Chapter 3 Low Density MDS Codes and Factors of Complete Graphs

3.1 Introduction

In this chapter, we describe another new family of MDS array codes of distance 3 with optimal update complexity. This family is called the B-Code. The B-Code is of size $n \times l$ over an Abelian group $G(q)$ with an addition operation $+$, where $l = 2n$ or $2n+1$, and q is size of the group $G(q)$. As the X-Code in Chapter 2, the B-Code uses only group additions for its encoding and decoding operations as well. The error model is also the same as that of the X-Code: erasures (errors) are column erasures (errors). Its distance is also defined over columns.

The novelty of this chapter is to use a graph approach to describe the code, making the design of the code easier and more direct. Figure 3.1(a) shows $\hat{B}_6$, the *dual B-Code* of length 6. In addition to the usual representation of a code as an array of information

a_1	a_2	a_3	a_4	a_5	a_6
$a_2 + a_3$	$a_3 + a_4$	$a_4 + a_5$	$a_5 + a_6$	$a_6 + a_1$	$a_1 + a_2$
$a_4 + a_6$	$a_5 + a_1$	$a_6 + a_2$	$a_1 + a_3$	$a_2 + a_4$	$a_3 + a_5$

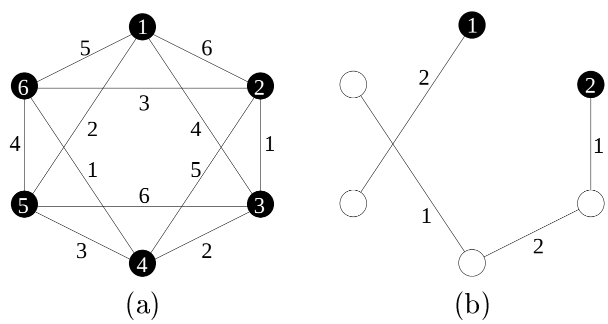


Figure 3.1: $\hat{B}_6$, the dual *B-Code* of length 6, is a 3×6 MDS array code or a $(6, 2, 5)$ MDS code over $G(2^3)$. The a_i's are the information bits. (a) the graph representation of $\hat{B}_6$, (b) a decoding path for the erasure of columns 3, 4, 5 and 6 (i.e., only columns 1 and 2 are available).

and parity bits, the B-Code can be represented by a labeled graph in which every vertex corresponds to an information bit and each edge represents a parity bit: each parity bit is simply the sum of the two information bits that constitute the edge. The edges and vertices of the graph are labeled with a column index: the ith column of the code consists of the information bit a_i and the parity bits with the column index i. The same notation will be used hereafter in this chapter. $\hat{B}_6$ has distance 5 and can therefore tolerate any erasure of 4 columns. Figure 3.1(b) shows a decoding path for the erasure of columns 3 through 6. Use a_2 (from column 2) together with parity $a_2 + a_3$ (from column 1) to recover a_3. Use the latter along with parity $a_3 + a_4$ (from column 2) to recover a_4, etc. For any 4-columns erasure, such a decoding path exists.

By using this new graph description, it will be proven in this chapter that constructing a B-Code is equivalent to a 3-decade old graph theory problem, the *perfect one-factorizations* of complete graphs[29], denoted **P1F**. Using results on P1F, we can construct two infinite families of B-Codes, one of which can be shown to be the construction of [37]. In addition, there are a number of values for which P1Fs exist that are not in the two infinite families; these result in constructions of the B-Codes of all lengths up to 49. The existence of perfect one-factorizations for *every* complete graph with an even number of nodes is a 35-year old conjecture in graph theory. An affirmative answer to this conjecture will provide the B-Code constructions of arbitrary length. Alternately, the construction of the B-Codes of arbitrary odd length will provide an affirmative answer to the conjecture.

The main contributions of this chapter are:

1. proving the *equivalence* of the perfect one-factorization of complete graphs and the MDS code constructions;
2. providing *constructions* for a new class of low-density MDS array codes;
3. proving that in general, the dual of an MDS array code is still MDS.

The chapter is organized as follows. In Section 3.2, we describe the B-Code and its dual using a new graph representation. In Section 3.3, we reveal the relation between the B-Code and the P1F problem. We also give efficient *erasure* and *error* decoding algorithms for the B-Code. In Section 3.4, we further discuss the equivalence between the B-Code and P1F. In Section 3.5, we conclude the chapter and present some future research directions.

3.2 B-Code and its Dual

As already described, a B-Code is an MDS code of size $n \times l$, with distance 3. The MDS property of the B-Code implies that out of nl bits, exactly $2n$ bits should be parity bits. In this section, we describe the B-Code and its dual code using graphs. We also prove that in general the dual of an MDS array code is also MDS.

3.2.1 Structure of the B-Code

Let B_l denote the B-Code of length l, where $l = 2n$ or $2n+1$. For B_{2n}, the first $n-1$ rows are information rows, and the last row is a parity row, i.e., all the bits in the first $n-1$ rows are information bits, while the $2n$ bits in the last row are parity bits. The structure of B_{2n+1} can be derived from that of B_{2n} simply by adding one more information column as the last column. Their structures are shown in Figure 3.2.

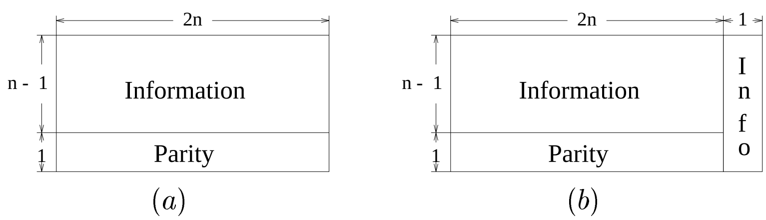


Figure 3.2: Structures of (a) B_{2n} and (b) B_{2n+1}.

Intuitively, if the roles of the information and parity bits of the B-Code are exchanged, i.e., the parity bits are placed in the entries which originally were for the information bits and vice versa, then we get the dual code of the B-Code for length l, denoted $\hat{B}_l$. We will soon give a more rigorous definition of the dual code for general array codes, and prove that the dual of a general MDS array code is also MDS. In particular, the dual B-Code is also an MDS array code; it has distance $l-1$, i.e., the dual B-Code can be recovered from any two of its columns. Figure 3.3 shows the structures of $\hat{B}_{2n}$ and $\hat{B}_{2n+1}$.

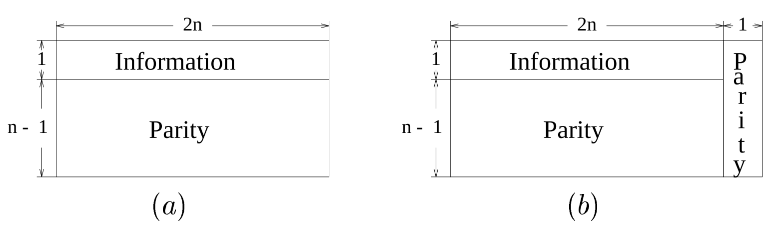


Figure 3.3: Structures of (a) $\hat{B}_{2n}$ and (b) $\hat{B}_{2n+1}$

3.2.2 Dual Array Codes

Array codes are linear codes which can be described by *parity check* or *generator* matrices. Consider an array code of size $n \times l$ over $G(q)$. A codeword of this code can be represented by a vector of length nl over $G(q)$: it consists of l blocks, each of which includes n components. The correspondence between the vector description and the array description is obvious: the ith block of the vector corresponds to the ith column of the array, and the n components within a block are just the n symbols within the corresponding column. A codeword c of a 2×4 array code is shown in both array form and vector form in Figure 3.4.

a_0	a_1	a_2	a_3
p_0	p_1	p_2	p_3

(a)

$$c = (\; a_0 \quad p_0 \mid a_1 \quad p_1 \mid a_2 \quad p_2 \mid a_3 \quad p_3 \;)$$

(b)

Figure 3.4: A codeword of 2×4 code in (a) array form and (b) vector form

Using this vector form, an array code of size $n \times l$ with nr parity bits can be described by its parity check matrix $\mathbf{H}$, of size $nr \times nl$, or its generator matrix $\mathbf{G}$, of size $n(l-r) \times nl$; here r is the number of parity (*redundant*) columns as if some columns consist of only parity bits. Like for other 1-dimensional linear block codes, it is easy to observe that for a codeword c of the array code and an information vector m of length $n(l-r)$, the identities that $c = m\mathbf{G}$ and $c\mathbf{H}^T = \mathbf{0}$ still hold, or equivalently $\mathbf{G}\mathbf{H}^T = \mathbf{0}$. In Figure 3.4, let the a_i's be information bits and p_i's be parity bits. Specifically, when $p_i = a_{(i+1)mod4} + a_{(i+2)mod4}$, for $i = 0, 1, 2, 3$, we get a B-Code of length 4, i.e. B_4, with $n = 2$, $l = 4$ and $r = 2$. Its parity check matrix can be described as follows

$$\mathbf{H} = \left(\begin{array}{cc|cc|cc|cc} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

Accordingly, its generator matrix is as follows

$$\mathbf{G} = \left(\begin{array}{cc|cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right)$$

Using the vector form of array codes, we can define dual of array codes as for a conventional 1-dimensional linear block code, i.e.,

Definition 3.1 (*dual array code*) Let C be a linear array code of size $n \times l$ over $G(q)$, then its dual code $C^{\perp}$ is defined as $C^{\perp} = \{\ \mathbf{u}{\in}G(q)^{nl}\text{:}\ \mathbf{u}{\cdot}\mathbf{v} = 0 \text{ for all } \mathbf{v} \in \mathrm{C}\ \}$, where $\cdot$ is the conventional *dot product* of vectors.

It follows that, as with 1-dimensional linear block codes, the parity check matrix of an array code is the generator matrix of its dual code. One would expect that other properties of dual codes that hold for 1-dimensional linear block codes also hold for array codes. In particular, the dual of MDS array code is also MDS. ([8] gives a proof for the above statement, but it implicitly assumes that information bits and parity bits are not mixed in a same column.) However, for general array codes, since information and parity bits can be mixed in the same column, it is not as obvious that this property holds as it seems to be. Fortunately, this property can be generalized to general linear array codes, and we will prove it here.

Theorem 3.1 *The dual code of an MDS array code is also MDS.*

Proof: Consider an MDS array code C of size $n \times l$. Suppose its distance is $r+1$ with respect to columns. The parity check matrix of C can then be written as $\mathbf{H} = (h_1\ h_2\ \cdots\ h_l)$, where h_i is a submatrix of size $nr \times n$ that corresponds to the ith block in the vector form of a codeword or to the ith column in the array $(1 \leq i \leq l)$. Since C is MDS, any combination of r submatrices (h_i's) is *linearly independent*, in terms of their columns.

Since $\mathbf{H}$ is the generator matrix of the dual code $C^{\perp}$, let a nonzero codeword $c \in C^{\perp}$ have s nonzero columns, where $s \leq l - r$, thus c has zero-columns in some set of r blocks h_i. Without loss of generality, let these blocks be $(h_1\ h_2\ \cdots\ h_r)$. Since c is by definition a linear combination of the r rows of $\mathbf{H}$ (this still holds for any linear array code), the $nr \times nr$ square submatrix formed by $(h_1\ h_2\ \cdots\ h_r)$ must be *singular*, which contradicts the fact that any combination of r blocks (h_i's) are *linearly independent*. Thus the minimum column weight of any codeword of $C^{\perp}$ must be greater than $l - r$, i.e., the minimum distance of $C^{\perp}$ is greater than $l - r$. By the Singleton bound[31], this shows the dual code $C^{\perp}$ is also MDS. $\square$

Since 1-dimensional linear block codes are just a special case of array codes, the above theorem certainly holds and the proof above reduces to one of many proofs for 1-dimensional block codes[31].

3.2.3 A New Graph Description of the B-Code

Typically, an array code is described by its *geometrical construction lines* [4][5][7][15], or by its *parity check matrix* [8][37]. Constructions of array codes are difficult to get using these descriptions. In this chapter, we describe the B-Code and its dual using a new graph approach. By relating the graph conditions for constructing the B-Code to a classical graph problem, *perfect one-factorization* of complete graphs, we obtain new constructions.

For any array code, each parity bit is the sum of some information bits; for binary codes, the addition is just the simple XOR (binary *exclusive OR*) operation. If a parity bit P is the sum of an information bit I and other information bits, then we say that the information bit I *appears* in the parity bit P. Now consider the dual B-Code $\hat{B}_l$. Because of its MDS and optimal encoding properties, each information bit must appear *exactly* $l-2$ times in the parity bits. Since the numbers of the total information and parity bits are $2n$ and $nl-2n$ respectively, each parity bit must be the sum of $\frac{2n(l-2)}{nl-2n}$ or *exactly* 2 information bits. (This is reflected in the parity check matrix by the fact that the weight of each row is exactly 3). So if we represent an information bit as a vertex, then a parity bit can be represented by an edge, where the parity bit is the sum of the two information bits whose vertices form the edge. This is the key idea of describing the B-Code and its dual with graphs.

Since the construction of $\hat{B}_{2n}$ can be obtained from $\hat{B}_{2n+1}$ simply by deleting the last parity column, here we focus on the graph description of $\hat{B}_{2n+1}$. $\hat{B}_{2n+1}$ has $2n$ information bits and $n(2n-1)$ parity bits, which can be represented exactly by a *complete* graph of $2n$ vertices, K_{2n}, which also has exactly $\binom{2n}{2} = n(2n-1)$ edges. The mapping is simple: one information bit can be represented by one vertex, and the parity bit that is the sum of 2 information bits can be represented by the edge that links the 2 corresponding vertices. So the only remaining problem is to define on K_{2n} the grouping relation that determines which information and parity bits occupy the same column of the code. This can be thought of as labeling the vertices and edges of the complete graph K_{2n} in such a way that information bit and parity bits in the same column are labeled with the same label. Since $\hat{B}_{2n+1}$ has $2n+1$ columns, we need $2n+1$ labels. Notice that the each of the first $2n$ columns has exactly 1 information bit and $n-1$ parity bits, and that the last column has n parity bits. A formal way of describing the $\hat{B}_{2n+1}$ is as follows:

Description 3.1 Graph Description of $\hat{B}_{2n+1}$

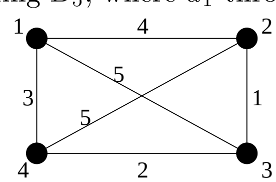
Given a complete graph K_{2n} with 2n vertices, which are labeled with integers from 1 to 2n, find an edge labeling scheme such that

1) each edge is labeled exactly once by an integer from 1 to 2n+1

2) For any pair of vertices (i, j) and any other vertex k, where $i, j, k \in [1, 2n]$, there is always a path to k from either i or j, using only the edges labeled with i or j.

3) For any vertex i and any other vertex k, where $i, k \in [1, 2n]$, there is always a path from i to k, using only the edges labeled with i or $2n + 1$.

With the above description, it is easy to see that the vertex and edges with the label i in the K_{2n} represent the information bit and parity bits in the ith column of $\hat{B}_{2n+1}$. The properties 2) and 3) ensure that any two columns of the code can recover the information bits in all other columns, thus the code is of column distance $2n$. Figure 3.5 shows such a labeling of K_4 and the corresponding $\hat{B}_5$, where a_1 through a_4 are the information bits.



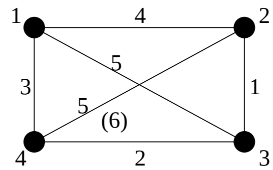
(a)

a_1	a_2	a_3	a_4	$a_1 + a_3$
$a_2 + a_3$	$a_3 + a_4$	$a_4 + a_1$	$a_1 + a_2$	$a_2 + a_4$

(b)

Figure 3.5: (a) graph and (b) array representations of $\hat{B}_5$

Naturally, if the edges of K_{2n} are used to represent information rather than parity bits, and vertices to represent the parity bits, it should be expected that by using such a labeling scheme and reindexing the edges, such a complete graph can represent B_{2n+1}, i.e., the B-Code itself. And in fact this is true. In the graph representation of B_{2n+1}, a parity bit is the sum of all the information bits whose edges are incident with its vertex. B_{2n} can easily be obtained from B_{2n+1} by setting all the information bits in the last column to zero and then deleting them after the parity bits are changed accordingly. B_5 is shown in Figure 3.6, where the edge labeled with (6) represents the information bit a_6 in the 5th column. It is also interesting to point out that B_5 happens to be a *perfect code* too, i.e., it achieves the *Hamming Bound*[31].



(a)

a_1	a_2	a_3	a_4	a_5
$a_3 + a_4 + a_5$	$a_4 + a_6 + a_1$	$a_5 + a_1 + a_2$	$a_6 + a_2 + a_3$	a_6

(b)

Figure 3.6: (a) graph and (b) array representations of B_5

3.3 B-Code and P1F

As already described in Section 3.2, constructing the B-Code amounts to the same problem as designing an edge labeling scheme such as in Description 3.1 for a complete graph K_{2n}. Fortunately this can be related to another graph theory problem, namely the *perfect one-factorization* problem.

3.3.1 Perfect One-Factorization of Complete Graphs

Definition 3.2 [30] Let *G=(V,E)* be a graph. A *factor* or *spanning subgraph* of G is a subgraph with vertex set V. In particular, a *one-factor* is a factor which is a regular graph of degree 1. A *factorization* of G is a set of factors of G which are pairwise *edge disjoint*, and whose union is all of G. A *one-factorization* of G is a factorization of G whose factors are all one-factors. In particular, a one-factorization is *perfect* if the union of any pair of its one-factors is a *Hamilton cycle*, a cycle that passes through every vertex of G.

Figure 3.7 shows a perfect one-factorization of K_4. A perfect one-factorization of K_6 is shown in Figure 3.8(b), where edges with the same label form a one-factor.

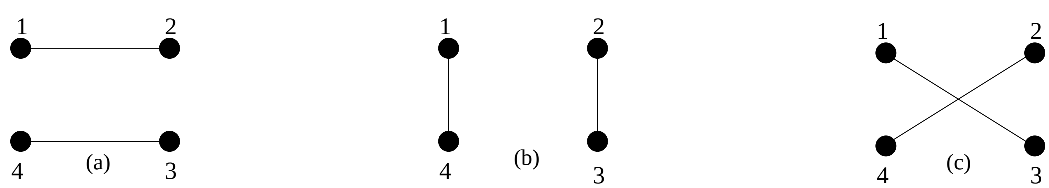


Figure 3.7: (a)(b)(c) are 3 one-factors, that together form a perfect one-factorization of K_4

The perfect one-factorization of complete graphs has been studied for many years since its introduction in [16]. It is now known that[30]:

Theorem 3.2 *If p is an odd prime, then K_{p+1} and K_{2p} have perfect one-factorizations.*

Constructions of P1F for K_{p+1} and K_{2p} can be found in [2] and [29]. Additionally, constructions of P1F for K_{2n}'s whose n's are some other sporadic integers have also been found[29][30]. However it still remains a conjecture [29][30] that:

Conjecture 3.1 *For any positive integer n, K_{2n} has perfect one-factorization(s).*

3.3.2 Equivalence between the B-Code and P1F

Let P_{2n+2} be a P1F for K_{2n+2}. Recall that $\hat{B}_{2n+1}$ has $2n+1$ columns, and P_{2n+2} also has $2n+1$ one-factors. So, if we are able to find a 1–to–1 mapping between the columns and one-factors, then we can get constructions for $\hat{B}_{2n+1}$ from P_{2n+2}, and vice versa. Luckily enough, such a mapping does exist. The following two algorithms give such a 1–to–1 mapping.

Algorithm 3.1 *Constructing $\hat{B}_{2n+1}$ from P_{2n+2}*

Step 1. Label the vertices of K_{2n+2} with $0, 1, \cdots, 2n, \infty$;

Step 2. If a P1F exists for K_{2n+2}, then let F_i denote the one-factor which contains the edge $0i$, where $i = 1, 2, \cdots, 2n, \infty$;

Step 3. In each F_i, delete the two vertices 0 and ∞ and all the edges which are incident with either of them; For $i = 1, 2, \cdots, 2n$, label all the remaining edges in F_i with i, and label all the remaining edges of F_∞ with $2n+1$.

Figure 3.8 shows the construction of $\hat{B}_5$ from P_6, where in (a) ∞ is replaced with 5, and the edges with the same label i form the one-factor F_i.

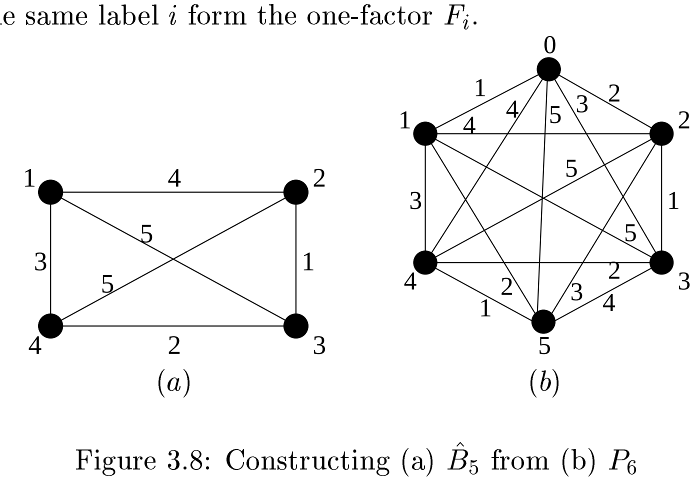


Figure 3.8: Constructing (a) $\hat{B}_5$ from (b) P_6

Theorem 3.3 Algorithm 3.1 *gives a graph as described in* Description 3.1*, i.e., a construction of* $\hat{B}_{2n+1}$.

Proof: First observe that in a P1F of K_{2n+2}, each edge appears exactly once in the whole set of the one-factors, thus step 2 is feasible. Now check the conditions of the graph description of $\hat{B}_{2n+1}$:

1) obviously holds;

2) Since F_i and F_j $(i, j \in [1, 2n])$ are two one-factors of a P_{2n+2}, their union is a Hamilton cycle of $2n+2$ vertices, after deleting the vertices 0 and ∞ and the four edges incident with them, the Hamilton cycle breaks into two paths , covering all the remaining vertices from 1 to $2n$. These paths start from i or j, thus this condition holds.

3) Since F_i and F_∞ $(i \in [1, 2n])$ form a Hamilton cycle of $2n + 2$ vertices where 0∞ is an edge, after the deletion of the vertices 0 and ∞ and the *three* edges incident with them, the Hamilton cycle becomes *one* path starting from i, thus this condition holds. □

Since B_{2n+1} and $\hat{B}_{2n+1}$ can be described with the same complete graph K_{2n}, both B_{2n+1} and $\hat{B}_{2n+1}$ can be constructed from P_{2n+2}. Additionally, B_{2n} and $\hat{B}_{2n}$ can be easily obtained from B_{2n+1} and $\hat{B}_{2n+1}$, so the B-Code and its dual (of size $n \times l$) can be constructed from the known P1F constructions of K_{2n+2}. In particular, from *Theorem 3.2*

Theorem 3.4 *For any odd prime p, a B-Code and its dual code of size $n \times l$ can be constructed, where n is either $\frac{p-1}{2}$ or $p - 1$.*

When n=$\frac{p-1}{2}$, the corresponding B-Code is the code in [37][8]. The B-Code of $n = p-1$ was not known before. The next natural question is : Can we get P_{2n+2} from B_{2n+1}? The answer is *yes* and the following algorithm can do it.

Algorithm 3.2 *Constructing P_{2n+2} from $\hat{B}_{2n+1}$*

Step 1. If $\hat{B}_{2n+1}$ exists, use Description 3.1 *of $\hat{B}_{2n+1}$, let $\tilde{F}_i$ denote the set of edges with the label i, where $i = 1, 2, \cdots, 2n$, and let $\tilde{F}_\infty$ denote the set of the edges with the label $2n + 1$;*

Step 2. Add two vertices 0 and ∞ to K_{2n};

Step 3. For $i = 1, 2, \cdots, 2n$ and ∞, add the edges $i0$ and $k\infty$ to $\tilde{F}_i$, where k is integer from 1 to 2n such that the expanded set, F_i, is a one-factor of the complete graph K_{2n+2} of vertices $0, 1, 2, \cdots, 2n, \infty$.

Theorem 3.5 Algorithm 3.2 *gives a* P_{2n+2}.

Proof: Observe that because of the MDS and optimal encoding properties of $\hat{B}_{2n+1}$, in each of the first $2n$ columns of $\hat{B}_{2n+1}$:

1) each information bit appears at most once;

2) there is exactly one bit which does *not* appear; also *no* pair of the columns miss the same bit, since otherwise that bit can *not* be recovered from just these two columns;

3) in the last column, each bit appears exactly once.

Thus 2) guarantees that in Step 3, there exists a unique k. Further, 1) ensures that for any pair of columns i and j, where $i, j = 1, 2, \cdots, 2n$, the two vertices i and j can only be the endpoints of the two paths in the graph description. Thus Step 3 of the above algorithm makes the union of any pair of F_i and F_j, where $i, j = 1, 2, \cdots, 2n$, into a Hamilton cycle. Step 3 also makes the union of any F_i ($i = 1, 2, \cdots, 2n$) and F_∞ a Hamilton cycle. Thus $\{F_1, F_2, \cdots, F_{2n}, F_\infty\}$ is a P1F of K_{2n+2}, i.e., it is P_{2n+2}. $\square$

Theorem 3.3 and Theorem 3.5 reveal a surprising result:

Theorem 3.6 *Constructing* $\hat{B}_{2n+1}$ (*or equivalently* B_{2n+1}) *is equivalent to constructing* P_{2n+2}*, i.e.,* $P_{2n+2} \Longleftrightarrow B_{2n+1}$.

Note that the equivalence does not include the B-Codes of even length, i.e., B_{2n}. This equivalence, however, already shows that any progress in P1F gives a new B-Code, and vice versa.

3.3.3 Erasure Decoding of the B-Code

Obviously the encoding of the B-Code can be done using Algorithm 3.1. Now consider erasure decoding for the B-Code. Recall that the dual B-Code can recover all information bits from any two columns. Erasure decoding for the dual B-Code is almost obvious from its graph description (*Description 3.1*). The two paths, starting from i and j and leading to all the other vertices in the graph, give the decoding chain used in recovering a $\hat{B}$-Code from its ith and jth columns. Figure 3.9 shows the decoding chain used in recovering $\hat{B}_5$ from its 1st column and its 2nd, 3rd and 5th columns, respectively.

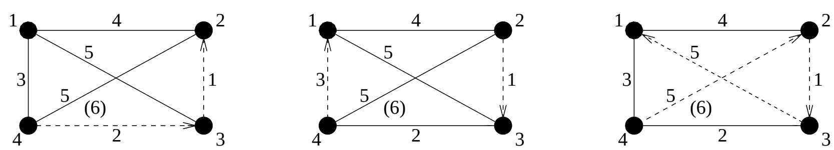
The B-Code can recover any two missing columns. Decoding for the B-Code itself is almost the same as for its dual, except that the roles of edges and vertices are exchanged. Figure 3.10 shows the decoding chains for recovering B_5's 1st column and 2nd, 3rd, and 5th

$1, 2 \to 3 \to 4$ $1 \to 4, 3 \to 2$ $1 \to 3 \to 2 \to 4$

(a) (b) (c)

Figure 3.9: Erasure decoding of $\hat{B}_5$: recovering from its 1st and (a) 2nd (b) 3rd and (c) 5th columns. The decoding chains for each case are also listed. 1 through 4 are the information bits in the corresponding columns.

columns respectively. Comparing the decoding sequences here with those of $\hat{B}_5$, it is easy to obverse that the decoding chains for recovering the ith and jth columns of B_l are just the reversed sequences of those for recovering its dual code $\hat{B}_l$ from its ith and jth columns. This also shows that the two codes are dual to each other, since their graph descriptions are dual to each other.



$4 \to 2 \to 1$ $4 \to 3, 2 \to 1$ $4 \to 5(6) \to 1 \to 5$

(a) (b) (c)

Figure 3.10: Erasure decoding of B_5: recovering its 1st and (a) 2nd (b) 3rd and (c) 5th columns. The decoding chains for each case are also listed. 1 through 6 are the information bits in the corresponding columns, except that 6 is also in the 5th column.

Finally, the above decoding algorithms can be summarized as follows:

Algorithm 3.3 *Erasure decoding of the dual B-Code*

The edges labeled i and j create two paths which span the vertex set. Starting at vertex i and at vertex j, use these paths, by adding a known information bit and a known parity bit to recover a new unknown information bit. Repeating this step along each path recovers the dual B-code from its ith and jth columns.

Algorithm 3.4 *Erasure decoding of the B-Code*

To recover the ith and jth columns of the B-Code, use the same paths with edges labeled with i and j. This time, traverse the paths in the opposite directions of the corresponding paths for the dual B-Code. Along each path, add all known information and parity bits to *get a new unknown information bit. Repeating this step along each path recovers the ith and jth columns of the B-Code from the other $n-2$ columns.*

3.3.4 Error Decoding of the B-Code

Recall that a B-Code of size $n \times l$ is of distance 3, so it can correct one error. To do this, the key is again to locate the error location; the error value can easily be determined once the location is found. One way to find the error location is to make a table that maps syndromes to single-error locations, and then do a table lookup after calculating the syndrome of a received array. The drawbacks are 1) such a table is needed for each B-Code and 2) table lookup is not efficient in both computation time and space (since the total number of 1-error syndromes is 2^n). Another rather straightforward algorithm is to consider the ith column and $(i+1)$th column to be erasures (where $i = 1, 3, 5, \cdots, 2n-1$ for $l = 2n$; if $l = 2n+1$, the lth column and the 1st column are also included), and then recover those columns. If exactly one of the recovered columns differs from the original ones, then that discrepant column is the error column. This algorithm can correct one error. The algorithm requires *on average* $\frac{n}{2}$ erasure decodings, each of which needs $2n(l-3)$ additions, thus the average total number of additions is $n^2(l-3)$, which is in the order of n times of $n \times l$. Another shortcoming is that the algorithm will give a *false* decoding result if more than one error occurs.

We present here a more efficient decoding algorithm for correcting one error. Observe the relation between a B-Code and its dual from their graph descriptions: if an information bit i of a B-Code appears in a set P of parity-bit positions, then in its dual code, the elements of P will be information bits and i will be then a parity bit; further, all the elements of P appear in the parity bit i. Thus, if there is a single column error in a received array of a B-Code, use the syndrome of this received array as the information vector of the dual code. In the obtained dual codeword, the parity bits in the error column of the B-Code should be zero, while other parity bits are nonzero because of the structural properties of the B-Code observed in the proof of Theorem 3.5. This differentiates the error location from other columns. The decoding algorithm can be described semi-formally as follows:

Algorithm 3.5 *Error Decoding of the B-Code*

1. *Given a received array R of size $n \times l$, calculate its syndrome, denoted as S (which*

is a vector of length $2n$ *);*

2. If S *is a zero vector, then the received array* R *is a codeword of* B_l*; otherwise go to next step;*

3. Use S *as the information vector of the dual B-Code* $\hat{B}_l$*, encode to get a codeword* C *of* $\hat{B}_l$*;*

4. If the weight of the syndrome S *is* even, *and if there is a* unique *all-zero column in* C*, then this is the error column of the original received array* R*. On the other hand, if the weight of syndrome* S *is* odd, *and if there is a* unique *column whose information bit is nonzero and whose parity bits are all zero, then this is the error column of* R*;*

5. If the error column of R *is found in the above step, regard this column as an erasure and recover it; otherwise declare decoding failure: there are at least two error columns in* R*.*

Notice that the above property holds only when the B-Code is defined in $G(2^m)$, i.e., each cell of the B-Code consists of a block of m *binary* bits. However, this decoding algorithm can be modified to work for the general B-Codes defined in $G(q^m)$, where q is not a power of 2. Here we will stick to the case where $q = 2$ and $m = 1$.

Before we prove the correctness of the above algorithm, we show an example of it.

Example 3.1 *Error-correcting for the B-Code*

Consider B_6, whose graph description is shown in Fig. 3.1. (Its array description is easy to get from either its graph description or the array description of $\hat{B}_6$, as in Fig. 3.1.) If two received arrays are as follows:

$$R_1 = \begin{array}{|c|c|c|c|c|c|}\hline 1&0&0&0&0&0\\\hline 1&0&0&0&0&0\\\hline\hline 0&0&0&0&0&0\\\hline\end{array} \qquad R_2 = \begin{array}{|c|c|c|c|c|c|}\hline 1&0&0&0&0&0\\\hline 1&0&0&0&0&0\\\hline\hline 1&0&0&0&0&0\\\hline\end{array}$$

Then the two syndromes are respectively:

$$S_1 = \begin{array}{|c|c|c|c|c|c|}\hline 0&1&1&1&0&1\\\hline\end{array} \qquad S_2 = \begin{array}{|c|c|c|c|c|c|}\hline 1&1&1&1&0&1\\\hline\end{array}$$

Since neither S_1 or S_2 is a zero vector, both R_1 and R_2 have errors. Now use S_1 and S_2 as information vectors of $\hat{B}_6$, whose graph and array descriptions are shown in Fig. 3.1. We get two codewords of $\hat{B}_6$

$$C_1 = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 1 \\ \hline \end{array} \qquad C_2 = \begin{array}{|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Since the weight of S_1 is even, and column 1 is the unique all-zero column in C_1, column 1 is the error column of R_1; on the other hand, since the weight of S_2 is odd, and column 1 is the unique column in C_2 whose information bit is nonzero and whose parity bits are all zeros, column 1 is the error column of R_2 too. Once the error column of R_1 (R_2) is found, the error value is easy to get. The corrected arrays of R_1 and R_2 are both all-zero arrays. □

Now we prove the correctness of the decoding algorithm.

Proof: The following can be observed from the graph description of the B-Code: each information bit appears in exactly *two* parity bits, and this information bit and the two parity bits are in three different columns; in addition, two information bits from the same column can *not* appear in the same parity bit, thus all possible errors in the information bits of a single column contribute *even* weight to its syndrome vector. On the other hand, single parity-bit error adds exactly *one* to the weight of the syndrome vector, i.e., a parity-bit makes the weight of the syndrome *odd*.

Suppose there is only one error column in a received array of a B-Code. Call this column the ith column. Consider the following two cases:

Case 1: *All errors occur in information bits.* Then the syndrome should be of *even* weight. Now we prove that the ith column of the obtained codeword of the dual B-Code is the only all-zero column:

1). *The ith column is an all-zero column.* This is true because of the relation between the B-Code and its dual : in the B-Code, an information bit i appears in two parity bits P_1 and P_2, and in the dual B-Code these two bits $\hat{P}_1$ and $\hat{P}_2$ are information bits, and the bit $\hat{i}$ is a parity bit such that both $\hat{P}_1$ and $\hat{P}_2$ appear in $\hat{i}$. Since $P_1 = P_2 = i$ and $\hat{P}_1 = \hat{P}_2$, $\hat{i} = 0$. Thus all parity bits of the ith column of the dual B-Code are zero. Additionally, since there is no error in the parity bit of the B-Code, the information bit of the ith column of the dual B-Code is also zero. So the entire ith column of the dual B-Code is zero.

2). *All the other columns are nonzero columns.* If there were at least one more all-zero column in the codeword, then the weight of the codeword would be no greater than $l-2$, which contradicts the fact the minimum distance of the dual B-Code is $l-1$. Here l is the length of the codeword.

Case 2: *An error also occurs in the parity bit as well.* In the dual B-Code, among all the columns which have both an information bit and parity bits (if the length of the dual B-Code is odd, there is one column containing no information bit), by the linearity of the dual B-Code, the ith column has a nonzero information bit, and all its parity bits are zero. No other column can have a nonzero information bit and all zero parity bits. The reason is as follows: the weight of the information bits in the dual B-Code is *odd*, and all the information bits appear in the ith column. Since each information bit is missing from exactly one of the columns, the number of nonzero information bits that appear in the jth column ($j \neq i$) is *even*. Thus if the information bit of the jth column is nonzero, then at least one of its parity bits is nonzero, since each parity bit is the sum of two information bits, and the total number of information bits which appear in the parity bits is now *odd*.

When multiple column errors occur, there can be multiple all-zero columns or multiple columns with the first component nonzero and all other components zero.

This concludes the proof for the correctness of the decoding algorithm. □

The complexity of the above decoding algorithm is easy to analyze. For a received array of size $n \times l$, the syndrome calculation requires $2(l-2)n$ additions; the encoding of the dual B-Code requires $(l-2)n$ additions; finally, correcting one erasure requires $(l-3)n$ additions. This adds up to $(4l-9)n$ additions, which is linear in the number of total bits in an array of size $n \times l$. The same trick used here cannot be applied directly to correct multiple errors for the dual B-Code, since multiple errors can weave together and cannot be easily separated. In general, it still remains a challenge to correct multiple errors efficiently (total additions linear in total number of bits in an array) for array codes.

3.4 Further Equivalence Discussion

The equivalence between the B-Code and P1F has been shown in the above section. It is quite clear that B_{2n} can be constructed from B_{2n+1} simply by *shortening*, namely, setting all the information bits in the last column to *zero*. Similarly, $\hat{B}_{2n}$ can be derived from $\hat{B}_{2n+1}$

by *puncturing*, i.e., deleting the last parity row. The relations among P_{2n+2}, B_{2n+1} and B_{2n} can be described as follows, where $\Longrightarrow$ means to *lead to*:

$$P_{2n+2} \Longleftrightarrow B_{2n+1} \Longrightarrow B_{2n}$$

A further question is whether B_{2n+1} (or $\hat{B}_{2n+1}$) can be constructed from a known construction of B_{2n} (or $\hat{B}_{2n}$), i.e., whether the last $\Longrightarrow$ can be replaced with $\Longleftrightarrow$. Our conjecture is *yes*.

Conjecture 3.2 *For any positive integer n, $\hat{B}_{2n+1}$ (or B_{2n+1}) can be constructed from $\hat{B}_{2n}$ (or B_{2n}) using* Algorithm 3.6*.*

Algorithm 3.6 *Constructing $\hat{B}_{2n+1}$ from $\hat{B}_{2n}$*
Extend a given $\hat{B}_{2n}$ by adding one more column, which contains, as parity bits, all the unused *or* unlabeled *edges in the graph description of $\hat{B}_{2n}$.*

The B-Codes shown in Figure 3.1, Figure 3.5 and Figure 3.6 are all what we call *shift codes*, and it is easy to verify *Conjecture 2* is true for these examples.

Definition 3.3 (*Shift Code*) An array code (of size $n \times l$) is called a *shift* code if any row of its parity check matrix is just a cyclic shift of the first row, i.e., the remaining columns of the code can be constructed by cyclically shifting the first column.

In general, for a shift B-Code, *Conjecture 2* can be proven true, namely,

Theorem 3.7 *For any shift B-Code, $\hat{B}_{2n+1}$ (or B_{2n+1}) can be constructed from $\hat{B}_{2n}$ (or B_{2n}) using* Algorithm 3.6*.*

Proof: Given a shift dual B-Code, $\hat{B}_{2n}$, notice that the missing edges are the diagonals, $(i, i+n)$, (addition is modulo $2n$). Indeed, if $(i, i+n)$ were present in column j of $\hat{B}_{2n}$, then it would be included in column $n+j$ as well, because of the shift property, making the code non-MDS.

To complete the proof, we need to show that by using an arbitrary column, j, of $\hat{B}_{2n}$ together with the diagonals $(i, i+n)$, $0 \leq i < n$, one can recover all remaining $2n-1$ columns, i.e. we indeed have $\hat{B}_{2n+1}$. Suppose that is not true. There exists a column j, in which a set of edges, combined with the diagonals, form a loop:

$$a_1 + n + a_2 + n + ... + a_q + n = 0 \bmod 2\text{n}$$

where q is the number of edges involved in the loop, a_i's are their lengths and n is the length of the diagonals. For example let $n = 6$, $q = 3$, $a_1 = 1$, $a_2 = 2$ and $a_3 = 3$:

$$1 + 6 + 2 + 6 + 3 + 6 = 24 = 0 \bmod 12$$

We will show that this cannot happen. We will show that column $j + n$ and column j form a loop and therefore the original code is not $\hat{B}_{2n}$. Using the above equation:

$$\sum_{i=1}^{q} a_i = qn$$

Because column $j + n$ is a cyclic shift of column j, it contains a set of edges of lengths $b_i = a_i$ such that A_1 connects to B_2, which in turn connects to A_3, etc.

$$\sum_{i=1}^{q} a_i + \sum_{i=1}^{q} b_i = 2qn = 0 \bmod 2\text{n}$$

There is a loop. □

If *Conjecture 2* can be proven true for any arbitrary B-Code, then we can get a *strong* equivalence between the B-Codes and P1F, i.e, the B-Code construction is completely equivalent to the P1F construction.

3.5 Summary

In this chapter we have presented the B-Code and its dual, a new class of optimal MDS array codes of size $n \times l$ (where $l = 2n$ or $2n+1$) with distance 3 (or $l-1$ for the dual code). We proved an equivalence between the B-Code and perfect one-factorizations using a new graph description of the B-Code. We also described encoding and decoding algorithms for the B-Code and its dual based on their graph descriptions. There are a number of open problems: (i) are the B-Code constructions strongly equivalent to perfect one-factorizations? (ii) can the graph description of the B-Codes be extended to design optimal array codes of arbitrary distance? (iii) how does one efficiently correct multiple errors for the dual B-Code (or other array codes)? and the ultimate question, (iv) can coding theory techniques be used to solve the P1F conjecture?