Low Density MDS Codes and Factors of Complete Graphs *

Lihao Xu   Vasken Bohossian   Jehoshua Bruck
California Institute of Technology
Mail Code 136-93
Pasadena CA 91125
Email: {lihao,vincent,bruck}@paradise.caltech.edu

David G. Wagner
University of Waterloo
Department of Combinatorics and Optimization
Waterloo, Ontario, Canada N2L 3G1
dgwagner@math.uwaterloo.ca

Abstract

We reveal an equivalence relation between the construction of a new class of low density MDS array codes, that we call B-Code, and a combinatorial problem known as *perfect one-factorization* of complete graphs. We use known perfect one-factors of complete graphs to create constructions and decoding algorithms for both B-Code and its *dual code*. B-Code and its dual are optimal in the sense that (i) they are *MDS*, (ii) they have an *optimal encoding* property, i.e., the number of the parity bits that are affected by change of a single information bit is *minimal* and (iii) they have *optimal length*. The existence of perfect one-factorizations for *every* complete graph with an even number of nodes is a 35 years long conjecture in graph theory. The construction of B-codes of arbitrary odd length will provide an affirmative answer to the conjecture.

Keywords: MDS Codes, array codes, update complexity, low density, perfect one-factorization

*Supported in part by the NSF Young Investigator Award CCR-9457811, by an IBM Partnership Award, by the Sloan Research Fellowship, and by DARPA through an agreement with NASA/OSAT.

1 Introduction

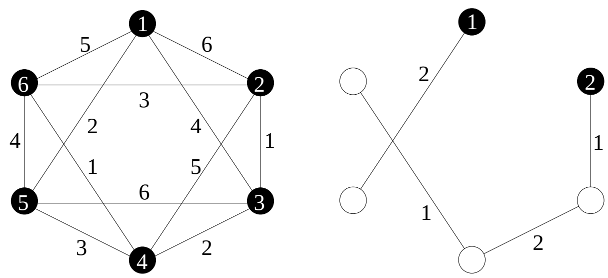
Array codes have important applications in communication and storage systems [6] [7], and have been studied extensively [2] [3] [4] [9]. A common property of these codes is that the encoding and decoding procedures use only simple XOR (*exclusive OR*) operations and thus are more efficient than Reed-Solomon codes in terms of computation complexity [6]. In this paper, we present a new class of linear code, called *B-Code*, an MDS (*Maximum Distance Separable*) array code 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 usual, the dimension of the code is defined as $k = log_{q^n} N$, where N is the number of its codewords. It can also be viewed as an (l, k) code over $G(q^n)$, and its distance is also defined over $G(q^n)$, i.e., over the columns of the array. Throughout this paper, for simplicity, we will assume $q = 2$, i.e., the code is binary, and the addition is just the binary XOR. Within suitable contexts, we will use bits and symbols interchangeably, since the entries in the array can be replaced by symbols of any Abelian group. The B-Code is of distance 3 and its dual code is of distance $l - 1$. Similar to the codes in [2][3][4], the error model of the B-Code is that errors or erasures are columns of the array, i.e., if one bit in a column is an error or erasure, then the whole column is considered to be an error or erasure.

The novelty of this paper is to use a graph approach to describe the code, making the design of the code easier. Figure 1(a) shows, $\hat{B}_6$, the *dual B-Code* of length 6. It can be represented as an array of information and parity bits as well as by a labeled graph in which every vertex corresponds to an information bit and each edge represents a parity bit summing the two information bits that are its vertices. The edges and vertices of the graph are labeled by the index of the column to which the corresponding information and parity bits belong, and a_1 through a_6 are the information bits. The same notation will be used hereafter in this paper. $\hat{B}_6$ has distance 5 and can therefore tolerate the erasure of 4 columns. Figure 1(b) shows a decoding path for the erasures 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.

One important parameter of array codes is the average number of parity bits affected by a change of a single information bit in the codes, called the *update complexity* in this paper. This parameter is particularly crucial when the codes are used in storage applications that need frequent updates of information. Research has been done to reduce this parameter or to make the density of parity check matrix of codes as low as possible [8]. The codes in [4] use two *dependent* parity columns to make the distance of the codes to be 3. But the dependency between the two parity columns makes update of one information bit affecting virtually all the parity bits. So the update complexity of the codes in [4] increases linearly with the number of the columns of the array codes, just similar to the Reed-Solomon codes. To overcome this drawback, the $EVENODD$ codes [2] and their generalizations [3] were designed based on *independent* parity columns resulting in a more efficient information update. The update complexity of $EVENODD$ codes approaches 2 as length (number of the columns) of the codes increases. But it was proven in [3] that for any linear array codes with only parity columns, the update complexity is always *strictly* larger than 2 (the obvious lower bound).

One can ask the following question: Is the update complexity of 2 achievable for general array codes? A positive answer to the foregoing question was given more than a decade ago [16], and the code in [16] was described by its *parity check matrix* with lower density and represented recently

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



(a) (b)

Figure 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 $G(2^3)$. a_i's are the information bits. (a) the graph representation of $\hat{B}_6$, (b) a decoding path for the erasures of columns 3, 4, 5 and 6 (i.e. only columns 1 and 2 are available).

in a general form, also by parity check matrix, in [5]. Another class of MDS array codes with this property was also found recently using geometrical construction[15]. Here we generalize the family of array codes in [16], by using a new graph description and proving the equivalence of B-code constructions to a 3-decade graph theory problem, the *perfect one-factorizations* of complete graphs [12]. Using results on perfect one-factorizations, denoted as **P1F**, we can construct two infinite families of B-Codes, that one of them can be shown to be the construction of [16]. In addition, there are a number of isolated values for which P1F exists resulting in constructions of all B-codes of lengths continuously up to 49. The existence of perfect one-factorizations for *every* complete graph with an even number of nodes is a 35 years long conjecture in graph theory. An affirmative answer to this conjecture will provide B-code constructions of arbitrary length. In addition, the construction of B-codes of arbitrary odd length will provide an affirmative answer to the conjecture.

As already proven in [16], the B-Code achieves the *maximum* length that MDS codes with optimal update property can have, thus the B-Code has *optimal length*, twice of that of the code in [15] with a same column size. In addition, since the parity bits are evenly distributed over all columns, and each parity bit requires the same amount of *XOR* operations, the computation complexity for computing parity bits is *balanced*, i.e., the B-Code features *balanced computation* as well. The properties of the B-Code are summarized in Table 1, together with a comparison with *Reed-Solomon* and *EVENODD* codes.

Codes \ Properties	MDS	XOR	Optimal Update	Optimal Length	Balanced Computation
Reed Solomon	Yes	No	No	Yes	No
EVENODD	Yes	Yes	No	No	No
B	Yes	Yes	Yes	Yes	Yes

Table 1: B-Code vs. Reed-Solomon and EVENODD.

The main contributions of this paper are:

1. proving an *equivalence relation* between 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 for a general array code, the dual of an MDS array code is still MDS.

The paper is organized as follows. In Section 2, we describe the B-Code and its dual using a new graph representation. In Section 3, we reveal the relation between the B-Code and the P1F problem. Efficient *erasure* and *error* decoding algorithms for the B-Code are also given. Section 4 further discusses the equivalence between B-Code and P1F. Section 5 concludes the paper and presents some future research directions.

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 B-Code requires 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 for general array codes it is still true that dual of an MDS array code is also MDS.

2.1 Structure of the B-Code

Denote B_l as the B-Codes 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 parity row, i.e., all 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 the B_{2n} simply by adding one more information column as the last column. Their structures are shown in Figure 2.

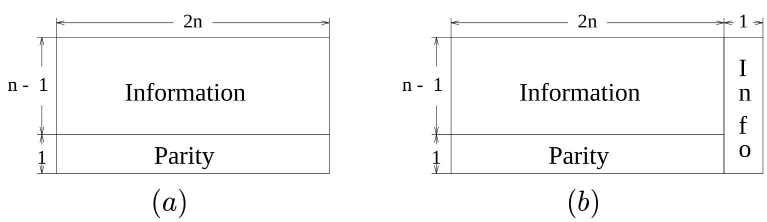
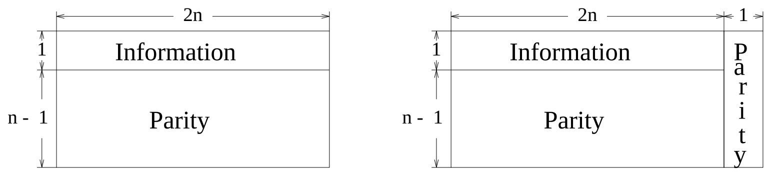


Figure 2: Structures of (a) B_{2n} and (b) B_{2n+1}.

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 may get the dual code of the B-Code, denoted as $\hat{B}_l$ hereafter for length l. We will soon give a more rigorous definition of dual code for general array codes, and prove that for general array codes it still holds that dual of an MDS code is also MDS, particularly the dual B-Code is also an MDS array code of distance $l - 1$, i.e., the dual B-Code can be recovered from any two of its columns. Figure 3 shows structures of $\hat{B}_{2n}$ and $\hat{B}_{2n+1}$.

2.2 Dual Array Codes

As other linear codes, array codes can also be described by *parity check* or *generator* matrices. A codeword of an array code of size $n \times l$ over $G(q)$ can be represented by a vector of length nl over



(a) (b)

Figure 3: Structures of (a) $\hat{B}_{2n}$ and (b) $\hat{B}_{2n+1}$

$G(q)$: 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 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 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 $n(l-r) \times nl$. As with 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 $n(l-r)$, it still holds that $c = m\mathbf{G}$ and $c\mathbf{H}^T = \mathbf{0}$ or equivalently $\mathbf{G}\mathbf{H}^T = \mathbf{0}$. In Figure 4, let the a_i's be information bits and p_i's be parity bits, and specially 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 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

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

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

Definition 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 naturally 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 for 1-dimensional linear block codes also hold for array codes. In particular, dual of MDS array code is also MDS. However, for general array codes, since parity bits can be mixed with information bits in a same column, it is not so obvious for the property to hold as it seems to be. ([5] gives a proof for the above statement, but it implicitly assumes that information bits and parity bits are not mixed in a same column.) Fortunately, this property can be generalized to general linear array codes, and we will prove it here.

Theorem 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, then the parity check matrix of C can be written as $\mathbf{H} = (\ h_1\ h_2\ \cdots\ h_l\ )$, where h_i is a submatrix of size $nr \times n$, which corresponds to the ith block in the vector form or ith column in the array form of a codeword ($1 \leq i \leq l$). Since C is MDS, any combination of r h_i's are *linearly independent*.

Now since $\mathbf{H}$ is the generator matrix of its 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's, without loss of generality, let them 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 codes), the $nr \times nr$ square submatrix formed by $(\ h_1\ h_2\ \cdots\ h_r\ )$ must be *singular*, which contradicts the fact that any combinations of r h_i's are *linearly independent*. Thus 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[14], this shows the dual code $C^{\perp}$ is also MDS. □

Since the 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[14].

2.3 A New Graph Description of the B-Code

Typically, an array code can be described by its *geometrical constructions lines* [2][3][4][9], or by its *parity check matrix* [5][16]. The constructions of array codes are difficult to get using these descriptions. In this paper, we manage to 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, i.e., *perfect one-factorization* of complete graphs, we obtain new constructions.

For any array code, each parity bit is the sum of some information bits, where addition is just the simple XOR (binary *exclusive OR*) operation for binary codes. 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, (which is reflected in its parity check matrix that the weight of each row is exactly 3). So if we represent a vertex as an information bit, then a parity bit can be represented by an edge, where the parity bit is the sum of the two information bits whose vertices are incident with 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 easily obtained from the $\hat{B}_{2n+1}$ simply by deleting the last parity column, here we focus on the graph description of the $\hat{B}_{2n+1}$. $\hat{B}_{2n+1}$ has $2n$ information bits and $n(2n-1)$ parity bits, which can exactly be represented by a *complete* graph of $2n$ vertices K_{2n}, that also has exactly $\binom{2n}{2} = n(2n-1)$ edges. So the only remaining problem is to define on the K_{2n} the grouping relation of information and parity bits in a same column of the code. This can be done by labeling the vertices and edges of the complete graph K_{2n}. A formal way of describing the $\hat{B}_{2n+1}$ is as follows:

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

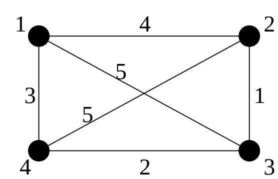
Given a complete graph K_{2n} with 2n vertices which are labeled with integers from 1 to 2n, find an edge labeling scheme such that

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 from either i or j to k, 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 the edges with the label i in the K_{2n} represent the information bit and parity bits in the ith column of the $\hat{B}_{2n+1}$. The properties 2) and 3) ensure that any two columns of the code can recover information bits in all other columns, thus the code is of column distance $2n$. Figure 5 shows such a K_4 and its corresponding $\hat{B}_5$, where a_1 through a_4 are the information bits.



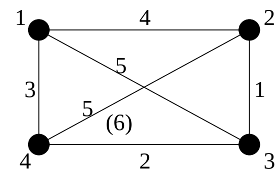
(a)

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

Naturally if the edges of a K_{2n} with such a labeling scheme are used to represent information rather than parity bits, and vertices to represent the parity bits, it should be expected that after reindexing the edges, such a complete graph can represent B_{2n+1}, i.e., the B-Code itself. And in fact it is true. Now 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. The B_5 is shown in Figure 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 the B_5 happens to be a *perfect code* too, which meets the *Hamming Bound*[14].



(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 6: (a) graph and (b) array representations of B_5

3 B-Code and P1F

As already described in Section 2, the construction problem of B-Code becomes how to design such an edge labeling scheme as in Description 1 for a complete graph K_{2n}. Fortunately this can be related to another graph theory problem, namely the *perfect one-factorization* problem.

3.1 Perfect One-Factorization of Complete Graphs

Definition 2 [13] Let G=*(V,E)* be a graph, then a *factor* or *spanning subgraph* of G is a subgraph with vertex set V, particularly 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*, which is a cycle passes through every vertex of G.

Figure 7 shows a perfect one-factorization of K_4. A perfect one-factorization of K_6 is shown in Figure 8(b), where edges with the same label form a one-factor.

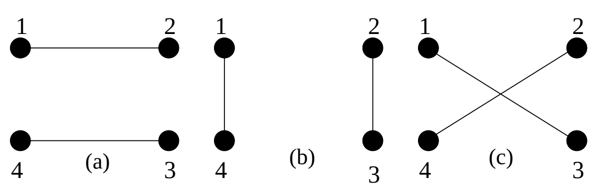


Figure 7: A perfect one-factorization of K_4: (a)(b)(c) are such 3 one-factors

The perfect one-factorization of complete graphs has been studied for many years since its introduction in [10]. It is now known that[13]:

Theorem 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 [1][12]. Besides, constructions of P1F for K_{2n}'s where n's are some other sporadic integers have also been found[12][13]. However it still remains a conjecture that [12][13]:

Conjecture 1 *For any positive integer n, K_{2n} has perfect one-factorization(s).*

3.2 Equivalence between B-Code and P1F

Denote P_{2n+2} as a P1F for the 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 1-1 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 1-1 mapping.

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

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

Step 2. If a P1F exists for the K_{2n+2}, then denote F_i as 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, then label all the remaining edges in F_i with i where $i = 1, 2, \cdots, 2n$, and label all the remaining edges of F_∞ with $2n+1$.

Figure 8 shows constructing $\hat{B}_5$ from P_6, where in (a) ∞ is replaced with 5, and the edges with the same label i form the one-factor F_i.

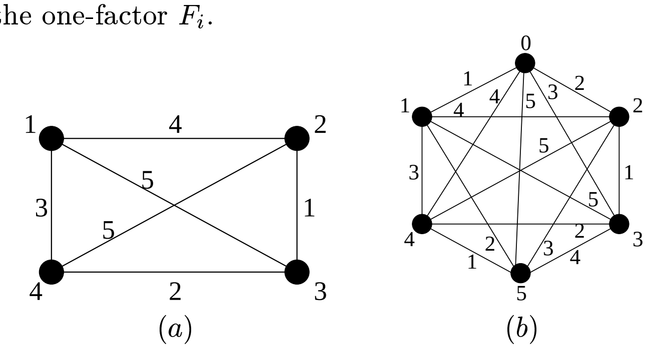


Figure 8: Constructing (a) $\hat{B}_5$ from (b) P_6

Theorem 3 Algorithm 1 *gives a graph as described in* Description 1*, i.e., a construction of the* $\hat{B}_{2n+1}$.

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

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 deletion of the vertices 0 and ∞ and four edges incident with them, the Hamilton cycle breaks into two paths (a simple vertex can be considered as a special case of a path), covering all the remaining vertices from 1 to $2n$, which start from i and j. Thus this condition holds.

3) holds with the similar reason as 2). □

Since the B_{2n+1} and $\hat{B}_{2n+1}$ can be described with the same complete graph K_{2n}, both the B_{2n+1} and $\hat{B}_{2n+1}$ can be constructed from the P_{2n+2}. Besides B_{2n} and $\hat{B}_{2n}$ can be easily obtained from B_{2n+1} and $\hat{B}_{2n+1}$, 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 2*

Theorem 4 *For any odd prime p, 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 [16][5]. 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 positive and the following algorithm can do it.

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

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

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

Step 3. In each $\tilde{F}_i$ where $i = 1, 2, \cdots, 2n$ and ∞, add the edges $i0$ and $k\infty$, where k is such an integer from 1 to 2n 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 5 Algorithm 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 can appear at most once;

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

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

Thus 2) guarantees that in Step 3, there exists a unique such k. 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, so Step 3 of the above algorithm makes the union of any pair of F_i and F_j where $i, j = 1, 2, \cdots, 2n$ 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., P_{2n+2}. □

Theorem 3 and Theorem 5 reveal a surprising result:

Theorem 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}. However, this equivalence already shows that any progress in P1F gives a new B-Code, and vice versa.

3.3 Erasure Decoding of B-Code

Obviously the encoding of B-Code can be done using Algorithm 1. Now consider the erasure decoding of B-Code. Erasure decoding of dual B-Code is almost obvious from its graph description (*Description 1*). The two paths starting from i and j to all other vertices in the graph give the decoding sequence of recovering a $\hat{B}$-Code from its ith and jth columns. Figure 9 shows the decoding sequences for recovering $\hat{B}_5$ from its 1st column and its 2nd, 3rd and 5th columns respectively.

Decoding for B-Code itself is almost the same as its dual except the roles of edges and vertices are just exchanged. Figure 10 shows the decoding sequences for recovering B_5's 1st column and 2nd, 3rd, and 5th columns respectively. Comparing the decoding sequences here with those of $\hat{B}_5$,

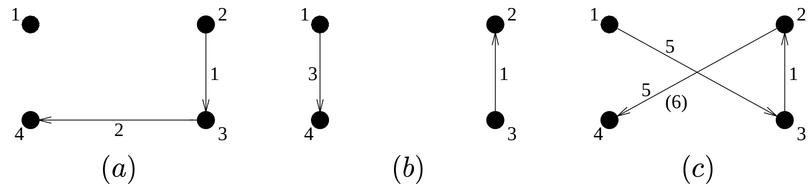


Figure 9: Erasure decoding of $\hat{B}_5$: recovering from its 1st and (a) 2nd (b) 3rd and (c) 5th columns. 1 through 4 are the information bits in the corresponding columns.

it is easy to obverse that the decoding sequences 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.

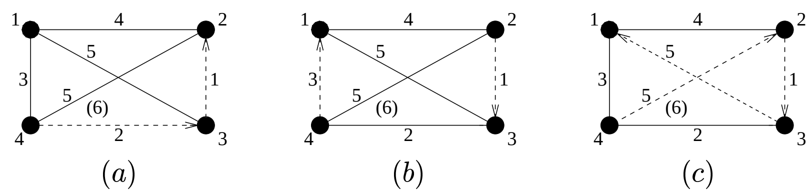


Figure 10: Erasure decoding of B_5: recovering its 1st and (a) 2nd (b) 3rd and (c) 5th columns. 1 through 6 are the information bits in the corresponding columns, except that 6 is also in the 5th column.

Finally the above decoding algorithm can be summarized as follows:

Algorithm 3 *Erasure decoding of B-Code*

Traverse the two paths starting from *vertices i and j using only edges labeled with i and j. Along the paths, add all known information and parity bits to get a new unknown information bit. This recovers the dual B-Code from its ith and jth columns.*

To recover the ith and jth columns of B-Code, just traverse the two paths ending at *vertices i and j using only edges labeled with i and j. Along the paths, add all known information and parity bits to get a new unknown information bit.*

3.4 Error Decoding of B-Code

Recall that B-Code of size $n \times l$ is of distance 3, so it can correct one error. To do this, the key is to locate the error location, and the error value can be easily determined once the location is found. One way to find the error location is to make a table that maps syndromes to one-error locations, and then do table lookup after calculating the syndrome of a received array. The drawbacks are 1) for each B-Code, such a table is needed 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 as erasures (where $i = 1, 3, 5, \cdots, 2n-1$ for $l = 2n$, and add the lth column and the 1st column if $l = 2n+1$), and then recover them. If the distance between the recovered columns and the original ones is 1, then the discrepant column is the error column. This algorithm can correct one error. The algorithm needs *in 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 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, which can also distinguish multiple errors, i.e, it is a *complete* decoding algorithm. Observe the relation between a B-Code and its dual from their graph descriptions: if an information bit of B-Code is disseminated to a set of parity-bit positions, then in its dual code, the now-information bits in the same set of positions convene in the parity bit where the information bit of the B-Code was. Thus if there is a single column error in a received array of a B-Code, use the syndrome of this received array as an information vector of its dual code, then in the obtained dual code, the parity bits in the error column of the B-Code should be zeros, while other parity bits are nonzeros because of the structural properties of B-Code as observed in the proof for Theorem 5. This differentiates the error location from other columns. The decoding algorithm can be described semi-formally as follows:

Algorithm 4 *Error Decoding of 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 $\hat{B}_l$, do encoding and 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 all the rest parity bits are zeros, then this is the error column of R;*

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

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

Example 1 *Consider B_6 whose graph description is shown in Fig. 1. If two received array are as follows:*

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

Then the two syndrome are as follows 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 of the S_1 and S_2 is a zero vector, both R_1 and R_2 have errors. Now use the S_1 and S_2 as information vectors of $\hat{B}_6$ whose graph and array descriptions are shown in Fig. 1, we get two codewords of $\hat{B}_6$ respectively as follows:

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

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 all parity bits are 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 decide. The corrected arrays of R_1 and R_2 are both all-zero arrays. □

Now we prove the correctness of the decoding algorithm.

Proof: It 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 different columns; in addition, two information bits at a same column can *NOT* appear in a same parity bit, thus all information-bit errors of a single column contribute *even* weight to its syndrome vector. On the other hand, one parity-bit error can add only *one* more to the weight of the syndrome vector, i.e., a parity-bit makes the weight of its syndrome *odd*.

Suppose there is only one error column in a received array of a B-Code, denote this column as 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 appears in two parity bits, and in the dual B-Code these two now-information bits add to the same parity bit where the information bit of the B-Code was, thus the parity bit of the dual B-Code is zero, i.e, all parity bits of the ith column of the dual B-Code are zeros. Besides, 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 ith column of the dual B-Code is zero.

2). All the other columns are nonzero columns: if there was at least one more all-zero column in the codeword, then the weight of the codeword is no greater than $l-2$, contradicts the fact the minimum distance of dual B-Code is $l-1$, where l is the length of the codeword.

Case 2: the error also occurs at the parity bit, then in the dual B-Code, among all the columns which have both an information bit and parity bits (there is one column containing no information bit when the length of the dual B-Code is odd), because of the linearity of the dual B-Code, the ith column has the information bit nonzero, and all the parity bits zeros. Any other column can*not* have its information bit nonzero while all its parity bits being zeros. The reason is as follows: the weight of the information bits in the dual B-Code is *odd*, and all the information bits already appear at the ith column. Since each information bit does *not* appear exact *once* in one of the columns, the number of the nonzero information bits that appear in 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 the 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 the other components zeros.

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 needs $2(l-2)n$ additions; the encoding of the dual B-Code needs $(l-2)n$ additions; finally the one erasure correcting needs $(l-3)n$ additions, all summing 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 directly apply 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 efficiently (total additions needed linear in total number of bits in an array) correct multiple errors for array codes.

4 Further Equivalence Discussion

The equivalence between 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*:

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

Now 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 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 5.

Algorithm 5 *Constructing $\hat{B}_{2n+1}$ from $\hat{B}_{2n}$*
For a given $\hat{B}_{2n}$, extend it by adding one more column with all the unused *or* unlabeled *edges in its graph description as parity bits.*

The B-Codes shown in Figure 1, Figure 5 and Figure 6 all have the property that we call *shift codes*, and it is easy to verify *Conjecture 2* is true for these examples.

Definition 3 (*Shift Code*) An array Code (of size $n \times l$) is called as a *shift* code if any other row of its parity check matrix is just a cyclic-shifted version of the first row, i.e., the rest columns of the code can be constructed by shifting from the first column.

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

Theorem 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 5.

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 $(i, i+n)$ was 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:

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

Where q is the number of edges involved in the loop, a_i 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. Consider column $j + n$, we will show that together with column j it forms 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 shifted version 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.

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

There is a loop. □

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

5 Conclusions

We have presented 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). We proved an equivalence between the B-Code and perfect one-factorization using a new graph description. We also described encoding and decoding algorithms for B-Code and its dual based on their graph descriptions. There are a number of open problems: (i) are B-Code constructions strongly equivalent to perfect one-factorizations? (ii) can the graph description for B-Code be extended to design optimal array codes of arbitrary distance? (iii) how to correct multiple errors efficiently for dual B-code (or other array codes)? and the ultimate question, (iv) can coding theory techniques be used to solve the P1F conjecture?

Acknowledgment

We are grateful to Dr. Mario Blaum of IBM Almaden Research Center for his helpful discussions with us.

References

[1] B. A. Anderson, "Symmetry Groups of Some Perfect 1-Factorizations of Complete Graphs", *Discrete Mathematics*, 18, 227-234, 1977

[2] M. Blaum, J. Brady, J. Bruck and J. Menon, "EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures", *IEEE Trans. on Computers*, 44(2), 192-202, Feb. 1995.

[3] M. Blaum, J. Bruck, A. Vardy, "MDS Array Codes with Independent Parity Symbols", *IEEE Trans. on Information Theory*, 42(2), 529-542, March 1996.

[4] M. Blaum, R. M. Roth, "New Array Codes for Multiple Phased Burst Correction", *IEEE Trans. on Information Theory*, 39(1), 66-77, Jan. 1993.

[5] M. Blaum, R. M. Roth, "On Lowest-Density MDS Codes", *to appear in IEEE Transactions on Information Theory*

[6] M. Blaum, P. G. Farrell and H. C. A. van Tilborg, "Chapter on Array Codes", Handbook of Coding Theory, edited by V. S. Pless and W. C. Huffman, to appear.

[7] P. G. Farrell, "A Survey of Array Error Control Codes", *ETT* , Vol.3, No.5, 441-454, 1992.

[8] R. G. Gallager, "Low-Density Parity-Check Codes", MIT Press, Cambridge, Massachusetts, 1963

[9] R. M. Goodman, R. J. McEliece and M. Sayano, "Phased Burst Error Correcting Arrays Codes", *IEEE Trans. on Information Theory*, 39, 684-693,1993.

[10] A. Kotzig, "Hamilton Graphs and Hamilton Circuits", *Theory of Graphs and Its Applications* (Proc. Sympos. Smolenice), 63-82, 1963

[11] R. M. Tanner, "A Recursive Approach to Low Complexity Codes", *IEEE Trans. on Information Theory*, 27(5), 533-547, Sep. 1981

[12] D. G. Wagner, "On the Perfect One-Factorization Conjecture", *Discrete Mathematics*, 104, 211-215, 1992

[13] W. D. Wallis, *One-Factorizations*, Kluwer Academic Publisher, 1997

[14] Stephen B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice-Hall Inc., 1995

[15] L. Xu, J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding", *to appear in IEEE Transactions on Information Theory*

[16] G. V. Zaitsev, V. A. Zinov'ev, and N. V. Semakov, "Minimum-Check-Density Codes for Correcting Bytes of Errors, Erasures, Or Defects", *Problems of Information Transmission*, 19(3), 197-204, 1983