 796-800.

[10] Jeffrey S. Leon, "A probabilistic algorithm for computing minimum weights of larger error-correcting codes" ,IEEE Trans. On Information theory, Vol. 34,No.5 , September 1988, pp. 1354-1358.

[11] N. J. A. Sloane, " A short course on error correcting codes", International center for Mechanical sciences, Springer Verlag Wien-NewYork, 1975.

[12] P. E. Allard, S. G. S. Shiva and S. E. Tavares, " A note on the decomposition of cyclic codes into cyclic classes", Information and control 22,1973, pp. 100-106.

[13] P. J. N. deRooij and J.H. Van Lint, "More on the minimum distance of cyclic codes ", IEEE Trans. On Information Theory, Vol. 37, No. 1, January1 991,p p.1 87-189.

[14] Richard E. Blahut, "Theory and practice of error control codes ",Addison-Wesley, London,, 1983.

[15] Robert Sedgewick, "Algorithms", second edition, Addison-Wesley, Tyokyo,1988.

[16] ShuLin and Daniel J. Castello,Jr., " Error-control coding: fundamentals and applications", Prentice Hall, Inc. , New Jersey, 1983.

[17] S. Gao and J.D.Key, "Bases of minimum Weight Vectors Codes from Designs" Finite Fields Appl., vol.4, 1998.

[18] N.J Calkin JD.key, and M.J.D.E. Resmini, "Minimum Weights and Dimension Formulas for Some Geometric Codes" , Design, and Cryptography, vpl, 17, Sep, 1998.

[19]T.Baicheva, S.Dodunekov and P. Kazakov, "Undetected error probability performance of cyclic redundancy- check codes of 16-bit redundancy" , IEE proc-commun,, vol.147.No. 5, October2000.

[20] J. Quistorff, "Some Remarks on the Plotkin Bound", the Electronic Journal of Combinatorics, Vol. 10, 2003.

[21] J.I. Hall, 'Notes on Coding Theory", Departments of Mathematics, Michigan State University, Jan .2003.

[22] Philip Koopman, Tridib Chakravarty, "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks", The International Conference on Dependable Systems and Networks, DSN-2004.

[23] Gunsheng Ding, "The Weight Distribution of Some Irreducible Cyclic Codes", IEEE Xplore. Downloaded on February 25, 2009 at 23:36 from IEEE Xplore. Restrictions apply .

[24] M.Pujol and M .villanueva," Computing the Minimum Hamming Distance ", University de Barcelona, 20l2 .

[25] S.Draft, "Coping with Bit Error Using Error Correction Codes", MIT leeture Notes, Sep. 32, 2012.

[26] E.Pasalic, "Coding theory and Applications", University of primoreska. 2013.

[27] P.Srelatha et al, "Extended (10, 5) Binary Hamming Code Generator for Tele commanding Applications", International Journal of Soft Computing and Engineering volume-4, Issue-2, May 2014.

Appendix

Some cyclic code with their minimum weight calculated by the algorithm. The generator polynomials are given by the power of their non-zero terms.

n	k	d	g(x)
65	**64**	**2**	**10**
65	**53**	**5**	**12 10 9 8 6 4 3 2 0**
65	**61**	**2**	**4 3 2 1 0**
65	**53**	**5**	**12 10 7 6 5 2 0**
65	**53**	**5**	**12 8 7 6 5 4 0**
65	**53**	**5**	**12 11 9 7 6 5 3 10**
65	**53**	**2**	**12 11 10 9 8 7 6 5 4 3 2 1 0**
65	**52**	**6**	**13 12 11 8 7 6 5 2 1 0**
65	**60**	**2**	**5 0**
65	**52**	**6**	**13 12 11 10 8 5 3 2 1 0**
65	**52**	**6**	**13 12 9 4 1 0**
65	**52**	**6**	**13 11 10 9 8 5 4 3 2 0**
65	**52**	**2**	**13 0**
65	**49**	**6**	**16 15 13 11 9 8 7 5 3 1 0**
65	**41**	**5**	**24 21 18 14 13 12 11 10 6 3 0**
65	**41**	**5**	**24 22 21 19 18 17 16 14 13 12 11 10 8 7 6 5 3 2 0**
65	**41**	**5**	**18 16 15 12 9 8 6 5 3 2 1 0**
65	**41**	**8**	**24 23 21 18 17 15 12 9 7 6 3 10**
65	**49**	**6**	**16 15 10 9 8 7 6 1 0 16 15 14 13 10**
65	**49**	**6**	**8 6 3 2 1 0**
65	**49**	**4**	**16 14 12 11 10 9 8 7 6 5 4 2 0**
65	**41**	**5**	**24 22 20 18 17 13 12 11 7 6 4 2 0**
65	**41**	**5**	**24 23 22 19 18 16 15 14 13 12 11 10 9 8 6 5 2 1 0**
65	**41**	**8**	**24 23 19 17 16 15 12 9 8 7 5 1 0**
65	**41**	**8**	**24 21 20 18 15 14 12 10 9 6 4 3 0**
65	**48**	**6**	**17 15 14 13 12 11 10 7 6 5 4 3 2 0**
65	**40**	**8**	**25 24 22 21 19 18 15 10 7 6 4 3 1 0**