Chapter 5
Good Binary Linear Codes

5.1 Introduction

Two of the important performance indicators for a linear code are the minimum Hamming distance and the weight distribution. Efficient algorithms for computing the minimum distance and weight distribution of linear codes are explored below. Using these methods, the minimum distances of all binary cyclic codes of length 129–189 have been enumerated. The results are presented in Chap. 4. Many improvements to the database of best-known codes are described below. In addition, methods of combining known codes to produce good codes are explored in detail. These methods are applied to cyclic codes, and many new binary codes have been found and are given below.

The quest of achieving Shannon's limit for the AWGN channel has been approached in a number of different ways. Here we consider the problem formulated by Shannon of the construction of good codes which maximise the difference between the error rate performance for uncoded transmission and coded transmission. For uncoded, bipolar transmission with matched filtered reception, it is well known (see for example Proakis [20]) that the bit error rate, p_b, is given by

$$p_b = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right). \tag{5.1}$$

Comparing this equation with the equation for the probability of error when using coding, viz. the probability of deciding on one codeword rather than another, Eq. (1.4) given in Chap. 1, it can be seen that the improvement due to coding, the coding gain is indicated by the term $d_{min}.\frac{k}{n}$, the product of the minimum distance between codewords and the code rate. This is not the end of the story in calculating the overall probability of decoder error because this error probability needs to be multiplied by the number of codewords distance d_{min} apart.

For a linear binary code, the Hamming distance between two codewords is equal to the Hamming weight of the codeword formed by adding the two codewords together. Moreover, as the probability of decoder error at high $\frac{E_b}{N_0}$ values depends on the minimum Hamming distance between codewords, for a linear binary code,

M. Tomlinson et al., *Error-Correction Coding and Decoding*,
Signals and Communication Technology,
DOI 10.1007/978-3-319-51103-0_5

the performance of the code depends on the minimum Hamming weight codewords of the code, the d_{min} of the code and the number of codewords with this weight (the multiplicity). For a given code rate $(\frac{k}{n})$ and length n, the higher the weight of the minimum Hamming weight codewords of the code, the better the performance, assuming the multiplicity is not too high. It is for this reason that a great deal of research effort has been extended, around the world in determining codes with the highest minimum Hamming weight for a given code rate $(\frac{k}{n})$ and length n. These codes are called the best-known codes with parameters (n, k, d), where d is understood to be the d_{min} of the code, and the codes are tabulated in a database available online [12] with sometimes a brief description or reference to their method of construction.[1]

In this approach, it is assumed that a decoding algorithm either exists or will be invented which realises the full performance of a best-known code. For binary codes of length less than 200 bits the Dorsch decoder described in Chap. 15 does realise the full performance of the code.

Computing the minimum Hamming weight codewords of a linear code is, in general, a Nondeterministic Polynomial-time (NP) hard problem, as conjectured by [2] and later proved by [24]. Nowadays, it is a common practice to use a multi-threaded algorithm which runs on multiple parallel computers (grid computing) for minimum Hamming distance evaluation. Even then, it is not always possible to evaluate the exact minimum Hamming distance for large codes. For some algebraic codes, however, there are some shortcuts that make it possible to obtain the lower and upper bounds on this distance. But knowing these bounds are not sufficient as the whole idea is to know explicitly the exact minimum Hamming distance of a specific constructed code. As a consequence, algorithms for evaluating the minimum Hamming distance of a code are very important in this subject area and these are described in the following section.

It is worth mentioning that a more accurate benchmark of how good a code is, in fact its Hamming weight distribution. Whilst computing the minimum Hamming distance of a code is in general NP-hard, computing the Hamming weight distribution of a code is even more complex. In general, for two codes of the same length and dimension but of different minimum Hamming distance, we can be reasonably certain that the code with the higher distance is the superior code. Unless we are required to decide between two codes with the same parameters, including minimum Hamming distance, it is not necessary to go down the route of evaluating the Hamming weight distribution of both codes.

[1] Multiplicities are ignored in the compiling of the best, known code Tables with the result that sometimes the best, known code from the Tables is not the code that has the best performance.

5.2 Algorithms to Compute the Minimum Hamming Distance of Binary Linear Codes

5.2.1 The First Approach to Minimum Distance Evaluation

For a $[n, k, d]$ linear code over $\mathbb{F}_2$ with a reduced-echelon generator matrix $\boldsymbol{G}_{sys} = [\boldsymbol{I}_k|\boldsymbol{P}]$, where $\boldsymbol{I}_k$ and $\boldsymbol{P}$ are $k \times k$ identity and $k \times (n-k)$ matrices respectively, a codeword of this linear code can be generated by taking a linear combination of some rows of $\boldsymbol{G}_{sys}$. Since the minimum Hamming distance of a linear code is the minimum non-zero weight among all of the 2^k codewords, a brute-force method to compute the minimum distance is to generate codewords by taking

$$\binom{k}{1}, \binom{k}{2}, \binom{k}{3}, \ldots, \binom{k}{k-1}, \text{ and } \binom{k}{k}$$

linear combinations of the rows in $\boldsymbol{G}_{sys}$, noting the weight of each codeword generated and returning the minimum weight codeword of all $2^k - 1$ non-zero codewords. This method gives not only the minimum distance, but also the weight distribution of a code. It is obvious that as k grows larger this method becomes infeasible. However, if $n - k$ is not too large, the minimum distance can still be obtained by evaluating the weight distribution of the $[n, n-k, d']$ dual code and using the MacWilliams Identities to compute the weight distribution of the code. It should be noted that the whole weight distribution of the $[n, n-k, d']$ dual code has to be obtained, not just the minimum distance of the dual code.

In direct codeword evaluation, it is clear that there are too many unnecessary codeword enumerations involved. A better approach which avoids enumerating large numbers of unnecessary codewords can be devised. Let

$$\boldsymbol{c} = (\boldsymbol{i}|\boldsymbol{p}) = (c_0, c_1, \ldots, c_{k-1}|c_k, \ldots, c_{n-2}, c_{n-1})$$

be a codeword of a binary linear code of minimum distance d. Let $\boldsymbol{c}' = (\boldsymbol{i}'|\boldsymbol{p}')$ be a codeword of weight d, then if $\mathrm{wt}_H(\boldsymbol{i}') = w$ for some integer $w < d$, $\mathrm{wt}_H(\boldsymbol{p}') = d - w$. This means that at most

$$\sum_{w=1}^{\min\{d-1,k\}} \binom{k}{w} \tag{5.2}$$

codewords are required to be enumerated.

In practice, d is unknown and an upper bound d_{ub} on the minimum distance is required during the evaluation and the minimum Hamming weight found thus far can be used for this purpose. It is clear that once all $\sum_{w'=1}^{w} \binom{k}{w'}$ codewords of information weight w' are enumerated,

- we know that we have considered all possibilities of $d \leq w$; and
- if $w < d_{ub}$, we also know that the minimum distance of the code is at least $w + 1$.

Therefore, having an upper bound, a lower bound $d_{lb} = w + 1$ on the minimum distance can also be obtained. The evaluation continues until the condition $d_{lb} \geq d_{ub}$ is met and in this event, d_{ub} is the minimum Hamming distance.

5.2.2 *Brouwer's Algorithm for Linear Codes*

There is an apparent drawback of the above approach. In general, the minimum distance of a low-rate linear code is greater than its dimension. This implies that $\sum_{w=1}^{k} \binom{k}{w}$ codewords would need to be enumerated. A more efficient algorithm was attributed to Brouwer[2] and the idea behind this approach is to use a collection of generator matrices of mutually disjoint information sets [11].

Definition 5.1 (*Information Set*) Let the set $S = \{0, 1, 2, \ldots, n-\}$ be the coordinates of an $[n, k, d]$ binary linear code with generator matrix $\boldsymbol{G}$. The set $\mathscr{I} \subseteq S$ of k elements is an information set if the corresponding coordinates in the generator matrix is linearly independent and the submatrix corresponding to the coordinates in $\mathscr{I}$ has rank k, hence, it can be transformed into a $k \times k$ identity matrix.

In other words, we can say, in relation to a codeword, the k symbols user message is contained at the coordinates specified by $\mathscr{I}$ and the redundant symbols are stored in the remaining $n - k$ positions.

An information set corresponds to a reduced-echelon generator matrix and it may be obtained as follows. Starting with a reduced-echelon generator matrix $\boldsymbol{G}_{sys}^{(1)} = \boldsymbol{G}_{sys} = [\boldsymbol{I}_k|\boldsymbol{P}]$, Gaussian elimination is applied to submatrix $\boldsymbol{P}$ so that it is transformed to reduced-echelon form.

The resulting generator matrix now becomes $\boldsymbol{G}_{sys}^{(2)} = [\boldsymbol{A}|\boldsymbol{I}_k|\boldsymbol{P}']$, where $\boldsymbol{P}'$ is a $k \times (n - 2k)$ matrix. Next, submatrix $\boldsymbol{P}'$ is put into reduced-echelon form and the process continues until there exists a $k \times (n - lk)$ submatrix of rank less than k, for some integer l. Note that column permutations may be necessary during the transformation to maximise the number of disjoint information sets.

Let $\mathscr{G}$ be a collection of m reduced-echelon generator matrices of disjoint information sets, $\mathscr{G} = \left\{\boldsymbol{G}_{sys}^{(1)}, \boldsymbol{G}_{sys}^{(2)}, \ldots, \boldsymbol{G}_{sys}^{(m)}\right\}$.

Using these m matrices means that after $\sum_{w'=1}^{w} \binom{k}{w'}$ enumerations

- all possibilities of $d \leq mw$ have been considered; and
- if $mw < d_{ub}$, the minimum distance of the code is at least $m(w + 1)$, i.e. $d_{lb} = m(w + 1)$.

We can see that the lower bound has been increased by a factor of m, instead of 1 compared to the previous approach. For $w \leq k/2$, we know that $\binom{k}{w} \gg \binom{k}{w-1}$ and this lower bound increment reduces the bulk of computations significantly.

If d is the minimum distance of the code, the total number of enumerations required is given by

[2] Zimmermann attributed this algorithm to Brouwer in [25].

$$\sum_{w=1}^{\min\{\lceil d/m\rceil-1,k\}} m\binom{k}{w}. \tag{5.3}$$

Example 5.1 (*Disjoint Information Sets*) Consider the $[55, 15, 20]_2$ optimal binary linear, a shortened code of the Goppa code discovered by [15]. The reduced-echelon generator matrices of disjoint information sets are given by

$$\boldsymbol{G}_{sys}^{(1)} = \left[\begin{array}{c|c}
1000000000000000 & 0101101111110101001110101010101010100000000 \\
0100000000000000 & 1000111001101011110100100111001000110000 \\
0010000000000000 & 1011001111011010111010111111001001000001 \\
0001000000000000 & 1010101011101101101111111010001001010101 \\
0000100000000000 & 0011110100111111011011010001100010101011101 \\
0000010000000000 & 0101000010100101111111000111000101000111 0 \\
0000001000000000 & 1001001110100010100110011001010010101001 11 \\
0000000100000000 & 1011000100110100000001110010111001111010 1 \\
0000000010000000 & 1010110101111111010100100101110111010000 1 \\
0000000001000000 & 1010000101001011000001110101111101100010 \\
0000000000100000 & 1101101011110001001011111011100101010100 \\
0000000000010000 & 1101101111010111111001110111111000001101 1 \\
0000000000001000 & 0000010001010011101110011001100101101101 \\
0000000000000100 & 0101100000011101011110011000111001100000 11 \\
0000000000000010 & 0011111001011000111001010001000000111011
\end{array}\right],$$

$$\boldsymbol{G}_{sys}^{(2)} = \left[\begin{array}{c|c|c}
1011010010110 01 & 1000000000000000 & 0011110011110111110110000 \\
0000000110001 10 & 0100000000000000 & 1011000111011111000111101 0 \\
0011111001000 11 & 0010000000000000 & 0011101001010100111000001 01 \\
0101101001111 01 & 0001000000000000 & 1100101001010101011100111 011 \\
1111110010001 00 & 0000100000000000 & 0110011001100101101000100 \\
1111100101010 01 & 0000010000000000 & 0000000100011111110011000 1 \\
1111001000111 10 & 0000001000000000 & 1010010110110101011000111 01 \\
0000011001111 11 & 0000000100000000 & 1101000100101011100010001 \\
0000001010000 01 & 0000000010000000 & 1110110000011111100111101 \\
1110011001001 00 & 0000000001000000 & 1100100111100111011010111 \\
1000111111001 11 & 0000000000100000 & 0100001100100001000101110 \\
0101100001101 11 & 0000000000010000 & 1101110101101101011100100 \\
0010110111111 11 & 0000000000001000 & 0101010011001011111111110 \\
1001010110010 11 & 0000000000000100 & 0110000111001000110010011 \\
1101001011101 01 & 0000000000000010 & 0010100011000100001111100
\end{array}\right],$$

and

$$\boldsymbol{G}_{sys}^{(3)} = \left[\begin{array}{c|c|c}
0100101001101000101111110001110 & 1000000000000000 & 1100010001 \\
1111010111100001111101111110011 & 0100000000000000 & 1000110111 \\
0111011011110010111001110110100 & 0010000000000000 & 1101000101 \\
0011000100101010100100011111101 & 0001000000000000 & 0110011100 \\
0001111011010101011011010101011 & 0000100000000000 & 1011000101 \\
1011101010010001001101010010 00 & 0000010000000000 & 0110101011 \\
0101010001101000000101111001 10 & 0000001000000000 & 0011001111 \\
1001011011101111100001101101010 & 0000000100000000 & 0100000001 \\
1001001101010111100111001 11000 & 0000000010000000 & 1101111001 \\
0110011000111001111101110111111 & 0000000001000000 & 0111111010 \\
1011100011111000111011011011 11 & 0000000000100000 & 0111010111 \\
1000011011010100001011101101 10 & 0000000000001000 & 1000110010 \\
0111010100000100010111011010 00 & 0000000000000100 & 1101100011 \\
0100111111010001100010011001 & 0000000000000010 & 1101111111 \\
1101001111100111001100111101101 & 0000000000000001 & 0000011110
\end{array}\right].$$

Brouwer's algorithm requires 9948 codewords to be evaluated to prove the minimum distance of this code is 20. In contrast, for the same proof, 32767 codewords would need to be evaluated if only one generator matrix is employed.

5.2.3 Zimmermann's Algorithm for Linear Codes and Some Improvements

A further refinement to the minimum distance algorithm is due to Zimmermann [25]. Similar to Brouwer's approach, a set of reduced-echelon generator matrices are required. While in Brouwer's approach the procedure is stopped once a non-full-rank submatrix is reached; Zimmermann's approach proceeds further to obtain submatrices with overlapping information sets. Let $\boldsymbol{G}_{sys}^{(m)} = [\boldsymbol{A}_m | \boldsymbol{I}_k | \boldsymbol{B}_{m+1}]$ be the last generator matrix which contains a disjoint information set. To obtain matrices with overlapping information sets, Gaussian elimination is performed on the submatrix $\boldsymbol{B}_{m+1}$ and this yields

$$\boldsymbol{G}_{sys}^{(m+1)} = \left[\hat{\boldsymbol{A}}_m \left| \begin{array}{c|c} \boldsymbol{0} & \boldsymbol{I}_{r_{m+1}} \\ \hline \boldsymbol{I}_{k-r_{m+1}} & \boldsymbol{0} \end{array} \right| \boldsymbol{B}_{m+2}\right],$$

where $r_{m+1} = \text{Rank}\,(\boldsymbol{B}_{m+1})$. Next, $\boldsymbol{G}_{sys}^{(m+2)}$ is produced by carrying out Gaussian elimination on the submatrix $\boldsymbol{B}_{m+2}$ and so on.

From $\boldsymbol{G}_{sys}^{(3)}$ of Example 5.1, we can see that the last 10 coordinates do not form an information set since the rank of this submatrix is clearly less than k. Nonetheless, a "partial" reduced-echelon generator matrix can be obtained from $\boldsymbol{G}_{sys}^{(3)}$,

$$\boldsymbol{G}_{sys}^{(4)} = \left[\begin{array}{c|c|c}
0111111001011010010100110001100101010000 & 00000 & 1000000000 \\
1100110000010110000010111110000110101110 & 00000 & 0100000000 \\
1010010000110001011100111001110100001110 & 00000 & 0010000000 \\
0010010010001100001001001001111011101111 & 00000 & 0001000000 \\
1011110010101000101110010011100001100010 & 00000 & 0000100000 \\
1010001011011101111000001000101101010100 & 00000 & 0000010000 \\
0100000100011011101111011111110010001111 & 00000 & 0000001000 \\
1011101011001011101101001111111100111011 & 00000 & 0000000100 \\
0111010111111100010011111110100011111010111 & 00000 & 0000000010 \\
0101101011111001000001100100100111010101010 & 00000 & 0000000001 \\
1100001101000000101011001011001001111101 & 10000 & 0000000000 \\
1001001100000111011010111010001110010001 & 01000 & 0000000000 \\
0111000001101001110100010110011000101110 & 00100 & 0000000000 \\
0001001111000011011010000101010110011110 & 00010 & 0000000000 \\
1111111100111111111000001111011010000101111 & 00001 & 0000000000
\end{array}\right].$$

From $\boldsymbol{G}_{sys}^{(4)}$, we can see that the last k columns is also an information set, but $k - \text{Rank}\left(\boldsymbol{G}_{sys}^{(4)}\right)$ coordinates of which overlap with those in $\boldsymbol{G}_{sys}^{(3)}$. The generator matrix $\boldsymbol{G}_{sys}^{(4)}$ then may be used to enumerate codewords with condition that the effect of overlapping information set has to be taken into account.

Assuming that all codewords with information weight $\leq w$ have been enumerated, we know that

- for all $\boldsymbol{G}_{sys}^{(i)}$ of full-rank, say there are m of these matrices, all cases of $d \leq mw$ have been considered and each contributes to the lower bound.
  As a result, the lower bound becomes $d_{lb} = m(w+1)$.
- for each $\boldsymbol{G}_{sys}^{(i)}$ that do not have full-rank, we can join $\boldsymbol{G}_{sys}^{(i)}$ with column subsets of $\boldsymbol{G}_{sys}^{(j)}$, for $j < i$, so that we have an information set $\mathscr{I}_i$, which of course overlaps with information set $\mathscr{I}_j$. Therefore, for all of these matrices, say M, all cases of $d \leq Mw$ have been considered, but some of which are attributed to other information sets, and considering these would result in double counting.

According to Zimmermann [25], for each matrix $\boldsymbol{G}_{sys}^{(m+j)}$ with an overlapping information set unless $w \geq k - \text{Rank}\left(\boldsymbol{B}_{m+j}\right)$ for which the lower bound becomes $d_{lb} = d_{lb} + \left\{w - \left(k - \text{Rank}\left(\boldsymbol{B}_{m+j}\right)\right) + 1\right\}$, there is no contribution to the lower bound.

Let the collection of full rank-reduced echelon matrices be denoted by, as before, $\mathscr{G} = \left\{\boldsymbol{G}_{sys}^{(1)}, \boldsymbol{G}_{sys}^{(2)}, \ldots, \boldsymbol{G}_{sys}^{(m)}\right\}$, and let $\mathscr{G}'$ denote the collection of M rank matrices with overlapping information sets $\mathscr{G}' = \left\{\boldsymbol{G}_{sys}^{(m+1)}, \boldsymbol{G}_{sys}^{(m+2)}, \ldots, \boldsymbol{G}_{sys}^{(m+M)}\right\}$. All $m + M$ generator matrices are needed by the [25] algorithm. Clearly, if the condition $w \geq k - \text{Rank}\left(\boldsymbol{B}_{m+j}\right)$ is never satisfied throughout the enumeration, the corresponding generator matrix contributes nothing to the lower bound and, hence, can be excluded [11]. In order to accommodate this improvement, we need to know w_{max} the maximum information weight that would need to be enumerated before the minimum distance is found. This can be accomplished as follows: Say at information weight w, a lower weight codeword is found, i.e. new d_{ub}, starting from $w' = w$, we let $\mathscr{X} = \mathscr{G}'$, set $d_{lb} = m(w' + 1)$ and then increment it by $(w' - (k - \text{Rank}(\boldsymbol{B}_{m+j})) + 1)$ for each matrix in $\mathscr{G}'$ that satisfies $w' \geq k - \text{Rank}\left(\boldsymbol{B}_{m+j}\right)$. Each matrix that satisfies this condition is also excluded from $\mathscr{X}$. The weight w' is incremented, d_{lb} is recomputed and at the point when $d_{lb} \geq d_{ub}$, we have w_{max} and all matrices in $\mathscr{X}$ are those to be excluded from codeword enumeration.

In some cases, it has been observed that while enumerating codewords of information weight w, a codeword, whose weight coincides with the lower bound obtained at enumeration step $w - 1$, appears. Clearly, this implies that the newly found codeword is indeed a minimum weight codeword; any other codeword of lower weight, if they exist, would have been found in the earlier enumeration steps. This suggests that the enumeration at step w may be terminated immediately. Since the bulk of computation time increases exponentially as the information weight is increased, this termination may result in a considerable saving of time.

Without loss of generality, we can assume that $\text{Rank}(\boldsymbol{B}_{m+j}) > \text{Rank}(\boldsymbol{B}_{m+j+1})$. With this consideration, we can implement the Zimmermann approach to minimum distance evaluation of linear code over $\mathbb{F}_2$–with the improvements, in Algorithm 5.1. The procedure to update w_{max} and $\mathscr{X}$ is given in Algorithm 5.2.

If there is additional code structure, the computation time required by Algorithm 5.1 can be reduced. For example, in some cases it is known that the binary code considered has even weight codewords only, then at the end of codeword enumeration at each step, the lower bound d_{lb} that we obtained may be rounded down to the next multiple of 2. Similarly, for codes where the weight of every codeword is divisible by 4, the lower bound may be rounded down to the next multiple of 4.

5.2.4 Chen's Algorithm for Cyclic Codes

Binary cyclic codes, which were introduced by Prange [19], form an important class of block codes over $\mathbb{F}_2$. Cyclic codes constitute many well-known error-

Algorithm 5.1 Minimum distance algorithm: improved Zimmermann's approach

Input: $\mathscr{G} = \left\{ \boldsymbol{G}_{sys}^{(1)}, \boldsymbol{G}_{sys}^{(2)}, \ldots, \boldsymbol{G}_{sys}^{(m)} \right\}$ where $|\mathscr{G}| = m$
Input: $\mathscr{G}' = \left\{ \boldsymbol{G}_{sys}^{(m+1)}, \boldsymbol{G}_{sys}^{(m+2)}, \ldots, \boldsymbol{G}_{sys}^{(m+M)} \right\}$ where $|\mathscr{G}'| = M$
Output: d (minimum distance)

$d' \leftarrow d_{ub} \leftarrow w_{max} \leftarrow k$
$d_{lb} \leftarrow w \leftarrow 1$
$\mathscr{X} = \emptyset$
repeat
  $M \leftarrow M - |\mathscr{X}|$
  for all $\boldsymbol{i} \in \mathbb{F}_2^k$ where $\mathrm{wt}_H(\boldsymbol{i}) = w$ **do**
    for $1 \leq j \leq m$ **do**
      $d' \leftarrow \mathrm{wt}_H(\boldsymbol{i} \cdot \boldsymbol{G}_{sys}^{(j)})$
      if $d' < d_{ub}$ **then**
        $d_{ub} \leftarrow d'$
        if $d_{ub} \leq d_{lb}$ **then**
          Goto Step 36
        end if
        $w_{max}, \mathscr{X} \leftarrow$ Update w_{max} and $\mathscr{X}\left(d_{ub}, k, m, \mathscr{G}'\right)$
      end if
    end for
    for $1 \leq j \leq M$ **do**
      $d' \leftarrow \mathrm{wt}_H(\boldsymbol{i} \cdot \boldsymbol{G}_{sys}^{(m+j)})$
      if $d' < d_{ub}$ **then**
        $d_{ub} \leftarrow d'$
        if $d_{ub} \leq d_{lb}$ **then**
          Goto Step 36
        end if
        $w_{max}, \mathscr{X} \leftarrow$ Update w_{max} and $\mathscr{X}\left(d_{ub}, k, m, \mathscr{G}'\right)$
      end if
    end for
  end for
  $d_{lb} \leftarrow m(w+1)$
  for $1 \leq j \leq M$ **do**
    if $w \geq \left\{k - \mathrm{Rank}\left(\boldsymbol{B}_{m+j}\right)\right\}$ **then**
      $d_{lb} = d_{lb} + \left\{w - \left(k - \mathrm{Rank}\left(\boldsymbol{B}_{m+j}\right)\right) + 1\right\}$
    end if
  end for
  $w \leftarrow w + 1$
until $d_{lb} \geq d_{ub}$ OR $w > k$
$d \Leftarrow d_{ub}$

correcting codes, such as the quadratic-residue codes and the commonly used in practice Bose–Chaudhuri–Hocquenghem (BCH) and Reed–Solomon (RS) codes. A binary cyclic code of length n, where n is necessarily odd, has the property that if $c(x) = \sum_{i=0}^{n-1} c_i x^i$, where $c_i \in \mathbb{F}_2$ is a codeword of the cyclic code, then $x^j c(x) \pmod{x^n - 1}$, for some integer j, is also a codeword of that cyclic code. That is to say that the automorphism group of a cyclic code contains the coordinate permutation $i \rightarrow i + 1 \pmod{n}$.

Algorithm 5.2 $w_{max}, \mathscr{X} = \text{Update } w_{max} \text{ and } \mathscr{X}\left(d_{ub}, k, m, \mathscr{G}'\right)$

Input: $d_{ub},\ k,\ m$
Input: $\mathscr{G}'\left\{\boldsymbol{G}_{sys}^{(m+1)}, \boldsymbol{G}_{sys}^{(m+2)}, \ldots, \boldsymbol{G}_{sys}^{(m+M)}\right\}$
Output: w_{max} and $\mathscr{X}$
1: $\mathscr{X} \leftarrow \mathscr{G}'$
2: $w_{max} \leftarrow 1$
3: **repeat**
4: $\quad d_{lb} \leftarrow m(w_{max}+1)$
5: $\quad$ **for** $1 \le j \le |\mathscr{G}'|$ **do**
6: $\quad\quad$ **if** $w_{max} \ge \left\{k - \text{Rank}\left(\boldsymbol{B}_{m+j}\right)\right\}$ **then**
7: $\quad\quad\quad$ Remove $\boldsymbol{G}_{sys}^{(m+j)}$ from $\mathscr{X}$ if $\boldsymbol{G}_{sys}^{(m+j)} \in \mathscr{X}$
8: $\quad\quad\quad d_{lb} = d_{lb} + \left\{w_{max} - \left(k - \text{Rank}\left(\boldsymbol{B}_{m+j}\right)\right) + 1\right\}$
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: $\quad w_{max} \leftarrow w_{max} + 1$
12: **until** $d_{lb} \ge d_{ub}$ OR $w_{max} > k$
13: **return** w_{max} and $\mathscr{X}$

An $[n, k, d]$ binary cyclic code is defined by a generator polynomial $g(x)$ of degree $n-k$, and a parity-check polynomial $h(x)$ of degree k, such that $g(x)h(x) = 0 \pmod{x^n - 1}$. Any codeword of this cyclic code is a multiple of $g(x)$, that is $c(x) = u(x)g(x)$, where $u(x)$ is any polynomial of degree less than k. The generator matrix $\boldsymbol{G}$ can be simply formed from the cyclic shifts of $g(x)$, i.e.

$$\boldsymbol{G} = \begin{bmatrix} g(x) & (\text{mod } x^n - 1) \\ xg(x) & (\text{mod } x^n - 1) \\ & \vdots \\ x^{k-1}g(x) & (\text{mod } x^n - 1) \end{bmatrix}. \tag{5.4}$$

Since for some integer i, $x^i = q_i(x)g(x) + r_i(x)$ where $r_i(x) = x^i \pmod{g(x)}$, we can write

$$x^k\left(x^{n-k+i} - r_{n-k+i}(x)\right) = x^k q_i(x)g(x)$$

and based on this, a reduced-echelon generator matrix $\boldsymbol{G}_{sys}$ of a cyclic code is obtained:

$$\boldsymbol{G}_{sys} = \left[\begin{array}{c|c} & -x^{n-k} \pmod{g(x)} \\ & -x^{n-k+1} \pmod{g(x)} \\ \boldsymbol{I}_k & -x^{n-k+2} \pmod{g(x)} \\ & \vdots \\ & -x^{n-1} \pmod{g(x)} \end{array}\right]. \tag{5.5}$$

The matrix $\boldsymbol{G}_{sys}$ in (5.5) may contain several mutually disjoint information sets. But because each codeword is invariant under a cyclic shift, a codeword generated by information set $\mathscr{I}_i$ can be obtained from information set $\mathscr{I}_j$ by means of a simple cyclic shift. For an $[n, k, d]$ cyclic code, there always exists $\lfloor n/k \rfloor$ mutually disjoint information sets. As a consequence of this, using a single information set is sufficient to improve the lower bound to $\lfloor n/k \rfloor (w+1)$ at the end of enumeration step w. However, Chen [7] showed that this lower bound could be further improved by noting that the average number of non-zeros of a weight w_0 codeword in an information set is $w_0 k/n$. After enumerating $\sum_{i=1}^{w} \binom{k}{i}$ codewords, we know that the weight of a codeword restricted to the coordinates specified by an information set is at least $w+1$. Relating this to the average weight of codeword in an information set, we have an improved lower bound of $d_{lb} = \lceil (w+1)n/k \rceil$. Algorithm 5.3 summarises Chen's [7] approach to minimum distance evaluation of a binary cyclic code. Note that Algorithm 5.3 takes into account the early termination condition suggested in Sect. 5.2.3.

Algorithm 5.3 Minimum distance algorithm for cyclic codes: Chen's approach

Input: $\boldsymbol{G}_{sys} = [\boldsymbol{I}_k | \boldsymbol{P}]$ {see (5.5)}
Output: d (minimum distance)

$d_{ub} \leftarrow k$
$d_{lb} \leftarrow 1$
$w \leftarrow 1$
repeat
  $d' \leftarrow k$
  for all $\boldsymbol{i} \in \mathbb{F}_2^k$ where $\mathrm{wt}_H(\boldsymbol{i}) = w$ **do**
    $d' \leftarrow \mathrm{wt}_H(\boldsymbol{i} \cdot \boldsymbol{G}_{sys})$
    if $d' < d_{ub}$ **then**
      $d_{ub} \leftarrow d'$
      if $d_{ub} \leq d_{lb}$ **then**
        Goto Step 18
      end if
    end if
  end for
  $d_{lb} \leftarrow \left\lceil \frac{n}{k}(w+1) \right\rceil$
  $w \leftarrow w+1$
until $d_{lb} \geq d_{ub}$ OR $w > k$
$d \Leftarrow d_{ub}$

It is worth noting that both minimum distance evaluation algorithms of Zimmermann [25] for linear codes and that of Chen [7] for cyclic codes may be used to compute the number of codewords of a given weight. In evaluating the minimum distance d, we stop the algorithm after enumerating all codewords having information weight i to w, where w is the smallest integer at which the condition $d_{lb} \geq d$ is reached. To compute the number of codewords of weight d, in addition to enumerating all codewords of weight i to w in their information set, all codewords having weight $w+1$ in their information set, also need to be enumerated. For Zimmermann's

method, we use all of the available information sets, including those that overlap, and store all codewords whose weight matches d. In contrast to Chen's algorithm, we use only a single information set but for each codeword of weight d found, we accumulate this codeword and all of the $n-1$ cyclic shifts. In both approaches, it is necessary to remove the doubly-counted codewords at the end of the enumeration stage.

5.2.5 *Codeword Enumeration Algorithm*

The core of all minimum distance evaluation and codeword counting algorithms lies in the codeword enumeration. Given a reduced-echelon generator matrix, codewords can be enumerated by taking linear combinations of the rows in the generator matrix. This suggests the need for an efficient algorithm to generate combinations.

One of the most efficient algorithm for this purpose is the revolving-door algorithm, see [4, 13, 17]. The efficiency of the revolving-door algorithm arises from the property that in going from one combination pattern to the next, there is only one element that is exchanged. An efficient implementation of the revolving-door algorithm is given in [13], called *Algorithm R*, which is attributed to [18].[3]

In many cases, using a single-threaded program to either compute the minimum distance, or count the number of codewords of a given weight, of a linear code may take a considerable amount of computer time and can take several weeks.

For these long codes, we may resort to a multi-threaded approach by splitting the codeword enumeration task between multiple computers. The revolving-door algorithm has a nice property that allows such splitting to be neatly realised. Let $a_t a_{t-1} \ldots a_2 a_1$, where $a_t > a_{t-1} > \ldots > a_2 > a_1$ be a pattern of an t out of s combinations–C_t^s. A pattern is said to have rank i if this pattern appears as the $(i+1)$th element in the list of all C_t^s combinations.[4] Let $\text{Rank}(a_t a_{t-1} \ldots a_2 a_1)$ be the rank of pattern $a_t a_{t-1} \ldots a_2 a_1$, the revolving-door algorithm has the property that [13]

$$\text{Rank}(a_t a_{t-1} \ldots a_2 a_1) = \left[\binom{a_t + 1}{t} - 1 \right] - \text{Rank}(a_{t-1} \ldots a_2 a_1) \tag{5.6}$$

and, for each integer N, where $0 \leq N \leq \binom{s}{t} - 1$, we can represent it uniquely with an ordered pattern $a_t a_{t-1} \ldots a_2 a_1$. As an implication of this and (5.6), if all $\binom{k}{t}$ codewords need to be enumerated, we can split the enumeration into $\left\lceil \binom{k}{t} / M \right\rceil$ blocks, where in each block only at most M codewords need to be generated. In

[3] This is the version that the authors implemented to compute the minimum distance and to count the number of codewords of a given weight of a binary linear code.

[4] Here it is assume that the first element in the list of all C_t^s combinations has rank 0.

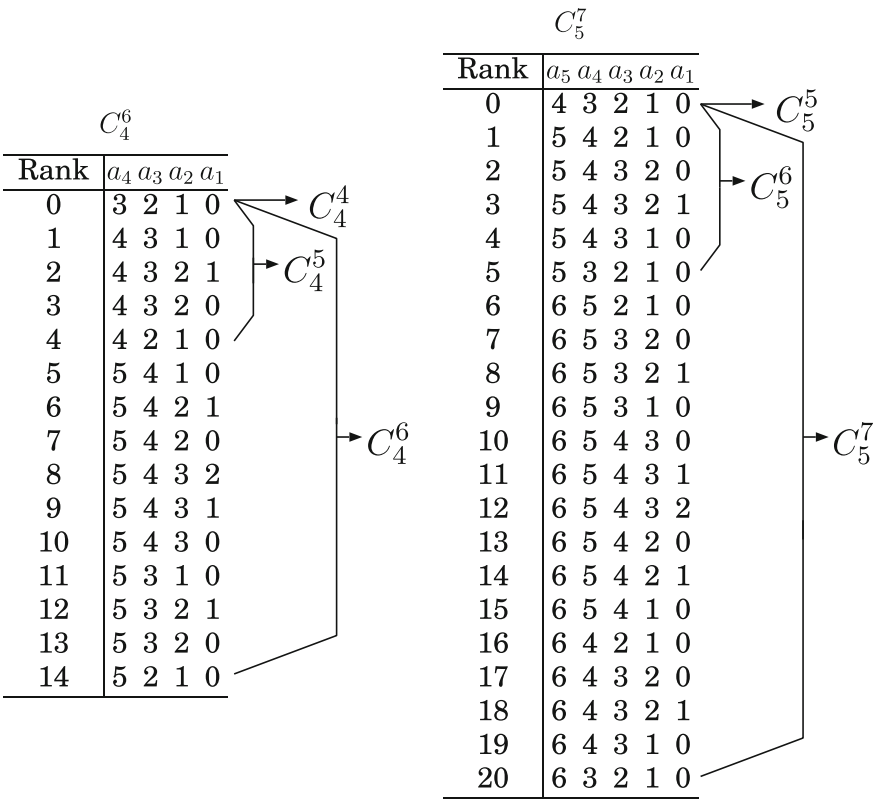


Fig. 5.1 C_4^6 and C_5^7 revolving-door combination patterns

this way, we can do the enumeration of each block on a separate computer and this allows a parallelisation of the minimum distance evaluation, as well as the counting of the number of codewords of a given weight. We know that at the ith block, the enumeration would start from rank $(i-1)M$, and the corresponding pattern can be easily obtained following (5.6) and Lemma 5.1 below.

All $a_t a_{t-1} \ldots a_2 a_1$ revolving-door patterns of C_t^s satisfy the property that if the values in position a_t grow in an increasing order, then for fixed a_t, the values in position a_{t-1} grow in a decreasing order, moreover for fixed $a_t a_{t-1}$ the values in position a_{t-2} grow in an increasing order, and so on in an alternating order. This behaviour is evident by observing all revolving-door patterns of C_4^6 (left) and C_5^7 (right) shown in Fig. 5.1.

From this figure, we can also observe that

$$C_t^s \supset C_t^{s-1} \supset \ldots \supset C_t^{t+1} \supset C_t^t\,, \tag{5.7}$$

and this suggests the following lemma.

Lemma 5.1 (Maximum and Minimum Ranks) *Consider the $a_t a_{t-1} \ldots a_2 a_1$ revolving-door combination pattern, if we consider patterns with fixed a_t, the maximum and minimum ranks of such pattern are respectively given by*

$$\binom{a_t+1}{t} - 1 \qquad \textit{and} \qquad \binom{a_t}{t}.$$

Example 5.2 (*Maximum and Minimum Ranks*) Say, if we consider all C_4^6 revolving-door combination patterns (left portion of Fig. 5.1) where $a_4 = 4$. From Lemma 5.1, we have a maximum rank of $\binom{5}{4} - 1 = 4$, and a minimum rank of $\binom{4}{4} = 1$. We can see that these rank values are correct from Fig. 5.1.

Example 5.3 (*The Revolving-Door Algorithm*) Consider combinations C_5^7 generated by the revolving-door algorithm, we would like to determine the rank of combination pattern 17. We know that the combination pattern takes the ordered form of $a_5 a_4 a_3 a_2 a_1$, where $a_i > a_{i-1}$. Starting from a_5, which can takes values from 0 to 6, we need to find a_5 such that the inequality $\binom{a_5}{5} \leq 17 \leq \binom{a_5+1}{5} - 1$ is satisfied (Lemma 5.1). It follows that $a_5 = 6$ and using (5.6), we have

$$\begin{aligned} 17 &= \text{Rank}(6a_4a_3a_2a_1) \\ &= \left[\binom{6+1}{5} - 1\right] - \text{Rank}(a_4a_3a_2a_1) \\ \text{Rank}(a_4a_3a_2a_1) &= 20 - 17 = 3\,. \end{aligned}$$

Next, we consider a_4 and as before, we need to find $a_4 \in \{5, 4, 3, 2, 1, 0\}$ such that the inequality $\binom{a_4}{4} \leq \text{Rank}(a_4a_3a_2a_1) \leq \binom{a_4+1}{4} - 1$ is satisfied. It follows that $a_4 = 4$ and from (5.6), we have

$$\begin{aligned} 3 &= \text{Rank}(4a_3a_2a_1) \\ &= \left[\binom{4+1}{4} - 1\right] - \text{Rank}(a_3a_2a_1) \\ \text{Rank}(a_3a_2a_1) &= 4 - 3 = 1\,. \end{aligned}$$

Next, we need to find a_3, which can only take a value less than 4, such that the inequality $\binom{a_3}{3} \leq \text{Rank}(a_3a_2a_1) \leq \binom{a_3+1}{3} - 1$ is satisfied. It follows that $a_3 = 3$ and from (5.6), $\text{Rank}(a_2a_1) = \left[\binom{3+1}{3} - 1\right] - 1 = 2$.

So far we have $643a_2a_1$, only a_2 and a_1 are unknown. Since $a_3 = 3$, a_2 can only take a value less than 3. The inequality $\binom{a_2}{2} \leq \text{Rank}(a_2a_1) \leq \binom{a_2+1}{2} - 1$ is satisfied if $a_2 = 2$ and correspondingly, $\text{Rank}(a_1) = \left[\binom{2+1}{2} - 1\right] - 2 = 0$.

For the last case, the inequality $\binom{a_1}{1} \leq \text{Rank}(a_1) \leq \binom{a_1+1}{1} - 1$ is true if and only if $a_1 = 0$. Thus, we have 64320 as the rank 17 C_5^7 revolving-door pattern. Cross-checking this with Fig. 5.1, we can see that 64320 is indeed of rank 17.

From (5.6) and Example 5.3, we can see that given a rank N, where $0 \leq N \leq \binom{s}{t} - 1$, we can construct an ordered pattern of C_t^s revolving-door combinations $a_t a_{t-1} \ldots a_2 a_1$, recursively. A software realisation of this recursive approach is given in Algorithm 5.4.

Algorithm 5.4 Recursively Compute a_i $(\text{Rank}(a_i a_{i-1} \ldots a_2 a_1), i)$

Input: i and $\text{Rank}(a_i a_{i-1} \ldots a_2 a_1)$
Output: a_i

1: Find a_i, where $0 \leq a_i < a_{i+1}$, such that $\binom{a_i}{i} \leq \text{Rank}(a_i a_{i-1} \ldots a_2 a_1) \leq \left[\binom{a_i+1}{i} - 1\right]$
2: **if** $i > i$ **then**
3: Compute $\text{Rank}(a_{i-1} \ldots a_2 a_1) = \left[\binom{a_i+1}{i} - 1\right] - \text{Rank}(a_i a_{i-1} \ldots a_2 a_1)$
4: RecursiveCompute a_i $(\text{Rank}(a_{i-1} \ldots a_2 a_1), i - 1)$
5: **end if**
6: **return** a_i

5.3 Binary Cyclic Codes of Lengths $129 \leq n \leq 189$

The minimum distance of all binary cyclic codes of lengths less than or equal to 99 has been determined by Chen [7, 8] and Promhouse et al. [21].

This was later extended to longer codes with the evaluation of the minimum distance of binary cyclic codes of lengths from 101 to 127 by Schomaker et al. [22]. We extend this work to include all cyclic codes of odd lengths from 129 to 189 in this book. The aim was to produce a Table of codes as a reference source for the highest minimum distance, with the corresponding roots of the generator polynomial, attainable by all cyclic codes over $\mathbb{F}_2$ of odd lengths from 129 to 189. It is well known that the coordinate permutation $\sigma : i \to \mu i$, where μ is an integer relatively prime to n, produces equivalent cyclic codes [3, p. 141f]. With respect to this property, we construct a list of generator polynomials $g(x)$ of all inequivalent and non-degenerate [16, p. 223f] cyclic codes of $129 \leq n \leq 189$ by taking products of the irreducible factors of $x^n - 1$. Two trivial cases are excluded, namely $g(x) = x + 1$ and $g(x) = (x^n - 1)/(x + 1)$, since these codes have trivial minimum distance and exist for any odd integer n. The idea is for each $g(x)$ of cyclic codes of odd length n; the systematic generator matrix is formed and the minimum distance of the code is determined using Chen's algorithm (Algorithm 5.3). However, due to the large number of cyclic codes and the fact that we are only interested in those of

largest minimum distance for given n and k, we include a threshold distance d_{th} in Algorithm 5.3. Say, for given n and k, we have a list of generator polynomials $g(x)$ of all inequivalent cyclic codes. Starting from the top of the list, the minimum distance of the corresponding cyclic code is evaluated. If a codeword of weight less than or equal to d_{th} is found during the enumeration, the computation is terminated immediately and the next $g(x)$ is then processed. The threshold d_{th}, which is initialised with 0, is updated with the largest minimum distance found so far for given n and k.

Table 4.3 in Sect. 4.5 shows the highest attainable minimum distance of all binary cyclic codes of odd lengths from 129 to 189. The number of inequivalent and non-degenerate cyclic codes for a given odd integer n, excluding the two trivial cases mentioned above, is denoted by $N_{\mathscr{C}}$.

Note that Table 4.3 does not contain entries for primes $n = 8m \pm 3$. This is because for these primes, 2 is not a quadratic residue modulo n and hence, $\text{ord}_2(n) = n - 1$. As a consequence, $x^n - 1$ factors into two irreducible polynomials only, namely $x + 1$ and $(x^n - 1)/(x + 1)$ which generate trivial codes. Let β be a primitive nth root of unity, the roots of $g(x)$ of a cyclic code (excluding the conjugate roots) are given in terms of the exponents of β. The polynomial $m(x)$ is the minimal polynomial of β and it is represented in octal format with most significant bit on the left. That is, $m(x) = 166761$, as in the case for $n = 151$, represents $x^{15} + x^{14} + x^{13} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^5 + x^4 + 1$.

5.4 Some New Binary Cyclic Codes Having Large Minimum Distance

Constructing an $[n, k]$ linear code possessing the largest minimum distance is one of the main problems in coding theory. There exists a database containing the lower and upper bounds of minimum distance of binary linear codes of lengths $1 \leq n \leq 256$. This database appears in [6] and the updated version is accessible online.[5]

The lower bound corresponds to the largest minimum distance for a given $[n, k]_q$ code that has been found to date. Constructing codes which improves Brouwer's lower bounds is an on-going research activity in coding theory. Recently, Tables of lower- and upper-bounds of not only codes over finite-fields, but also quantum error-correcting codes, have been published by Grassl [12]. These bounds for codes over finite-fields, which are derived from MAGMA [5], appear to be more up-to-date than those of Brouwer.

We have presented in Sect. 5.3, the highest minimum distance attainable by all binary cyclic codes of odd lengths from 129 to 189 and found none of these cyclic codes has larger minimum distance than the corresponding Brouwer's lower bound for the same n and k. The next step is to consider longer length cyclic codes, $191 \leq$

[5]The database is available at http://www.win.tue.nl/~aeb/voorlincod.html.

Note that, since 12th March 2007, A. Brouwer has stopped maintaining his database and hence it is no longer accessible. This database is now superseded by the one maintained by Grassl [12].

$n \leq 255$. For these lengths, unfortunately, we have not been able to repeat the exhaustive approach of Sect. 5.3 in a reasonable amount of time. This is due to the computation time to determine the minimum distance of these cyclic codes and also, for some lengths (e.g. 195 and 255), there are a tremendous number of inequivalent cyclic codes. Having said that, we can still search for improvements from lower rate cyclic codes of these lengths for which the minimum distance computation can be completed in a reasonable time. We have found many new cyclic codes that improve Brouwer's lower bound and before we present these codes, we should first consider the evaluation procedure.

As before, let β be a primitive nth root of unity and let Λ be a set containing all distinct (excluding the conjugates) exponents of β. The polynomial $x^n - 1$ can be factorised into irreducible polynomials $f_i(x)$ over $\mathbb{F}_2$, $x^n - 1 = \prod_{i \in \Lambda} f_i(x)$. For notational purposes, we denote the irreducible polynomial $f_i(x)$ as the minimal polynomial of β^i. The generator and parity-check polynomials, denoted by $g(x)$ and $h(x)$ respectively, are products of $f_i(x)$. Given a set $\Gamma \subseteq \Lambda$, a cyclic code $\mathscr{C}$ which has β^i, $i \in \Gamma$, as the non-zeros can be constructed. This means the parity-check polynomial $h(x)$ is given by

$$h(x) = \prod_{i \in \Gamma} f_i(x)$$

and the dimension k of this cyclic code is $\sum_{i \in \Gamma} \deg(f_i(x))$, where $\deg(f(x))$ denotes the degree of $f(x)$. Let $\Gamma' \subseteq \Lambda \setminus \{0\}$, $h'(x) = \prod_{i \in \Gamma'} f_i(x)$ and $h(x) = (1+x)h'(x)$. Given $\mathscr{C}$ with parity-check polynomial $h(x)$, there exists an $[n, k-1, d']$ expurgated cyclic code, $\mathscr{C}'$, which has parity-check polynomial $h'(x)$. For this cyclic code, $\mathrm{wt}_H(\boldsymbol{c}) \equiv 0 \pmod 2$ for all $\boldsymbol{c} \in \mathscr{C}'$. For convenience, we call $\mathscr{C}$ the augmented code of $\mathscr{C}'$.

Consider an $[n, k-1, d']$ expurgated cyclic code $\mathscr{C}'$, let the set $\boldsymbol{\Gamma} = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_r\}$ where, for $1 \leq j \leq r$, $\Gamma_j \subseteq \Lambda \setminus \{0\}$ and $\sum_{i \in \Gamma_j} \deg(f_i(x)) = k - 1$. For each $\Gamma_j \in \boldsymbol{\Gamma}$, we compute $h'(x)$ and construct $\mathscr{C}'$. Having constructed the expurgated code, the augmented code can be easily obtained as shown below. Let $\boldsymbol{G}$ be a generator matrix of the augmented code $\mathscr{C}$, and without loss of generality, it can be written as

$$\boldsymbol{G} = \begin{array}{|c|}\hline \boldsymbol{G}' \\ \hline \mathbf{v} \\ \hline \end{array} \tag{5.8}$$

where $\boldsymbol{G}'$ is a generator matrix of $\mathscr{C}'$ and the vector $\mathbf{v}$ is a coset of $\mathscr{C}'$ in $\mathscr{C}$. Using the arrangement in (5.8), we evaluate d' by enumerating codewords $c \in \mathscr{C}'$ from $\boldsymbol{G}'$. The minimum distance of $\mathscr{C}$, denoted by d, is simply $\min_{\boldsymbol{c} \in \mathscr{C}'}\{d', \mathrm{wt}_H(\boldsymbol{c} + \mathbf{v})\}$ for all codewords $\boldsymbol{c}$ enumerated. We follow Algorithm 5.3 to evaluate d'. Let $d_{Brouwer}$

and $d'_{Brouwer}$ denote the lower bounds of [6] for linear codes of the same length and dimension as those of $\mathscr{C}$ and $\mathscr{C}'$ respectively. During the enumerations, as soon as $d \leq d_{Brouwer}$ and $d' \leq d'_{Brouwer}$, the evaluation is terminated and the next Γ_j in $\boldsymbol{\Gamma}$ is then processed. However, if $d \leq d_{Brouwer}$ and $d' > d'_{Brouwer}$, only the evaluation for $\mathscr{C}$ is discarded. Nothing is discarded if both $d' > d'_{Brouwer}$ and $d > d_{Brouwer}$. This procedure continues until an improvement is obtained; or the set in $\boldsymbol{\Gamma}$ has been exhausted, which means that there does not exist $[n, k-1]$ and $[n, k]$ cyclic codes which are improvements to Brouwer's lower bounds. In cases where the minimum distance computation is not feasible using a single computer, we switch to a parallel version using grid computers.

Table 5.1 presents the results of the search for new binary cyclic codes having lengths $195 \leq n \leq 255$. The cyclic codes in this table are expressed in terms of the parity-check polynomial $h(x)$, which is given in the last column by the exponents of β (excluding the conjugates). Note that the polynomial $m(x)$, which is given in octal with the most significant bit on the left, is the minimal polynomial of β. In many cases, the entries of $\mathscr{C}$ and $\mathscr{C}'$ are combined in a single row and this is indicated by "a/b" where the parameters a and b are for $\mathscr{C}'$ and $\mathscr{C}$, respectively. The notation "[0]" indicates that the polynomial $(1 + x)$ is to be excluded from the parity-check polynomial of $\mathscr{C}'$.

Some of these tabulated cyclic codes have a minimum Hamming distance which coincides with the lower bounds given in [12]. These are presented in Table 5.1 with the indicative mark "†".

In the late 1970s, computing the minimum distance of extended Quadratic Residue (QR) codes was posed as an open research problem by MacWilliams and Sloane [16]. Since then, the minimum distance of the extended QR code for the prime 199 has remained an open question. For this code, the bounds of the minimum distance were given as $16 - 32$ in [16] and the lower bound was improved to 24 in [9]. Since $199 \equiv -1 \pmod 8$, the extended code is a doubly even self-dual code and its automorphism group contains a projective special linear group, which is known to be doubly transitive [16]. As a result, the minimum distance of the binary [199, 100] QR code is odd, i.e. $d \equiv 3 \pmod 4$, and hence, $d = 23, 27$ or 31. Due to the cyclic property and the rate of this QR code [7], we can safely assume that a codeword of weight d has maximum information weight of $\lfloor d/2 \rfloor$. If a weight d codeword does not satisfy this property, there must exist one of its cyclic shifts that does. After enumerating all codewords up to (and including) information weight 13 using grid computers, no codeword of weight less than 31 was found, implying that d of this binary [199, 100] QR code is indeed 31.

Without exploiting the property that $d \equiv 3 \pmod 4$, an additional $\binom{100}{14} + \binom{100}{15}$ codewords (88,373,885,354,647,200 codewords) would need to be enumerated in order to establish the same result and beyond available computer resources. Accordingly, we now know that there exists the [199, 99, 32] expurgated QR code and the [200, 100, 32] extended QR code.

It is interesting to note that many of the code improvements are contributed by low-rate cyclic codes of length 255 and there are 16 cases of this. Furthermore, it is also interesting that Table 5.1 includes a [255, 55, 70] cyclic code and a [255, 63, 65]

Table 5.1 New binary cyclic codes

$[m(x)]_8$	n	k	d	$d_{Brouwer}$	$h(x)$
17277	195	† 66/67	42/41	40/40	[0], 3, 5, 9, 19, 39, 65, 67
		† 68/69	40/39	39/38	[0], 1, 3, 13, 19, 35, 67, 91
		† 73	38	37	0, 3, 7, 19, 33, 35, 47
		† 74/75	38/37	36/36	[0], 3, 7, 19, 33, 35, 47, 65
		78	36	35	3, 7, 9, 11, 19, 35, 39, 65
13237042705- 30057231362- 555070452551	199	99/100	32/31	28/28	[0], 1
6727273	205	† 60	48	46	5, 11, 31
		† 61	46	44	0, 3, 11, 31
3346667657	215	70/71	46/46	44/44	[0], 3, 13, 35
3705317547055	223	74/75	48/47	46/45	[0], 5, 9
3460425444467- 7544446504147	229	76	48	46	1
6704436621	233	† 58/59	60/60	56/56	[0], 3, 29
150153013	241	† 49	68	65	0, 1, 21
		73	54	53	0, 1, 3, 25
435	255	48/49	76/75	75/72	[0], 47, 55, 91, 95, 111, 127
		50/51	74/74	72/72	[0], 9, 13, 23, 47, 61, 85, 127
		52/53	72/72	71/68	[0], 7, 9, 17, 47, 55, 111, 127
		54/55	70/70	68/68	[0], 3, 7, 23, 47, 55, 85, 119, 127
		56/57	68/68	67/65	[0], 7, 27, 31, 45, 47, 55, 127
		58	66	64	7, 39, 43, 45, 47, 55, 85, 127
		60	66	64	7, 17, 23, 39, 45, 47, 55, 127
		62/63	66/65	64/63	[0], 11, 21, 47, 55, 61, 85, 87, 119, 127
		64/65	64/63	62/62	[0], 19, 31, 39, 47, 55, 63, 91, 127

cyclic code, which are superior to the BCH codes of the same length and dimension. Both of these BCH codes have minimum distance 63 only.

5.5 Constructing New Codes from Existing Ones

It is difficult to explicitly construct a new code with large minimum distance. However, the alternative approach, which starts from a known code which already has large minimum distance, seems to be more fruitful. Some of these methods are described below and in the following subsections, we present some new binary codes constructed using these methods, which improve on Brouwer's lower bound.

Theorem 5.1 (Construction X) *Let $\mathscr{B}_1$ and $\mathscr{B}_2$ be $[n, k_1, d_1]$ and $[n, k_2, d_2]$ linear codes over $\mathbb{F}_q$ respectively, where $\mathscr{B}_1 \supset \mathscr{B}_2$ ($\mathscr{B}_2$ is a subcode of $\mathscr{B}_1$). Let $\mathscr{A}$*

be an $[n', k_3 = k_1 - k_2, d']$ *auxiliary code over the same field. There exists an* $[n + n', k_1, \min\{d_2, d_1 + d'\}]$ *code* $\mathscr{C}_X$ *over* $\mathbb{F}_q$.

Construction X is due to Sloane et al. [23] and it basically adds a tail, which is a codeword of the auxiliary code $\mathscr{A}$, to $\mathscr{B}_1$ so that the minimum distance is increased. The effect of Construction X can be visualised as follows. Let $\boldsymbol{G}_{\mathscr{C}}$ be the generator matrix of code $\mathscr{C}$. Since $\mathscr{B}_1 \supset \mathscr{B}_2$, we may express $\boldsymbol{G}_{\mathscr{B}_1}$ as

$$\boldsymbol{G}_{\mathscr{B}_1} = \left[\begin{array}{c} \boldsymbol{G}_{\mathscr{B}_2} \\ \hline \boldsymbol{V} \end{array}\right],$$

where $\boldsymbol{V}$ is a $(k_1 - k_2) \times n$ matrix which contains the cosets of $\mathscr{B}_2$ in $\mathscr{B}_1$. We can see that the code generated by $\boldsymbol{G}_{\mathscr{B}_2}$ has minimum distance d_2, and the set of codewords $\{\mathbf{v} + \boldsymbol{c}_2\}$, for all $\mathbf{v} \in \boldsymbol{V}$ and all codewords $\boldsymbol{c}_2$ generated by $\boldsymbol{G}_{\mathscr{B}_2}$, have minimum weight of d_1. By appending non-zero weight codewords of $\mathscr{A}$ to the set $\{\mathbf{v} + \boldsymbol{c}_2\}$, and all zeros codeword to each codeword of $\mathscr{B}_2$, we have a lengthened code of larger minimum distance, $\mathscr{C}_X$, whose generator matrix is given by

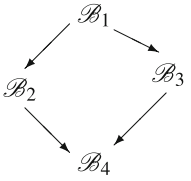
$$\boldsymbol{G}_{\mathscr{C}_X} = \left[\begin{array}{c|c} \boldsymbol{G}_{\mathscr{B}_2} & \mathbf{0} \\ \hline \boldsymbol{V} & \boldsymbol{G}_{\mathscr{A}} \end{array}\right]. \tag{5.9}$$

We can see that, for binary cyclic linear codes of odd minimum distance, code extension by annexing an overall parity-check bit is an instance of Construction X. In this case, $\mathscr{B}_2$ is the even-weight subcode of $\mathscr{B}_1$ and the auxiliary code $\mathscr{A}$ is the trivial $[1, 1, 1]_2$ code.

Construction X given in Theorem 5.1 considers a chain of two codes only. There also exists a variant of Construction X, called Construction XX, which makes use of Construction X twice and it was introduced by Alltop [1].

Theorem 5.2 (Construction XX) *Consider three linear codes of the same length,* $\mathscr{B}_1 = [n, k_1, d_1]$, $\mathscr{B}_2 = [n, k_2, d_2]$ *and* $\mathscr{B}_3 = [n, k_3, d_3]$ *where* $\mathscr{B}_2 \subset \mathscr{B}_1$ *and* $\mathscr{B}_3 \subset \mathscr{B}_1$. *Let* $\mathscr{B}_4$ *be an* $[n, k_4, d_4]$ *linear code which is the intersection code of* $\mathscr{B}_2$ *and* $\mathscr{B}_3$, *i.e.* $\mathscr{B}_4 = \mathscr{B}_2 \cap \mathscr{B}_3$. *Using auxiliary codes* $\mathscr{A}_1 = [n_1, k_1 - k_2, d_1']$ *and* $\mathscr{A}_2 = [n_2, k_1 - k_3, d_2']$, *there exists an* $[n + n_1 + n_2, k_1, d]$ *linear code* $\mathscr{C}_{XX}$, *where* $d = \min\{d_4, d_3 + d_1', d_2 + d_2', d_1 + d_1' + d_2'\}$.

The relationship among $\mathscr{B}_1$, $\mathscr{B}_2$, $\mathscr{B}_3$ and $\mathscr{B}_4$ can be illustrated as a lattice shown below [11].



Since $\mathscr{B}_1 \supset \mathscr{B}_2$, $\mathscr{B}_1 \supset \mathscr{B}_3$, $\mathscr{B}_4 \subset \mathscr{B}_2$ and $\mathscr{B}_4 \subset \mathscr{B}_3$, the generator matrices of $\mathscr{B}_2$, $\mathscr{B}_3$ and $\mathscr{B}_1$ can be written as

$$\boldsymbol{G}_{\mathscr{B}_2} = \left[\begin{array}{c} \boldsymbol{G}_{\mathscr{B}_4} \\ \hline \boldsymbol{V_2} \end{array}\right], \quad \boldsymbol{G}_{\mathscr{B}_3} = \left[\begin{array}{c} \boldsymbol{G}_{\mathscr{B}_4} \\ \hline \boldsymbol{V_3} \end{array}\right]$$

$$\text{and} \quad \boldsymbol{G}_{\mathscr{B}_1} = \left[\begin{array}{c} \boldsymbol{G}_{\mathscr{B}_4} \\ \hline \boldsymbol{V_2} \\ \hline \boldsymbol{V_3} \\ \hline \boldsymbol{V} \end{array}\right]$$

respectively, where $\boldsymbol{V}_i$, $i = 2, 3$, is the coset of $\mathscr{B}_4$ in $\mathscr{B}_i$, and $\boldsymbol{V}$ contains the cosets of $\mathscr{B}_2$ and $\mathscr{B}_3$ in $\mathscr{B}_1$. Construction XX starts by applying Construction X to the pair of codes $\mathscr{B}_1 \supset \mathscr{B}_2$ using $\mathscr{A}_1$ as the auxiliary code. The resulting code is $\mathscr{C}_X = [n + n_1, k_1, \min\{d_2, d_1 + d_1'\}]$, whose generator matrix is given by

$$\boldsymbol{G}_{\mathscr{C}_X} = \left[\begin{array}{c|c} \boldsymbol{G}_{\mathscr{B}_4} & \boldsymbol{0} \\ \hline \boldsymbol{V_2} & \\ \hline \boldsymbol{V_3} & \\ \hline \boldsymbol{V} & \boldsymbol{G}_{\mathscr{A}_1} \end{array}\right].$$

This generator matrix can be rearranged such that the codewords formed from the first n coordinates are cosets of $\mathscr{B}_3$ in $\mathscr{B}_1$. This rearrangement results in the following generator matrix of $\mathscr{C}_X$,

$$\boldsymbol{G}_{\mathscr{C}_X} = \left[\begin{array}{c|c} \boldsymbol{G}_{\mathscr{B}_4} & \boldsymbol{0} \\ \hline \boldsymbol{V_3} & \boldsymbol{G}^{(1)}_{\mathscr{A}_1} \\ \hline \boldsymbol{V_2} & \boldsymbol{0} \\ \hline \boldsymbol{V} & \boldsymbol{G}^{(2)}_{\mathscr{A}_1} \end{array}\right],$$

where $\boldsymbol{G}_{\mathscr{A}_1} = \left[\begin{array}{c} \boldsymbol{G}^{(1)}_{\mathscr{A}_1} \\ \hline \boldsymbol{G}^{(2)}_{\mathscr{A}_1} \end{array}\right]$. Next, using $\mathscr{A}_2$ as the auxiliary code, applying Construction X to the pair $\mathscr{B}_1 \supset \mathscr{B}_3$ with the rearrangement above, we obtain $\mathscr{C}_{XX}$ whose generator matrix is

$$\boldsymbol{G}_{\mathscr{C}_{XX}} = \left[\begin{array}{c|c|c} \boldsymbol{G}_{\mathscr{B}_4} & \boldsymbol{0} & \boldsymbol{0} \\ \hline \boldsymbol{V_3} & \boldsymbol{G}^{(1)}_{\mathscr{A}_1} & \\ \hline \boldsymbol{V_2} & \boldsymbol{0} & \\ \hline \boldsymbol{V} & \boldsymbol{G}^{(2)}_{\mathscr{A}_1} & \boldsymbol{G}_{\mathscr{A}_2} \end{array}\right].$$

While Constructions X and XX result in a code with increased length, there also exists a technique to obtain a shorter code with known minimum distance lower bounded from a longer code whose minimum distance and also that of its dual code are known explicitly. This technique is due to Sloane et al. [23] and it is called Construction Y1.

Theorem 5.3 (Construction Y1) *Given an* $[n, k, d]$ *linear code* $\mathscr{C}$*, which has an* $[n, n-k, d^\perp]$ $\mathscr{C}^\perp$ *as its dual, an* $[n - d^\perp, k - d^\perp + 1, \geq d]$ *code* $\mathscr{C}'$ *can be constructed.*

Given an $[n, k, d]$ code, with standard code shortening, we obtain an $[n-i, k-i, \geq d]$ code where i indicates the number of coordinates to shorten. With Construction Y1, however, we can gain an additional dimension in the resulting shortened code. This can be explained as follows. Without loss of generality, we can assume the parity-check matrix of $\mathscr{C}$, which is also the generator matrix of $\mathscr{C}^\perp$, $\boldsymbol{H}$ contains a codeword $\boldsymbol{c}^\perp$ of weight $d^\perp$. If we delete the coordinates which form the support of $\boldsymbol{c}^\perp$ from $\boldsymbol{H}$, now $\boldsymbol{H}$ becomes an $(n-k) \times n - d^\perp$ matrix and there is a row which contains all zeros among these $n-k$ rows. Removing this all zeros row, we have an $(n-k-1) \times (n-d^\perp)$ matrix $\boldsymbol{H}'$, which is the parity-check matrix of an $[n-d^\perp, n-d^\perp-(n-k-1), \geq d] = [n-d^\perp, k-d^\perp+1, \geq d]$ code $\mathscr{C}'$.

5.5.1 New Binary Codes from Cyclic Codes of Length 151

Amongst all of the cyclic codes in Table 4.3, those of length 151 have minimum distances that were found to have the highest number of matches against Brouwer's [6] lower bounds. This shows that binary cyclic codes of length 151 are indeed good codes. Since 151 is a prime, cyclic codes of this length are special as all of the irreducible factors of $x^{151} - 1$, apart from $1 + x$, have a fixed degree of 15. Having a fixed degree implies that duadic codes [14], which includes the quadratic residue codes, also exist for this length. Due to their large minimum distance, they are good candidate component codes for Constructions X and XX.

Table 5.2 Order of β in an optimum chain of $[151, k_i, d_i]$ cyclic codes

i	k_i	d_i	Roots of $g(x)$, excluding conjugate roots
1	150	2	β^0
2	135	6	β^0 β^1
3	120	8	β^0 β^1 β^3
4	105	14	β^0 β^1 β^3 β^5
5	90	18	β^0 β^1 β^3 β^5 β^{11}
6	75	24	β^0 β^1 β^3 β^5 β^{11} β^{15}
7	60	32	β^0 β^1 β^3 β^5 β^{11} β^{15} β^{37}
8	45	36	β^0 β^1 β^3 β^5 β^{11} β^{15} β^{23} β^{37}
9	30	48	β^0 β^1 β^3 β^5 β^{11} β^{15} β^{23} β^{35} β^{37}
10	15	60	β^0 β^1 β^3 β^5 β^7 β^{11} β^{15} β^{23} β^{35} β^{37}

Definition 5.2 (*Chain of Cyclic Codes*) A pair of cyclic codes, $\mathscr{C}_1 = [n, k_1, d_1]$ and $\mathscr{C}_2 = [n, k_2, d_2]$ where $k_1 > k_2$, is nested, denoted $\mathscr{C}_1 \supset \mathscr{C}_2$, if all roots of $\mathscr{C}_1$ are contained in $\mathscr{C}_2$. Here, the roots refer to those of the generator polynomial. By appropriate arrangement of their roots, cyclic codes of the same length may be partitioned into a sequence of cyclic codes $\mathscr{C}_1 \supset \mathscr{C}_2 \supset \ldots \supset \mathscr{C}_t$. This sequence of codes is termed a chain of cyclic codes.

Given all cyclic codes of the same length, it is important to order the roots of these cyclic codes so that an optimum chain can be obtained. For all cyclic codes of length 151 given in Table 4.3, whose generator polynomial contains $1+x$ as a factor, an ordering of roots (excluding the conjugate roots) shown in Table 5.2 results in an optimum chain arrangement. Here β is a primitive 151^{st} root of unity. Similarly, a chain which contains cyclic codes, whose generator polynomial does not divide $1 + x$, can also be obtained.

All the constituent codes in the chain $\mathscr{C}_1 \supset \mathscr{C}_2 \supset \ldots \supset \mathscr{C}_{10}$ of Table 5.2 are cyclic. Following Grassl [10], a chain of non-cyclic subcodes may also be constructed from a chain of cyclic codes. This is because for a given generator matrix of an $[n, k, d]$ cyclic code (not necessarily in row-echelon form), removing the last i rows of this matrix will produce an $[n, k - i, \geq d]$ code which will no longer be cyclic. As a consequence, with respect to Table 5.2, there exists $[151, k, d]$ linear codes, for $15 \leq k \leq 150$.

Each combination of pairs of codes in the $[151, k, d]$ chain is a nested pair which can be used as component codes for Construction X to produce another linear code with increased distance. There is a chance that the minimum distance of the resulting linear code is larger than that of the best-known codes for the same length and dimension. In order to find the existence of such cases, the following exhaustive approach has been taken. There are $\binom{150-15+1}{2} = \binom{136}{2}$ distinct pair of codes in the above chain of linear codes, and each pair say $\mathscr{C}_1 = [n, k_1, d_1] \supset \mathscr{C}_2 = [n, k_2, d_1]$, is combined using Construction X with an auxiliary code $\mathscr{A}$, which is an $[n', k_1 - k_2, d']$ best-known linear code. The minimum distance of the resulting code $\mathscr{C}_X$ is then

compared to that of the best-known linear code for the same length and dimension to check for a possible improvement. Two improvements were obtained and they are tabulated in in the top half of Table 5.3.

In the case where $k_1 - k_2$ is small, the minimum distance of $\mathscr{C}_1$, i.e. d_1, obtained from a chain of linear codes, can be unsatisfactory. We can improve d_1 by augmenting $\mathscr{C}_1$ with a vector $\mathbf{v}$ of length n, i.e. add $\mathbf{v}$ as an additional row in $\boldsymbol{G}_{\mathscr{C}_2}$. In finding a vector $\mathbf{v}$ that can maximise the minimum distance of the enlarged code, we have adopted the following procedure. Choose a code $\mathscr{C}_2 = [n, k_2, d_2]$ that has sufficiently high minimum distance.

Assuming that $\boldsymbol{G}_{\mathscr{C}_2}$ is in reduced-echelon format, generate a vector $\mathbf{v}$ which satisfies the following conditions:

1. $v_i = 0$ for $0 \leq i \leq k - 1$ where v_i is the ith element of $\mathbf{v}$,
2. $\mathrm{wt}_H(\mathbf{v}) > d_1$, and
3. $\mathrm{wt}_H(\mathbf{v} + \boldsymbol{G}_r) > d_1$ for all $r \in \{0, 1, \ldots, k_2 - 1\}$ where $\boldsymbol{G}_{\mathscr{C}_2,r}$ denotes the rth row of $\boldsymbol{G}_{\mathscr{C}_2}$.

The vector $\mathbf{v}$ is then appended to $\boldsymbol{G}_{\mathscr{C}_2}$ as an additional row. The minimum distance of the resulting code is computed using Algorithm 5.1. A threshold is applied during the minimum distance evaluation and a termination is called whenever: $d_{ub} \leq d_1$, in which case a different $\mathbf{v}$ is chosen and Algorithm 5.1 is restarted; or $d_1 < d_{ub} \leq d_{lb}$ which means that an improvement has been found.

Using this approach, we found two new linear codes, $[151, 77, 20]$ and $[151, 62, 27]$, which have higher minimum distance than the corresponding codes obtained from a chain of nested cyclic codes. These two codes are obtained starting from the cyclic code $[151, 76, 23]$–which has roots $\{\beta, \beta^5, \beta^{15}, \beta^{35}, \beta^{37}\}$ and the cyclic code $[151, 61, 31]$–which has roots $\{\beta, \beta^3, \beta^5, \beta^{11}, \beta^{15}, \beta^{37}\}$, respectively and therefore

$$[151, 77, 20] \supset [151, 76, 23]$$

and

$$[151, 62, 27] \supset [151, 61, 31].$$

The second half of Table 5.3 shows the foundation codes for these new codes.

Note that when searching for the $[151, 62, 27]$ code, we exploited the property that the $[152, 61, 32]$ code obtained by extending the $[151, 61, 31]$ cyclic code is doubly even. We chose the additional vector $\mathbf{v}$ such that extending the enlarged code $[151, 62, d_1]$ yields again a doubly even code. This implies the congruence $d_1 = 0, 3 \bmod 4$ for the minimum distance of the enlarged code. Hence, it is sufficient to establish a lower bound $d_{lb} = 25$ using Algorithm 5.1 to show that $d_1 \geq 27$.

Furthermore, we also derived two different codes, $\mathscr{C}_2 = [151, 62, 27] \subset \mathscr{C}_1$ and $\mathscr{C}_3 = [151, 62, 27] \subset \mathscr{C}_1$, where $\mathscr{C}_1 = [151, 63, 23]$ and $\mathscr{C}_4 = \mathscr{C}_2 \cap \mathscr{C}_3 = [151, 61, 31]$. Using Construction XX, a $[159, 63, 31]$ code is obtained, see Table 5.4.

Table 5.3 New binary codes from Construction X and cyclic codes of length 151

$\mathscr{C}_1$	$\mathscr{C}_2$	$\mathscr{A}$	$\mathscr{C}_X$
Using chain of linear codes			
[151,72,24]	[151,60,32]	[23,12,7]	[174,72,31]
[151,60,32]	[151,45,36]	[20,15,3]	[171,60,35]
Using an improved subcode			
[151,77,20]	[151,76,23]	[3,1,3]	[154,77,23]
[151,62,27]	[151,61,31]	[4,1,4]	[155,62,31]

Table 5.4 New binary code from Construction XX and cyclic codes of length 151

$\mathscr{C}_1$	$\mathscr{C}_2$	$\mathscr{C}_3$	$\mathscr{C}_4 = \mathscr{C}_2 \cap \mathscr{C}_3$	$\mathscr{A}_1$	$\mathscr{A}_2$	$\mathscr{C}_{XX}$
[151, 63, 23]	[151, 62, 27]	[151, 62, 27]	[151, 61, 31]	[4, 1, 4]	[4, 1, 4]	[159, 63, 31]

5.5.2 *New Binary Codes from Cyclic Codes of Length ≥ 199*

We know from Table 5.1 that there exists an outstanding $[199, 100, 31]$ cyclic code. The extended code, obtained by annexing an overall parity-check bit, is a $[200, 100, 32]$ doubly even self-dual code. As the name implies, being self-dual we know that the dual code has minimum distance 32. By using Construction Y1 (Theorem 5.3), a $[168, 69, 32]$ new, improved binary code is obtained. The minimum distance of the $[168, 69]$ previously considered best-known binary linear code is 30.

Considering cyclic codes of length 205, in addition to a $[205, 61, 46]$ cyclic code (see Table 5.1), there also exists a $[205, 61, 45]$ cyclic code which contains a $[205, 60, 48]$ cyclic code as its even-weight subcode. Applying Construction X (Theorem 5.1) to the $[205, 61, 45] \supset [205, 60, 48]$ pair of cyclic codes with a repetition code of length 3 as the auxiliary code, a $[208, 61, 48]$ new binary linear code is constructed, which improves Brouwer's lower bound distance by 2.

Furthermore, by analysing the dual codes of the $[255, 65, 63]$ cyclic code in Table 5.1 and its $[255, 64, 64]$ even weight subcode it was found that both have minimum distance of 8. Applying Construction Y1 (Theorem 5.3), we obtain the $[247, 57, 64]$ and the $[247, 58, 63]$ new binary linear codes, which improves on Brouwer's lower bound distances by 2 and 1, respectively.

5.6 Concluding Observations on Producing New Binary Codes

In the search for error-correcting codes with large minimum distance, having a fast, efficient algorithm to compute the exact minimum distance of a linear code is important. The evolution of various algorithms to evaluate the minimum distance of a binary

linear code, from the naive approach to Zimmermann's efficient approach, have been explored in detail. In addition to these algorithms, Chen's approach in computing the minimum distance of binary cyclic codes is a significant breakthrough.

The core basis of a minimum distance evaluation algorithm is codeword enumeration. As we increase the weight of the information vector, the number of codewords grows exponentially. Zimmermann's very useful algorithm may be improved by omitting generator matrices with overlapping information sets that never contribute to the lower bound throughout the enumeration. Early termination is important in the event that a new minimum distance is found that meets the lower bound value of the previous enumeration step. In addition, if the code under consideration has the property that every codeword weight is divisible by 2 or 4, the number of codewords that need to be enumerated can be considerably reduced.

With some simple modifications, these algorithms can also be used to collect and hence, count all codewords of a given weight to determine all or part of the weight spectrum of a code.

Given a generator matrix, codewords may be efficiently generated by taking linear combinations of rows of this matrix. This implies the faster we can generate the combinations, the less time the minimum distance evaluation algorithm will take. One such efficient algorithm to generate these combinations is called the revolving-door algorithm. The revolving-door algorithm has a nice property that allows the problem of generating combinations to be readily implemented in parallel. Having an efficient minimum distance computation algorithm, which can be computed in parallel on multiple computers has allowed us to extend earlier research results [8, 21, 22] in the evaluation of the minimum distance of cyclic codes. In this way, we obtained the highest minimum distance attainable by all binary cyclic codes of odd lengths from 129 to 189. We found that none of these cyclic codes has a minimum distance that exceeds the minimum distance of the best-known linear codes of the same length and dimension, which are given as lower bounds in [6]. However there are 134 cyclic codes that meet the lower bounds, see Sect. 5.3 and encoders and decoders may be easier to implement for the cyclic codes.

Having an efficient, multiple computer based, minimum distance computation algorithm also allowed us to search for the existence of binary cyclic codes of length longer than 189 which are improvements to Brouwer's lower bounds. We found 35 of these cyclic codes, namely

[195, 66, 42], [195, 67, 41], [195, 68, 40], [195, 69, 39], [195, 73, 38], [195, 74, 38], [195, 75, 37], [195, 78, 36], [199, 99, 32], [199, 100, 32], [205, 60, 48], [205, 61, 46], [215, 70, 46], [215, 71, 46], [223, 74, 48], [223, 75, 47], [229, 76, 48], [233, 58, 60], [233, 59, 60], [255, 48, 76], [255, 49, 75], [255, 50, 74], [255, 51, 74], [255, 52, 72], [255, 53, 72], [255, 54, 70], [255, 55, 70], [255, 56, 68], [255, 57, 68], [255, 58, 66], [255, 60, 66], [255, 62, 66], [255, 63, 65], [255, 64, 64], [255, 65, 63].

From the cyclic codes above, using Construction X to lengthen the code or Construction Y1 to shorten the code, four additional improvements to [6] lower bound are found, namely

Table 5.5 Updated minimum distance lower bounds of linear codes $\mathscr{C} = [n, k]$ for $153 \leq n \leq 174$ and $58 \leq k \leq 77$

$n\backslash k$	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	k/n
153	32	32	32	32	29^P	28	28	27	26	26	26	26	25	24	24	24	24	24	24	22	153
154	32	32	32	32	30^P	28	28	28	27	26	26	26	26	24	24	24	24	24	24	23^X	154
155	32	32	32	32	31^X	28	28	28	28	27	26	26	26	25	24	24	24	24	24	24^E	155
156	32	32	32	32	32^E	28	28	28	28	28	27	26	26	26	25	24	24	24	24	24^E	156
157	32	32	32	32	32^E	29	28	28	28	28	28	26	26	26	26	24	24	24	24	24^E	157
158	32	32	32	32	32^E	30	29	28	28	28	28	26	26	26	26	25	24	24	24	24	158
159	32	32	32	32	32^E	31^{XX}	30	29	28	28	28	27	26	26	26	26	25	24	24	24	159
160	32	32	32	32	32^E	32^E	30	30	28	28	28	28	26	26	26	26	26	25	24	24	160
161	32	32	32	32	32^E	32^E	30	30	29	28	28	28	27	26	26	26	26	26	25	24	161
162	33	32	32	32	32	32^E	31^S	30	30	29	28	28	28	27	26	26	26	26	26	24	162
163	34	33	32	32	32	32	32^S	31^S	30	30	29	28	28	28	27	26	26	26	26	25	163
164	34	34	33	32	32	32	32	32^S	31^S	30	30	29	28	28	28	27	26	26	26	26	164
165	34	34	34	33	32	32	32	32	32^S	31^S	30	30	28	28	28	28	27	26	26	26	165
166	34	34	34	34	32	32	32	32	32^S	32^S	31^S	30	28	28	28	28	28	27	26	26	166
167	34	34	34	34	32	32	32	32	32^S	32^S	32^S	31^P	29	28	28	28	28	28	27	26	167
168	34	34	34	34	32	32	32	32	32^S	32^S	32^S	32^{Y1}	30	29^S	28	28	28	28	28	26	168
169	35^S	34	34	34	32	32	32	32	32^E	32^S	32^S	32^E	31^S	30^S	29^S	28	28	28	28	27	169
170	36^E	35^S	34	34	33	32	32	32	32	32^E	32^S	32^E	32^S	31^S	30^S	29^S	28	28	28	28	170
171	36	36^E	35^X	34	34	33	32	32	32	32	32^E	32^E	32^S	32^S	31^S	30^S	29^S	28	28	28	171
172	36	36	36^E	34	34	34	33	32	32	32	32	32^E	32^S	32^S	32^S	31^S	30^S	29^S	28	28	172
173	36	36	36	35	34	34	34	33	32	32	32	32	32^E	32^S	32^S	32^S	31^S	30^S	29^S	28	173
174	36	36	36	36	34	34	34	34	32	32	32	32	32	32^E	32^S	32^S	32^S	31^S	30^S	29^S	174

Table 5.6 Updated minimum distance lower bounds of linear codes $\mathscr{C} = [n, k]$ for $175 \leq n \leq 224$ and $56 \leq k \leq 78$

$n\backslash k$	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	k/n
175	38	36	36	36	36	36	34	34	34	34	33	32	32	32	32	32^E	32^E	32^S	32^S	32^S	31^S	30^S	29^S	175
176	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^E	32^S	32^S	32^S	32^S	31^S	30^S	176
177	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	32^S	32^S	32^S	32^S	31^S	177
178	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	32^S	32^S	32^S	32^S	178
179	39	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	32^S	32^S	32^S	179
180	40	38	38	38	38	36	36	36	36	36	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^S	180
181	40	39	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	32^S	32^S	181
182	40	40	39	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	32^S	182
183	40	40	40	39	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	32^S	183
184	41	40	40	40	39	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	32	184
185	42	41^S	40	40	40	38	38	38	38	37	36	36	36	36	35	34	34	34	34	33	32	32	32	185
186	42	42^S	41^S	40	40	39	38	38	38	38	37	36	36	36	36	34	34	34	34	34	32	32	32	186
187	42	42	42^S	41^S	40	40	39	38	38	38	38	37	36	36	36	35	34	34	34	34	33	32	32	187
188	42	42	42	42^S	41^S	40	40	39	38	38	38	38	37	36	36	36	35	34	34	34	34	33	32	188
189	43	42	42	42	42^S	41^S	40	40	39	38	38	38	38	37	36	36	36	35	34	34	34	34	33	189
190	44	42	42	42	42	42^S	41^S	40	40	38	38	38	38	38	37^S	36	36	36	35	34	34	34	34	190
191	44	43	42	42	42	42	42^S	41^S	40	39	38	38	38	38	38^S	37^S	36	36	36	35	34	34	34	191
192	44	44	43	42	42	42	42	42^S	41^S	40	39^P	38	38	38	38	38^S	37^S	36	36	36	35^S	34	34	192
193	44	44	44	43	42	42	42	42	42^S	41^S	40^P	39^P	38	38	38	38^S	38^S	37^S	36	36	36^S	35^S	34	193
194	44	44	44	44	43	42	42	42	42	42^S	41^P	40^P	39^P	38	38	38	38^S	38^S	37^P	36	36	36^S	35^P	194
195	44	44	44	44	44	43	42	42	42	42	42^C	41^C	40^C	39^C	38	38	38	38^C	38^C	37^C	36	36	36^C	195
196	44	44	44	44	44	44	42	42	42	42	42^E	42^E	40	40^E	38	38	38	38	38^E	38^E	36	36	36	196
197	45	44	44	44	44	44	42	42	42	42	42^E	42^E	40	40	39	38	38	38	38	38^E	36	36	36	197
198	46	44	44	44	44	44	42	42	42	42	42^E	42^E	40	40	40	38	38	38	38	38	36	36	36	198
199	46	45^S	44	44	44	44	43	42	42	42	42^E	42^E	40	40	40	38	38	38	38	38	36	36	36	199

(continued)

Table 5.6 (continued)

$n\backslash k$	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	k/n
200	47^S	46^S	45^S	44	44	44	44	42	42	42	42^E	42^E	40	40	40	38	38	38	38	38	37	36	36	200
201	48^S	47^S	46^S	45^S	44	44	44	42	42	42	42	42^E	40	40	40	38	38	38	38	38	38	37	36	201
202	48^S	48^S	47^S	46^S	45^P	44	44	43	42	42	42	42^E	40	40	40	39	38	38	38	38	38	38	37	202
203	48^S	48^S	48^S	47^S	46^P	44	44	44	43	42	42	42^E	40	40	40	40	39	38	38	38	38	38	38	203
204	48^S	48^S	48^S	48^S	47^P	45^P	44	44	44	43	42	42	41	40	40	40	40	39	38	38	38	38	38	204
205	48	48^S	48^S	48^S	48^C	46^C	45^S	44	44	44	42	42	42	41	40	40	40	40	39	38	38	38	38	205
206	48	48^S	48^S	48^S	48^E	46^E	46^S	45^S	44	44	43	42	42	42	41	40	40	40	40	39	38	38	38	206
207	48	48	48^S	48^S	48^E	47^P	46^S	46^S	45^S	44	44	43	42	42	42	41	40	40	40	40	38	38	38	207
208	48	48	48	48^S	48^E	48^X	46	46^S	46^S	45^S	44	44	43	42	42	42	41	40	40	40	39	38	38	208
209	49	48	48	48	48^E	48^E	46	46	46^S	46^S	45^S	44	44	43	42	42	42	41	40	40	40	39	38	209
210	50	48	48	48	48	48^E	47^S	46	46	46^S	46^S	45^S	44	44	43	42	42	42	40	40	40	40	39	210
211	50	49	48	48	48	48^E	48^S	47^S	46	46^S	46^S	46^S	45^S	44	44	43	42	42	41	40	40	40	40	211
212	50	50	49	48	48	48	48^S	48^S	47^S	46	46^S	46^S	46^S	45^S	44	44	43	42	42	41	40	40	40	212
213	50	50	50	49	48	48	48	48^S	48^S	47^S	46	46^S	46^S	46^S	45^S	44	44	43	42	42	41	40	40	213
214	51	50	50	50	49	48	48	48	48^S	48^S	47^S	46	46^S	46^S	46^S	45^P	44	44	43	42	42	41	40	214
215	52	50	50	50	50	48	48	48	48	48^S	48^S	47^S	46	46^S	46^C	46^C	44	44	44	43	42	42	40	215
216	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	47^S	46^S	46^E	46^E	44	44	44	44	43	42	41	216
217	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	47^S	46^E	46^E	44	44	44	44	44	43	42	217
218	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	47^S	46^E	45^S	44	44	44	44	44	43	218
219	53	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	47^S	46^S	45^S	44	44	44	44	44	219
220	54	52	52	52	52	50	50	50	50	48	48	48	48	48^S	48^S	48^S	47^S	46^S	45^P	44	44	44	44	220
221	54	53	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	48^S	47^S	46^P	45^P	44	44	44	221
222	54	54	53	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	48^S	47^P	46^P	44	44	44	222
223	54	54	54	53	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^S	48^C	47^C	44	44	44	223
224	55	54	54	54	53	52	52	52	51	50	50	50	49	48	48	48	48^S	48^S	48^E	48^E	45	44	44	224

Table 5.7 Updated minimum distance lower bounds of linear codes $\mathscr{C} = [n, k]$ for $175 \leq n \leq 224$ and $79 \leq k \leq 100$

$n\backslash k$	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	k/n
175	28	28	28	27	26	26	26	26	25	24	24	24	24	23	22	22	22	22	22	22	22	21	175
176	29^S	28	28	28	27	26	26	26	26	25	24	24	24	24	23	22	22	22	22	22	22	22	176
177	30^S	29^S	28	28	28	27	26	26	26	26	25	24	24	24	24	23	22	22	22	22	22	22	177
178	31^S	30^S	29^S	28	28	28	27	26	26	26	26	25	24	24	24	24	23	22	22	22	22	22	178
179	32^S	31^S	30^S	29^S	28	28	28	27	26	26	26	26	25	24	24	24	24	23	22	22	22	22	179
180	32^S	32^S	31^S	30^S	29^S	28	28	28	26	26	26	26	26	24	24	24	24	24	23	22	22	22	180
181	32^S	32^S	32^S	31^S	30^S	29^S	28	28	27	26	26	26	26	25	24	24	24	24	24	23	22	22	181
182	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	28	27	26	26	26	26	25	24	24	24	24	24	24	23	182
183	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	28	27	26	26	26	26	25	24	24	24	24	24	23	183
184	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	28	27	26	26	26	26	25	24	24	24	24	24	184
185	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	28	27	26	26	26	26	25	24	24	24	24	185
186	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	28	26	26	26	26	26	24	24	24	24	186
187	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27	26	26	26	26	25	24	24	24	187
188	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27	26	26	26	26	25	24	24	188
189	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27	26	26	26	26	25	24	189
190	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27	26	26	26	26	25	190
191	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27	26	26	26	26	191
192	34	34	32	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28	27^S	26	26	26	192
193	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28^S	27^S	26	26	193
194	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28^S	27^P	26	194
195	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^S	28^P	27^P	195
196	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^S	29^P	28^P	196
197	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^S	30^P	29^P	197
198	36	36	34	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	31^P	30^P	198
199	36	36	34	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	32^C	31^C	199

(continued)

Table 5.7 (continued)

$n\backslash k$	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	k/n
200	36	36	35	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	32^E	32^E	200
201	36	36	36	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	32^E	32^E	201
202	36	36	36	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^S	32^E	32^E	202
203	37	36	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^S	32^E	32^E	203
204	38	37	36	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^S	32^E	32^E	204
205	38	38	37	36	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^S	32^E	32^E	205
206	38	38	38	37	36	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^S	32^E	32^E	206
207	38	38	38	38	37	36	36	35	34	34	34	34	33	32	32	32	32	32	32^S	32^S	32^E	32^E	207
208	38	38	38	38	38	37	36	36	34	34	34	34	34	32	32	32	32	32	32^S	32^S	32^E	32^E	208
209	38	38	38	38	38	38	37	36	35	34	34	34	34	32	32	32	32	32	32	32^S	32^E	32^E	209
210	38	38	38	38	38	38	38	37	36	35	34	34	34	32	32	32	32	32	32	32^S	32^E	32^E	210
211	39	38	38	38	38	38	38	38	37	36	35	34	34	33	32	32	32	32	32	32^S	32^E	32^E	211
212	40	39	38	38	38	38	38	38	38	37	36	35	34	34	33	32	32	32	32	32^S	32^E	32^E	212
213	40	40	39	38	38	38	38	38	38	38	37	36	35	34	34	33	32	32	32	32	32^E	32^E	213
214	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	34	33	32	32	32	32	32^E	214
215	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	34	33	32	32	32	32	215
216	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	34	32	32	32	32	216
217	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	33	32	32	32	217
218	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	33	32	32	218
219	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	33	32	219
220	44	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	33	220
221	44	44	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	34	221
222	44	44	44	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	35	222
223	44	44	44	44	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	36	223
224	44	44	44	44	44	43	42	41	40	40	40	40	40	39	38	38	38	38	38	38	38	37	224

Table 5.8 Updated minimum distance lower bounds of linear codes $\mathscr{C} = [n, k]$ for $225 \leq n \leq 256$ and $48 \leq k \leq 62$

$n\backslash k$	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	k/n
225	60	60^S	60^S	60^S	59^S	58^S	57^S	56	56	54	54	54	54	52	52	225
226	60	60	60^S	60^S	60^S	59^S	58^S	57^S	56	55^S	54	54	54	52	52	226
227	60	60	60^S	60^S	60^S	60^S	59^S	58^S	57^S	56^S	55^S	54	54	52	52	227
228	61	60	60^S	60^S	60^S	60^S	60^S	59^S	58^S	57^S	56^S	55^P	54	53^S	52	228
229	62	60	60^S	60^S	60^S	60^S	60^S	60^S	59^S	58^S	57^S	56^P	54	54^S	53^S	229
230	62	60	60	60^S	60^S	60^S	60^S	60^S	60^S	59^S	58^S	57^P	54	54	54^S	230
231	63	61	60	60	60^S	60^S	60^S	60^S	60^S	60^S	59^S	58^P	54	54	54	231
232	64	62	60	60	60	60^S	60^S	60^S	60^S	60^S	60^S	59^P	54	54	54	232
233	64	62	60	60	60	60^S	60^S	60^S	60^S	60^S	60^C	60^C	54	54	54	233
234	64	62	61	60	60	60	60^S	60^S	60^S	60^S	60^E	60^E	55	54	54	234
235	64	63	62	61	60	60	60	60^S	60^S	60^S	60^E	60^E	56	55	54	235
236	65	64	62	62	61	60	60	60	60^S	60^S	60^E	60^E	56	56	54	236
237	66	64	63	62	62	61	60	60	60	60^S	60^E	60^E	56	56	55	237
238	66	65^P	64	63	62	62	61	60	60	60	60^E	60^E	57	56	56	238
239	67	66^P	64	64	63	62	62	61	60	60	60	60^E	58	57	56	239
240	68	67^P	64	64	64	62	62	62	61	60	60	60	58	58	56	240
241	68	68^C	64	64	64	62	62	62	62	61	60	60	58	58	57	241
242	68	68^E	65	64	64	63^S	62	62	62	62	61	60	59	58	58	242
243	68	68^E	66	65	64	64^S	63^S	62	62	62	62	61	60	59	58	243
244	69	68	66	66	65	64	64^S	63^S	62	62	62	62	61	60	59	244
245	70	68	67	66	66	65^S	64^S	64^S	63^S	62	62	62	62	61	60	245
246	70	68	68	67	66	66^S	65^S	64^S	64^S	63^P	62	62	62	62	61	246
247	71	68	68	68	67	66^S	66^S	65^S	64^S	64^{Y1}	63^{Y1}	62	62	62	62	247

(continued)

Table 5.8 (continued)

$n\backslash k$	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	k/n
248	72	69^S	68	68	68	66^S	66^S	66^S	65^S	64^E	64^E	62	62	62	62	248
249	72	70^S	69^S	68	68	66^P	66^S	66^S	66^S	65^S	64^E	63^S	62	62	62	249
250	72	71^S	70^S	69^P	68	67^P	66^S	66^S	66^S	66^S	65^S	64^S	63^S	62	62	250
251	73^S	72^S	71^S	70^P	69^S	68^P	67^S	66^P	66^S	66^S	66^S	65^S	64^S	63^S	62	251
252	74^S	73^S	72^S	71^P	70^S	69^P	68^S	67^P	66^S	66^S	66^S	66^S	65^S	64^S	63^P	252
253	74	74^S	73^S	72^P	71^S	70^P	69^S	68^P	67^S	66^P	66^S	66^S	66^S	65^S	64^P	253
254	75^P	74^P	74^S	73^P	72^S	71^P	70^S	69^P	68^S	67^P	66^S	66^S	66^S	66^S	65^P	254
255	76^C	75^C	74^C	74^C	72^C	72^C	70^C	70^C	68^C	68^C	66^C	66^S	66^C	66^S	66^C	255
256	76	76^E	74^E	74^E	72	72^E	70^E	70^E	68	68^E	66^E	66^S	66^E	66^S	66^E	256

Table 5.9 Updated minimum distance lower bounds of linear codes $\mathscr{C} = [n, k]$ for $225 \leq n \leq 256$ and $63 \leq k \leq 76$

$n\backslash k$	63	64	65	66	67	68	69	70	71	72	73	74	75	76	k/n
225	52	52	50	50	50	50	48	48	48	48	48^S	48^E	48^E	46	225
226	52	52	50	50	50	50	48	48	48	48	48^S	48^E	48^E	46	226
227	52	52	50	50	50	50	48	48	48	48	48^S	48^E	48^E	46	227
228	52	52	50	50	50	50	48	48	48	48	48	48^E	48^E	47^P	228
229	52	52	51	50	50	50	49	48	48	48	48	48^E	48^E	48^C	229
230	53^S	52	52	51	50	50	50	48	48	48	48	48^E	48^E	48^E	230
231	54^S	53^S	52	52	51	50	50	48	48	48	48	48	48^E	48^E	231
232	54	54^S	53^S	52	52	51	50	49	48	48	48	48	48	48^E	232
233	54	54	54^S	53^S	52	52	51	50	49	48	48	48	48	48	233
234	54	54	54	54^S	53^S	52	52	51	50	49	48	48	48	48	234
235	54	54	54	54	54^S	53^S	52	52	51	50	49	48	48	48	235
236	54	54	54	54	54	54^S	53^S	52	52	51	50	49	48	48	236
237	54	54	54	54	54	54	54^S	53^S	52	52	51	50	49	48	237
238	55	54	54	54	54	54	54	54^S	53^S	52	52	51	50	49	238
239	56	55	54	54	54	54	54	54	54^S	53^S	52	52	51	50	239
240	56	56	54	54	54	54	54	54	54	54^S	53^P	52	52	51	240
241	56	56	55	54	54	54	54	54	54	54	54^C	52	52	52	241
242	57	56	56	55	54	54	54	54	54	54	54	53	52	52	242
243	58	57	56	56	55	54	54	54	54	54	54	54	53	52	243
244	58	58	56	56	56	55	54	54	54	54	54	54	54	53	244
245	59	58	57	56	56	56	55	54	54	54	54	54	54	54	245
246	60	59	58	57	56	56	56	55	54	54	54	54	54	54	246
247	61	60	59	58	57	56	56	56	55	54	54	54	54	54	247
248	62	61	60	59	58	57	56	56	56	55	54	54	54	54	248
249	62	62	61	60	59	58	57	56	56	56	55	54	54	54	249
250	62	62	62	61	60	59	58	57	56	56	56	55	54	54	250
251	62	62	62	62	61	60	59	58	57	56	56	56	55	54	251
252	62	62	62	62	62	61	60	59	58	56	56	56	56	55	252
253	63^P	62	62	62	62	62	61	60	59	56	56	56	56	56	253
254	64^P	63^P	62	62	62	62	62	61	60	57	56	56	56	56	254
255	65^C	64^C	63^C	62	62	62	62	62	61	58	57	56	56	56	255
256	66^E	64^E	64^E	62	62	62	62	62	62	58	58	56	56	56	256

$$[168, 69, 32], [208, 61, 48], [247, 57, 64], [247, 58, 63] .$$

Five new linear codes, which are derived from cyclic codes of length 151, have also been constructed. These new codes, which are produced by Constructions X and XX, are

$$[154, 77, 23], [155, 62, 31], [159, 63, 31], [171, 60, 35], [174, 72, 31] .$$

Given an $[n, k, d]$ code $\mathscr{C}$, where d is larger than the minimum distance of the best-known linear code of the same n and k, it is possible to obtain more codes, whose minimum distance is still larger than that of the corresponding best-known linear code, by recursively extending (annexing parity-checks), puncturing and/or shortening $\mathscr{C}$. For example, consider the new code $[168, 69, 32]$ as a starting point. New codes can be obtained by annexing parity-check bits $[168 + i, 69, 32]$, for $1 \leq i \leq 3$. With puncturing by one bit a $[167, 69, 31]$ new code is obtained by shortening $[168 - i, 69 - i, 32]$, for $1 \leq i \leq 5$, 5 new codes are obtained with a minimum distance of 32. More improvements are also obtained by shortening these extended and punctured codes. Overall, with all of the new codes described and presented in this chapter, there are some 901 new binary linear codes which improve on Brouwer's lower bounds. The updated lower bounds are tabulated in Tables 5.5, 5.6, 5.7, 5.8 and 5.9 in Appendix "Improved Lower Bounds of the Minimum Distance of Binary Linear Codes".

5.7 Summary

Methods have been described and presented which may be used to determine the minimum Hamming distance and weight distribution of a linear code. These are the main tools for testing new codes which are candidates for improvements to currently known, best codes. Several efficient algorithms for computing the minimum distance and weight distribution of linear codes have been explored in detail. The many different methods of constructing codes have been described, particularly those based on using known good or outstanding codes as a construction basis. Using such methods, several hundred new codes have been presented or described which are improvements to the public database of best, known codes.

For cyclic codes, which have implementation advantages over other codes, many new outstanding codes have been presented including the determination of a table giving the code designs and highest attainable minimum distance of all binary cyclic codes of odd lengths from 129 to 189. It has been shown that outstanding cyclic codes may be used as code components to produce new codes that are better than the previously thought best codes, for the same code length and code rate.

Appendix

Improved Lower Bounds of the Minimum Distance of Binary Linear Codes

The following tables list the updated lower bounds of minimum distance of linear codes over $\mathbb{F}_2$. These improvements—there are 901 of them in total—are due to the new binary linear codes described above. In the tables, entries marked with C refer to cyclic codes, those marked with X, XX and $Y1$ refer to codes obtained from Constructions X, XX and Y1, respectively. Similarly, entries marked with E, P and S denote $[n, k, d]$ codes obtained by extending (annexing an overall parity-check bit) to $(n-1, k, d')$ codes, puncturing $(n+1, k, d+1)$ codes and shortening $(n+1, k+1, d)$ codes, respectively. Unmarked entries are the original lower bounds of Brouwer [6].

References

1. Alltop, W.O.: A method of extending binary linear codes. IEEE Trans. Inf. Theory **30**(6), 871–872 (1984)
2. Berlekamp, E., McEliece, R., van Tilborg, H.: On the inherent intractability of certain coding problems. IEEE Trans. Inf. Theory **24**, 384–386 (1978)
3. Berlekamp, E.R.: Algebraic Coding Theory. Aegean Park Press, Laguna Hills (1984). ISBN 0 894 12063 8
4. Bitner, J.R., Ehrlich, G., Reingold, E.M.: Efficient generation of the binary reflected gray code and its applications. Commun. ACM **19**(9), 517–521 (1976)
5. Bosma, W., Cannon, J.J., Playoust, C.P.: Magma Algebra Syst. User Lang. **24**, 235–266 (1997)
6. Brouwer, A.E.: Bounds on the size of linear codes. In: Pless, V.S., Huffman, W.C. (eds.) Handbook of Coding Theory, pp. 295–461. Elsevier, North Holland (1998)
7. Chen C.L. (1969) Some results on algebraically structured error-correcting codes. Ph.D Dissertation, University of Hawaii, USA
8. Chen, C.L.: Computer results on the minimum distance of some binary cyclic codes. IEEE Trans. Inf. Theory **16**(3), 359–360 (1970)
9. Grassl, M.: On the minimum distance of some quadratic residue codes. In: Proceedings of the IEEE International Symposium on Information and Theory, Sorento, Italy, p. 253 (2000)
10. Grassl, M.: New binary codes from a chain of cyclic codes. IEEE Trans. Inf. Theory **47**(3), 1178–1181 (2001)
11. Grassl, M.: Searching for linear codes with large minimum distance. In: Bosma, W., Cannon, J. (eds.) Discovering Mathematics with MAGMA - Reducing the Abstract to the Concrete, pp. 287–313. Springer, Heidelberg (2006)
12. Grassl M.: Code tables: bounds on the parameters of various types of codes. http://www.codetables.de
13. Knuth D.E. (2005) The Art of Computer Programming, Vol. 4: Fascicle 3: Generating All Combinations and Partitions, 3rd edn. Addison-Wesley, ISBN 0 201 85394 9
14. Leon, J.S., Masley, J.M., Pless, V.: Duadic codes. IEEE Trans. Inf. Theory **30**(5), 709–713 (1984)
15. Loeloeian, M., Conan, J.: A [55,16,19] binary Goppa code. IEEE Trans. Inf. Theory **30**, 773 (1984)

16. MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes. North-Holland, Amsterdam (1977)
17. Nijenhuis, A., Wilf, H.S.: Combinatorial Algorithms for Computers and Calculators, 2nd edn. Academic Press, London (1978)
18. Payne, W.H., Ives, F.M.: Combination generators. ACM Trans. Math. Softw. **5**(2), 163–172 (1979)
19. Prange E.: Cyclic error-correcting codes in two symbols. Technical report TN-58-103, Air Force Cambridge Research Labs, Bedford, Massachusetts, USA (1957)
20. Proakis, J.G.: Digital Communications, 3rd edn. McGraw-Hill, New York (1995)
21. Promhouse, G., Tavares, S.E.: The minimum distance of all binary cyclic codes of odd lengths from 69 to 99. IEEE Trans. Inf. Theory **24**(4), 438–442 (1978)
22. Schomaker, D., Wirtz, M.: On binary cyclic codes of odd lengths from 101 to 127. IEEE Trans. Inf. Theory **38**(2), 516–518 (1992)
23. Sloane, N.J., Reddy, S.M., Chen, C.L.: New binary codes. IEEE Trans. Inf. Theory **IT–18**, 503–510 (1972)
24. Vardy, A.: The intractability of computing the minimum distance of a code. IEEE Trans. Inf. Theory **43**, 1759–1766 (1997)
25. Zimmermann, K.H.: Integral hecke modules, integral generalized reed-muller codes, and linear codes. Technical report 3–96, Technische Universität Hamburg-Harburg, Hamburg, Germany (1996)