BULLETIN
of the
MALAYSIAN MATHEMATICAL SOCIETY

Bull. Malaysian Math. Soc. (Second Series) **21** (1998) 63-78

The Check Positions of Hamming Codes and the Construction of a 2EC-AUED Code

[1]HOW GUAN AUN AND [2]ANG MIIN HUEY
School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM Pulau Pinang, Malaysia
[1]gahow@cs.usm.my, [2]ang_miin_huey@hotmail.com

Abstract. Hamming Code is the oldest and the most commonly used single error correcting and double errors detecting code. For implication, it is constructed over the field $GF(2)$. For each $r \geq 2$, there is a $[n, k, 3]$ Hamming Code where $n = 2^r - 1$ and $k = 2^r - r - 1$. A message word of length *k* is encoded using a generating matrix *G* into a codeword of length *n*. This amounts to inserting *r* parity check digits into the message word. The positions of the parity check digits in the codeword are called the *check positions* of the code (with respect to *G*). A received word is then decoded using a parity check matrix *H*. If the check positions of the code are in the $i_1^{\text{th}}, i_2^{\text{th}}, \cdots, i_r^{\text{th}}$ coordinates of the codeword, then the $i_1^{\text{th}}, i_2^{\text{th}}, \cdots, i_r^{\text{th}}$ rows of *H* are called the *check rows* of the code. We proved in this paper that for any parity check matrix of a Hamming Code, there exists a generating matrix *G* of the code, such that the check rows of the code are linearly independent. We believe that this fact is contained implicitly in a paper of Hamming [7] but we cannot find any explicit proof in existing literature. Using the above fact, we construct a 2 EC-AUED code.

1. Introduction

More than 40 years, error control coding theory has been proved to have increased the reliability of computer or communication systems against errors [17]. Different system may be vulnerable to different types of errors. Error that changes digit 1 to 0 is called a *1-error* whereas error that changes digit 0 to 1 is called a *0-error*. If both 0-errors and 1-errors occur with equal probability in each received word, the errors are classified as *symmetry type*. If 0-errors and 1-errors are both likely to occur but not simultaneously in each received word of a system, the errors are classified as *unidirectional type*. It has been found that the most likely errors that occurred in VLSI memories are not of symmetric type but of unidirectional type [1, 6,12, 13, 14]. As a result, a receiver usually gets limited number of symmetry errors while the number of unidirectional errors can be very large. Therefore for the late 10 years, the aim of most research works in coding theory [2, 3, 4, 5, 11, 16] is to design codes for correcting up to *t* symmetry errors and detecting all occurrences of unidirectional error. These codes are called *t* EC-AUED codes. In [5], Bose and Rao have shown that constant weight codes of minimal distance $2t + 2$ are *t EC-AUED codes*. In this paper, we shall construct a 2 EC-AUED code, which is not of constant weight type. Before this, we need to show a property regarding the check rows of $[n, k, 3]$ Hamming Codes. The basic theories used in this paper can be found in [8, 9,15].

2. The check rows of Hamming Code

Let F be any finite field $GF(p^{\mu})$. A subset V of F^r is said to be a *pair-wise independent subset* if and only if any two distinct elements in V are linearly independent. V is a *maximal pair-wise independent subset* if and only if there does not exist any other pair-wise independent subset in F^r that contain V. Proposition below gives a property of the maximal pair-wise independent subset.

Proposition. *Let V be a pair-wise independent subset of* F^r, $r \geq 2$. *V is a maximal pair-wise independent subset of* F^r *if and only if* $|V| = \frac{|F|^r - 1}{|F| - 1}$.

Proof. Let K be a relation define on $F^r - \{\boldsymbol{0}\}$ such that $\forall\, \boldsymbol{a}, \boldsymbol{b} \in F^r - \{\boldsymbol{0}\}$, $\boldsymbol{a}K\boldsymbol{b}$ if and only if $\exists\, k \in F - \{0\}$ such that $\boldsymbol{b} = k\boldsymbol{a}$, i.e., if and only if $\{\boldsymbol{a}, \boldsymbol{b}\}$ is linearly dependent. Clearly, K is an equivalent relation. Let $\boldsymbol{a} \in F^r - \{\boldsymbol{0}\}$ and $E_{\boldsymbol{a}}$ be the equivalent class containing $\boldsymbol{a}$. Thus, $E_{\boldsymbol{a}} = \{\, \boldsymbol{b} \in F^r \mid \boldsymbol{a}K\boldsymbol{b} \,\} = \{ k\boldsymbol{a} \mid k \in F - \{0\}\}$. $|E_a| = |F| - \mathbf{1}$ and thus K has $\frac{|F|^r - 1}{|F| - 1}$ distinct equivalent classes.

Let V be a pair-wise independent subset of F^r. If $|V| > \frac{|F|^r - 1}{|F| - 1}$, $\exists\, \boldsymbol{b} \in F^r - \{\boldsymbol{0}\}$ such that $|\, V \cap E_{\boldsymbol{b}} \,| \geq 2$, contradicting the pair-wise independence of V. If $|V| < \frac{|F|^r - 1}{|F| - 1}$, then $\exists\, \boldsymbol{b} \in F^r - \{\boldsymbol{0}\}$ such that $V \cap E_{\boldsymbol{b}} = \{\}$. Choose any $\boldsymbol{a_j} \in E_{\boldsymbol{b}}$ then $V \cup \{\boldsymbol{a_j}\}$ is also a pair-wise independent subset of F^r. This shows that V is not a maximal pair-wise independent subset of F^r. Thus we have proved the theorem.

In term of maximal pair-wise independent subset, the definition of Hamming Codes over F is given as follows

Definition. *A linear code over F such that the rows of its parity check matrix forms a maximal pair-wise independent subset of* F^r, $r \geq 2$, *is called a* $[\boldsymbol{n}, \boldsymbol{k}, \mathbf{3}]$ *Hamming Code where* $n = \frac{|F|^r - 1}{|F| - 1}$ and $k = \frac{|F|^r - 1}{|F| - 1} - r$.

The only maximal pair-wise independent subset of $GF(2)^r$, is $V = GF(2)^r - \{\boldsymbol{0}\}$ with $|V| = 2^r - 1$. Thus over $GF(2)$, a parity check matrix of any $[n, k, 3]$ Hamming Code is simply a matrix whose rows are non-zero elements of

$GF(2)^r$. From now on, we assume $F = GF(2)$. Algorithm below gives a method of getting a generating matrix G for a $[n, k, 3]$ Hamming Code from its parity check matrix.

Algorithm. Let H be any parity check matrix of $[n, k, 3]$ Hamming Code.

(i) Permute the row of H to get H' of the form $\begin{pmatrix} X \\ I_r \end{pmatrix}$. Thus $H' = P_f H$ where f is some element in S_n.

(ii) Let $G' = (I_k \quad X)$.

(iii) Permute the column of G' according to f^{-1} to get G. Hence, $G = G'P_f$.

Obviously,

$$\mathrm{r}(G) = \mathrm{r}(G') = k \quad \text{and} \quad GH = (G'P_f)(P_f^{-1}H') = (I_k \quad X)\begin{pmatrix} X \\ I_r \end{pmatrix} = X + X = \boldsymbol{0}.$$

Thus G is a generating matrix of the $[n, k, 3]$ Hamming Code.

Example 1. Let $GF(2^4)$ be constructed using the irreducible polynomial $1 + x + x^4$ over F. Thus $GF(2^4) = F[x] / \langle 1 + x + x^4 \rangle = \{0, \beta, \beta^2, \cdots, \beta^{15} = 1\}$ and $1 + x + x^4$ is the minimal polynomial of β. We write $m_\beta(x) = 1 + x + x^4$.

Let H be a parity check matrix of $[3, 11, 15]$ cyclic Hamming Code having $m_\beta(x) = 1 + x + x^4$ as its generator polynomial. Permute the rows of H according to

$$f = (\,1 \quad 5 \quad 9 \quad 13 \quad 2 \quad 6 \quad 10 \quad 14 \quad 3 \quad 7 \quad 11 \quad 15 \quad 4 \quad 8 \quad 12\,),$$

we get $H' = \begin{pmatrix} X \\ I_4 \end{pmatrix}$, where

$$X = \begin{pmatrix} \beta^4 \\ \beta^5 \\ \beta^6 \\ \vdots \\ \beta^{14} \end{pmatrix}.$$

Then we get $G' = (I_{11} \quad X)$. Permute the column of G' according to f^{-1} and thus we get $G = (X \quad I_{11})$. Note that $\mathrm{r}(G) = 11$ and $GH = \boldsymbol{0}$. Hence, G is a generating matrix of the $[3, 11, 15]$ Hamming code having H as parity check matrix. Therefore $\forall\, m = (\boldsymbol{a}_0, \boldsymbol{a}_1, \cdots, \boldsymbol{a}_{10}) \in F^{11}$, $\boldsymbol{m}$ is encoded into

$$\boldsymbol{m}G = \left(\sum_{i=0}^{10} a_i\, \beta^{i+4},\ \boldsymbol{m} \right).$$

Now we shall prove our main theorem regarding the check rows of $[n, k, 3]$ Hamming Code.

Theorem. *Assume $r \geq 2$ and H is a parity check matrix for a $[n, k, 3]$ Hamming Code. Let the rows of H be $r_1, r_2, \cdots, r_n$. Then there is a generating matrix G for the code such that when encode using G, if the check positions of the codeword are in the $i_1{}^{\text{th}}, i_2{}^{\text{th}}, \cdots, i_r{}^{\text{th}}$ coordinates, then $\{\boldsymbol{r}_{i_1}, \boldsymbol{r}_{i_2}, \cdots, \boldsymbol{r}_{i_r}\} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_r\}$.*

Proof. Consider a parity check matrix of $[n, k, 3]$ Hamming Code with the form $H' = \begin{pmatrix} X \\ I_r \end{pmatrix}$, where X is a $k \times r$ matrix. Apparently $G' = (I_k \quad X)$ is a generating matrix of the code. When encoding using G', the check positions are in the last r coordinates and the theorem is obviously true for this particular case.

Let H be any parity check matrix of the $[n, k, 3]$ Hamming Code C. Then H could be obtained from H' by permuting the rows of H' according to some permutation in S_n. That is $H = P_\theta H'$, for some $\theta \in S_n$ and thus $G = G' P_{\theta^{-1}}$ is a generating matrix of C. While encoding using G, every message word $\boldsymbol{m} \in F^k$ is encoded into $\boldsymbol{m}G$. Hence,

$$\begin{aligned} mG &= (mG') P_{\vartheta^{-1}} \\ &= (\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_k,\ \boldsymbol{a}_{k+1}, \cdots, \boldsymbol{a}_n) P_{\vartheta^{-1}} \\ &= (\boldsymbol{a}_{\theta(1)}, \boldsymbol{a}_{\theta(2)}, \cdots, \boldsymbol{a}_{\theta(k)},\ \boldsymbol{a}_{\theta(k+1)}, \cdots, \boldsymbol{a}_{\theta(n)}). \end{aligned}$$

i_k is a check position (with respect to the generating matrix G)

$\Rightarrow a_{\theta(i_k)}$ is a parity check digit

$\Rightarrow a_{\theta(i_k)} = a_y$ for some y, $k+1 \leq y \leq n$

$\Rightarrow \vartheta(i_k) = y$ for some y, $k+1 \leq y \leq n$

$\Rightarrow i_k{}^{\text{th}}$ row of H is the y^{th} row of H' for some y, $k+1 \leq y \leq n$

(as $H = P_\theta H' \Rightarrow i_k^{\text{th}}$ row of H is the $\vartheta(i_k)^{\text{th}}$ row of H')

$\Rightarrow i_k{}^{\text{th}}$ row of H has the form of $\boldsymbol{e}_i$ for some i, $1 \leq i \leq r$.

Thus, we have proved the theorem.

Let H be a parity check matrix and G be a generating matrix of a $[n, k, d]$ linear code where $n - k = r$. We named the $i_1{}^{\text{th}}, i_2{}^{\text{th}}, \cdots, i_r{}^{\text{th}}$ rows of H as the *check rows* if the $i_1{}^{\text{th}}, i_2{}^{\text{th}}, \cdots, i_r{}^{\text{th}}$ coordinates are the check positions of the code (with respect to G). Therefore theorem above shows that given any parity check matrix H of a Hamming Code we can find a generating matrix such that when encoding using G, the check rows of the code are linearly independent.

3. The construction of a 2EC-AUED code

We shall now construct a code C of minimal distance 5. The message set M is a constant weight code of length k and each message word $\boldsymbol{m}$ is encoded into a codeword of the form $(\boldsymbol{y}, \boldsymbol{m}, \boldsymbol{u})$ where $\boldsymbol{y}$ and $\boldsymbol{u}$ are elements in F^t and F^p respectively.

Let us start by explaining how to get $\boldsymbol{u}$ from $\boldsymbol{m}$. Let $Q = \{\omega_0, \omega_1, \cdots, \omega_{k-1}\}$ be an additive abelian group of order k with ω_0 its identity element and N be a constant weight code of length p. We assume $|N| \geq k$. For every $\boldsymbol{m} = (m_0, m_1, \cdots, m_{k\text{-}1}) \in M$, we calculate $\sum_{i=0}^{k-1} m_i\, \omega_i$ where

$$m_i\, \omega_i = \begin{cases} \omega_i & \text{if } m_i = 1 \\ \omega_0 & \text{if } m_i = 0. \end{cases}$$

Let

$$g : Q \longrightarrow N$$

be any one to one function. Then we define $\boldsymbol{u} = g\left(\sum_{i=0}^{k-1} m_i\, \omega_i\right)$. This method is due to Rao and Bose [5].

Example 2. Let M be a 3 out of 7 code; $Q = \{\bar{0}, \bar{1}, \bar{2}, \bar{3}, \bar{4}, \bar{5}, \bar{6}\}$ be the additive group of $GF(7)$ and $N = \{\alpha_0 = 11000, \alpha_1 = 01100, \alpha_2 = 00110, \alpha_3 = 00011, \alpha_4 = 10001, \alpha_5 = 10100, \alpha_6 = 10010\}$. a subset of the 2 out of 5 code. Define

$$g : Q \longrightarrow N$$

such that $g(\bar{r}) = \alpha_r$. Then for $\boldsymbol{m} = 0101010 \in M$, we get

$$\boldsymbol{u} = g\left(\sum_{i=0}^{6} m_i\, \bar{i}\right) = g(\bar{0} + \bar{1} + \bar{0} + \bar{3} + \bar{0} + \bar{5} + \bar{0}) = \mathrm{g}(\bar{2}) = \boldsymbol{\alpha_2} = 00110\,.$$

Next we describe how $\boldsymbol{y}$ is obtained from $\boldsymbol{m}$. Let G be a generating matrix and H be a parity check matrix of a linear code having independent check rows. For every $\boldsymbol{m} = (m_0, m_1, \cdots, m_{k\text{-}1}) \in M$, we then permute the coordinates of $\boldsymbol{m}G$ so that the resulting word has the form $(\boldsymbol{y}, \boldsymbol{m})$. Thus we get $\boldsymbol{y}$.

Example 3. Let

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \quad \text{and} \quad H = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

be a generating matrix and parity check matrix of $[3, 4, 7]$ Hamming Code having independent check rows. Let $M = \{1100,\ 0110,\ 0011,\ 1001\}$. Then $\forall \boldsymbol{m} = (a, b, c, d) \in M$, $\boldsymbol{m}G = (a + c + d,\ a + b + c,\ c,\ a + b + d,\ d,\ b,\ a)$. Permute the coordinates of $\boldsymbol{m}$G according to $f^{-1} = (3 \quad 4 \quad 7 \quad 5 \quad 6)$ or compute $(\boldsymbol{m}G)P_f$ to get $\boldsymbol{y}$

$$\begin{array}{ccc} \boldsymbol{m} & & \boldsymbol{m}G \\ (a, b, c, d) & \longrightarrow & (a + c + d,\ a + b + c,\ c,\ a + b + d,\ d,\ b,\ a) \\ & & \Big\downarrow f^{-1} \\ & & (\boldsymbol{m}G)P_f \\ & & (a + c + d,\ a + b + c,\ a + b + d,\ a,\ b,\ c,\ d) = (\boldsymbol{y}, \boldsymbol{m}) \end{array}$$

Hence if $\boldsymbol{m} = 0101 \in M$, then $(0101)G = 1100110$ and thus $(1100110)P_f = 1100101$ and we get $\boldsymbol{y} = 110$.

Now we choose suitable Q, N, g, M, G, H to construct our code C. We take Q to be the additive group of $GF(11)$, write $Q = \{\ \bar{0},\ \bar{1},\ \bar{2},\ \bar{3},\ \bar{4},\ \bar{5},\ \bar{6},\ \bar{7},\ \bar{8},\ \bar{9},\ \overline{10}\ \}$; $N = \{\alpha_0 = 110000,\quad \alpha_1 = 011000,\quad \alpha_2 = 001100,\quad \alpha_3 = 000110,\quad \alpha_4 = 000011,$ $\alpha_5 = 101000,\ \ \alpha_6 = 100100,\ \alpha_7 = 100010,\ \alpha_8 = 100001,\ \alpha_9 = 101000,\ \alpha_{10} = 010010\ \}$ a subset of the 2 out of 6 code and the one to one map g is defined as follow:

$$g : Q \longrightarrow N$$

such that $\boldsymbol{g} : (\bar{\boldsymbol{r}}) = \boldsymbol{\alpha_r}$.

Let G and H be respectively the generating matrix and the parity check matrix of the $[3, 11, 15]$ cyclic Hamming Code, C' given in Example 1. Let M' be the 5 out of 11 code. $\boldsymbol{m} \in M'$ is then encoded into a codeword $\boldsymbol{c} = (\boldsymbol{y}, \boldsymbol{m}, \boldsymbol{u})$ as explained before. For our chosen generating matrix $G = (X \quad I_{11})$, we get $\boldsymbol{y} = \boldsymbol{m}X$ as $\boldsymbol{m}G = (\boldsymbol{y}, \boldsymbol{m})$. Hence $\forall \boldsymbol{c} = (\boldsymbol{y}, \boldsymbol{m}, \boldsymbol{u}) \in C$, $(\boldsymbol{y}, \boldsymbol{m}) \in C'$. It is clear from the choice of Q that $\forall \boldsymbol{m} = (m_0, m_1, \cdots, m_{10}) \in M'$, if $\sum_{i=0}^{10} \boldsymbol{m}_i \, \bar{i} = \bar{r}$, then $\boldsymbol{u} = g(\bar{r}) = \boldsymbol{\alpha_r}$.

Unfortunately the code C encoded from M' has minimal distance less than 5. To increase its distance we choose the set of message words to be M, a subset of M' that satisfies a further condition, namely if $\boldsymbol{a}, \boldsymbol{a}' \in M$, $\boldsymbol{a} \neq \boldsymbol{a}'$, which are encoded into $(\boldsymbol{y}, \boldsymbol{a}, \boldsymbol{u})$ and $(\boldsymbol{y}', \boldsymbol{a}', \boldsymbol{u}')$ respectively, then $\boldsymbol{u} = \boldsymbol{u}'$ and $\boldsymbol{y} = \boldsymbol{y}'$ implies $\mathrm{d}(\boldsymbol{a}, \boldsymbol{a}') \geq 6$. There exist many such subsets M, we exhibit one in appendix. Now let C be the code encoded from M instead of from M' and we shall prove $\mathrm{d}(C) = 5$.

Let $\boldsymbol{c} = (\boldsymbol{y}, \boldsymbol{a}, \boldsymbol{u})$, $\boldsymbol{c}' = (\boldsymbol{y}', \boldsymbol{a}', \boldsymbol{u}') \in C$, $\boldsymbol{c} \neq \boldsymbol{c}'$ and thus $\boldsymbol{a} \neq \boldsymbol{a}'$.

Case 1. $\boldsymbol{y} \neq \boldsymbol{y}'$ and $\boldsymbol{u} \neq \boldsymbol{u}'$: $\boldsymbol{u} \neq \boldsymbol{u}'$ implies $\mathrm{d}(\boldsymbol{u}, \boldsymbol{u}') \geq 2$. As $(\boldsymbol{y}, \boldsymbol{a})$, $(\boldsymbol{y}', \boldsymbol{a}') \in C'$, $\mathrm{d}\big((\boldsymbol{y}, \boldsymbol{a}), (\boldsymbol{y}', \boldsymbol{a}')\big) \geq 3$. Thus, $\mathrm{d}(c, c') \geq 3 + 2 = 5$.

Case 2. $\boldsymbol{y} = \boldsymbol{y}'$ and $\boldsymbol{u} \neq \boldsymbol{u}'$: Similar to Case 1, $\mathrm{d}(\boldsymbol{u}, \boldsymbol{u}') \geq 2$ and $\mathrm{d}((\boldsymbol{y}, \boldsymbol{a}), (\boldsymbol{y}', \boldsymbol{a}')) \geq 3$. Thus $\mathrm{d}(\boldsymbol{c}, \boldsymbol{c}') \geq 3 + 2 = 5$.

Case 3. $\boldsymbol{y} \neq \boldsymbol{y}'$ and $\boldsymbol{u} = \boldsymbol{u}'$: We claim that $\mathrm{d}(\boldsymbol{a}, \boldsymbol{a}') \geq 4$. Let $\boldsymbol{a} = (a_0, a_1, \cdots, a_{10})$ and $\boldsymbol{a} = (a'_0, a'_1, \cdots, a'_{10})$. Assume that $\mathrm{d}(\boldsymbol{a}, \boldsymbol{a}') = 2$ when $\boldsymbol{u} = \boldsymbol{u}'$. Then we have $\sum_{i=0}^{10} a_i \, \bar{i} = \sum_{i=0}^{10} a'_i \, \bar{i}$, where at position j and k, $j \neq k$, we have $a_j = 1$, $a'_j = 0$ and $a_k = 0$, $a'_k = 1$. This results in $\bar{j} = \bar{k}$ for $j \neq k$, which is impossible. Hence if $\boldsymbol{u} = \boldsymbol{u}'$ then $\mathrm{d}(\boldsymbol{a}, \boldsymbol{a}') \geq 4$. Therefore $\mathrm{d}(\boldsymbol{c}, \boldsymbol{c}') \geq 4 + 1 = 5$ as $\mathrm{d}(\boldsymbol{y}, \boldsymbol{y}') \geq 1$.

Case 4. $\boldsymbol{y} = \boldsymbol{y}'$ and $\boldsymbol{u} = \boldsymbol{u}'$: Apparently $\mathrm{d}(\boldsymbol{c}, \boldsymbol{c}') \geq 5$ by the further condition satisfied by M.

Exhausting all possible cases, we see that C is of minimal distance five and thus is a 2 EC code.

Let $\boldsymbol{w} = (\boldsymbol{y}', \boldsymbol{a}, \boldsymbol{u}')$ be any received word. As N and M are constant weight codes, the occurrences of unidirectional errors in $\boldsymbol{a}$ and $\boldsymbol{u}'$ are always detected by C. On the other hand, if unidirectional errors occur only in $\boldsymbol{y}'$ which is also the check positions of C', then the errors can be detected by computing $(\boldsymbol{y}', \boldsymbol{a})H$, since the check rows of C' are linearly independent. Therefore C is a 2 EC-AUED code. Below is the decoding algorithm of C.

Decoding algorithm. Assume $\boldsymbol{w} = (\boldsymbol{y}', \boldsymbol{a}, \boldsymbol{u}')$ is received, where $\boldsymbol{a} = (a_0, a_1, \cdots, a_{10})$. Let $w(x)$ be the check polynomial of $(\boldsymbol{y}', \boldsymbol{a})$ and $\boldsymbol{u}'' = g(\sum_{i=0}^{10} a_i\, \bar{i}\,)$.

Step 1. $\mathrm{wt}(\boldsymbol{u}') \neq 2$.

(i) $\mathrm{wt}(\boldsymbol{a}) = 5$: $\boldsymbol{a}$ is the transmitted message word.

(ii) $\mathrm{wt}(\boldsymbol{a}) \neq 5$: If $\boldsymbol{w}(\beta) = \beta^i$, $4 \le i \le 14$ then $\boldsymbol{a} + \boldsymbol{e}_{i-3}$ is the decoded message word. Else we detect an uncorrectable error pattern in $\boldsymbol{w}$.

Step 2. $\mathrm{wt}(\boldsymbol{u}') \neq 2$.

(i) $\mathrm{wt}(\boldsymbol{a}) = 5$ and $\boldsymbol{u}' = \boldsymbol{u}''$: $\boldsymbol{a}$ is the transmitted message word.

(ii) $\mathrm{wt}(\boldsymbol{a}) = 5$ and $\boldsymbol{u}' \neq \boldsymbol{u}''$:

(a) $\boldsymbol{w}(\boldsymbol{\beta}) = \boldsymbol{0}$: $\boldsymbol{a}$ is the transmitted message word,

(b) $\boldsymbol{w}(\boldsymbol{\beta}) \neq \boldsymbol{0}$: Find

$$Q = \left\{ \boldsymbol{m} = (m_0,\ m_1,\ \cdots,\ m_{10}) \in M \;\middle|\; g\left(\sum_{i=0}^{10} \boldsymbol{m}_i\ \bar{\boldsymbol{i}}\right) = \boldsymbol{u}' \ \text{ and } \ \boldsymbol{m}X = \boldsymbol{y}' \right\}.$$

If $\exists\, \boldsymbol{m} \in Q$ such that $\mathrm{d}(\boldsymbol{m}, \boldsymbol{a}) = 2$, then $\boldsymbol{m}$ is the decoded message word. Else we detect an uncorrectable error pattern in $\boldsymbol{w}$.

(iii) $\mathrm{wt}(\boldsymbol{a}) = 5 \pm 1$ and $g^{-1}(\boldsymbol{u}') \neq \{\}$: Compute

$$s = \pm\left[\left(\sum_{i=0}^{10} a_i\ \bar{i} \right) - g^{-1}(\boldsymbol{u}') \right].$$

If $s = \overline{i-1}$, then $\boldsymbol{a} + \boldsymbol{e}_i$ is the decoded message word. Else an uncorrectable error pattern detected in $\boldsymbol{w}$.

(iv) $\mathrm{wt}(\boldsymbol{a}) = 5 \pm 2$: Find

$$Q = \left\{ \boldsymbol{m} = (m_0,\ m_1,\ \cdots,\ m_{10}) \in M \;\middle|\; g\left(\sum_{i=0}^{10} \boldsymbol{m}_i\ \bar{\boldsymbol{i}}\right) = \boldsymbol{u}' \ \text{ and } \ \boldsymbol{m}X = \boldsymbol{y}' \right\}.$$

If $\exists\, \boldsymbol{m} \in Q$ such that $\mathrm{d}(\boldsymbol{m}, \boldsymbol{a}) = 2$, then $\boldsymbol{m}$ is the decoded message word. Else we detect an uncorrectable error pattern in $\boldsymbol{w}$.

Step 3. Other conditions besides Steps 1 and 2, we detect uncorrectable error pattern in $\boldsymbol{w}$.

The decoding algorithm given above is capable of correcting t errors if $t \le 2$. Suppose no error occurred in $\boldsymbol{w}$. Then $\mathrm{wt}(\boldsymbol{a}) = 5$, $\mathrm{wt}(\boldsymbol{u}') = 2$ and $\boldsymbol{u}' = \boldsymbol{u}''$. Hence Step 1 of the decoding algorithm fails and we go on to Step 2. We get $\boldsymbol{a}$ as the decoded message word according to Step 2 (i).

Assume that an error has occurred in $\boldsymbol{w}$. If the error is in $\boldsymbol{y}'$, then $\mathrm{wt}(\boldsymbol{a}) = 5$, $\mathrm{wt}(\boldsymbol{u}') = 2$ and $\boldsymbol{u}' = \boldsymbol{u}''$. Again Step 1 fail and we get $\boldsymbol{a}$ as the decoded message word according to Step 2(i). However if the error is in the k^{th} position of $\boldsymbol{a}$, then $\mathrm{wt}(\boldsymbol{a}) = 5 \pm 1$ with $\mathrm{wt}(\boldsymbol{u}') = 2$. Obviously

$$s = \pm\left[\left(\sum_{i=0}^{10} a_i\,\bar{i}\right) - g^{-1}(\boldsymbol{u}')\right] = \overline{k-1}\ .$$

Thus, according to Step 2(iii), $\boldsymbol{a} + \boldsymbol{e_k}$ is the decoded message word. If the error happened to be in $\boldsymbol{u}'$, we get $\mathrm{wt}(\boldsymbol{u}') = 2 \pm 1$ with $\mathrm{wt}(\boldsymbol{a}) = 5$. Then Step 1(i) in the decoding algorithm propose $\boldsymbol{a}$ to be the decoded message word.

Assume that double errors have occurred in $\boldsymbol{w}$ during the transmission. If the errors occurred in

(i) $\boldsymbol{y}'$ and the j^{th} position of $\boldsymbol{a}$, then $\mathrm{wt}(\boldsymbol{u}') = 2$ and $\mathrm{wt}(\boldsymbol{a}) = 5 \pm 1$. Apparently $s = \pm\left[(\sum_{i=0}^{10} a_i\,\bar{i}) - g^{-1}(\boldsymbol{u}')\right] = \overline{j-1}$ and thus $\boldsymbol{a} + \boldsymbol{e_j}$ is the decoded message word according to Step 2(iii).

(ii) $\boldsymbol{y}'$ and $\boldsymbol{u}'$, then $\mathrm{wt}(\boldsymbol{u}') = 2 \pm 1$ together with $\mathrm{wt}(\boldsymbol{a}) = 5$. According to Step 1(i), $\boldsymbol{a}$ is the decoded message word.

(iii) $\boldsymbol{u}'$ and the j^{th} position of $\boldsymbol{a}$ (which is the $(j+4)^{\text{th}}$ position of $(\boldsymbol{y}', \boldsymbol{a}) \in C'$), then $\mathrm{wt}(\boldsymbol{u}') = 2 \pm 1$, $\mathrm{wt}(\boldsymbol{a}) = 5 \pm 1$ and $w(\beta) = \beta^{(j+4)-1}$. Let $\beta^i = \beta^{(j+4)-1}$. Apparently $4 \le i \le 14$. Thus $\boldsymbol{a} + \boldsymbol{e_{i\text{-}3}}$ is the decoded message word according to Step 1(ii).

For cases where double errors have occurred simultaneously in $\boldsymbol{y}'$, $\boldsymbol{a}$ or $\boldsymbol{u}'$, the errors involved might be of unidirectional or symmetry types. Assume that two errors have occurred in $\boldsymbol{y}'$. Whether the errors are of unidirectional or symmetry type, we get $\mathrm{wt}(\boldsymbol{a}) = 5$, $\mathrm{wt}(\boldsymbol{u}') = 2$ and $\boldsymbol{u}' = \boldsymbol{u}''$. Hence for both cases, we take $\boldsymbol{a}$ as the decoded message word as proposed in Step 1(i).

If the errors are in $\boldsymbol{u}'$, then $\mathrm{wt}(\boldsymbol{u}') = 2 \pm 2$ if the errors are of unidirectional type or $\mathrm{wt}(\boldsymbol{u}') = 2$ if the errors are of symmetry type with $\mathrm{wt}(\boldsymbol{a}) = 5$. In the first case, $\boldsymbol{a}$ will be taken as decoded message word as given in Step 1(i). For the later case, the two symmetry errors in $\boldsymbol{u}'$ will cause $\boldsymbol{u}' \neq \boldsymbol{u}''$ with $w(\beta) = \boldsymbol{0}$. Hence by Step 2(ii)(a) in the decoding algorithm $\boldsymbol{a}$ is the decoded message word.

Suppose both errors are in $\boldsymbol{a}$, then $\mathrm{wt}(\boldsymbol{a}) = 5 \pm 2$ if the errors are unidirectional errors or $\mathrm{wt}(\boldsymbol{a}) = 5$ if the errors are symmetry errors with $\mathrm{wt}(\boldsymbol{u}') = 2$. The two symmetry errors in $\boldsymbol{a}$ will cause $\boldsymbol{u}' \neq \boldsymbol{u}''$ with $w(\beta) \neq \boldsymbol{0}$. Thus the errors can be corrected by Step 2(ii)(b) in the decoding algorithm. For the remaining case, we will use Step 2(iv) which is similar to Step 2(ii)(b) to correct the errors.

Refer to Steps 2(ii)(b) and 2(iv), we now describe how to construct the set

$$Q = \left\{ \boldsymbol{m} = (m_0,\ m_1,\ \cdots,\ m_{10}) \in M \ \middle|\ g\left(\sum_{i=0}^{10} \boldsymbol{m}_i\ \bar{\boldsymbol{i}}\right) = \boldsymbol{u}' \ \text{ and } \ \boldsymbol{m}X = \boldsymbol{y}' \right\}.$$

We partition M into a number of equivalent classes, each denoted by V_{ij} for $i = 0, 1, 2, \cdots, 15$ and $j = 0, 1, 2, \cdots, 10$ using two equivalent relations, S and Z as given below

$$\forall\, \boldsymbol{a}, \boldsymbol{b} \in M, \quad \boldsymbol{a}S\boldsymbol{b} \quad \text{if and only if} \quad \boldsymbol{a}X = \boldsymbol{b}X.$$

Obviously S is an equivalent relation and thus M is partitioned into 16 equivalent classes, denoted by $V_0, V_1, \cdots, V_{15}$, where

$$V_i = \{\boldsymbol{a} \in M \mid \boldsymbol{a}X \text{ is the binary representation of integer } i\}.$$

Let Z be an equivalent relation defined on V_i such that $\forall\, \boldsymbol{a} = (a_0,\ a_1,\ \cdots,\ a_{10}),\quad \boldsymbol{b} = (b_0,\ b_1,\ \cdots,\ b_{10}) \in V_i$,

$$\boldsymbol{a}Z\boldsymbol{b} \quad \text{if and only if} \quad \sum_{k=0}^{10} a_k\, \bar{k} = \sum_{k=0}^{10} b_k\, \bar{k}.$$

Apparently Z is an equivalent relation on V_i $\forall\, i$. Thus, each V_i can be further partitioned into 11 equivalent classes, denoted by V_{ij}, $j = 0, 1, 2, \cdots, 10$ where $V_{ij} = \left\{(a_0,\ a_0,\ \cdots, a_{10}) \in V_i \mid \sum_{k=0}^{10} a_k\, \bar{k} = \bar{j}\right\}$. Therefore

$$V_i = \bigcup_{j=0}^{10} V_{ij} \quad \text{and} \quad M = \bigcup_{i=0}^{15} V_i = \bigcup_{i=0}^{15} \bigcup_{j=0}^{10} V_{ij}.$$

Thus for a received word $\boldsymbol{w} = (\boldsymbol{y'}, \boldsymbol{a}, \boldsymbol{u'})$, if $\boldsymbol{y'}$ is the binary form of integer i and $g^{-1}(\boldsymbol{u'}) = \bar{j}$, then $Q = V_{ij}$. The list of all elements in each V_{ij}, $i = 0, 1, 2, \cdots, 15$ and $j = 0, 1, 2, \cdots, 10$ is given in appendix.

Assume that there exist $\boldsymbol{m_r}, \boldsymbol{m_s} \in V_i$, $\boldsymbol{m_r} \neq \boldsymbol{m_s}$ such that $\mathrm{d}(\boldsymbol{m_r}, \boldsymbol{a}) = \mathrm{d}(\boldsymbol{m_s}, \boldsymbol{a}) = 2$. Then $\mathrm{d}(\boldsymbol{m_r}, \boldsymbol{m_s}) \leq \mathrm{d}(\boldsymbol{m_s}, \boldsymbol{a}) + \mathrm{d}(\boldsymbol{m_r}, \boldsymbol{a}) = 4$, which is a contradiction as each V_{ij}, is chosen to be a distance 6 constant weight code. Hence if two errors have occurred in $\boldsymbol{a}$, there is an unique $\boldsymbol{m} \in V_{ij}$ such that $\mathrm{d}(\boldsymbol{a}, \boldsymbol{m}) = 2$.

Example 4. Assume that $\boldsymbol{w} = 0000\ 01100111000\ 011000$ is received. Let $\boldsymbol{y'} = 0000$, $\boldsymbol{a} = 01100111000$ and $\boldsymbol{u'} = 011000$. Note that $\mathrm{wt}(\boldsymbol{a}) = 5$, $\mathrm{wt}(\boldsymbol{u'}) = 2$ and $\boldsymbol{u'} \neq \boldsymbol{u''}$ as $\boldsymbol{u''} = \boldsymbol{\alpha}_{10}\left(\sum_{i=0}^{10} a_i\,\bar{i} = \bar{0} + \bar{1} + \bar{2} + \bar{0} + \bar{0} + \bar{5} + \bar{6} + \bar{7} + \bar{0} + \bar{0} + \bar{0} = \overline{10}\right)$ and $\boldsymbol{u'} = \boldsymbol{\alpha}_1$. Since

$$\begin{aligned} w(\beta) &= \beta^5 + \beta^6 + \beta^9 + \beta^{10} + \beta^{11} \\ &= 0110 + 0011 + 0101 + 1110 + 0111 \\ &= 1001 \neq \boldsymbol{0}, \end{aligned}$$

$\boldsymbol{a}$ is compared to each message word in $V_{0\,1}$ by Step 2(ii)(b) (as $\boldsymbol{y'}$ is the binary representation of integer 0 and $g^{-1}(\boldsymbol{u'}) = \bar{1}$). From appendix, we get $\mathrm{d}(01001111000, \boldsymbol{a}) = 2$ and thus 01001111000 is the decoded message word according to Step 2(ii)(b).

Assume that $\boldsymbol{w} = 0000\ 01011111000\ 011000$ is received. Let $\boldsymbol{y'} = 0000$, then $\boldsymbol{a} = 01011111000$ and $\boldsymbol{u'} = 011000$. Note that $\mathrm{wt}(\boldsymbol{a}) = 6$ and $\mathrm{wt}(\boldsymbol{u'}) = 2$. By Step 2(iii), compute $s = \pm\left[\left(\sum_{i=0}^{10} a_i\,\bar{i}\right) - g^{-1}(\boldsymbol{u'})\right]$. As $g^{-1}(\boldsymbol{u'}) = \bar{1}$ and $\sum_{i=0}^{10} a_i\,\bar{i} = \bar{0} + \bar{1} + \bar{0} + \bar{3} + \bar{4} + \bar{5} + \bar{6} + \bar{7} + \bar{0} + \bar{0} + \bar{0} = \bar{4}$, $s = \bar{4} - \bar{1} = \bar{3}$. Thus $\boldsymbol{a} + \boldsymbol{e_4} = 01001111000$ is the decoded message word according to Step 2(iii).

Assume that $\boldsymbol{w} = 0000\ 01001111010\ 010000$ is received. Let $\boldsymbol{y'} = 0000$, then $\boldsymbol{a} = 01001111010$ and $\boldsymbol{u'} = 010000$. Note that $\mathrm{wt}(\boldsymbol{a}) = 6$ and $\mathrm{wt}(\boldsymbol{u'}) = 1$. Obviously two errors have occurred separately in $\boldsymbol{a}$ and $\boldsymbol{u'}$. As

$$\begin{aligned} w(\beta) &= \beta^5 + \beta^8 + \beta^9 + \beta^{10} + \beta^{11} + \beta^{13} \\ &= 0110 + 1010 + 0101 + 1110 + 0111 + 1011 = 1011 \\ &= 1011 = \beta^{13}. \end{aligned}$$

Then $\boldsymbol{a} + \boldsymbol{e}_{10} = 01001111000$ is the decoded message word according to Step 1(ii).

4. Conclusion

We make a few remarks to conclude this paper.

(i) The information rate of the code we constructed is 0.4048, which is good compare to most commonly used codes.
(ii) The main theorem we proved is also true over any arbitrary finite field.
(iii) The check rows of Golay code, C_{23} , are also independent. Using our method, a 4 EC-AUED code could be constructed.
(iv) We do not have an efficient algorithm to compute V_{ij}. This may be a future research problem.

References

1. D.A. Anderson, *Design of Self-checking Digital Networks Using Coding Techniques*, Univ. Illinois, Urbana, CSL Rep. R-527, 1971.
2. M. Blaum and H.C.A. Van Tilborg, On *t*-error correcting/all unidirectional error detecting codes, *IEEE Trans. Comp.* **C-38** (1989), 1493-1501.
3. F.J.H. Bonick and H.C.A. Van Tilborg, Constructions and bounds for systematic *t*-EC/AUED codes, *IEEE Trans. Inf. Theory* **IT-36** (1990), 1381-1390.
4. B. Bose and D.K. Pradhan, Optimum unidirectional error detecting/correcting codes, *IEEE Trans. Comp.* **C-31** (1982), 521-530.
5. B. Bose and T.R.N. Rao, Theory of unidirectional error correcting/detecting codes, *IEEE Trans. Comp.* **C-31** (1982), 564-568.
6. R.W. Cook, W.H. Sisson, T.G. Stoney and W.N. Toy, Design of self-checking microprogram control, *IEEE Trans. Comp.* **C-22** (1973), 225-262.
7. R.W. Hamming, Error detecting and error correcting codes, *Bell Syst. Tech. J.* **29** (1950), 147-160.
8. I.N. Herstein, *Topic in Algebra*, Wiley, New York, 1975.
9. G.A. How, *Algebra with Coding Theory*, Tung Hua Book Co., Taiwan, 1996. (in Chinese)
10. M.C. Lin, Constant weight codes for correcting symmetry errors and detecting unidirectional errors, *IEEE Trans. Comp.* **42** (1993), 1294-1302.
11. D. Nilolos, N. Gaitanis, and G. Philokyprou, Systematic *t*-error correcting/all unidirectional error detecting codes, *IEEE Trans. Comp.* **C-35** (1986), 394-402.
12. B. Parhami and A. Avizienis, Detection of storage errors in mass memories using low-cost arithmetic error codes, *IEEE Trans. Comp.* **C-27** (1978), 302-308.
13. D.K.Pradhan and J.J. Stiffler, Error correcting codes and self-checking circuits in fault-tolerant computers, *Computer* (1980), 27-37.
14. R.W. Sahni, Reliability of integration circuits, *Proceedings of the IEEE International Computers Group Conference*, Washington (1970), 213-219.
15. N.J.A. Sloane and F.J. MacWilliam, *The Theory of Error Correcting Code*, North-Holland, Amsterdam, The Netherlands, 1978.
16. D.L. Tao, C.R.P. Hartmann, and P.K. Lala, An efficient class of unidirectional error detecting/correcting codes, *IEEE Trans. Comp.* **C-37** (1988), 879-882.
17. T.M. Thompson, From error-correcting codes through sphere packings to simple group, *The Carus Math. Monographs* **21**, The Math. Assoc. of America, 1983.

Appendix

$$M=\bigcup_{i=0}^{15}\bigcup_{j=0}^{10}V_{ij},\quad i=0,1,\cdots,15\ \text{ and }\ j=0,1,\cdots,10.\qquad V_i=\bigcup_{j=0}^{10}V_{ij},\quad i=0,1,\cdots,15.$$

j	V_{0j}
0	0 0 0 1 0 0 1 1 1 1 0
1	0 1 0 0 1 1 1 1 0 0 0
2	1 1 0 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 1 1
3	1 0 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 0 0 1 1 0
4	0 1 0 1 0 1 0 1 0 0 1 0 0 1 1 1 0 0 0 1 1 0
5	0 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0
6	0 0 1 0 0 1 1 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1
7	0 1 1 0 0 0 0 1 0 1 1 1 0 0 1 1 1 1 0 0 0 0
8	0 1 0 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 0 0 1 1
9	0 0 0 1 1 1 0 0 0 1 1 1 1 0 1 0 0 1 0 0 0 1
10	0 1 1 1 0 0 0 1 1 0 0 0 0 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 0

j	V_{1j}
0	1 1 0 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0 0
1	1 1 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 0 0 0 1 1 0
2	0 1 1 0 1 0 0 1 0 0 1 0 0 1 1 1 0 1 0 0 1 0
3	0 1 0 0 1 1 0 1 1 0 0 0 0 0 0 1 0 1 1 0 1 1
4	1 0 0 1 0 0 1 1 0 0 1
5	0 1 0 1 0 0 1 0 1 1 0
6	0 1 0 0 0 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 0 1 1 0 0 0 1 1
7	1 1 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 0 1
8	0 1 1 1 0 0 1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 0 0 1 1 1 1 0
9	1 0 0 1 1 1 0 0 1 0 0
10	0 1 1 0 1 1 0 0 0 1 0

j	V_{2j}
0	1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 1 0 0 1 0 1 1 1 0 1 0 1 0 1 0 0 0 1
1	0 0 1 1 0 1 1 1 0 0 0 1 0 0 0 1 1 1 0 1 0 0
2	1 1 0 0 1 0 0 0 0 1 1
3	0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0
4	1 0 0 1 0 1 0 0 1 0 1
5	0 1 1 0 0 1 0 0 0 1 1 0 0 1 1 1 0 0 0 1 0 1 1 0 0 1 0 0 0 1 1 1 0
6	0 0 1 0 1 1 0 1 0 0 1
7	1 0 1 0 0 0 0 0 1 1 1
8	0 1 1 1 0 1 0 0 1 0 0 0 0 1 1 0 0 1 0 0 1 1
9	0 1 0 0 0 1 0 1 1 0 1 0 0 0 1 0 1 1 0 1 1 0 1 0 0 1 1 0 1 1 0 0 0
10	-

j	V_{3j}
0	1 1 0 0 0 1 0 1 0 1 0 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 0 1
1	0 0 0 0 1 1 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0
2	1 0 1 0 1 0 0 0 1 0 1 1 1 1 1 0 0 0 1 0 0 0 1 0 0 1 0 1 1 0 0 0 1
3	1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1 1 0 1 0 1 1 0 0
4	0 0 0 1 1 1 1 0 1 0 0
5	1 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 0 0 1 1 0 1 0 1 1 0 0 0 0 1 1
6	1 0 0 0 0 1 1 0 1 1 0 0 1 1 1 0 1 1 0 0 0 0
7	1 0 1 1 1 0 0 0 0 1 0 0 1 0 0 0 1 1 1 0 0 1
8	1 0 0 0 1 0 0 1 0 1 1 1 0 1 1 0 0 1 0 1 0 0
9	1 0 1 0 0 1 1 1 0 0 0 0 1 0 1 1 1 0 1 0 0 0
10	1 1 0 1 0 0 0 0 1 1 0

j	V_{4j}
0	0 1 1 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 1 1 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0 1 0 0 0 1 1 0
1	1 0 1 1 0 0 0 0 1 0 1 0 1 0 1 1 0 1 0 0 1 0

Appendix (*Cont'd*)

2	1 0 1 0 0 1 0 1 0 0 1
3	0 1 1 0 0 1 0 0 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0 1
4	0 0 1 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 0 1 0
5	-
6	0 0 1 1 0 0 1 0 1 1 0 0 1 1 0 0 0 0 1 1 0 1
7	0 0 1 0 0 1 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 1 0 1 0 1 0 1
8	0 0 0 1 1 1 0 0 1 0 1
9	1 0 1 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 1 1 1 0
10	0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1
j	V_{5j}
0	1 0 0 1 0 1 1 0 1 0 0
1	0 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 0
2	0 0 1 0 1 1 1 1 0 0 0 1 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 1 1
3	0 1 0 1 1 0 0 0 1 1 0 1 0 1 0 0 0 1 0 1 1 0
4	0 1 0 0 1 1 0 1 0 1 0 0 1 1 0 0 0 1 1 0 0 1
5	1 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1
6	1 0 0 0 1 0 0 1 1 1 0 0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 1
7	0 0 0 1 1 0 1 1 0 1 0 1 1 1 0 0 1 0 0 0 0 1

8	1 0 0 0 0 1 1 0 0 1 1 1 1 1 0 0 0 0 1 0 1 0
9	0 0 1 0 1 0 1 0 0 1 1 0 1 1 0 1 1 0 0 1 0 0
10	0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 0 0 1 0 0 0 1
j	V_{6j}
0	1 0 1 1 0 0 0 0 1 1 0
1	0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 1 0 1 0 1 0
2	0 0 0 1 0 0 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 0 0 1 0
3	-
4	0 1 1 0 0 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 1 1 0 1 0 0 0 0
5	1 0 0 0 1 0 1 1 0 0 1 0 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 1 1 1 0
6	0 0 1 1 1 0 0 0 0 1 1 0 1 0 0 1 0 1 0 1 1 0
7	0 0 1 1 0 0 1 0 1 0 1 0 0 0 1 1 1 0 0 1 1 0
8	0 0 1 0 0 1 1 1 0 0 1
9	0 1 0 1 0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 0 0 0
10	0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 0 1 0 0 0
j	V_{7j}
0	0 0 0 0 1 1 1 0 1 0 1 0 1 0 1 0 1 1 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0
1	1 0 0 1 1 0 0 1 0 1 0 0 0 0 0 1 0 1 1 1 1 0

2	1 0 0 1 0 0 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1
3	1 0 1 0 1 0 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 0
4	1 0 0 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 1 0 1 0 1 0 1 0
5	0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1
6	1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1
7	1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1 0 1 0 0 0
8	0 0 0 1 1 0 1 1 0 0 1
9	1 0 1 1 0 0 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1
10	1 0 0 0 0 0 1 1 0 1 1 1 1 0 0 0 1 0 1 1 0 0
j	V_{8j}
0	1 1 0 0 0 1 1 0 0 0 1
1	0 1 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 0 1 1 0 1 0
2	0 0 0 0 0 1 1 1 1 1 0 1 1 0 1 1 1 0 0 0 0 0
3	0 0 1 1 1 0 0 1 0 1 0
4	0 0 1 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 0 1 1
5	0 1 0 1 0 1 0 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0
6	0 1 0 1 0 0 0 1 1 1 0 0 0 1 0 1 1 0 0 1 1 0 1 1 1 0 0 0 1 0 1 0 0
7	1 0 1 1 0 1 0 0 1 0 0

Appendix (*Cont'd*)

8	1 0 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 1 1 1
9	1 1 0 0 1 0 0 1 1 0 0 0 0 0 1 0 1 1 1 0 0 1
10	0 1 0 1 1 0 1 1 0 0 0
j	$V_{9\,j}$
0	0 0 0 1 0 1 0 1 1 0 1 1 0 1 0 1 0 0 1 0 1 0
1	1 0 1 0 0 0 1 1 1 0 0 0 1 0 1 1 0 0 1 1 0 0
2	1 1 0 0 0 1 0 0 1 0 1
3	0 1 1 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 0 1 1 0 0 0 1
4	0 1 0 1 0 0 1 1 0 1 0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 1 0 0 0 0 1 1
5	1 0 0 1 0 0 1 0 1 0 1
6	1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1
7	0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 0 0 0 1 1
8	0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1 0
9	0 1 0 0 1 0 0 1 0 1 1 0 1 1 1 0 0 1 0 1 0 0
10	0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 1 1 0 1 0 1 1 0 0 0 0 1
j	$V_{10\,j}$
0	1 0 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0
1	0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 0 0 0 0

2	0 0 1 1 0 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0 0 1 1 0 0 1
3	0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 1 0 1
4	0 0 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1 0 0
5	0 0 0 1 1 1 0 1 1 0 0
6	1 0 1 0 0 0 0 1 0 1 1
7	0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 0 1 0 0 1 1
8	0 1 0 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0
9	1 0 1 1 0 0 0 1 1 0 0
10	1 0 0 1 1 0 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1 1
j	$V_{11\,j}$
0	1 1 1 0 0 0 0 0 0 1 1
1	1 0 1 0 1 0 0 1 0 0 1 1 1 0 0 0 1 0 0 1 1 0
2	1 0 0 0 1 1 0 1 1 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0
3	1 1 1 1 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0 0
4	1 0 0 1 0 0 1 0 1 1 0 1 1 1 0 0 1 0 1 0 0 0
5	1 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1 0 0 0 1
6	0 0 1 0 1 0 1 1 0 1 0
7	1 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 1

8	1 1 0 1 0 1 0 0 0 0 1
9	1 0 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0 1 0 1 1
10	1 0 1 0 0 1 1 0 1 0 0 0 0 0 1 0 1 0 1 1 1 0 0 0 0 1 1 0 1 0 0 1 1
j	$V_{12\,j}$
0	0 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 1 1 1 0 0 0 0 1 1 1 0 0
1	0 1 1 1 0 0 0 0 1 1 0 1 0 0 1 0 1 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 1
2	0 1 1 0 0 1 0 1 0 1 0 0 0 1 1 1 0 0 1 1 0 0
3	0 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 1
4	1 0 1 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 0 1
5	0 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 1 0
6	0 1 0 1 0 1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 1
7	1 1 1 0 0 0 1 0 0 1 0 0 0 0 1 1 1 0 1 0 0 1
8	0 0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 1 0 1 1 0 0 0
9	1 0 0 0 0 1 0 1 0 1 1
10	0 0 1 0 1 0 0 1 0 1 1
j	$V_{13\,j}$
0	0 0 1 1 1 0 1 1 0 0 0 0 1 0 0 0 0 1 1 0 1 1
1	0 1 0 1 1 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 1

Appendix (*Cont'd*)

2	0 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 1 0 0 0 1 1 0 1 0
3	0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 1 0 0 0 1 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 0 0
4	1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 0 0 1 1
5	0 1 0 0 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 0 1
6	1 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 1
7	0 1 1 1 1 0 0 0 1 0 0
8	0 1 1 0 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 1 1 0
9	0 0 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0 1 1 0
10	1 0 1 0 1 0 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0 1 0 0 1 0 1 1 1 0 0 0

j	$V_{14\,j}$
0	0 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0
1	0 0 0 0 1 1 0 1 1 0 1
2	0 1 1 1 0 0 0 0 1 0 1 1 0 1 0 0 1 0 0 1 1 0
3	0 0 0 1 0 0 1 0 1 1 1 0 1 1 0 0 1 0 1 0 0 1
4	0 1 0 0 1 1 1 0 0 0 1
5	1 0 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 1 1 0 1 0
6	0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0
7	0 0 0 1 0 1 1 1 1 0 0
8	0 1 0 1 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0 1 0 0 0 1
9	1 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 1 1 0 1 0 0
10	0 0 1 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0

j	$V_{15\,j}$
0	0 1 1 1 0 0 1 0 0 0 1 0 1 0 1 1 1 0 0 0 1 0 1 0 1 0 0 1 1 0 0 1 0
1	0 1 0 1 0 1 1 0 1 0 0
2	1 0 0 1 1 0 0 0 1 1 0
3	0 1 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 0 1 0 1 0
4	0 1 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 1 1 1 0 0
5	1 1 0 0 0 0 0 1 0 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0
6	0 1 0 0 1 1 0 0 1 0 1
7	0 1 0 0 1 0 0 1 1 1 0
8	1 1 0 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 1 0 0
9	0 1 0 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 0 0
10	1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1