Enumeration of the Binary Self-Dual Codes of Length 34

R. T. Bilous
Department of Computer Science, University of Manitoba
Winnipeg, Manitoba, Canada R3T 2N2
umbiloul@cs.umanitoba.ca

Abstract

An (n, k) binary self-orthogonal code is an (n, k) binary linear code C that is contained in its orthogonal complement $C^{\perp}$. A self-orthogonal code C is self-dual if $C = C^{\perp}$. Two codes, C_1 and C_2, are equivalent if and only if there exists a coordinate permutation of C_1 that takes C_1 into C_2. The automorphism group of a code C is the set of all coordinate permutations of C that takes C into itself.

This paper is a continuation of the work presented in [2], in which we described an algorithm for enumerating inequivalent binary self-dual codes. We used our algorithm to enumerate the self-dual codes of length up to and including 32. Our algorithm also found the size of the automorphism group of each code.

We have since made several improvements to our algorithm. It now generally runs faster. It also now finds generators for the automorphism group of each code. We have used our improved algorithm to enumerate the self-dual codes of length 34. We have also found the automorphism groups for each of our self-dual codes of length less than or equal to 34. The list of length 34 codes are new, as are the lists of automorphism groups for the length 32 and length 34 codes. We have found there are 19914 inequivalent length 34 codes with distance 4 and 938 length 34 codes with distance 6.

1 Introduction

Undefined coding theory terms and examples can be found in MacWilliams and Sloane [5]. Let $V_n(2)$ denote the vector space of all binary n-vectors. An (n, k) binary linear code C is a k dimensional subspace of $V_n(2)$. (We only consider binary linear codes in this paper, and thus, whenever we use

JCMCC **59** (2006), pp. 173-211

the term code we will be referring to this class of codes only.) The integers n and k are the length and dimension of C, respectively. The n-vectors in C are called codewords. The weight of a codeword $\vec{c}$, which is denoted by $w(\vec{c})$, is the number of ones in $\vec{c}$. The weight distribution of C is a count of the number of codewords in C with weight i, for $i = 0, 1, \ldots, n$. The weight distribution is represented by the sequence $(A_0, A_1, \ldots, A_n)$, where A_i is the number of weight i codewords in C. The distance of C is the smallest weight of any non-zero codeword in C. An (n, k, d) code is an (n, k) code with distance d.

Two codes C_1 and C_2 are equivalent if and only if there exists a coordinate permutation of C_1 that takes C_1 into C_2. If C_1 and C_2 are not equivalent then C_1 and C_2 are said to be inequivalent. The equivalence class of a code C is the set of all codes that are equivalent to C. An automorphism of a code C is a coordinate permutation that takes C into itself. The set of all automorphisms of C, which is a group, is called the automorphism group of C.

A generator matrix for an (n, k) code C is a $k \times n$ binary matrix whose rows form a basis for C. We typically represent our codes with generator matrices. For any code C, there exists a unique generator matrix that is in row-reduced echelon form. Furthermore, there exists a code that is equivalent to C and is generated by a matrix of the form $[I_k|A]$, where I_k is a $k \times k$ identity matrix and A is an $k \times (n-k)$ binary matrix.

Let $\vec{u} = (u_1, u_2, \ldots, u_n)$ and $\vec{v} = (v_1, v_2, \ldots, v_n)$ be any two n-vectors in $V_n(2)$. The dot product of $\vec{u}$ and $\vec{v}$, written $\vec{u} \bullet \vec{v}$, is defined as $(\sum_{i=1}^{n} u_i v_i) \bmod 2$. Two vectors $\vec{u}$ and $\vec{v}$ are called orthogonal if $\vec{u} \bullet \vec{v} = 0$. The orthogonal complement of an (n, k) code C, written $C^{\perp}$, is the set of all n-vectors that are orthogonal to every codeword in C. It is well known that $C^{\perp}$ is an $(n, n-k)$ code. If $C^{\perp} \subseteq C$ then C is called self-orthogonal. If $C^{\perp} = C$ then C is called self-dual. A self-orthogonal code is a self-dual code if and only if $n = 2k$.

In this paper, we extend the research presented in [2], in which we described an algorithm for enumerating inequivalent self-dual codes. We used our algorithm to enumerate the self-dual codes of length up to and including 32. The algorithm also found the size of the automorphism group of each code produced. We have since made several improvements to our algorithm. We have used our improved algorithm to enumerate the self-dual codes of length 34. The number of such codes is summarized in the following theorem:

Theorem 1.1 There are 24147 equivalence classes of $(34, 17)$ binary self-dual codes, of which 3295 consist of codes with distance 2, 19914 consist of codes with distance 4, and 938 consist of codes with distance 6.

Our algorithm now also finds generators for the automorphism group of each code produced. We have used it to find the automorphism groups of all codes we have enumerated (i.e. all codes with length up to and including 34). The enumeration of the length 34 codes and the lists of generators for the automorphism groups of the length 32 and length 34 codes are new.

Our enumeration consists of two main stages. In the first stage, we produce a list of $(2k, k)$ self-dual codes that contains at least one code from each equivalence class of $(2k, k)$ self-dual codes. A review of this stage of our algorithm is given in Section 2. In the second stage of our algorithm, we eliminate all the equivalent codes produced in the first stage of the algorithm. This is accomplished by running a program on each code C, produced in the first stage of the enumeration, that produces a unique code C_u in the equivalence class of C. The automorphism group of the code C_u is also found by our program. In Section 3, we give a somewhat detailed description of our program. We conclude with Section 4, in which we give a summary of the results of our enumeration. Included is a table of all the different weight distributions for the length 34 codes, and the number of equivalence classes whose codes have a given distribution. The length 34 codes with distance 6, along with their automorphism groups, are available on the world wide web at:

www.cs.umanitoba.ca/~umbiloul/

Further information on the length 34 codes can be obtained from the author of this paper.

2 Enumerating the Codes

Our enumeration of the self-dual codes is split into three cases: the distance 2 codes, the distance 4 codes, and the codes with distance greater than 4. Enumeration of the distance 2 codes is trivial, so we will only discuss our enumerations of the other two cases. Both enumerations are recursive.

2.1 The Distance 4 Codes

We can enumerate the $(2k, k, 4)$ self-dual codes provided we have available a complete list of $(2k-4, k-2)$ self-dual codes. (Such a list of

$$\mathcal{A}(A_0,\vec{v}) = \left[\begin{array}{cc|c} 1 & 0 & \vec{v} \\ 0 & 1 & \vec{v} \\ \hline & & \\ & F & A_0 \\ & & \end{array}\right]$$

Figure 1: The $k \times k$ matrix $\mathcal{A}(A_0,\vec{v})$. The $(k-2)$-vector $\vec{v}$ has even weight greater than or equal to two. The $(k-2) \times (k-2)$ matrix A_0 is such that $[I_{k-2}|A_0]$ generates a $(2k-4, k-2)$ self-dual code. The rows in $\mathcal{A}(A_0,\vec{v})$ must be orthogonal which means the rows of the $(k-2) \times 2$ matrix F are uniquely determined by $\vec{v}$ and A_0.

$(2k-4, k-2)$ self-dual codes can be found, of course, by running both stages of our enumeration algorithm on the length $k-2$ codes.) Our algorithm for enumerating a list of $(2k, k, 4)$ self-dual codes that contains at least one representative from each equivalence class is based on the following result:

Theorem 2.1 Let C be a $(2k, k, 4)$ binary self-dual code. Then there exists a code C', equivalent to C, that is generated by a matrix of the form $[I_k|\mathcal{A}(A_0,\vec{v})]$, where $\vec{v}$ is a $(k-2)$-vector with even weight and A_0 is a $(k-2) \times (k-2)$ binary matrix such that $[I_{k-2}|A_0]$ generates a $(2k-4, k-2)$ binary self-dual code. The $k \times k$ matrix $\mathcal{A}(A_0,\vec{v})$ is as given in Figure 1.

Proof See [2]. □

The converse of Theorem 2.1 is also true. That is, given a generator matrix $G_0 = [I_{k-2}|A_0]$ for a $(2k-4, k-2)$ self-dual code C, and a $(k-2)$-vector $\vec{v}$ with even weight, the matrix $G = [I_k|\mathcal{A}(A_0,\vec{v})]$ generates a $(2k, k)$ self-dual code. This leads us to the following simple algorithm for enumerating a list of $(2k, k, 4)$ self-dual codes that contains at least one representative from each equivalence class:

Algorithm 2.2 :

Input: A list L_{in} of $(k-2) \times 2(k-2)$ matrices $[I_{k-2}|A_0]$ such that for any $(2k-2, k-2)$ self-dual code C_0, there exists one and only one matrix in L_{in} that generates a code equivalent to C_0.

Output: A list L_{out} of $k \times 2k$ matrices $[I_k|A]$ such that for any $(2k, k, 4)$ self-dual code C, there exists at least one matrix in L_{out} that generates a code equivalent to C.

```
begin
  Clear L_out.
  for each [I_{k-2}|A_0] in L_in do:
    for each even weight (k-2)-vector v do:
      Set A = 𝒜(A_0, v).
      if [I_k|A] generates a code with distance 4 then:
        Insert [I_k|A] in L_out (unless its already present).
      end if
    end for
  end for
end
```

We refer to the process of producing the generator matrix $G = [I_k|A]$, from the generator matrix $G_0 = [I_{k-2}|A_0]$ and $(k-2)$-vector $\vec{v}$, as *extending* the code C_0 to the code C using the vector $\vec{v}$, where C and C_0 are the codes generated by G and G_0, respectively. We refer to the vector $\vec{v}$ as the *extension vector.*

Due to the exponential number of vectors $\vec{v}$, as is, Algorithm 2.2 is too slow and produces too many equivalent codes to be of practical use. However, many improvements can be made to this algorithm that results in both a reduction in the running time and the number of equivalent codes produced. A substantial reduction in the running time was achieved by examining the different ways the weight 4 words of a self-dual code can interact. This led us to a classification of the distance 4 codes based on the weight 4 structures they contain. Each class of codes was enumerated separately. This resulted in a reduction in the number of codes input into our algorithm, the number of vectors $\vec{v}$ used by our algorithm, and the number of equivalent codes produced by the algorithm. We were also able to reduce the number of equivalent codes produced by restricting the form of the generator matrices produced. For example, whenever we produced a matrix $[I_k|A]$, we sorted the rows and columns of A in *descending* order (i.e. in order of decreasing binary value when each row/column is considered a binary integer). For further information see [1].

We have used our algorithm to produce a *manageable* list of the distance 4 self-dual codes of length 34. The enumeration took approximately 15 hours. (Note: all our programs were run on a Solaris 2.5.) Our list of length 34 codes contained, on average, 3.4 equivalent codes.

2.2 The Distance ≥ 6 Codes

Once we have our list of $(2k, k, 4)$ self-dual codes that contain at least one code from each equivalence class, we then execute the second stage of our algorithm: removing the equivalent codes from our list (which we discuss in Section 3). This leaves us with a list of $(2k, k, 4)$ self-dual codes that contain one and only one code from each equivalence class. This list is used to enumerate the $(2k, k, d \geq 6)$ self-dual codes.

The $(2k, k, d \geq 6)$ self-dual codes come from the $(2k, k, 4)$ self-dual codes that do not contain any intersecting weight 4 words (i.e. codes that do not contain weight 4 words that have a value of one in a common coordinate.) Our algorithm for enumerating the $(2k, k, d \geq 6)$ codes is based on the following result:

Theorem 2.3 Let C_0' be a $(2k, k, d \geq 6)$ self-dual code. Then there exists a code C_0 generated by $G_0 = [I_k | A_0]$, where C_0 is equivalent to C_0', and a k-vector $\vec{v}$, such that $G = [I_{k+2} | \mathcal{A}(A_0, \vec{v})]$ generates a $(2k+4, k+2, 4)$ self-dual code C that contains one and only one weight 4 word. Furthermore, there are at most two other inequivalent codes that can be extended to a code equivalent to C. One of these codes is a $(2k, k, 4)$ self-dual code that does not contain any intersecting weight 4 words. The other is a $(2k, k, d')$ self-dual code, where $4 \leq d' \leq d - 2$.

Proof See [2]. □

Given any one of the three smaller codes in Theorem 2.3, we can find the other two codes as follows: Let C_1, with generator matrix $G_1 = [I_k | A_1]$, be any one of the three $(2k, k)$ codes that can be extended to a code equivalent to the $(2k+4, k+2, 4)$ code C in Theorem 2.3, and let $1\vec{u}$ be an extension vector that extends C_1 to a code equivalent to C. Then the matrices $G_2 = [I_k | \mathcal{R}(A_1, 1\vec{u})]$ and $G_3 = [I_k | \mathcal{S}(A_1, 1\vec{u})]$, where $\mathcal{R}(A_1, 1\vec{u})$ and $\mathcal{S}(A_1, 1\vec{u})$ are defined in Figure 2, generate the other two codes. This leads us to the following algorithm for enumerating a list of $(2k, k, d \geq 6)$ self-dual codes that contains at least one representative from each equivalence class:

$$A_1 = \left[\begin{array}{c|c} 0 & \vec{w} \\ \hline 1 & X_0 \\ 0 & X_1 \\ 1 & X_2 \\ 0 & X_3 \end{array}\right] \quad \begin{array}{l} \vec{w} \bullet \vec{u} \equiv 1 \bmod 2 \\ \text{for any } \vec{x} \in X_0,\ \vec{x} \bullet \vec{u} \equiv 0 \bmod 2 \\ \text{for any } \vec{x} \in X_1,\ \vec{x} \bullet \vec{u} \equiv 1 \bmod 2 \\ \text{for any } \vec{x} \in X_2,\ \vec{x} \bullet \vec{u} \equiv 1 \bmod 2 \\ \text{for any } \vec{x} \in X_3,\ \vec{x} \bullet \vec{u} \equiv 0 \bmod 2 \end{array}$$

$$\mathcal{R}(A_1, 1\vec{u}) = \left[\begin{array}{c|c} 1 & \vec{u} \oplus \vec{w} \\ \hline 0 & X_0 \oplus \vec{u} \\ 1 & X_1 \oplus \vec{u} \\ 1 & X_2 \\ 0 & X_3 \end{array}\right] \qquad \mathcal{S}(A_1, 1\vec{u}) = \left[\begin{array}{c|c} 0 & \vec{u} \\ \hline 1 & X_0 \oplus \vec{u} \oplus \vec{w} \\ 1 & X_1 \oplus \vec{w} \\ 0 & X_2 \oplus \vec{u} \\ 0 & X_3 \end{array}\right]$$

Figure 2: The $k \times k$ matrices $\mathcal{R}(A_1, 1\vec{u})$ and $\mathcal{S}(A_1, 1\vec{u})$ are produced from the $k \times k$ matrix A_1 and the $(k-1)$-vector $\vec{u}$. The notation $X \oplus \vec{u}$ means add modulo 2, component by component, $\vec{u}$ to each vector (i.e. row segment) in X.

Algorithm 2.4 :

Input: A list L_{in} of $k \times 2k$ matrices $[I_k|A_1]$ such that for any $(2k, k, 4)$ self-dual code C_1, that does not contain any intersecting weight 4 words, there exists one and only one matrix in L_{in} that generates a code equivalent to C_1.

Output: A list L_{out} of $k \times 2k$ matrices $[I_k|A_0]$ such that for any $(2k, k, d \geq 6)$ self-dual code C_0, there exists at least one matrix in L_{out} that generates a code equivalent to C_0.

begin
  Clear L_{out}.
  for each $[I_k|A_1]$ in L_{in} **do:**
    for each odd weight $(k-1)$-vector $\vec{u}$ **do:**
      if $[I_{k+2}|A(A_1, 1\vec{u})]$ generates a $(2k, k, 4)$ code with only one weight 4 word **then:**
        Set $A_2 = \mathcal{R}(A_1, 1\vec{u})$.
        Set $A_3 = \mathcal{S}(A_1, 1\vec{u})$.
        Let $d_2 =$ the distance of the code generated by $[I_k|A_2]$.
        Let $d_3 =$ the distance of the code generated by $[I_k|A_3]$.
        if $d_2 \neq d_3$ **then:**
          if $d_2 > d_3$ **then:**
            Set $A_0 = A_2$.
          else
            Set $A_0 = A_3$.

```
        end if-then-else
        Insert [I_k|A_0] in L_out (unless already present).
      end if
    end if
  end for
 end for
end
```

As with our algorithm for the distance 4 codes, this simple algorithm is too slow and produces too many equivalent codes to be of practical use. However, as with the distance 4 case, many improvements can be made to this algorithm that result in both a reduction in the running time and the number of equivalent codes produced. Again, this can be accomplished by examining the structures of the weight d words in the codes and by limiting the number of different codes produced. For further information see [1].

We have used our algorithm to produce a *manageable* list of the distance $d \geq 6$ self-dual codes of length 34. The enumeration took approximately 150 hours. Our list of length 34 codes contained, on average, 5.8 equivalent codes.

3 Eliminating Equivalent Codes

Once we have our list L of $(2k, k)$ self-dual codes that contains at least one representative from each equivalence class of interest, we then remove all of the equivalent codes from L. We accomplish this by running, on each code C in L, a program that produces a unique code C_u in C's equivalence class. That is, if C_1 and C_2 are equivalent codes, then when we run our program on C_1, the code C_u produced is the same code that is produced when we run our program on C_2. By replacing each code C in L with the equivalent code C_u produced by our program, and then removing all duplicates from L, we end up with a list L of codes that contains one and only one code from each equivalence class of interest. Our program also finds generators for the automorphism group of C_u.

We refer to the algorithm used by our program as a *unique representative algorithm.* We refer to the code C_u produced by the program as a *unique representative* for the equivalence class of the input code C. In this section, we will describe our unique representative algorithm.

3.1 A Simple Unique Representative Algorithm

Due to the complexity of our algorithm, we will begin by describing a simplified version of our unique representative algorithm. Though this simplified algorithm becomes impractical for relatively small codes, it does form a basis for our final algorithm.

Let C be a $(2k, k)$ self-dual code. Let $\mathcal{G}(C)$ denote the set of all $k \times 2k$ binary matrices $G = [I_k|A]$ that generate a code equivalent to C. Given any $G \in \mathcal{G}(C)$, our unique representative algorithm produces a unique $G_u \in \mathcal{G}(C)$ (that, of course, generates a unique code C_u that is equivalent to C). Before we can describe which matrix G_u our simplified unique representative algorithm produces, we must first introduce some terminology involving comparisons of various structures.

Let G be a generator matrix for a $(2k, k)$ self-dual code and let W_i denote the number of rows with weight i in G, for $i = 2, 4, \ldots, 2k$. We will refer to the sequence $(W_2, W_4, \ldots, W_{2k})$ as the *row weight distribution* of G. Let G and G' be matrices with row weight distributions $(W_2, W_4, \ldots, W_{2k})$ and $(W'_2, W'_4, \ldots, W'_{2k})$, respectively. We will say the row weight distribution of G is *less than* the row weight distribution of G' if $W_i < W'_i$ where i is the smallest integer in which W_i and W'_i differ. Let v and v' be binary strings of length $2k$. We will say v is *less than* v' if the binary value of v is less than the binary value of v'. Let M and M' be two binary matrices with equal dimensions. We will say M is *less than* M' if row i of M (which is a binary string) is less than row i of M', where row i is the first row (from top to bottom) in which M and M' differ. (Note, for each of the comparison operations we have just defined, we have only described what we mean by *less than*. We will also use phrases such as *equal to* and *greater than* with the obvious interpretations.)

The matrix G_u produced by our simplified unique representative algorithm is the unique matrix in $\mathcal{G}(C)$ that possesses the following two properties: 1) G_u has a minimal row weight distribution (i.e. the row weight distribution of G_u is less than or equal to the row weight distribution of all other matrices in $\mathcal{G}(C)$), and 2) G_u is greater than all other matrices in $\mathcal{G}(C)$ that have a minimal row weight distribution.

Given any $G = [I_k|A]$ in $\mathcal{G}(C)$, our unique representative algorithm produces G_u by applying a sequence of column permutations and elementary row operations to G. Of course, any sequence of such operations results in a matrix that is an element of $\mathcal{G}(C)$. In describing our algorithm, we will make use of the following conventions and notation regarding these operations: We will use permutations π of the integers $1, 2, \ldots, 2k$ to denote both permutations of the columns of G and permutations of the coordinates of C. That is, given the permutation:

$$\pi = \begin{pmatrix} 1 & 2 & \cdots & 2k \\ a_1 & a_2 & \cdots & a_{2k} \end{pmatrix}$$

we will use the notation πC to denote the code C' that is obtained by moving coordinate a_i of C to coordinate i of C', for $i = 1, 2, \ldots, 2k$. Similarly, we will use the notation πG to denote the matrix G' that is obtained by inserting column a_i of G into column i of G', for $i = 1, 2, \ldots, 2k$. The notation $\mathcal{RREF}(\pi G)$ will be used to denote the row-reduced echelon form of πG. Let π_L denote any permutation of the form:

$$\begin{pmatrix} 1 & 2 & \cdots & k & k+1 & k+2 & \cdots & 2k \\ a_1 & a_2 & \cdots & a_k & k+1 & k+2 & \cdots & 2k \end{pmatrix}$$

and let π_R denote any permutation of the form:

$$\begin{pmatrix} 1 & 2 & \cdots & k & k+1 & k+2 & \cdots & 2k \\ 1 & 2 & \cdots & k & b_1 & b_2 & \cdots & b_k \end{pmatrix}.$$

We will use Π_L to denote the set of all such permutations π_L and Π_R to denote the set of all such permutations π_R. Finally, given a generator matrix $G' = [I_k|A']$ and a permutation $\pi_L\pi_R$, we will refer to the permutations π_r and π_c,

$$\pi_r = \begin{pmatrix} 1 & 2 & \cdots & k \\ a_1 & a_2 & \cdots & a_k \end{pmatrix} \text{ and } \pi_c = \begin{pmatrix} 1 & 2 & \cdots & k \\ b_1 - k & b_2 - k & \cdots & b_k - k \end{pmatrix},$$

as the permutations of the rows and columns of A', respectively, that correspond to the permutations π_L and π_R, respectively. The reason for this is that when we find the row-reduced echelon form of the matrix $\pi_L\pi_R G$, the matrix that results is $[I_k|A'']$, where A'' is obtained by applying π_r to the rows of A' and π_c to the columns of A'.

Now, given the matrix $G = [I_k|A]$, our unique representative algorithm finds G_u in two stages. In the first stage, it produces a set L of pairs (G', Π), where $G' \in \mathcal{G}(C)$ and Π is a set of permutations π in which $\mathcal{RREF}(\pi G) = G'$. (Note, the set Π is used to find the automorphism group of C_u, which we will discuss later.) The set L produced has the property that if $G'' = [I_k|A'']$ is any matrix in $\mathcal{G}(C)$ with a minimal row weight distribution, then there exists at least one $G' = [I_k|A']$ in L in which the rows and columns of A' are a permutation of the rows and columns of A''. For each $G' = [I_k|A']$ in L, the second stage of our algorithm finds the largest matrix $G'' = [I_k|A'']$ in which the rows and columns of A'' are permutations of the rows and columns of A'. The largest $G'' = [I_k|A'']$ found in the second stage gives us our unique matrix G_u. For reasons that should soon be clear, we refer to the first stage of our algorithm as the *combination algorithm* and the second stage of our algorithm as the *permutation algorithm.*

3.1.1 Our Simplified Combination Algorithm

The input to our combination algorithm is a generator matrix $G = [I_k|A]$ for a $(2k,k)$ self-dual code C. The output is a set L of pairs (G', Π), where G' is a generator matrix for a code equivalent to C and Π is a set of permutations π in which $\mathcal{RREF}(\pi G) = G'$.

Our combination algorithm is based on a partition of the set of all $(2k)!$ permutations of the integers $1, 2, \ldots, 2k$ into $\binom{2k}{k}$ disjoint sets $\Pi(T)$, which we define as follows: Let $S_{2k,k}$ denote the set of all k-combinations of the set of integers $\{1, 2, \ldots, 2k\}$. That is, $S_{2k,k}$ is the set of all subsets T of $\{1, 2, \ldots, 2k\}$ in which $|T| = k$. For each $T \in S_{2k,k}$, we define $\Pi(T)$ as the set of all permutations π that move the k integers in T to the first k integers. In other words, $\Pi(T)$ is the set of all $(k!)^2$ permutations of the form:

$$\begin{pmatrix} 1 & 2 & \cdots & k & k+1 & k+2 & \cdots & 2k \\ a_1 & a_2 & \cdots & a_k & b_1 & b_2 & \cdots & b_k \end{pmatrix}$$

where $\{a_1, a_2, \ldots, a_k\} = T$. Note that the set $\Pi(T)$ is equal to the set $\Pi_L \Pi_R \pi$, where π is any permutation in $\Pi(T)$.

For each $T \in S_{2k,k}$, we will use the notation $\mathcal{G}(T)$ to denote the set of all matrices $\mathcal{RREF}(\pi G)$, where $\pi \in \Pi(T)$. Given any $G' \in \mathcal{G}(T)$, we can generate the set $\mathcal{G}(T)$ by producing each matrix $\mathcal{RREF}(\pi_L \pi_R G')$, where $\pi_L \in \Pi_L$ and $\pi_R \in \Pi_R$. This fact can be used to easily deduce the following three properties of the sets $\mathcal{G}(T)$:

P1 Either every matrix in $\mathcal{G}(T)$ begins with an identity matrix (i.e. is of the form $[I_k|A']$), or none of the matrices in $\mathcal{G}(T)$ begin with an identity matrix.

P2 If the matrices in $\mathcal{G}(T)$ begin with I_k then, given any $G' = [I_k|A'] \in \mathcal{G}(T)$, the matrix $G'' = [I_k|A'']$ is in $\mathcal{G}(T)$ if and only if the rows and columns of A'' are permutations of the rows and columns of A'.

P3 Either $\mathcal{G}(T_1) = \mathcal{G}(T_2)$ or $\mathcal{G}(T_1) \cap \mathcal{G}(T_2) = \emptyset$, where $T_1, T_2 \in S_{2k,k}$.

For each $T \in S_{2k,k}$, our combination algorithm produces one matrix G' in $\mathcal{G}(T)$. Property P1 tells us that if the matrix G' produced does not begin with an identity matrix then none of the matrices in $\mathcal{G}(T)$ begin with an identity matrix, and thus, cannot contain G_u. Property P2 tells us that if $G' = [I_k|A']$ then every matrix in $\mathcal{G}(T)$ can be produced by simply permuting the rows and columns of A'. Furthermore, every matrix in $\mathcal{G}(T)$ will share any characteristic that does not depend on the arrangement of the rows and columns of A'. In particular, the row weight distributions of every matrix in $\mathcal{G}(T)$ will be the same. Property P3 tells us that whenever our algorithm produces matrices $G_1 = [I_k|A_1]$ and $G_2 = [I_k|A_2]$ in which $A_1 = A_2$, where $G_1 \in \mathcal{G}(T_1)$ and $G_2 \in \mathcal{G}(T_2)$ for some $T_1, T_2 \in S_{2k,k}$, then the sets $\mathcal{G}(T_1)$ and $\mathcal{G}(T_2)$ will be the same, and thus, only one needs to be searched for G_u.

For each $T \in S_{2k,k}$, the combination algorithm performs the following action: First, it produces a permutation $\pi_0 \in \Pi(T)$ and the matrix $G_0 = \mathcal{RREF}(\pi_0 G)$. If G_0 does not begin with I_k, it is discarded and the algorithm continues on to the next combination in $S_{2k,k}$. If $G_0 = [I_k|A_0]$, the algorithm will then sort the rows and columns of A_0 in descending order by calling a function *QuickOrderMatrix*(A_0, π_0, A', π). This function uses a simple, non-exhaustive algorithm to sort the rows and columns of A_0 in descending order, producing the matrix A'. It also returns a permutation $\pi \in \Pi(T)$ in which $\mathcal{RREF}(\pi G) = [I_k|A']$. *QuickOrderMatrix* sorts the rows and columns of A_0 in linear time with respect to the number of rows in A_0. However, since it is not an exhaustive algorithm (and since there may exist many different matrices whose rows and columns are in descending order that differ in only the arrangement of their rows and columns), given any two matrices A_0 and A_0', whose rows and columns are permutations of one another, *QuickOrderMatrix* may not necessarily produce the same matrix. The reason we call *QuickOrderMatrix* is to increase the chances of our algorithm producing equal matrices. That is, if T_1 and T_2 are such that $\mathcal{G}(T_1) = \mathcal{G}(T_2)$, then by sorting the rows and columns of our matrices, we increase the chances of our algorithm producing matrices $G_1 \in \mathcal{G}(T_1)$ and $G_2 \in \mathcal{G}(T_2)$ in which $G_1 = G_2$, which in turn reduces the number of matrices we need to consider in our permutation algorithm.

Once our algorithm has produced a matrix $G' = \mathcal{RREF}(\pi G) = [I_k|A']$, in which the rows and columns of A' are in descending order, it adjusts the set of pairs L as follows: First, it compares the row weight distribution of G' to the row weight distribution of the matrices already in L. If G' has a larger row weight distribution, it is discarded. If G' has a smaller row weight distribution, L is replaced with the set consisting of the single pair $(G', \{\pi\})$. If G' has an equal row weight distribution then the algorithm will first determine if L contains a pair (G'', Π) in which $G'' = G'$. If this is the case then π is added to the set Π. If this is not the case then the pair $(G', \{\pi\})$ is added to L. Note that the reason we include the permutation π in the set L is that is allows us to find the set of permutations $\Pi(T)$, which in turn will allow us to find the automorphism group of G_u.

At the end of our combination algorithm, we will have produced a set L of pairs (G', Π) with the property that if T is any k-combination in

$S_{2k,k}$ in which the matrices in $\mathcal{G}(T)$ begin with I_k and have a minimal row weight distribution, then there exists one and only one pair (G', Π) in L in which Π contains a permutation $\pi \in \Pi(T)$. We summarize the combination algorithm for our simplified unique representative algorithm in Algorithm 3.1.

Algorithm 3.1 : *CombinationAlgorithm*(G, L)

Input: A generator matrix $G = [I_k|A]$ for a $(2k, k)$ self-dual code C.
Output: A set L of pairs (G', Π), where G' generates a code equivalent to C and Π is a set of permutations π in which $\mathcal{RREF}(\pi G) = G'$.

```
begin
  Initialize L = ∅.
  for each T ∈ S_{2k,k} do
    Pick any permutation π_0 in Π(T).
    Set G_0 = RREF(π_0 G).
    if G_0 = [I_k|A_0] then
      QuickOrderMatrix(A_0, π_0, A', π).
      Let G' = [I_k|A'].
      if the row weight distribution of G' is less than the
      row weight distribution of the matrices in L then
        Set L = {(G', {π})}.
      else if the row weight distribution of G' is equal to the
      row weight distribution ofthe matrices in L then
        if the pair (G', Π) is an element of L then
          Replace Π with Π ∪ {π}.
        else
          Insert the pair (G', {π}) into L.
        end if-then-else
      end if-then-else-if
    end if
  end for
end
```

3.1.2 Our Simplified Permutation Algorithm

The input to our permutation algorithm is the set L of pairs output by the combination algorithm. For each pair (G', Π) in L, where $G' = [I_k|A']$, the permutation algorithm finds the unique matrix $G'' = [I_k|A'']$, where A'' is the largest matrix whose rows and columns are permutations of the rows and columns of A'. The largest such matrix G'' produced by the algorithm gives us our generator matrix G_u for the unique representative C_u.

Given any $G' = [I_k|A']$, the permutation algorithm finds the largest matrix $G'' = [I_k|A'']$, where the rows and columns of A'' are permutations of the rows and columns of A', by calling a function *UniqueOrderMatrix*(A', A'', Π''). This function is based on the following two facts: First, the rows and columns of A'' must be in descending order (otherwise, we could swap any pair of offending rows (or columns) and get a larger matrix). Second, for any permutation π_r of the rows of A' there exists at most one column permutation π_c in which the rows and columns of $\pi_r\pi_c A'$ are in descending order. (The reason there is at most one such column permutation is that, provided $[I_k|A']$ generates a self-dual code with distance greater than 2, the columns of A' are distinct.) The function *UniqueOrderMatrix* uses a recursive algorithm to consider all the different row permutations of A' that may lead to A''. At the start of the i^{th} level of the recursion, the first $i-1$ rows and first $i-1$ components of the columns of A' are in descending order. The algorithm then tries each possibility for row i of A' (where row i is selected from the last $k-i$ rows of A') that may lead us to A''. For each possibility, the algorithm moves the selected row to row i of A', sorts the first i components of the columns of the resulting matrix in descending order, and then proceeds to the $(i+1)^{th}$ level of the recursion. The function *UniqueOrderMatrix* also returns the set Π'' of all permutations $\pi_L\pi_R$, where $\pi_L \in \Pi_L$ and $\pi_R \in \Pi_R$, in which $\mathcal{RREF}(\pi_L\pi_R G') = [I_k|A'']$. The set Π'' is found by the algorithm with simple bookkeeping.

The permutation algorithm also finds the automorphism group of C_u. It does this by finding the set Π_u of all permutations π in which $\mathcal{RREF}(\pi G) = G_u$ (where G is the matrix input into the combination algorithm). This immediately gives us the automorphism group of C_u since $\mathcal{AUT}(C_u) = \Pi_u\pi^{-1}$, where π is any permutation in Π_u.

The permutation algorithm finds Π_u by producing two sets of permutations Π_c and Π_p. Let S_u denote the set of all k-combinations T in which $G_u \in \mathcal{G}(T)$. The set Π_c consists of one and only one permutation π_c from each set $\Pi(T)$, where $T \in S_u$, such that $\mathcal{RREF}(\pi_c G) = G_u$. The set Π_p consists of all permutations $\pi_L\pi_R$, where $\pi_L \in \Pi_L$ and $\pi_R \in \Pi_R$, such that $\mathcal{RREF}(\pi_L\pi_R G_u) = G_u$. For each $\pi_c \in \Pi_C$, the set $\Pi_p\pi_c$ gives us the set of all permutations $\pi \in \Pi(T)$ in which $\mathcal{RREF}(\pi G) = G_u$, where T is the k-combination whose set $\Pi(T)$ contains π_c. This is due to the fact that, as mentioned, given any $\pi_c \in \Pi(T)$ the set $\Pi(T)$ is equal to the set of all permutations $\Pi_L\Pi_R\pi_c$. Therefore, since $\mathcal{RREF}(\pi_c G) = G_u$, the set $\Pi_p\pi_c$ will give us the set of all permutations in $\Pi(T)$ that give us G_u. Thus, $\Pi_u = \Pi_p\Pi_c$.

The sets of permutations Π_p and Π_c are found as follows: Let (G', Π) be a pair in our input set L in which the rows and columns of the matrix

A' in $G' = [I_k|A']$ are permutations of the rows and columns of the matrix A_u in $G_u = [I_k|A_u]$. Then when we run *UniqueOrderMatrix* on A', it will return A_u along with the set Π'' of all permutations $\pi_L\pi_R$ in which $\mathcal{RREF}(\pi_L\pi_R G') = G_u$. This immediately gives us Π_p since $\Pi_p = \Pi''\pi^{-1}$ where π is any permutation in Π''. For each π' in the set Π, if T is the k-combination in S_u in which $\pi' \in \Pi(T)$, the set of permutations Π'' can also be used to find a permutation $\pi_c \in \Pi(T)$ in which $\mathcal{RREF}(\pi_c G) = G_u$. That is, if π is any permutation in Π'' then $\pi_c = \pi\pi'$ gives us a permutation in $\Pi(T)$ in which $\mathcal{RREF}(\pi_c G) = G_u$. Finding such a permutation π_c, for each π' in L that is an element of a set $\Pi(T)$ in which $G_u \in \mathcal{G}(T)$, gives us our set Π_C.

We summarize the permutation algorithm for our simplified unique representative algorithm in Algorithm 3.2. Together, Algorithms 3.1 and 3.2 give us our simplified unique representative algorithm.

Algorithm 3.2 : *PermutationAlgorithm*$(L, G_u, \mathcal{AUT}(C_u))$

Input: The set L output by our combination algorithm. L is a set of pairs (G', Π'), where $G' = [I_k|A']$ generates a code equivalent to the code generated by G (the matrix input into the combination algorithm) and Π is a set of permutations π of the columns of G in which $\mathcal{RREF}(\pi G) = G'$.
Output: The matrix G_u and the group of permutations $\mathcal{AUT}(C_u)$. The matrix G_u generates the unique representative C_u of the equivalence class of C, the code generated by G. The group $\mathcal{AUT}(C_u)$ is the automorphism group of C_u.

begin
  Initialize G_u to *nil*.
  for each pair $([I_k|A'], \Pi)$ in L **do**
    UniqueOrderMatrix(A', A'', Π'').
    Set $G'' = [I_k|A'']$.
    if $G_u =$ *nil* **or** G'' is greater than G_u **then**
      Set $G_u = G''$.
      Pick any permutation $\pi \in \Pi''$.
      Set $\Pi_p = \Pi''\pi^{-1}$.
      Set $\Pi_c = \pi\Pi$.
    else if G'' is equal to G_u **then**
      Pick any permutation $\pi \in \Pi''$.
      Set $\Pi_c = \Pi_c \cup \pi\Pi$.
    end if-then-else
  end for each
  Set $\Pi_u = \Pi_p\Pi_c$.

Pick any permutation $\pi \in \Pi_u$.
Set $\mathcal{AUT}(C_u) = \Pi_u \pi^{-1}$.
end

Note that due to the size of the automorphism groups of some of our codes, the sets of permutations found in our unique representative algorithm can get quit large. In some cases, too large to store the entire set. Time and space limitations prevent us from giving details on how we get around this problem, so we will only briefly discuss how we get around it in our simplified combination algorithm. Whenever the number of permutations produced and stored in the combination algorithm reaches a certain limit, we break from the combination algorithm and run the permutation algorithm on the partial list L thus far produced by the combination algorithm. The permutation algorithm will find $G_u = [I_k|A_u]$, the largest matrix with minimal row weight distributions in the sets $\mathcal{G}(T)$ that our combination algorithm has thus far considered, and $\mathcal{AUT}(C_u)$, the automorphisms of C_u that are derived from the sets $\Pi(T)$ considered thus far by our combination algorithm. The permutation algorithm will now also return a set L' consisting of a single pair $([I_k|A'], \{\pi\})$, from our partial list L, in which the rows and columns of A' are permutations of the rows and columns of A_u. We then return to the point where we broke from the combination algorithm, replace the set L with L', and then continue on from where we left off. Note that we do not have problems with the sizes of our groups produced by our algorithm since we only store generators for the group. Further, our method for storing groups requires us to store at most $(2k(2k-1))/2$ generators.

3.2 Improvements

As mentioned, our simplified unique representative algorithm becomes impractical for relatively small codes. Reasons for this include the facts that our combination algorithm needs to produce $\binom{2k}{k}$ permutations, one for each set $\Pi(T)$, and the function *UniqueOrderMatrix* in the permutation algorithm may need to consider, in the worst case, $k!$ row permutations. By examining different structures of codewords that may occur in a self-dual code, we have managed to improve our unique representative algorithm to the point that we have been able to use it to find unique representatives for each of our codes of length up to and including 34. We will conclude this section by discussing the improvements we have made to our algorithm.

$$\left[\begin{array}{cccc|ccccc} 1 & 0 & 0 & 0^* & 1 & 1 & 1 & 0 & 0^* \\ 0 & 1 & 0 & 0^* & 1 & 1 & 0 & 1 & 0^* \\ 0 & 0 & 1 & 0^* & 1 & 0 & 1 & 1 & 0^* \end{array}\right] \qquad \left[\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & 0^* & 1 & 1 & 1 & 0 & 0^* \\ 0 & 1 & 0 & 0 & 0^* & 1 & 1 & 0 & 1 & 0^* \\ 0 & 0 & 1 & 0 & 0^* & 1 & 0 & 1 & 1 & 0^* \\ 0 & 0 & 0 & 1 & 0^* & 0 & 1 & 1 & 1 & 0^* \end{array}\right]$$

the e_3-block the e_4-block

$$\left.\begin{array}{cccc|cccccc} \overbrace{\begin{matrix}1 & 0 & \cdots & 0\end{matrix}}^{i} & 0^* & 1 & 1 & \overbrace{\begin{matrix}1 & 0 & \cdots & 0\end{matrix}}^{i} & 0^* \\ \begin{matrix}0 & 1 & \cdots & 0\end{matrix} & 0^* & 1 & 1 & \begin{matrix}0 & 1 & \cdots & 0\end{matrix} & 0^* \\ \begin{matrix}\vdots & \vdots & & \vdots\end{matrix} & \vdots & \vdots & \vdots & \begin{matrix}\vdots & \vdots & & \vdots\end{matrix} & \vdots \\ \begin{matrix}0 & 0 & \cdots & 1\end{matrix} & 0^* & 1 & 1 & \begin{matrix}0 & 0 & \cdots & 1\end{matrix} & 0^* \end{array}\right\} i$$

the d_i-blocks, $i \geq 2$

Figure 3: The different weight 4 row blocks, up to row and column rearrangement, that may occur in a generator matrix $[I_k|A]$ for a $(2k, k, 4)$ self-dual code. The vertical bar separates the columns that occur in I_k and A. The notation 0^* means enough zeroes to fill out the rows.

3.2.1 The Distance 4 Codes

We will begin by discussing the improvements that were realized by investigating the different ways weight 4 rows can interact in a generator matrix for a self-dual code with distance 4.

Let C be a $(2k, k, 4)$ self-dual code. It can be shown that the set $\mathcal{G}(C)$ must contain matrices with weight 4 rows (see [1]). So, let G be a matrix in $\mathcal{G}(C)$ that contains at least one weight 4 row. We group the weight 4 words in G into one or more disjoint sets we refer to as *weight 4 row blocks.* A weight 4 row block is a set W of rows in G with the properties that: 1) for any row r_1 in W there exists at least one row r_2 in W such that r_1 and r_2 intersect, 2) if r_1 is a weight 4 row in G that is not in W then r_1 does not intersect any of the rows in W, and 3) W cannot be partitioned into two non-empty subsets both of which posses the first two properties. The different weight 4 row blocks, up to row and column rearrangement, that may occur in a generator matrix of the form $[I_k|A]$ are the e_3-block, the e_4-block, and the d_i-blocks, $i \geq 1$. These blocks are listed in Figure 3.

The unique matrix G_u returned by our unique representative algorithm for the $(2k, k, 4)$ self-dual codes contains what we refer to as a *maximal com-*

bination of weight 4 row blocks. Further, the row and column arrangement of the weight 4 row blocks in G_u will be in what we refer to as *standard order*. In order to define these concepts, we must first establish a precedence relation for the different weight 4 row blocks. ¿From highest to lowest, this precedence is $e_4, e_3, \ldots, d_i, \ldots, d_2, d_1$. We consider a generator matrix G in $\mathcal{G}(C)$ as having a maximal combination of weight 4 row blocks if for every $G' \in \mathcal{G}(C)$, either G and G' have the same number of each of the different weight 4 row blocks, or, G has more B-blocks than G', where B is the highest precedenced weight 4 row block in which G and G' contain a different amount. We consider the weight 4 row blocks in $G' = [I_k|A']$ to be in standard order if they occur in the top right-hand corners of I_k and A' and are ordered in descending order of precedence. More formally, we have the following: Let m denote the number of weight 4 rows in G. Then the weight 4 row blocks in G are in standard order if: 1) the m weight 4 rows in G occur in the first m rows of G, 2) the rows/non-zero columns of each weight 4 row block occur before the rows/non-zero columns of all lower precedenced blocks, and 3) the $m \times 2k$ submatrix in the first m rows of G is greater than or equal to the submatrix in the first m rows of any matrix in $\mathcal{G}(C)$ that has a maximal combination of weight 4 row blocks and possesses the first two properties.

Of course, $\mathcal{G}(C)$ may contain many matrices that have a maximal combination of weight 4 row blocks that are in standard order. Let $\mathcal{G}_{max}(C)$ denote the set of all matrices in $\mathcal{G}(C)$ that have a maximal combination of weight 4 row blocks that are in standard order. The matrix G_u returned by our unique representative algorithm for the distance 4 codes is the unique matrix in $\mathcal{G}_{max}(C)$ in which: 1) the row weight distribution of G_u is less than or equal to the row weight distribution of all other matrices in $\mathcal{G}_{max}(C)$, and 2) G_u is greater than all other matrices in $\mathcal{G}_{max}(C)$ that have a minimal row weight distribution.

Time and space limitations prevent us from giving the details of how we use the maximal combinations of weight 4 row blocks to reduce the work our unique representative algorithm performs in finding G_u. So, we will only give brief outlines of how our revised combination and permutation algorithms work. More information can be found in [1].

Let $T_{max}(C)$ denote the set of all k-combinations in which the matrices in $\mathcal{G}(C)$ have a maximal combination of weight 4 row blocks. Our revised combination algorithm only produces permutations from the sets $\Pi(T)$ in which $T \in T_{max}(C)$ (along with permutations from some of the unwanted sets $\Pi(T)$ in which the matrices in $\mathcal{G}(T)$ do not have the form $[I_k|A']$). It accomplishes this by first producing a set $L_{max}(C)$ of pairs (G_0, Π_0), where G_0 has a maximal combination of weight 4 row blocks in standard order and

Π_0 is a set of permutations π in which $\mathcal{RREF}(\pi C) = G_0$. The set $L_{max}(C)$ has the property that for any $T \in T_{max}(C)$, there exists one and only one pair $(G_0, \Pi_0) \in L_{max}(C)$, and permutation $\pi \in \Pi_0$, for which there exists a permutation π' such that $\pi'\pi \in \Pi(T)$, where π' is a permutation that only involves the columns of G_0 that do *not* have a value of one in any of the weight 4 rows in G_0. For each pair $(G_0, \Pi_0) \in L_{max}(C)$, let $T(G_0, \Pi_0)$ denote the set of all $T \in T_{max}(C)$ for which such a permutation $\pi'\pi$ exists. The combination algorithm uses the pair (G_0, Π_0) to find permutations for the subset of k-combinations T in $T_{max}(C)$ in which $T \in T(G_0, \Pi_0)$. More specifically, for each $(G_0, \Pi_0) \in L_{max}(C)$, the algorithm produces a set L'' of pairs (G'', Π'') with the property that for any $T \in T(G_0, \Pi_0)$, in which $\mathcal{G}(T)$ may contain G_u, there exists a $(G'', \Pi'') \in L''$ and $\pi'' \in \Pi''$ such that $\pi'' \in \Pi(T)$ and $\mathcal{RREF}(\pi'' C) = G''$. It accomplishes this by first running our simplified combination algorithm on the matrix G_0, but instead of considering all k-combinations in $S_{2k,k}$, it only considers the k-combinations T in which $\Pi(T)$ contains permutations that do not move the columns in G_0 that have a value of one in at least one of the weight 4 rows in G_0 (i.e. T is considered if, in $G_0 = [I_k|A_0]$, the columns that are moved by the permutations in $\Pi(T)$ from I_k to A_0, and vice-versa, do not have a value of one in any of the weight 4 rows in G_0). The output of this run is a set L' of pairs (G'', Π'), in which the row weight distribution of G'' is minimal (over all matrices produced during the run). The permutations π' in the set Π' are such that $\mathcal{RREF}(\pi' G_0) = G''$. For each pair (G'', Π') in L', the algorithm then replaces the set of permutations Π' with a set of permutations Π''. The set Π'' consists of all permutations $\pi\pi'$ where $\pi \in \Pi_0$ and $\pi' \in \Pi'$. This gives us our set L'', of pairs (G'', Π''), for the pair (G_0, Π_0). Combining each of the sets L'' produced during the algorithm gives us our set L of pairs (G', Π) output by our combination algorithm.

The savings we realize with our revised combination algorithm are due to the fact that we do not produce permutations π from the sets $\Pi(T)$ in which the matrices in $\mathcal{G}(T)$ have the form $[I_k|A']$, but do not have a maximal combination of weight 4 row blocks. The algorithm also avoids producing permutations from some of the sets $\Pi(T)$ in which the matrices in $\mathcal{G}(T)$ do not have the form $[I_k|A']$.

Let L denote the set of pairs (G', Π) produced by our revised combination algorithm. We can use the fact that each G' in L has a maximal combination of weight 4 row blocks in standard order to improve our permutation algorithm. For each $G' = [I_k|A']$ in L, our revised permutation algorithm finds the largest matrix $G'' = [I_k|A'']$, where the rows and columns of A'' are permutations of the rows and columns of A', such that G'' has a maximal combination of weight 4 row blocks in standard order. Note that, as with our simplified permutation algorithm, the columns of the matrix A''

$$\left.\begin{array}{ccccc|cccccccccccccccc}
\multicolumn{4}{c}{\overbrace{\hphantom{1\;0\;\cdots\;0}}^{i}} & & \multicolumn{4}{c}{\overbrace{\hphantom{1\;1\;\cdots\;1}}^{w-i-2}} & \multicolumn{4}{c}{\overbrace{\hphantom{0\;1\;\cdots\;1}}^{i}} & \multicolumn{7}{c}{\overbrace{\hphantom{1\;1\;0\;0\;\cdots\;0\;0}}^{2i}} & \\
1 & 0 & \cdots & 0 & 0^* & 1 & 1 & \cdots & 1 & 0 & 1 & \cdots & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0^* \\
0 & 1 & \cdots & 0 & 0^* & 1 & 1 & \cdots & 1 & 1 & 0 & \cdots & 1 & 0 & 0 & 1 & 1 & \cdots & 0 & 0 & 0^* \\
\vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0^* & 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 0^*
\end{array}\right\} i$$

Figure 4: The $f_{w,i}$-blocks, $w \geq 6$, $i \geq 1$.

must be in descending order. Again, this means we only need to consider the different permutations of the rows of A''. Now, let m denote the number of weight 4 rows in G'. Since the weight 4 row blocks in G' are in standard order, the m weight 4 rows in G' occur in its first m rows. Therefore, any permutation of the rows of A' that results in A'' cannot move any of the first m rows of A' to the last $k-m$ rows of A', and vice-versa. This means the only row permutations our *UniqueOrderMatrix* function needs to consider are those of the form $\pi_1\pi_2$, where π_1 only moves the first m rows of A' and π_2 only moves the last $k-m$ rows of A'. Thus, in the worst case, the call to the function *UniqueOrderMatrix* in our revised permutation algorithm now needs to consider at most $m!(k-m)!$ row permutations, instead of $k!$.

We can realize even more savings in both our revised permutation and combination algorithms by considering permutations of a weight 4 row block W that are automorphisms in any code C that contain W. How we can use such permutations to reduce the amount of work we do is discussed in [1].

3.2.2 Code With Distance Greater Than Four

Our improvements based on the weight 4 words in a code C offer relatively litter help if C contains only a few weight 4 words and no help if C has distance greater than 4. For these codes, we consider some of the different ways the weight w rows can interact in a generator matrix of the form $[I_k|A]$, for $w \geq 6$. In particular, we consider blocks of weight w rows that from what we call $f_{w,i}$-blocks, for $i \geq 1$, which we give in Figure 4.

For the codes C with distance greater than 4, the unique matrix G_u returned by our unique representative algorithm contains what we refer to as a ***maximal combination of non-intersecting $f_{w,i}$-blocks***. Further, the $f_{w,i}$-blocks in G_u will be in *standard order*. In order to define this concept, we first need to define the related concept of a ***combination of non-intersecting $f_{w,i}$-blocks***. We also need to establish a precedence among the $f_{w,i}$-blocks.

Let $G = [I_k|A]$ be a generator matrix for a $(2k, k, d \geq 6)$ self-dual code. We consider a set S of rows in G as forming a combination of non-intersecting $f_{w,i}$-blocks if S can be partitioned into $m \geq 1$ disjoint sets $S_1, S_2, \ldots, S_m$ in which: 1) the rows in S_x form an f_{w_x,i_x}-block, for $0 \leq x \leq m$, and 2) the sets T_x and T_y are disjoint, where T_x and T_y are the sets of non-zero columns in S_x and S_y, respectively, for $0 \leq x < y \leq m$. The precedence we establish among the $f_{w,i}$-blocks is based on the weight of the rows in the block and the number of rows in the block. We consider an f_{w_1,i_1}-block as having higher precedence that an f_{w_2,i_2}-block if either $w_1 < w_2$, or $w_1 = w_2$ and $i_1 > i_2$. We consider a matrix $G \in \mathcal{G}(C)$ as having a maximal combination of non-intersecting $f_{w,i}$-blocks if for every $G' \in \mathcal{G}(C)$, either G and G' have the same number of each of the different $f_{w,i}$-blocks, or, G has more B-blocks than G', where B is the highest precedenced $f_{w,i}$-block in which G and G' contain a different amount. The maximal combination of non-intersecting $f_{w,i}$-blocks in G is in standard order if the $f_{w,i}$-blocks occur at the start of G, the rows of each block occur in consecutive rows, the rows and columns of the blocks are order as in Figure 4, and each block occurs before all other blocks with lower precedence.

The revisions we make to our combination and permutation algorithms for the codes with distance greater than 4 mirror the revisions we made for the distance 4 code, with one slight change. The change we make is necessitated by the fact that a generator matrix may contain more than one maximal combination of non-intersecting $f_{w,i}$-blocks. That is, it is possible for a generator matrix G to contain two sets of rows S_1 and S_2, where $S_1 \neq S_2$, in which both S_1 and S_2 form maximal combinations of non-intersecting $f_{w,i}$-blocks. To get around this problem, our combination algorithm logically partitions each set $\Pi(T)$ into $t \geq 1$ sets $\Pi_1(T), \Pi_2(T), \ldots, \Pi_{t-1}(T), \Pi_{rem}(T)$. The set $\Pi_{rem}(T)$ consists of all permutations $\pi \in \Pi(T)$ in which $\mathcal{RREF}(\pi G)$ does not contain a maximal combination of non-intersecting $f_{w,i}$-blocks that are in standard order. Each set $\Pi_i(T)$, where $1 \leq i \leq t-1$, consists of all permutations $\pi \in \Pi(T)$ in which the matrices in $\mathcal{G}(T)$ contain corresponding maximal combinations of non-intersecting $f_{w,i}$-blocks that are in standard order. That is, if $\pi \in \Pi_i(T)$ and S is the set of columns in $G' = \mathcal{RREF}(\pi G)$ that are non-zero in the rows of the maximal combination of non-intersecting $f_{w,i}$-blocks in G', then $\pi_L \pi_R \pi$ is an elment of $\Pi_i(T)$ if and only if π_L and π_R do not move any of the columns in S out of S, where $\pi_L \in \Pi_L$ and $\pi_R \in \Pi_R$. Now, our combination algorithm will produce one permutation π from each of the sets $\Pi_i(T)$, $1 \leq i \leq t$. So, for each set T in which $\mathcal{T}(G)$ contains a matrix $G' = [I_k|A']$ that has a maximal combination of non-intersecting $f_{w,i}$-blocks, the combination algorithm is essentially finding one matrix $G'' = [I_k|A'']$, where the rows and columns of A'' are permutations of the rows and columns of A', for each of the different sets of rows in G' that form a maximal combination of non-intersecting $f_{w,i}$-blocks. Thus, in the permutation algorithm, given a generator matrix $G' = [I_k|A']$ whose first $m \geq 1$ rows contain a maximal combination of non-intersecting $f_{w,i}$-blocks in standard order, the only row permutations of A' that the function *UniqueOrderMatrix* will consider are those that do not move any the first m rows of A' to the last $k - m$ rows of A' and vice-versa. In other words, in the permutation algorithm, we do not have to consider the fact that G' may contain more than one maximal combination of non-intersecting $f_{w,i}$-blocks.

We also make use of the $f_{w,i}$-blocks in the codes with distance four that contain only a few weight 4 words. For these codes, we simply combine the concepts of maximal combinations of weight 4 row blocks and maximal combinations of non-intersecting $f_{w,i}$-blocks. We do so by only considering, for each G that contains a maximal combination of weight 4 row blocks, the $f_{w,i}$-blocks in G whose non-zero columns occur in the columns of G that have a value of zero in every weight 4 row in G.

3.3 Summary

We have used our unique representative algorithm to eliminate the equivalent codes from our lists of self-dual codes with length 34. We have also used our algorithm to find generators for the automorphism group of each of our codes with length less than or equal to 34. Our list of length 34 codes with distance greater than 2 contains 20,852 codes, which means there are 20,852 equivalence classes of $(34, 17)$ self-dual codes with distance greater than or equal to 4. Our algorithm took approximately one month to produce the unique representative and automorphism group for each of the codes produced in the first stage of our enumeration (including the duplicates). The algorithm took over 30 minutes for thirteen of the codes (excluding duplicates), with the worst case taking 104 minutes.

4 Results

In Table 1, we give the number of equivalence classes of $(2k, k, d)$ binary self-dual codes for $2k \leq 34$ and $d = 2, 4, 6, 8$. (There are no codes within this length range whose distance is greater than 8.) The results for $2k = 34$ are new. Among the equivalence classes for the $(34, 17, 4)$ self-dual codes, 66 of the classes consist of composed codes, and 9936 of the classes consist of codes that do not contain weight 4 words that intersect.

$2k$	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
$d=2$	1	1	1	1	2	2	3	4	7	9	16	25	55	103	261	731	3295
$d=4$	0	0	0	1	0	1	1	3	2	7	8	28	47	155	457	2482	19914
$d=6$	0	0	0	0	0	0	0	0	0	0	1	1	1	3	13	74	938
$d=8$	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	8	0

Table 1: The number of equivalence classes of $(2k, k, d)$ self-dual codes.

In Table 2, we list all the different weight distributions for the length 34 codes with distance greater than 2. Included with each distribution is the number of equivalence classes whose codes have the distribution, and, the minimum and maximum automorphism group sizes for the codes with the given distribution. Note that for each distribution, we only list the number of weight $2i$ words, for $2i = 4, 6, \dots, 16$. For all codes, the number of weight 0 words is 1, the number of weight 2 words is 0, and the number of weight $2i$ words, $2i \geq 18$ is equal to the number of weight $34 - 2i$ words.

We have found that 159 of the inequivalent (34, 17) self-dual codes have an automorphism group that is the trivial group. All of these codes have distance 6. The length 34 codes with distance 6, along with their automorphism groups, are available on the world wide web at:

www.cs.umanitoba.ca/~umbilou1/

Further information on the length 34 codes can be obtained from the author of this paper.

Weight Distributions							Number	Automorphism Group Sizes	
4	6	8	10	12	14	16	of Codes	Minimum	Maximum
0	2	287	2081	8306	21100	33759	1	64	64
0	6	283	2061	8326	21140	33719	22	2	1536
0	6	411	1165	10886	17556	35511	11	8	11520
0	10	279	2041	8346	21180	33679	82	1	5760
0	14	275	2021	8366	21220	33639	183	1	768
0	18	271	2001	8386	21260	33599	257	1	1152
0	22	267	1981	8406	21300	33559	214	1	768
0	26	263	1961	8426	21340	33519	111	1	128
0	30	259	1941	8446	21380	33479	41	2	384
0	34	255	1921	8466	21420	33439	16	12	34560
1	3	297	2091	8257	21051	33835	4	96	2304
1	7	293	2071	8277	21091	33795	27	8	12288
1	7	421	1175	10837	17507	35587	21	12	258048
1	11	289	2051	8297	21131	33755	114	4	1536
1	15	285	2031	8317	21171	33715	328	4	5760
1	19	281	2011	8337	21211	33675	448	4	1152
1	23	277	1991	8357	21251	33635	446	4	6144
1	27	273	1971	8377	21291	33595	272	4	384
1	31	269	1951	8397	21331	33555	139	4	18432
1	35	265	1931	8417	21371	33515	33	4	128
1	39	261	1911	8437	21411	33475	15	32	6144
2	4	307	2101	8208	21002	33911	6	192	2048
2	8	303	2081	8228	21042	33871	38	16	6144
2	8	431	1185	10788	17458	35663	24	32	12288
2	12	299	2061	8248	21082	33831	150	16	61440
2	16	295	2041	8268	21122	33791	350	16	9216
2	20	291	2021	8288	21162	33751	545	16	6144
2	24	287	2001	8308	21202	33711	529	16	3072
2	28	283	1981	8328	21242	33671	441	16	18432
2	32	279	1961	8348	21282	33631	223	16	2048
2	36	275	1941	8368	21322	33591	116	16	1024
2	40	271	1921	8388	21362	33551	30	32	1536
2	44	267	1901	8408	21402	33511	9	384	4096
3	1	321	2131	8139	20913	34027	1	18432	18432
3	5	317	2111	8159	20953	33987	5	128	1024
3	9	313	2091	8179	20993	33947	44	64	73728
3	9	441	1195	10739	17409	35739	35	128	73728
3	13	309	2071	8199	21033	33907	132	64	13824

Table 2: Weight Distributions of the $(34, 17, d \geq 4)$ Self-Dual Codes.

Weight Distributions							Number	Automorphism Group Sizes	
4	6	8	10	12	14	16	of Codes	Minimum	Maximum
3	17	305	2051	8219	21073	33867	332	48	245760
3	21	301	2031	8239	21113	33827	458	24	27648
3	25	297	2011	8259	21153	33787	553	24	49152
3	29	293	1991	8279	21193	33747	410	24	4608
3	33	289	1971	8299	21233	33707	339	48	147456
3	37	285	1951	8319	21273	33667	144	48	9216
3	41	281	1931	8339	21313	33627	85	64	24576
3	45	277	1911	8359	21353	33587	16	128	1536
3	49	273	1891	8379	21393	33547	17	1024	1032192
4	2	331	2141	8090	20864	34103	2	4096	18432
4	6	327	2121	8110	20904	34063	11	512	110592
4	10	323	2101	8130	20944	34023	43	256	552960
4	10	451	1205	10690	17360	35815	39	256	552960
4	14	319	2081	8150	20984	33983	132	192	3870720
4	18	315	2061	8170	21024	33943	262	96	110592
4	22	311	2041	8190	21064	33903	406	96	552960
4	26	307	2021	8210	21104	33863	417	96	55296
4	30	303	2001	8230	21144	33823	435	96	184320
4	34	299	1981	8250	21184	33783	270	96	18432
4	38	295	1961	8270	21224	33743	216	96	18432
4	42	291	1941	8290	21264	33703	90	96	9216
4	46	287	1921	8310	21304	33663	51	256	16384
4	50	283	1901	8330	21344	33623	12	512	4096
4	54	279	1881	8350	21384	33583	7	4096	24576
5	3	341	2151	8041	20815	34179	2	4096	73728
5	7	337	2131	8061	20855	34139	10	1024	18432
5	11	333	2111	8081	20895	34099	43	768	98304
5	11	461	1215	10641	17311	35891	35	768	221184
5	15	329	2091	8101	20935	34059	92	384	12288
5	19	325	2071	8121	20975	34019	204	384	110592
5	23	321	2051	8141	21015	33979	282	384	24576
5	27	317	2031	8161	21055	33939	356	384	2949120
5	31	313	2011	8181	21095	33899	301	384	24576
5	35	309	1991	8201	21135	33859	297	384	73728
5	39	305	1971	8221	21175	33819	154	384	8192
5	43	301	1951	8241	21215	33779	155	384	196608
5	47	297	1931	8261	21255	33739	43	384	18432
5	51	293	1911	8281	21295	33699	40	1024	98304

Table 2 continued: Weight Distributions of the $(34, 17, d \geq 4)$ Self-Dual Codes.

Weight Distributions							Number	Automorphism Group Sizes	
4	6	8	10	12	14	16	of Codes	Minimum	Maximum
5	55	289	1891	8301	21335	33659	7	4096	18432
5	59	285	1871	8321	21375	33619	8	8192	196608
6	0	355	2181	7972	20726	34295	1	1105920	1105920
6	4	351	2161	7992	20766	34255	3	12288	331776
6	8	347	2141	8012	20806	34215	11	3072	49152
6	12	343	2121	8032	20846	34175	31	1536	131072
6	12	471	1225	10592	17262	35967	41	1536	23224320
6	16	339	2101	8052	20886	34135	91	1536	510935040
6	20	335	2081	8072	20926	34095	127	1536	65536
6	24	331	2061	8092	20966	34055	250	1536	98304
6	28	327	2041	8112	21006	34015	226	1152	1105920
6	32	323	2021	8132	21046	33975	279	1152	73728
6	36	319	2001	8152	21086	33935	178	1152	110592
6	40	315	1981	8172	21126	33895	192	1152	41472
6	44	311	1961	8192	21166	33855	81	1536	36864
6	48	307	1941	8212	21206	33815	92	1536	2211840
6	52	303	1921	8232	21246	33775	22	1536	331776
6	56	299	1901	8252	21286	33735	20	3072	18432
6	60	295	1881	8272	21326	33695	3	6144	81920
6	64	291	1861	8292	21366	33655	4	32768	2211840
7	5	361	2171	7943	20717	34331	3	24576	1474560
7	9	357	2151	7963	20757	34291	10	6144	82944
7	13	353	2131	7983	20797	34251	34	6144	196608
7	13	481	1235	10543	17213	36043	41	6144	23224320
7	17	349	2111	8003	20837	34211	58	4608	73728
7	21	345	2091	8023	20877	34171	133	2304	1179648
7	25	341	2071	8043	20917	34131	157	2304	73728
7	29	337	2051	8063	20957	34091	211	1536	294912
7	33	333	2031	8083	20997	34051	187	1536	193536
7	37	329	2011	8103	21037	34011	205	1536	30965760
7	41	325	1991	8123	21077	33971	115	1536	49152
7	45	321	1971	8143	21117	33931	141	2304	786432
7	49	317	1951	8163	21157	33891	48	2304	49152
7	53	313	1931	8183	21197	33851	69	4608	884736
7	57	309	1911	8203	21237	33811	14	6144	82944
7	61	305	1891	8223	21277	33771	17	12288	442368
7	65	301	1871	8243	21317	33731	1	24576	24576
7	69	297	1851	8263	21357	33691	8	131072	2949120

Table 2 continued: Weight Distributions of the $(34, 17, d \geq 4)$ Self-Dual Codes.

Weight Distributions							Number	Automorphism Group Sizes	
4	6	8	10	12	14	16	of Codes	Minimum	Maximum
9	71	317	1871	8165	21259	33843	3	147456	294912
9	75	313	1851	8185	21299	33803	1	884736	884736
9	79	309	1831	8205	21339	33763	5	884736	17694720
10	0	399	2241	7756	20490	34639	2	162570240	232243200
10	4	395	2221	7776	20530	34599	2	294912	884736
10	8	391	2201	7796	20570	34559	4	294912	1179648
10	12	387	2181	7816	20610	34519	13	147456	884736
10	16	383	2161	7836	20650	34479	13	98304	1179648
10	16	511	1265	10396	17066	36271	29	55296	19585843200
10	20	379	2141	7856	20690	34439	33	49152	589824
10	24	375	2121	7876	20730	34399	44	49152	21233664
10	28	371	2101	7896	20770	34359	69	36864	663552
10	32	367	2081	7916	20810	34319	46	18432	1179648
10	36	363	2061	7936	20850	34279	108	18432	1769472
10	40	359	2041	7956	20890	34239	46	36864	1769472
10	44	355	2021	7976	20930	34199	73	18432	589824
10	48	351	2001	7996	20970	34159	49	10752	3538944
10	52	347	1981	8016	21010	34119	54	36864	1179648
10	56	343	1961	8036	21050	34079	10	49152	2359296
10	60	339	1941	8056	21090	34039	37	49152	589824
10	64	335	1921	8076	21130	33999	4	98304	11059200
10	68	331	1901	8096	21170	33959	12	147456	393216
10	72	327	1881	8116	21210	33919	1	589824	589824
10	76	323	1861	8136	21250	33879	4	294912	589824
10	84	315	1821	8176	21330	33799	2	3538944	162570240
11	9	401	2211	7747	20521	34635	2	442368	4718592
11	13	397	2191	7767	20561	34595	6	221184	2654208
11	17	393	2171	7787	20601	34555	18	196608	4718592
11	17	521	1275	10347	17017	36347	25	147456	4718592
11	21	389	2151	7807	20641	34515	15	73728	884736
11	25	385	2131	7827	20681	34475	41	73728	141557760
11	29	381	2111	7847	20721	34435	33	73728	884736
11	33	377	2091	7867	20761	34395	72	73728	4718592
11	37	373	2071	7887	20801	34355	62	43008	884736
11	41	369	2051	7907	20841	34315	70	73728	2359296
11	45	365	2031	7927	20881	34275	40	73728	884736
11	49	361	2011	7947	20921	34235	75	64512	5308416
11	53	357	1991	7967	20961	34195	21	73728	442368

Table 2 continued: Weight Distributions of the $(34, 17, d \geq 4)$ Self-Dual Codes.

Weight Distributions							Number of Codes	Automorphism Group Sizes	
4	6	8	10	12	14	16		Minimum	Maximum
13	43	389	2071	7809	20743	34467	30	245760	5308416
13	47	385	2051	7829	20783	34427	15	122880	2654208
13	51	381	2031	7849	20823	34387	54	258048	56623104
13	55	377	2011	7869	20863	34347	13	245760	7962624
13	59	373	1991	7889	20903	34307	19	245760	5308416
13	63	369	1971	7909	20943	34267	13	258048	5308416
13	67	365	1951	7929	20983	34227	27	491520	28311552
13	71	361	1931	7949	21023	34187	3	1327104	2654208
13	75	357	1911	7969	21063	34147	9	884736	5308416
13	83	349	1871	8009	21143	34067	3	2359296	18874368
13	87	345	1851	8029	21183	34027	2	5308416	9289728
13	99	333	1791	8089	21303	33907	4	28311552	169869312
14	0	443	2301	7540	20254	34983	1	10616832	10616832
14	4	439	2281	7560	20294	34943	2	21233664	24772608
14	8	435	2261	7580	20334	34903	1	5308416	5308416
14	12	431	2241	7600	20374	34863	1	10616832	10616832
14	16	427	2221	7620	20414	34823	6	1769472	10616832
14	20	423	2201	7640	20454	34783	9	1769472	10616832
14	20	551	1305	10200	16870	36575	17	1769472	10616832
14	24	419	2181	7660	20494	34743	10	983040	10616832
14	28	415	2161	7680	20534	34703	15	983040	4128768
14	32	411	2141	7700	20574	34663	24	1474560	31850496
14	36	407	2121	7720	20614	34623	22	589824	10616832
14	40	403	2101	7740	20654	34583	36	368640	108380160
14	44	399	2081	7760	20694	34543	16	368640	10616832
14	48	395	2061	7780	20734	34503	24	983040	21233664
14	52	391	2041	7800	20774	34463	22	368640	15095808
14	56	387	2021	7820	20814	34423	21	983040	10616832
14	60	383	2001	7840	20854	34383	10	983040	3538944
14	64	379	1981	7860	20894	34343	16	983040	21233664
14	68	375	1961	7880	20934	34303	5	1769472	21233664
14	72	371	1941	7900	20974	34263	10	983040	21233664
14	76	367	1921	7920	21014	34223	2	1769472	24772608
14	80	363	1901	7940	21054	34183	5	1769472	10616832
14	84	359	1881	7960	21094	34143	1	3538944	3538944
14	88	355	1861	7980	21134	34103	1	7077888	7077888
14	104	339	1781	8060	21294	33943	1	42467328	42467328
15	5	449	2291	7511	20245	35019	1	47185920	47185920

Table 2 continued: Weight Distributions of the $(34, 17, d \geq 4)$ Self-Dual Codes.

Weight Distributions							Number of Codes	Automorphism Group Sizes	
4	6	8	10	12	14	16		Minimum	Maximum
15	9	445	2271	7531	20285	34979	1	15925248	15925248
15	13	441	2251	7551	20325	34939	2	14155776	15925248
15	21	433	2211	7591	20405	34859	7	5308416	63700992
15	21	561	1315	10151	16821	36651	18	1548288	928972800
15	25	429	2191	7611	20445	34819	5	5308416	15925248
15	29	425	2171	7631	20485	34779	16	1548288	14155776
15	33	421	2151	7651	20525	34739	5	737280	95551488
15	37	417	2131	7671	20565	34699	20	2654208	28311552
15	41	413	2111	7691	20605	34659	14	737280	5308416
15	45	409	2091	7711	20645	34619	24	737280	88473600
15	49	405	2071	7731	20685	34579	5	737280	31850496
15	53	401	2051	7751	20725	34539	38	737280	33030144
15	57	397	2031	7771	20765	34499	8	737280	31850496
15	61	393	2011	7791	20805	34459	23	2359296	31850496
15	65	389	1991	7811	20845	34419	7	1474560	3096576
15	69	385	1971	7831	20885	34379	7	1474560	15728640
15	73	381	1951	7851	20925	34339	3	5308416	31850496
15	77	377	1931	7871	20965	34299	13	2949120	33030144
15	81	373	1911	7891	21005	34259	1	15925248	15925248
15	85	369	1891	7911	21045	34219	2	9437184	14155776
15	93	361	1851	7951	21125	34139	7	14155776	127401984
15	97	357	1831	7971	21165	34099	1	31850496	31850496
15	109	345	1771	8031	21285	33979	1	20437401600	20437401600
16	6	459	2301	7462	20196	35095	1	21233664	21233664
16	10	455	2281	7482	20236	35055	1	10616832	10616832
16	14	451	2261	7502	20276	35015	2	10616832	10616832
16	18	447	2241	7522	20316	34975	7	6193152	63700992
16	22	443	2221	7542	20356	34935	5	5898240	21233664
16	22	571	1325	10102	16772	36727	18	5898240	63700992
16	26	439	2201	7562	20396	34895	6	5898240	21233664
16	30	435	2181	7582	20436	34855	10	2949120	17694720
16	34	431	2161	7602	20476	34815	12	2949120	382205952
16	38	427	2141	7622	20516	34775	15	2949120	21233664
16	42	423	2121	7642	20556	34735	26	2064384	21233664
16	46	419	2101	7662	20596	34695	12	2949120	10616832
16	50	415	2081	7682	20636	34655	12	2949120	21233664
16	54	411	2061	7702	20676	34615	17	2322432	24772608
16	58	407	2041	7722	20716	34575	9	2949120	10616832

Table 2 continued: Weight Distributions of the $(34, 17, d \geq 4)$ Self-Dual Codes.

Weight Distributions							Number	Automorphism Group Sizes	
4	6	8	10	12	14	16	of Codes	Minimum	Maximum
20	26	611	1365	9906	16576	37031	9	35389440	637009920
20	30	479	2241	7366	20200	35199	1	106168320	106168320
20	34	475	2221	7386	20240	35159	7	35389440	106168320
20	38	471	2201	7406	20280	35119	1	106168320	106168320
20	42	467	2181	7426	20320	35079	4	35389440	212336640
20	46	463	2161	7446	20360	35039	11	41287680	173408256
20	50	459	2141	7466	20400	34999	4	8847360	106168320
20	54	455	2121	7486	20440	34959	3	106168320	212336640
20	58	451	2101	7506	20480	34919	7	8847360	148635648
20	62	447	2081	7526	20520	34879	2	106168320	212336640
20	66	443	2061	7546	20560	34839	1	35389440	35389440
20	70	439	2041	7566	20600	34799	9	74317824	212336640
20	74	435	2021	7586	20640	34759	3	17694720	212336640
20	78	431	2001	7606	20680	34719	1	106168320	106168320
20	82	427	1981	7626	20720	34679	2	35389440	74317824
20	86	423	1961	7646	20760	34639	1	106168320	106168320
20	94	415	1921	7686	20840	34559	3	74317824	212336640
21	27	493	2271	7297	20111	35315	3	169869312	1019215872
21	27	621	1375	9857	16527	37107	8	70778880	339738624
21	35	485	2231	7337	20191	35235	5	99090432	141557760
21	43	477	2191	7377	20271	35155	9	35389440	2038431744
21	47	473	2171	7397	20311	35115	1	30965760	30965760
21	51	469	2151	7417	20351	35075	4	35389440	141557760
21	59	461	2111	7457	20431	34995	11	33030144	396361728
21	67	453	2071	7497	20511	34915	4	35389440	141557760
21	71	449	2051	7517	20551	34875	2	65028096	260112384
21	75	445	2031	7537	20591	34835	6	283115520	2038431744
21	83	437	1991	7577	20671	34755	2	141557760	141557760
21	91	429	1951	7617	20751	34675	7	70778880	2038431744
21	107	413	1871	7697	20911	34515	2	283115520	660602880
21	139	381	1711	7857	21231	34195	1	6115295232	6115295232
22	0	531	2421	7108	19782	35671	2	1337720832	1911029760
22	12	519	2361	7168	19902	35551	1	445906944	445906944
22	16	515	2341	7188	19942	35511	1	1274019840	1274019840
22	20	511	2321	7208	19982	35471	1	212336640	212336640
22	24	507	2301	7228	20022	35431	1	1040449536	1040449536
22	28	503	2281	7248	20062	35391	1	212336640	212336640
22	28	631	1385	9808	16478	37183	8	106168320	2675441664

Table 2 continued: Weight Distributions of the $(34, 17, d \geq 4)$ Self-Dual Codes.

Weight Distributions							Number	Automorphism Group Sizes	
4	6	8	10	12	14	16	of Codes	Minimum	Maximum
25	15	549	2391	7021	19755	35779	3	1698693120	3397386240
25	31	533	2311	7101	19915	35619	2	1698693120	3397386240
25	31	661	1415	9661	16331	37411	8	424673280	7134511104
25	39	525	2271	7141	19995	35539	1	424673280	424673280
25	47	517	2231	7181	20075	35459	3	330301440	1698693120
25	55	509	2191	7221	20155	35379	1	424673280	424673280
25	63	501	2151	7261	20235	35299	11	198180864	3567255552
25	71	493	2111	7301	20315	35219	2	424673280	424673280
25	79	485	2071	7341	20395	35139	5	1698693120	3397386240
25	87	477	2031	7381	20475	35059	2	424673280	990904320
25	95	469	1991	7421	20555	34979	2	1321205760	1698693120
25	127	437	1831	7581	20875	34659	2	3397386240	5662310400
26	16	559	2401	6972	19706	35855	1	6242697216	6242697216
26	24	551	2361	7012	19786	35775	1	4246732800	4246732800
26	32	671	1425	9612	16282	37487	3	1486356480	2123366400
26	40	535	2281	7092	19946	35615	3	1486356480	2123366400
26	48	527	2241	7132	20026	35535	1	2123366400	2123366400
26	52	523	2221	7152	20066	35495	3	1734082560	12485394432
26	56	519	2201	7172	20106	35455	1	4246732800	4246732800
26	64	511	2161	7212	20186	35375	4	123863040	1486356480
26	88	487	2041	7332	20426	35135	4	1486356480	4246732800
27	25	561	2371	6963	19737	35851	2	2548039680	5096079360
27	33	553	2331	7003	19817	35771	1	5096079360	5096079360
27	33	681	1435	9563	16233	37563	8	1189085184	5662310400
27	49	537	2251	7083	19977	35611	3	2548039680	5662310400
27	57	529	2211	7123	20057	35531	3	1415577600	5096079360
27	65	521	2171	7163	20137	35451	4	594542592	2831155200
27	73	513	2131	7203	20217	35371	1	2548039680	2548039680
27	77	509	2111	7223	20257	35331	1	2601123840	2601123840
27	81	505	2091	7243	20297	35291	3	495452160	1698693120
27	97	489	2011	7323	20457	35131	2	2831155200	5096079360
27	113	473	1931	7403	20617	34971	1	16647192576	16647192576
27	121	465	1891	7443	20697	34891	2	2548039680	5096079360
28	30	567	2361	6934	19728	35887	2	7283146752	10404495360
28	34	691	1445	9514	16184	37639	3	1486356480	4954521600
28	42	555	2301	6994	19848	35767	1	2229534720	2229534720
28	50	547	2261	7034	19928	35687	1	743178240	743178240
28	66	531	2181	7114	20088	35527	3	1040449536	4954521600

Table 2 continued: Weight Distributions of the $(34, 17, d \geq 4)$ Self-Dual Codes.

Weight Distributions							Number of Codes	Automorphism Group Sizes	
4	6	8	10	12	14	16		Minimum	Maximum
28	98	499	2021	7274	20408	35207	1	1486356480	1486356480
28	114	483	1941	7354	20568	35047	1	4459069440	4459069440
29	35	573	2351	6905	19719	35923	1	4246732800	4246732800
29	35	701	1455	9465	16135	37715	3	2972712960	23781703680
29	43	565	2311	6945	19799	35843	1	2972712960	2972712960
29	59	549	2231	7025	19959	35683	1	4246732800	4246732800
29	67	541	2191	7065	20039	35603	5	1981808640	11890851840
29	75	533	2151	7105	20119	35523	1	4246732800	4246732800
29	99	509	2031	7225	20359	35283	1	33973862400	33973862400
29	107	501	1991	7265	20439	35203	1	4246732800	4246732800
29	115	493	1951	7305	20519	35123	5	11890851840	23781703680
30	20	599	2441	6776	19510	36159	1	14863564800	14863564800
30	36	583	2361	6856	19670	35999	1	4459069440	4459069440
30	36	711	1465	9416	16086	37791	5	9364045824	20808990720
30	44	575	2321	6896	19750	35919	2	10404495360	14863564800
30	84	535	2121	7096	20150	35519	1	4459069440	4459069440
31	21	609	2451	6727	19461	36235	1	17836277760	17836277760
31	37	721	1475	9367	16037	37867	3	5945425920	17836277760
31	45	585	2331	6847	19701	35995	2	17836277760	25480396800
31	53	577	2291	6887	19781	35915	3	5945425920	8493465600
31	61	569	2251	6927	19861	35835	1	50960793600	50960793600
31	69	561	2211	6967	19941	35755	2	4161798144	5945425920
31	85	545	2131	7047	20101	35595	1	3963617280	3963617280
31	93	537	2091	7087	20181	35515	4	17836277760	50960793600
31	101	529	2051	7127	20261	35435	1	5945425920	5945425920
31	105	525	2031	7147	20301	35395	2	54623600640	76473040896
32	70	571	2221	6918	19892	35831	1	5202247680	5202247680
32	94	547	2101	7038	20132	35591	1	15606743040	15606743040
33	39	613	2391	6709	19523	36227	1	23781703680	23781703680
33	39	741	1495	9269	15939	38019	3	11890851840	23781703680
33	71	581	2231	6869	19843	35907	2	3963617280	11890851840
33	87	565	2151	6949	20003	35747	3	19818086400	35672555520
33	95	557	2111	6989	20083	35667	1	29727129600	29727129600
33	135	517	1911	7189	20483	35267	1	71345111040	71345111040
34	0	663	2601	6460	19074	36703	1	208089907200	208089907200
34	40	751	1505	9220	15890	38095	1	31213486080	31213486080
34	72	591	2241	6820	19794	35983	2	31213486080	43698880512
35	25	649	2491	6531	19265	36539	1	169869312000	169869312000

Table 2 continued: Weight Distributions of the $(34, 17, d \geq 4)$ Self-Dual Codes.

Weight Distributions							Number of Codes	Automorphism Group Sizes	
4	6	8	10	12	14	16		Minimum	Maximum
35	41	761	1515	9171	15841	38171	2	23781703680	84934656000
35	57	617	2331	6691	19585	36219	1	23781703680	23781703680
35	73	601	2251	6771	19745	36059	1	35672555520	35672555520
35	89	585	2171	6851	19905	35899	1	169869312000	169869312000
35	97	577	2131	6891	19985	35819	1	249707888640	249707888640
35	105	569	2091	6931	20065	35739	1	11890851840	11890851840
35	121	553	2011	7011	20225	35579	1	41617981440	41617981440
36	42	771	1525	9122	15792	38247	3	72831467520	218494402560
37	43	653	2431	6513	19327	36531	1	142690222080	142690222080
37	43	781	1535	9073	15743	38323	2	118908518400	178362777600
37	75	621	2271	6673	19647	36211	5	83235962880	428070666240
37	91	605	2191	6753	19807	36051	1	23781703680	23781703680
37	107	589	2111	6833	19967	35891	1	118908518400	118908518400
38	28	679	2521	6384	19118	36767	1	218494402560	218494402560
39	45	673	2451	6415	19229	36683	1	178362777600	178362777600
39	45	801	1555	8975	15645	38475	2	124853944320	178362777600
39	77	641	2291	6575	19549	36363	1	83235962880	83235962880
41	47	821	1575	8877	15547	38627	2	166471925760	237817036800
41	63	677	2391	6397	19291	36675	1	237817036800	237817036800
41	79	661	2311	6477	19451	36515	1	499415777280	499415777280
41	95	645	2231	6557	19611	36355	1	71345111040	71345111040
41	175	565	1831	6957	20411	35555	1	2378170368000	2378170368000
43	49	841	1595	8779	15449	38779	1	1248539443200	1248539443200
43	81	681	2331	6379	19353	36667	2	1248539443200	1747955220480
45	35	749	2591	6041	18775	37299	1	1664719257600	1664719257600
45	51	861	1615	8681	15351	38931	4	428070666240	1664719257600
45	67	717	2431	6201	19095	36979	1	428070666240	428070666240
45	147	637	2031	6601	19895	36179	1	4994157772800	4994157772800
49	135	693	2151	6325	19539	36643	2	2996494663680	4280706662400
51	105	745	2331	6067	19121	37115	1	2140353331200	2140353331200
55	61	961	1715	8191	14861	39691	1	7491236659200	7491236659200
57	63	981	1735	8093	14763	39843	2	20975462645760	29964946636800
59	49	889	2731	5355	18089	38363	1	104877313228800	104877313228800
61	115	845	2431	5577	18631	37875	1	47087773286400	47087773286400
65	71	1061	1815	7701	14371	40451	1	78479622144000	78479622144000
85	91	1261	2015	6721	13391	41971	1	4284987369062400	4284987369062400

Table 2 continued: Weight Distributions of the $(34, 17, d \geq 4)$ Self-Dual Codes.

References

[1] R. T. Bilous. A recursive enumeration of the inequivalent binary $(2k, k)$ self-dual codes, for $2k \leq 32$. Master's thesis, University of Manitoba, 1998.

[2] R. T. Bilous and G. H. J. van Rees. An enumeration of binary self-dual codes of length 32. *Designs, Codes and Cryptography, Volume 26*, pages 61–86, 2002.

[3] J. H. Conway and V. Pless. On the enumeration of self-dual codes. *J. Combin. Theory Ser. A 28*, pages 26–53, 1980.

[4] J. H. Conway, V. Pless, and N. J. A. Sloane. The binary self-dual codes of length up to 32: a revised enumeration. *J. Combin. Theory Ser. A 60*, pages 183–195, 1992.

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

[6] V. Pless. A classification of self-orthogonal codes over $gf(2)$. *Discrete Math. 3*, pages 209–246, 1972.

[7] V. Pless and N. J. A. Sloane. On the classification and enumeration of self-dual codes. *J. Combin. Theory Ser. A 18*, pages 313–335, 1975.