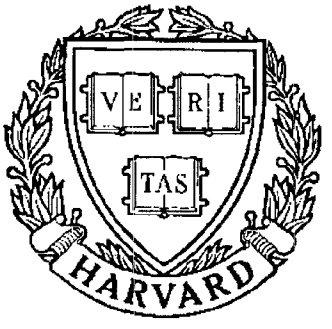
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Distributed Decoding of Cyclic Block Codes Using a Generalization of Majority-Logic Decoding

by A.H. Murad and T.E. Fuja

TR 92-55

Distributed Decoding of Cyclic Block Codes Using a Generalization of Majority-Logic Decoding

Ahsun H. Murad and Thomas E. Fuja[†]
Department of Electrical Engineering
Systems Research Center
University of Maryland
College Park, MD 20742

Abstract

One-step majority-logic decoding is one of the simplest algorithms for decoding cyclic block codes. However, it is an effective decoding scheme for very few codes. This paper presents a generalization based on the "common-symbol decoding problem." Suppose one is given M (possibly corrupted) codewords from M (possibly different) codes over the same field; suppose further that the codewords share a single symbol in common. The common-symbol decoding problem is that of estimating the symbol in the common position. (This is equivalent to one-step majority logic decoding when each of the "constituent" codes is a simple parity check.) This paper formulates conditions under which this decoding is possible and presents a simple algorithm that accomplishes the same. When applied to decoding cyclic block codes, this technique yields a decoder structure ideal for parallel implementation. Furthermore, this approach frequently results in a decoder capable of correcting more errors than one-step majority-logic decoding. To demonstrate the simplicity of the resulting decoders, an example is presented.

[†] Supported in part by National Science Foundation grant NCR-8957623; also by the NSF Engineering Research Centers Program, CDR-8803012. Portions of this paper were presented at the 1991 International Symposium on Information Theory, June 23-28, 1991, Budapest, Hungary

Captions:

Figure 1: M codewords that share a common symbol.

Figure 2: M codewords sharing a common symbol being transmitted over a noisy channel.

Figure 3: Block diagram of the common-symbol decoding algorithm.

Figure 4: Common-symbol decoder for cyclic codes.

Figure 5: Common-symbol decoder for the $(21, 8, 6)$ cyclic code.

Table 1: Some cyclic codes for which the common-symbol decoding algorithm corrects/detects more errors than one-step majority logic decoding.

Table 2: The truth table for common-symbol decoding of the $(21, 8, 6)$ cyclic code.

I. Introduction

One-step majority-logic decoding (first studied in detail by Massey [1]), is perhaps the simplest decoding strategy for decoding linear block codes. Unfortunately, it does not apply effectively to many codes. On the contrary, there are only several small classes of codes for which one-step majority-logic decoding (MLD) is capable of correcting $\lfloor (d_{\min} - 1)/2 \rfloor$ errors [1–5].

A lot of effort has gone into modifying one-step majority-logic decoding so that it can be used to decode a larger class of codes. Usually, a little of the simplicity is sacrificed to achieve wider applicability. In [1], Massey generalized his one-step majority-logic decoding algorithm to an L-step algorithm and demonstrated its applicability to a larger class of codes. This algorithm was shown to be formally equivalent to the Reed algorithm for decoding Muller codes by Gore [6]. Chen [7], and Kasami and Lin [8] showed that the L-step majority-logic decoding algorithm is optimal for decoding finite-geometry codes. Rudolph [9–10] suggested a different generalization of majority-logic decoding involving non-orthogonal parity-checks and a weighted thresholding. Though applicable to all cyclic codes over $\mathbf{F}_p$, its use has been limited by the need to find an appropriate set of parity-checks and a corresponding set of weights. Another generalization was proposed by Gore [11] that decodes Reed-Solomon codes optimally, and others by Assmus et.al. [12], and Gore [13].

The work in this paper can be viewed as yet another generalization of majority-logic decoding. One-step majority-logic decoding can be crudely described as follows: First, find J parity-checks orthogonal on the first symbol position of the code. Each of these parity-checks acts as a vote on the error value in that position. The majority vote is accepted as the error in the first position. This is repeated for every symbol position of the code to construct the error in the codeword. (For a more rigorous description, see [1].)

Upon closer examination, it can be seen that the J orthogonal parity-checks describe J simple parity check codes; that is, if one looks only at the symbol positions indicated by a particular parity check, the set of symbols that can be legitimately placed in those positions forms a simple linear code with minimum distance two. Thus, one-step majority logic decoding may be viewed as the process of decoding the one symbol common to J codewords from J different parity-check codes.

An immediate generalization springs to mind: What if codes other than simple parity-check codes are used in this same manner? It is reasonable to assume that doing so might allow us to decode a larger class of codes optimally. In other cases, we might expect that this decoding strategy, though sub-optimal, would correct more errors than the one-step majority-logic decoding algorithm. Furthermore, the proposed generalization potentially retains the most appealing aspect of MLD: It takes a difficult problem—decoding a powerful block code—and breaks it down into several easier problems—decoding less powerful codes—that can be done in parallel.

In [14], Fuja, Heegard and Goodman considered the following special case of decoding a symbol common to two codewords: Suppose one is given two (possibly corrupted) codewords from two (possibly different) linear block codes over $\mathbf{F}_q$—an (n_1, k_1, d_1) code $\mathcal{C}_1$ and an (n_2, k_2, d_2) code $\mathcal{C}_2$. Suppose further that they share a single symbol in common—that is, the two codewords together consist of $n_1 + n_2 - 1$ symbols, with $n_1 - 1$ symbols belonging to the codeword from $\mathcal{C}_1$ alone, $n_2 - 1$ symbols belonging to the codeword from $\mathcal{C}_2$ alone, and a single symbol belonging to both. The problem addressed was: When can the common symbol be corrected? In [14], it is shown that as long as no more than $\lfloor (d_t - 1)/2 \rfloor$ errors occur in the $n_1 + n_2 - 1$ symbols making up the two codewords, the common symbol can be recovered; here $d_t = d_1 + d_2 - 1$.

In Section 3, we consider the generalization of this problem to the case where the symbol is common to $M \geq 2$ codewords. This problem is called the *common-symbol decoding problem*. In Section 4, a decoding algorithm that accomplishes this goal is developed. The algorithm is conveniently structured in that each "constituent" codeword is decoded in isolation, and the partial results are then "pooled" to obtain a final estimate of the common symbol. In Section 5, we demonstrate the applicability of the common-symbol decoding problem to decoding linear cyclic block codes. The resulting decoder structure is highly suited to a parallel implementation. Section 6 presents a list of some of the codes that we found for which the common-symbol decoding algorithm corrects more errors than the one-step majority-logic decoding algorithm. As an example, Section 7 describes a simple implementation of a decoder for a $(21, 8, 6)$ linear, cyclic, binary block-code that corrects all double errors. Section 8 concludes this paper. First however, we introduce some notation, and preliminary material.

II. Notation and Background

Throughout this paper, we shall denote the Galois Field with q elements and the associated set by $\mathbf{F}_q$. Furthermore, we shall use $\mathbf{F}_q^n$ to represent the product space $\mathbf{F}_q \times \mathbf{F}_q \times \cdots \times \mathbf{F}_q$.

Unless otherwise stated, in this paper, we shall assume a *minimum-distance decoding strategy* as defined below.

Definition 1 (Minimum-Distance Decoding): *Let $\mathcal{C}$ be a linear block code of block-length n over $\mathbf{F}_q$. For any vector $\mathbf{r} = (r_1, r_2, \ldots, r_n) \in \mathbf{F}_q^n$, and for each $i = 1, 2, \ldots, n$, define a mapping*

$$\Phi_i : \mathbf{F}_q^n \longmapsto \mathbf{F}_q \cup \{?\}$$

as follows: Whenever there exists a unique solution u_i to

$$u_i = \underset{\alpha \in \mathbf{F}_q}{\operatorname{argmin}} \quad \min_{\substack{\mathbf{c}=(c_1,c_2,\ldots,c_n)\in\mathcal{C}: \\ c_i=\alpha}} \mathrm{d_H}(\mathbf{r},\mathbf{c}). \tag{2.1}$$

define $\Phi_i(\mathbf{r}) \stackrel{\triangle}{=} u_i$ as that unique solution. If however there exists more than one solution to (2.1), we define $\Phi_i(\mathbf{r}) \stackrel{\triangle}{=}$?. Then, the mapping Φ_i defines the minimum-distance decoder for the ith symbol-position of the code $\mathcal{C}$. The mapping onto a "?" implies a decoder failure.

Next, we define the *symbol-protection distance* of a code and state (without proof) some associated lemmas. These results are drawn from the work on unequal error protection by Masnick and Wolf [15].

Definition 2 : *Let $\mathcal{C}$ be a linear block code of block-length n defined over $\mathbf{F}_q$. For each $i = 1, 2, \ldots, n$, the symbol-protection distance of the i^{th} symbol is defined as*

$$d_i \stackrel{\triangle}{=} \min_{\substack{\mathbf{x}=(x_1,x_2,\ldots,x_n), \\ \mathbf{y}=(y_1,y_2,\ldots,y_n)\in\mathcal{C} \\ x_i\neq y_i}} \mathrm{d_H}(\mathbf{x},\mathbf{y}).$$

That is, d_i is the minimum distance between codewords of $\mathcal{C}$ that differ in the ith position. (It is possible that the minimum above may not be defined—i.e., if *all* of the codewords in $\mathcal{C}$ contain a zero in the i^{th} position. In such a case the i^{th} symbol is useless, and so we shall ignore this possibility.)

Lemma 1. (The Symbol-Protection Lemma) *Let $\mathbf{r} = \mathbf{c} + \mathbf{e}$, where $\mathbf{c} = [c_1, c_2, \ldots, c_n] \in \mathcal{C}$, and let d_i be the symbol-protection distance of the i^{th} symbol of $\mathcal{C}$. Then minimum distance decoding will recover c_i from $\mathbf{r}$ provided*

$$\mathrm{wt}(\mathbf{e}) \leq \lfloor \frac{d_i - 1}{2} \rfloor.$$

Furthermore, if d_i is even, and $\mathrm{wt}(\mathbf{e}) = d_i/2$, then it is possible to either correctly decode c_i or to detect the presence of an uncorrectable error.

Lemma 2. *There exists an error pattern with weight $\mathrm{wt}(\mathbf{e}) = \lfloor d_i/2 \rfloor + 1$ for which the ith symbol is decoded erroneously.*

Proofs of Lemmas 1 and 2 follow from recognizing that the symbol-protection distance of the ith symbol (d_i), is related to the *unequal error protection* (f_i) of Masnick and Wolf [15] by $f_i =$

$\lfloor (d_i - 1)/2 \rfloor$.

The decoding algorithms we shall consider in this paper are of a symbol-by-symbol decoding form. Therefore, the relevant parameter of the code is not its minimum distance, but rather the symbol-protection distance (or equivalently, the unequal error protection) of the associated symbol.

III. The Common-Symbol Decoding Problem

Consider M linear block codes $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_M$, each defined over $\mathbf{F}_q$; assume that $\mathcal{C}_i$ has block-length n_i. Suppose that we select one codeword from each code for transmission over a noisy channel; suppose further that we always select the codewords for transmission so that the first symbol in each codeword is the same, and we transmit this common symbol only once. To illustrate, let

$$\mathbf{c}_i = (c_{i,1}, c_{i,2}, \ldots, c_{i,n_i}) \in \mathcal{C}_i, \qquad i = 1, 2, \ldots, M$$

be such a set of M codewords. Then for each $i = 1, 2, \ldots, M$, $c_{i,2}, c_{i,3}, \ldots, c_{i,n_i}$ are the $n_i - 1$ symbols that belong only to $\mathbf{c}_i$. The first symbol in each codeword $c^* \stackrel{\Delta}{=} c_{1,1} = c_{2,1} = \cdots = c_{M,1}$ is common to each of them. (See Fig. 1.)

Let $\mathbf{r}_i = \mathbf{c}_i + \mathbf{e}_i$ be the (possibly corrupted) received n_i-tuple for $i = 1, 2, \ldots, M$. The $\mathbf{e}_i$'s therefore represent the error patterns. Since the common symbol was transmitted only once, any possible error in the common symbol affects the first symbol of every codeword similarly (i.e. $e_{1,1} = e_{2,1} = \cdots = e_{M,1}$), and therefore the received n_i-tuples also share the first symbol in common (i.e., $r_{1,1} = r_{2,1} = \cdots = r_{M,1}$). (See Fig. 2.)

The Common-Symbol Decoding Problem: *Suppose one is given M (possibly corrupted) codewords $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_M$, one from each of the codes $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_M$; suppose further that these M codewords have a common first symbol, as described above. Under what conditions can the common symbol be correctly estimated?*

Our approach to solving the common-symbol decoding problem shall be as follows: We shall construct a code $\mathcal{C}$ whose codewords are equivalent to all the possible selections of M codewords from the M codes $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_M$. We shall then describe necessary and sufficient conditions under which a minimum distance decoder for the common position in $\mathcal{C}$ will always yield a correct estimate.

Theorem 1. (The Common-Symbol Decoding Theorem) *Given M (possibly corrupted) codewords $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_M$ as described above, it is possible to correctly estimate the common symbol provided no more than $\lfloor (\delta - 1)/2 \rfloor$ errors have occurred throughout the $n = n_1 + n_2 + \cdots + n_M -$*

$(M-1)$ *transmitted symbols. Here*

$$\delta \triangleq \left(\sum_{i=1}^{M} \delta_i\right) \quad -(M-1),$$

$$\textit{where} \qquad \delta_i = \min_{\substack{\mathbf{c}=(c_1,c_2,\ldots,c_{n_i})\in\mathcal{C}_i \\ c_1\neq 0}} \mathrm{wt}(\mathbf{c}).$$

Further, if δ is even and $\delta/2$ errors occur, then it is possible to either correctly decode the common symbol, or to detect the presence of an uncorrectable error.

Proof. Consider a set $\mathbf{T} \subset \mathbf{F}_q^{n_1} \times \mathbf{F}_q^{n_2} \times \cdots \times \mathbf{F}_q^{n_M}$ defined as

$$\mathbf{T} \triangleq \{(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M) : \mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,M}) \in \mathbf{F}_q^{n_i} \text{and } x_{1,1} = x_{2,1} = \cdots = x_{M,1}\}.$$

Therefore each element of $\mathbf{T}$ is a collection of M vectors over $\mathbf{F}_q$ that share the first symbol between them. Now define an invertible mapping

$$\mathcal{A} : \mathbf{T} \longmapsto \mathbf{F}_q^n,$$

where $n = n_1 + n_2 + \ldots n_M - (M-1)$, by

$$\mathcal{A}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M) = (\ x^*, x_{1,2}, x_{1,3}, \ldots, x_{1,n_1}, x_{2,2}, \ldots, x_{2,n_2}, x_{3,2} \cdots \ldots, x_{M-1,n_{M-1}}, x_{M,2}, \ldots, x_{M,n_M}\ )$$

where $x^* = x_{1,1} = x_{2,1} = \cdots = x_{M,1}$. That is, $\mathcal{A}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M)$ represents a concatenation of the M vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$ into the n–tuple $\mathbf{x}$: the common symbol appears only once—at the first position in $\mathbf{x}$.

Now construct a code $\mathcal{C}$ as follows:

$$\mathcal{C} \triangleq \{\mathbf{c} = \mathcal{A}(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_M) : \mathbf{c}_i \in \mathcal{C}_i, \quad \text{and} \quad c_{1,1} = c_{2,1} = \cdots = c_{M,1}\}.$$

Then $\mathcal{C}$ is also a block code over $\mathbf{F}_q$ with block-length n. The common-symbol decoding problem therefore reduces to performing minimum-distance decoding of the first symbol of $\mathcal{C}$ and Lemma 1 and 2 can be applied.

Indeed, let $\mathcal{H}_i = (\mathbf{h}_{i,1}, \mathbf{h}_{i,2}, \ldots, \mathbf{h}_{i,n_i})$ be the parity-check matrix for the code $\mathcal{C}_i$. Here $\mathbf{h}_{i,j}$, $j = 1, 2, \ldots, n_i$ represent the n_i column-vectors of the parity-check matrix $\mathcal{H}_i$. Then, it can easily be shown that the parity-check matrix $\mathcal{H}$ of the code $\mathcal{C}$ is given by

$$\mathcal{H} = \begin{pmatrix} \mathbf{h}_{1,1} & \mathbf{h}_{1,2} & \cdots & \mathbf{h}_{1,n_1} & \mathbf{0} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \mathbf{0} \\ \mathbf{h}_{2,1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{h}_{2,2} & \cdots & \mathbf{h}_{2,n_2} & \mathbf{0} & \cdots & \cdots & \cdots & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ \mathbf{h}_{M,1} & \mathbf{0} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \mathbf{0} & \mathbf{h}_{M,2} & \cdots & \mathbf{h}_{M,n_M} \end{pmatrix}$$

where the **0**'s are all-zero column vectors of appropriate lengths.

For each $i = 1, 2, \ldots, M$, let $\mathbf{a}_i$ be a codeword from $\mathcal{C}_i$ with $\mathrm{wt}(\mathbf{a}_i) = \delta_i$ and $a_{i,1} = 1$. By definition of δ_i, such a codeword is guaranteed to exist. Then $\mathbf{a} = \mathcal{A}(\mathbf{a}_i, \mathbf{a}_2, \ldots, \mathbf{a}_M)$ is a codeword from $\mathcal{C}$. Furthermore, this is the (possibly non-unique) minimum-weight codeword of $\mathcal{C}$ containing a non-zero symbol in the first position. Its weight is therefore, by linearity, the symbol-protection distance of the first symbol in $\mathcal{C}$. Showing that

$$\begin{aligned}\mathrm{wt}(\mathbf{a}) &= \sum_{i=1}^{M}(\mathrm{wt}(\mathbf{a}_i) - 1) \quad + 1 \\ &= \sum_{i=1}^{M}(\delta_i - 1) \quad + 1 \\ &= \delta\end{aligned}$$

and invoking Lemma 1 completes the proof. ❐

Theorem 2. *There exists an error pattern* $\mathbf{e}$ *with* $\mathrm{wt}(\mathbf{e}) = \lfloor \delta/2 \rfloor + 1$ *for which the common symbol is decoded erroneously.*

Proof. In the proof of Theorem 1 we showed that the symbol protection distance of the first symbol in $\mathcal{C}$ is δ. So the result follows by invoking Lemma 2. ❐

Theorem 1 places a restriction on the number of errors that may occur in the n symbols, and still guarantee correct decoding of the common symbol, and Theorem 2 shows that this condition cannot be made less restrictive.

Finally, we should note that Theorems 1 and 2 are consistent with the analogous results for one-step majority logic decoding. If each of the codes are simple parity checks then $\delta_i = 2$ for all i; this in turn implies $\delta = M + 1$ and so we have the following familiar result: If we can find M orthogonal parity checks on a position, then it is possible to correctly estimate the value in that position in the presence of up to $\lfloor M/2 \rfloor$ errors. Further, if M is odd, then the presence of any $(M + 1)/2$ errors can be detected.

IV. The Common-Symbol Decoding Algorithm for Binary Codes

In Section 3 we showed that it is possible to correctly estimate a symbol common to M codewords provided no more than $\lfloor (\delta - 1)/2 \rfloor$ errors have occurred; here $\delta = \delta_1 + \delta_2 + \ldots + \delta_M - (M - 1)$ and δ_i is the minimum Hamming distance between any two codewords in the i^{th} constituent code differing in the common position. Unfortunately, the method used to prove this result assumed

the existence of a large "super-decoder" for the "super code" constructed from the constituent codes. Since our ultimate goal is to use these constituent codes to simplify decoding of the "super code"—just as in majority logic decoding we decode a powerful code by decoding several parity check codes—the method used in Section 3 is unacceptable.

In this section we develop an algorithm for decoding a symbol common to binary M codewords that is naturally distributed. That is, the algorithm consists of first decoding the constituent codes and then "pooling" their collective results in a simple way to estimate the common symbol.

Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_M$ be M linear binary block codes. Assume the block-length of $\mathcal{C}_i$ is n_i, and let δ_i be the symbol-protection distance of its first symbol position. That is,

$$\delta_i = \min_{\substack{\mathbf{c}=(c_1,c_2,\ldots,c_M)\in\mathcal{C}_i \\ c_1\neq 0}} \mathrm{wt}(\mathbf{c}).$$

Furthermore, let $\mathbf{c}_1 \in \mathcal{C}_1, \mathbf{c}_2 \in \mathcal{C}_2, \ldots, \mathbf{c}_M \in \mathcal{C}_M$ be M codewords that share the first symbol between them i.e. $c_{1,1} = c_{2,1} = \cdots = c_{M,1}$. The $n \stackrel{\Delta}{=} n_1 + n_2 + \ldots + n_M - (M-1)$ symbols that make up these codewords are transmitted over a noisy channel, Let $\mathbf{r}_i = \mathbf{c}_i + \mathbf{e}_i$ represent the received n_i-tuples. As was pointed out earlier, the common symbol is transmitted only once. Therefore the $\mathbf{e}_i$'s and the $\mathbf{r}_i$'s also share the corresponding common-symbols. That is, $e_{1,1} = e_{2,1} = \cdots = e_{M,1}$ and $r_{1,1} = r_{2,1} = \cdots = r_{M,1}$. (Note: $\mathbf{c}_i, \mathbf{r}_i, \mathbf{e}_i \in \mathbf{F}_2^{n_i}$ for each $i = 1, 2, \ldots, M$.) We now define the following terms:

(i) $\mathbf{e} = \mathcal{A}(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_M)$. That is, $\mathbf{e}$ consists of a concatenation of all the errors in all of the codewords, with the common errors occurring once in the first position. Let e_1 denote that value in the first position.

(ii) ζ: the number of errors in the common position—i.e., $\zeta = 1$ if $e_1 = 1$ and and $\zeta = 0$ if $e_1 = 0$. (The distinction: ζ is a number and e_1 is an element of $\mathbf{F}_2$.)

(iii) $\hat{\eta}$: the final estimate of e_1, the error in the common position. $\hat{\eta} \in \mathbf{F}_2$.

(iv) $\tau_i = \mathrm{wt}(\mathbf{e}_i)$: the number of errors that actually occurred in $\mathbf{c}_i$ for each $i = 1, 2, \ldots, M$.

(v) For each $i = 1, 2, \ldots, M$ let $\mathcal{D}_i : \mathbf{F}_2^{n_i} \mapsto \mathcal{C}_i \cup \{?\}$ be a decoder for the i^{th} code defined as follows:

$$\mathcal{D}_i(\mathbf{r}_i) \stackrel{\Delta}{=} \begin{cases} \operatorname{argmin} \{\mathrm{d_H}(\mathbf{x}, \mathbf{r}_i) : \mathbf{x} \in S_i(\mathbf{r}_i)\}, & \text{if } S_i(\mathbf{r}_i) \neq \varnothing\ ; \\ ?, & \text{if } S_i(\mathbf{r}_i) = \varnothing; \end{cases}$$

where

$$S_i(\mathbf{r}_i) \triangleq \{\mathbf{y} \in \mathcal{C}_i : \mathrm{d_H}(\mathbf{y}, \mathbf{r}_i) \leq \lfloor (\delta_i - 1)/2 \rfloor\}.$$

Remark: $\mathcal{D}_i$ represents a bounded distance decoder with a slight "twist". Since δ_i is the symbol-protection distance for the first position in $\mathcal{C}_i$ – and *not* the minimum distance – it follows that $S_i(\mathbf{r}_i)$ may contain more than one codeword; the decoder selects as its output the codeword in $S_i(\mathbf{r}_i)$ that is closest to the received n_i-tuple. Note that all of the elements in $S_i(\mathbf{r}_i)$ must have the same first symbol. Note also, it is *possible* that there are multiple solutions to the above equation—i.e., it is possible that there may exist many codewords in $S_i(\mathbf{r}_i)$ that are each "closest" to $\mathbf{r}_i$. In such a case we do not care which of the possible solutions is chosen.

(vi) $\hat{\mathbf{e}}_i$: the ith decoder's estimate of $\mathbf{e}_i$. Whenever $\mathcal{D}_i(\mathbf{r}_i) \neq \mathbf{?}$, let $\hat{\mathbf{c}}_i \triangleq \mathcal{D}_i(\mathbf{r}_i)$ for each $i = 1, 2, \ldots, M$. Then $\hat{\mathbf{e}}_i \triangleq (\mathbf{r}_i - \hat{\mathbf{c}}_i)$. Further define $\hat{\eta}_i \triangleq \hat{e}_{i,1}$ and $\hat{\tau}_i \triangleq \mathrm{wt}(\hat{\mathbf{e}}_i)$. Therefore $\hat{\eta}_i$ represents the ith decoder's estimate of the error in the common position and $\hat{\tau}_i$ represents its estimate of $\mathrm{wt}(\mathbf{e}_i)$. When $\mathcal{D}_i(\mathbf{r}_i) = \mathbf{?}$, this information is somehow conveyed. (One arbitrary way to do this is to set $\hat{\eta}_i = 1, \hat{\tau}_i = 0$.)

Remark: The decoding algorithm that we shall subsequently develop does not require a knowledge of $\hat{\mathbf{e}}_i$, but only of $\hat{\eta}_i$ and $\hat{\tau}_i$. This fact may in some cases result in a simplification of the decoder structure.

(vii) $\mathbf{U} \triangleq \{i : \mathcal{D}_i(\mathbf{r}_i) = \mathbf{?}\}$: the indices of all those decoders that detect an uncorrectable error. We will denote by $\mathbf{U}^c$ the complement of this set—i.e., the indices of all those decoders that come up with a codeword estimate (whether right or wrong).

(viii) $\mathbf{R} \triangleq \{i : \hat{\eta}_i = e_1, i \in \mathbf{U}^c\}$: the indices of all those decoders that come up with the correct estimate of the error in the common position.

(ix) $\mathbf{W} \triangleq \{i : \hat{\eta}_i \neq e_1, i \in \mathbf{U}^c\}$: the indices of all those decoders that come up with an incorrect estimate of the error in the common position.

(x) $\mathbf{E}_0 \triangleq \{i : \hat{\eta}_i = 0, i \in \mathbf{U}^c\}$: the indices of all those decoders that estimate a "0" for the error in the common position.

(xi) $\mathbf{E}_1 \triangleq \{i : \hat{\eta}_i = 1, i \in \mathbf{U}^c\}$: the indices of all those decoders that estimate a "1" for the error in the common position.

Remark: Note that $\mathbf{E}_0$ and $\mathbf{E}_1$ partition $\mathbf{U}^c$ – i.e., $\mathbf{E}_0 \cap \mathbf{E}_1 = \emptyset$, and $\mathbf{E}_0 \cup \mathbf{E}_1 = \mathbf{U}^c$. Similarly $\mathbf{R}$ and $\mathbf{W}$ also partition $\mathbf{U}^c$.

We develop the decoding algorithm through the following series of lemmas.

Lemma 3. *For all $i \in \mathbf{U}^c$, it holds that $\tau_i \geq \hat{\tau}_i$.*

Lemma 4. *For all $i \in \mathbf{U}$, it holds that $2\tau_i \geq \delta_i$.*

Lemma 5. *For all $i \in \mathbf{W}$, it holds that $\hat{\tau}_i \geq \delta_i - \tau_i$.*

Proof of Lemmas 3–5 follow from the properties of the Hamming metric, and by definition of the $\mathcal{D}_i$'s.

Lemma 6. *If $\mathbf{e}$ is an error pattern such that $\mathrm{wt}(\mathbf{e}) \leq \lfloor (\delta - 1)/2 \rfloor$ then*

$$\text{for } e_1 = 1: \quad \sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i) \leq M - 2$$

$$\text{for } e_1 = 0: \quad -\sum_{i \in \mathbf{U}^c} \delta_i \leq \sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i) \leq -M.$$

Proof. Let $\mathbf{e}$ satisfy $\mathrm{wt}(\mathbf{e}) \leq \lfloor (\delta - 1)/2 \rfloor$. Then

$$\Rightarrow \quad 2\,\mathrm{wt}\,(\mathbf{e}) + 1 \leq \left(\sum_{i=1}^{M} \delta_i \right) - M + 1$$

$$\Rightarrow \quad 2\left(\sum_{i=1}^{M} (\tau_i - \zeta) \right) + 2\zeta + 1 \leq \left(\sum_{i=1}^{M} \delta_i \right) - M + 1$$

$$\Rightarrow \quad \sum_{i \in \mathbf{U}} (2\tau_i - \delta_i) + \sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i) \leq M\,(2\zeta - 1) - 2\zeta \tag{4.1}$$

From Lemma 4 we have

$$\sum_{i \in \mathbf{U}} (2\tau_i - \delta_i) \geq 0.$$

Therefore, for $e_1 = 1 (\Leftrightarrow \zeta = 1)$, we have

$$\sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i) \leq M - 2.$$

For $e_1 = 0$, using the fact that $\tau_i \geq 0$, and (4.1) we have

$$-\sum_{i \in \mathbf{U}^c} \delta_i \leq \sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i) \leq -M.$$

This completes the proof of Lemma 6. ❐

Corollary 6. *If* $\sum_{i \in \mathbf{U}^c} \delta_i < M$ *then, either* $e_1 \neq 0$ *or* $\mathrm{wt}(\mathbf{e}) > \lfloor (\delta - 1)/2 \rfloor$.

Proof. Follows as the contra-positive of Lemma 6. ❐

Lemma 7. *If* δ *is even, and* $\mathbf{e}$ *is an error pattern such that* $\mathrm{wt}(\mathbf{e}) = \delta/2$, *then*

$$\text{for } e_1 = 1: \quad \sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i) \leq M - 1$$

$$\text{for } e_1 = 0: \quad -\sum_{i \in \mathbf{U}^c} \delta_i \leq \sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i) \leq -(M-1).$$

Proof. Similar to that of Lemma 6. ❐

Corollary 7. *If* $\sum_{i \in \mathbf{U}^c} \delta_i < M - 1$ *then either* $e_1 \neq 0$ *or* $\mathrm{wt}(\mathbf{e}) > \lfloor \delta/2 \rfloor$.

Proof. Follows from Corollary 6 and the contra-positive of Lemma 7. ❐

Lemma 8. *Define the quantity* α *as*

$$\alpha \stackrel{\triangle}{=} \sum_{i \in \mathbf{E}_1} (2\hat{\tau}_i - \delta_i) \quad - \sum_{i \in \mathbf{E}_0} (2\hat{\tau}_i - \delta_i).$$

Then α *satisfies the following inequalities:*

$$e_1 = 0 \quad \Rightarrow \quad \alpha \geq -\sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i)$$

$$e_1 = 1 \quad \Rightarrow \quad \alpha \leq \sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i).$$

Proof. For $e_1 = 0$: $\mathbf{E}_0 = \mathbf{R}$, and $\mathbf{E}_1 = \mathbf{W}$, and so

$$\alpha \stackrel{\triangle}{=} \sum_{i \in \mathbf{E}_1} (2\hat{\tau}_i - \delta_i) - \sum_{i \in \mathbf{E}_0} (2\hat{\tau}_i - \delta_i)$$

$$= \sum_{i \in \mathbf{W}} (2\hat{\tau}_i - \delta_i) - \sum_{i \in \mathbf{R}} (2\hat{\tau}_i - \delta_i)$$

Using Lemmas 3 and 5, we get

$$\alpha \geq \sum_{i \in \mathbf{W}} (\delta_i - 2\tau_i) - \sum_{i \in \mathbf{R}} (2\tau_i - \delta_i)$$

$$= \sum_{i \in \mathbf{R} \cup \mathbf{W}} (\delta_i - 2\tau_i)$$

$$= - \sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i)$$

For $e_1 = 1$: $\mathbf{E}_1 = \mathbf{R}$, and $\mathbf{E}_0 = \mathbf{W}$. Proceeding in a similar fashion as above, we get

$$\alpha \leq \sum_{i \in \mathbf{U}^c} (2\tau_i - \delta_i).$$

This completes the proof of Lemma 8. ❐

Define the quantity β as

$$\beta \stackrel{\Delta}{=} \sum_{i \in \mathbf{U}^c} \delta_i.$$

Then Corollary 6 and 7 represent constraints on β. Specifically, we can deduce that if $\mathbf{e}$ is an error pattern that satisfies $\text{wt}(\mathbf{e}) \leq \lfloor \delta/2 \rfloor$. Then

$$\beta < M - 1 \quad \Rightarrow \quad e_1 \neq 0 \tag{4.2}$$

$$\beta = M - 1 \quad \Rightarrow \quad \begin{array}{l} \text{Either } e_1 \neq 0 \text{ or} \\ \delta \text{ is even and } \text{wt}(\mathbf{e}) = \delta/2. \end{array} \tag{4.3}$$

Here, (4.2) follows from Corollary 6 and (4.3) from Corollary 7. Note that (4.3) implies that if $\beta = M - 1$ and $e_1 = 0$ then $\mathbf{e}$ is an uncorrectable error pattern. Further, from Lemma 6–8 it follows that

Case 1: for $\text{wt}(\mathbf{e}) \leq \lfloor (\delta - 1)/2 \rfloor$ it is true that

$$\begin{array}{rl} \text{for } e_1 = 1\text{:} & \alpha < M - 1 \\ \text{and for } e_1 = 0\text{:} & \alpha > M - 1. \end{array} \tag{4.4}$$

Case 2: for δ even and $\text{wt}(\mathbf{e}) = \delta/2$,

$$\begin{array}{rl} \text{for } e_1 = 1\text{:} & \alpha \leq M - 1 \\ \text{and for } e_1 = 0\text{:} & \alpha \geq M - 1. \end{array} \tag{4.5}$$

where (4.4) follows from Lemma 6 and 8, and (4.5) follows from Lemma 7 and 8.

Combining the constraints dictated by equations (4.2)– (4.5) the following decoding algorithm results.

(i) Receive the M (possibly corrupted) codewords $\mathbf{r}_1 = \mathbf{c}_1 + \mathbf{e}_1$, $\mathbf{r}_2 = \mathbf{c}_2 + \mathbf{e}_2, \ldots, \mathbf{r}_M = \mathbf{c}_M + \mathbf{e}_M$.

(ii) Decode the individual codes; i.e., for each $i = 1, 2, \ldots, M$, *evaluate* $(\hat{\eta}_i, \hat{\tau}_i)$.

(iii) Form the sets $\mathbf{E}_0, \mathbf{E}_1$ *and* $\mathbf{U}^c$.

(iv) Compute $\alpha = \sum_{i \in \mathbf{E}_1} (2\hat{\tau}_i - \delta_i) \quad - \sum_{i \in \mathbf{E}_0} (2\hat{\tau}_i - \delta_i)$.

(v) Compute $\beta = \sum_{i \in \mathbf{U}^c} \delta_i$.

(vi) If

$$\begin{cases} \beta < M-1 \text{ then} & \hat{\eta} = 1 \\ \beta = M-1 \text{ then if} & \begin{cases} \alpha < M-1 \text{ then} & \hat{\eta} = 1 \\ \alpha \geq M-1 \text{ then} & \hat{\eta} = ? \end{cases} \\ \beta > M-1 \text{ then if} & \begin{cases} \alpha < M-1 \text{ then} & \hat{\eta} = 1 \\ \alpha = M-1 \text{ then} & \hat{\eta} = ? \\ \alpha > M-1 \text{ then} & \hat{\eta} = 0 \end{cases} \end{cases}$$

A block-diagram representation of the algorithm appears in Fig. 3.

It is worth pointing out that the above algorithm is indeed equivalent to one-step majority logic decoding when each of the constituent codes is a simple parity check—i.e., $\delta_i = 2$ for all i. In such a case $\mathbf{E}_0 = \mathbf{U}^c$ contains the indices of the decoders that do not observe a parity violation, while $\mathbf{U}$ contains the indices of the decoders that *do*. Thus it follows that $\alpha = \beta = 2|\mathbf{E}_0| = 2(M - V)$, where V is the number of parity violations. Inserting these values into the above yields the following algorithm:

$$\text{If} \begin{cases} V < (M+1)/2, & \text{then } \hat{\eta} = 0; \\ V = (M+1)/2, & \text{then } \hat{\eta} = ?; \\ V > (M+1)/2, & \text{then } \hat{\eta} = 1. \end{cases}$$

This is exactly the one-step majority logic decoding algorithm.

The algorithm that we have developed for decoding of the common symbol has some very nice features:

(i) This algorithm is guaranteed to correct all errors of weight $\lfloor (\delta - 1)/2 \rfloor$ or less; and for δ even, it is guaranteed to either detect or correct all errors of weight $\delta/2$. However, there may be some errors of weight greater than $\lfloor \delta/2 \rfloor$ for which Lemma 6 may hold.

The common symbol can be corrected in the presence of all such errors. There may also be other such errors for which Lemma 7 (and not Lemma 6) holds. For all such errors, it is possible to either correctly decode the common symbol or if that is not possible, to declare an uncorrectable error has occurred. Therefore, the algorithm can possibly correct more errors than are dictated by Theorem 1.

(ii) The decoders $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_M$ can work in complete isolation from each other. All that each $\mathcal{D}_i$ needs is the n_i-tuple $\mathbf{r}_i$. Therefore, each decoder $\mathcal{D}_i$ can be implemented on a different processor, with no inter-processor communication required.

(iii) Individual decoders are required only to return their estimates $\hat{\eta}_i$, and $\hat{\tau}_i$ (rather than their estimate of the transmitted codeword). This may be used to simplify their structure.

(iv) The final decision $\hat{\eta}$ is a very simple threshold function of the decoder outputs. It requires no other information besides that which the decoders $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_M$ provide.

(v) There are no circuits in the data-dependency graph of the algorithm. Thus, the algorithm is also highly suitable for pipelined structures.

Thus the above algorithm solves the problem set forth in Section 3—that of correctly estimating a symbol common to M possibly corrupted codewords—and it does it in a way that is naturally distributed. That is, no "super-decoder" is necessary—just a (partial) decoder for each of the constituent codes and then a remarkably simple central unit that "pools" the partial results to arrive at the estimate of the common symbol.

V. Decoding Cyclic Codes

In this section, we demonstrate the applicability of the common-symbol decoding algorithm developed in Section 4 to decoding cyclic codes. First, we introduce the concept of a partition.

Definition 3 : *A partition* $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_M\}$ *on the coordinates* $\{1, 2, \ldots, n\}$ *is a collection whose elements satisfy the following properties:*

(i) $\mathbf{P}_i \subseteq \{1, 2, \ldots, n\} \quad \forall\ i = 1, 2, \ldots, M,$ *and*

(ii) $\mathbf{P}_i \cap \mathbf{P}_j = \{1\} \quad \forall\ i \neq j, 1 \leq i, j \leq M.$

The size (or cardinality) of the partition is $|\mathbf{P}| = M.$

Remark: In difference to the usual notion of a "partition", here we do not require that the union of the $\mathbf{P}_i$'s be equal to $\{1, 2, \ldots, n\}$, and (in addition) we require that "1" is common to every cell of the partition.

Assume a partition $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_M\}$ where $\mathbf{P}_i = \{p_{i,1} = 1, p_{i,2}, \ldots, p_{i,n_i}\}$, and $n_i = |\mathbf{P}_i|$. We will now use $\mathbf{P}$ to "break up" an n-tuple over $\mathbf{F}_q$ into M shorter vectors over $\mathbf{F}_q$ as follows: First, define a mapping

$$\mathcal{B}_{\mathbf{P}} : \mathbf{F}_q^n \longmapsto \mathbf{F}_q^{n_1} \times \mathbf{F}_q^{n_2} \times \cdots \times \mathbf{F}_q^{n_M}$$
$$\text{by} \quad \mathcal{B}_{\mathbf{P}}(\mathbf{x}) = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M), \tag{5.1}$$

where for any $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbf{F}_q^n$,

$$\mathbf{y}_i = (x_{p_{i,1}}, x_{p_{i,2}}, \ldots, x_{p_{i,n_i}}), \quad \forall\ i = 1, 2, \ldots, M.$$

Therefore $\mathcal{B}_{\mathbf{P}}(\mathbf{x}) = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M)$ essentially represents a "partitioning" of the symbols from the n-tuple $\mathbf{x}$ into $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M$ according to the partition $\mathbf{P}$; all symbols with positions that are included in $\mathbf{P}_1$ are used to form $\mathbf{y}_1$; those with positions in $\mathbf{P}_2$ form $\mathbf{y}_2$, and so on. Note that the first symbol of $\mathbf{x}$ is also the first symbol of each of $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M$.

Example 1: Let $\Theta = \{\Theta_1, \Theta_2\}$ be a partition of $\{1, 2, \ldots, 21\}$, with $\Theta_1 = \{1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$ and $\Theta_2 = \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$. Let $\mathbf{x} = (x_1, x_2, \ldots, x_{21}) \in \mathbf{F}_q^{21}$. Then $\mathcal{B}_{\Theta}(\mathbf{x}) = (\mathbf{y}_1, \mathbf{y}_2)$, where $\mathbf{y}_1 = \{x_1, x_2, x_4, x_6, x_8, x_{10}, x_{12}, x_{14}, x_{16}, x_{18}, x_{20}\}$, and $\mathbf{y}_2 = \{x_1, x_3, x_5, x_7, x_9, x_{11}, x_{13}, x_{15}, x_{17}, x_{19}, x_{21}\}$. Note that $y_{1,1} = y_{2,1} = x_1$. ❐

We shall use this idea of a partition to construct some codes. Let $\mathcal{C}$ be a $(n, k, d_{\min})$ cyclic block-code over $\mathbf{F}_q$, and let $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_M\}$ be a partition on $\{1, 2, \ldots, n\}$. Then for each i define $\mathcal{C}_{\mathbf{P}_i}$ to be the set of n_i-tuples obtained by deleting from every codeword in $\mathcal{C}$ every symbol *except* those whose coordinates are indicated by $\mathbf{P}_i$—i.e.,

$$\mathcal{C}_{\mathbf{P}_i} \stackrel{\Delta}{=} \{\mathbf{y}_i : (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M) = \mathcal{B}_{\mathbf{P}}(\mathbf{c}) \text{ for some } \mathbf{c} \in \mathcal{C}.\}.$$

Definition 4 : *The code $\mathcal{C}_{\mathbf{P}_i}$ as constructed above is termed the i'th constituent code of $\mathcal{C}$, for each $i = 1, 2, \ldots, M$.*

Since $\mathcal{C}$ is linear, each of its constituent codes is also linear for any partition.

An equivalent construction is as follows: Let $\mathcal{G} = (\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n)$ be the generator matrix of the code $\mathcal{C}$. Here $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n$ are the n column vectors of $\mathcal{G}$. Consider a partition $\mathbf{P}$ as described

above. Then the code $\mathcal{C}_{\mathbf{P}_i}$ is the row-space of the matrix

$$\mathcal{G}_{\mathbf{P}_i} \stackrel{\triangle}{=} (\mathbf{g}_{p_{i,1}}, \mathbf{g}_{p_{i,2}}, \ldots, \mathbf{g}_{p_{i,n_i}}).$$

Example 2: Consider the partition Θ of Example 1. Let $\mathcal{C}$ be the $(21, 8, 6)$ binary BCH code with generator polynomial $g(x) = 1 + x^2 + x^3 + x^5 + x^6 + x^7 + x^8 + x^{10} + x^{11} + x^{13}$. The corresponding generator matrix is

$$\mathcal{G} = \begin{pmatrix}
1&0&1&1&0&1&1&1&1&0&1&1&0&1&0&0&0&0&0&0&0\\
0&1&0&1&1&0&1&1&1&1&0&1&1&0&1&0&0&0&0&0&0\\
0&0&1&0&1&1&0&1&1&1&1&0&1&1&0&1&0&0&0&0&0\\
0&0&0&1&0&1&1&0&1&1&1&1&0&1&1&0&1&0&0&0&0\\
0&0&0&0&1&0&1&1&0&1&1&1&1&0&1&1&0&1&0&0&0\\
0&0&0&0&0&1&0&1&1&0&1&1&1&1&0&1&1&0&1&0&0\\
0&0&0&0&0&0&1&0&1&1&0&1&1&1&1&0&1&1&0&1&0\\
0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&1&0&1&1&0&1
\end{pmatrix}.$$

Then,

$$\mathcal{G}_{\Theta_1} = \begin{pmatrix}
1&0&1&1&1&0&1&1&0&0&0\\
0&1&1&0&1&1&1&0&0&0&0\\
0&0&0&1&1&1&0&1&1&0&0\\
0&0&1&1&0&1&1&1&0&0&0\\
0&0&0&0&1&1&1&0&1&1&0\\
0&0&0&1&1&0&1&1&1&0&0\\
0&0&0&0&0&1&1&1&0&1&1\\
0&0&0&0&1&1&0&1&1&1&0
\end{pmatrix}, \quad \text{and } \mathcal{G}_{\Theta_2} = \begin{pmatrix}
1&1&0&1&1&1&0&0&0&0&0\\
0&0&1&1&1&0&1&1&0&0&0\\
0&1&1&0&1&1&1&0&0&0&0\\
0&0&0&1&1&1&0&1&1&0&0\\
0&0&1&1&0&1&1&1&0&0&0\\
0&0&0&0&1&1&1&0&1&1&0\\
0&0&0&1&1&0&1&1&1&0&0\\
0&0&0&0&0&1&1&1&0&1&1
\end{pmatrix}.$$

Since the rows of $\mathcal{G}_{\Theta_1}$ and $\mathcal{G}_{\Theta_2}$ are linearly independent they can be used as generator matrices for the component codes. It can be shown that $\mathcal{C}_{\Theta_1}$ and $\mathcal{C}_{\Theta_2}$ are both $(11, 8, 2)$ codes. ❐

For the ith component-code $\mathcal{C}_{\mathbf{P}_i}$, define

$$\delta_{\mathbf{P}_i} \stackrel{\triangle}{=} \min_{\substack{\mathbf{x}=(x_1,x_2,\ldots,x_{n_i}),\\ \mathbf{y}=(y_1,y_2,\ldots,y_{n_i})\in\mathcal{C}_{\mathbf{P}_i}\\ x_1\neq y_1}} \mathrm{d_H}(\mathbf{x}, \mathbf{y}). \tag{5.2}$$

Note that as defined above, $\delta_{\mathbf{P}_i}$ is the symbol-protection distance of the first symbol of the ith component code $\mathcal{C}_{\mathbf{P}_i}$, for each $i = 1, 2, \ldots, M$.

Definition 5 : *The quantity* $\delta_{\mathbf{P}}$ *defined as*

$$\delta_{\mathbf{P}} \triangleq \left(\sum_{j=1}^{M} \delta_{\mathbf{P}_j} \right) - (M-1) \tag{5.3}$$

is called the common-symbol decoding distance of the code $\mathcal{C}$ *with respect to the partition* $\mathbf{P}$.

Example 3: For the codes developed in Example 2, it can be shown that $\delta_{\Theta_1} = \delta_{\Theta_2} = 3$. So the decoding distance of the partition Θ for the $(21, 8, 6)$ binary cyclic code of Example 2 is $\delta_{\Theta} = 5$.❐

We are now in a position to discuss the decoding of the linear cyclic block code $\mathcal{C}$. Let the codeword $\mathbf{c} = (c_1, c_2, \ldots, c_n) \in \mathcal{C}$ be transmitted, and an error $\mathbf{e} \in \mathbf{F}_q^n$ occurs. The received n-tuple is then $\mathbf{r} = \mathbf{c} + \mathbf{e}$.

Theorem 3. *It is possible to recover* $\mathbf{c}$ *from* $\mathbf{r}$ *provided*

$$\mathrm{wt}(\mathbf{e}) \leq \lfloor \frac{\delta_{\mathbf{P}} - 1}{2} \rfloor.$$

Further, if $\delta_{\mathbf{P}}$ *is even, and* $\mathrm{wt}(\mathbf{e}) = \delta_{\mathbf{P}}/2$, *then it is possible to either correctly decode* $\mathbf{c}$, *or to detect the presence of an uncorrectable error.*

Proof. From the construction above, we observe that the received n-tuple $\mathbf{r}$ can be considered as the concatenation of M codewords (one each from $\mathcal{C}_{\mathbf{P}_1}, \mathcal{C}_{\mathbf{P}_2}, \ldots, \mathcal{C}_{\mathbf{P}_M}$), with the first symbol in common. The common-symbol decoding theorem (Theorem 1) can therefore be applied, completing the proof. ❐

For linear, cyclic *binary* block codes, however, we now have a means of decoding the common-symbol, namely, the common-symbol decoding algorithm of Section 4. Hence, a decoding procedure for linear, cyclic binary block-codes is:

(i) Construct codes $\mathcal{C}_{\mathbf{P}_1}, \mathcal{C}_{\mathbf{P}_2}, \ldots, \mathcal{C}_{\mathbf{P}_M}$ according to (5.4). Compute $\delta_{\mathbf{P}_i}$, $\forall\ i = 1, 2, \ldots, M$, and $\delta_{\mathbf{P}}$ using (5.2) and (5.3) respectively. Further, let $\mathcal{D}_i$ be the decoder for the code $\mathcal{C}_{\mathbf{P}_i}$ as defined in Section 4.

(ii) Let $\mathbf{r}$ be the received n-tuple. Break up $\mathbf{r}$ into $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_M$ using $\mathbf{P}$ as follows:

$$(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_M) = \mathcal{B}_{\mathbf{P}}(\mathbf{r}).$$

(iii) Compute $\hat{\eta}$ (the estimate of the error in the common position) using the common-symbol decoding algorithm of Section 4.

(iv) If $\hat{\eta} \neq ?$ then $r_1 = r_1 - \hat{\eta}$, otherwise signal an uncorrectable error pattern, and abort processing.

(v) Repeat (ii)–(iv) for all $n-1$ cyclic shifts of $\mathbf{r}$. (Of course, if the code is systematic, it is sufficient to recover only the data symbols.)

A block diagram for the decoder appears in Fig. 4.

As was pointed out earlier, the common-symbol protection distance of a code is a function of the associated partition. If for some code $\mathcal{C}$ with minimum distance $d_{\min}$ there exists a partition $\mathbf{P}$ such that $d_{\min} = \delta_{\mathbf{P}}$, then the common-symbol decoding algorithm is optimal (i.e., it corrects up to $\lfloor (d_{\min} - 1)/2 \rfloor$ errors, and for $d_{\min}$ even, detects (or corrects) all occurrences of $d_{\min}/2$ errors). Such a partition is called an "optimal partition". Unfortunately, for a given code, an optimal partition does not always exist. The next best thing is to settle for a "maximal partition", i.e., one that maximizes the common-symbol decoding distance for a given code.

Additionally, it is favorable to choose a partition with the largest size. This has twin advantages: (i) the implementation complexity decreases with increasing partition size; and (ii) the degree of parallelism increases with increasing partition size. These advantages are seen by considering the largest possible partition – i.e., the partition corresponding to one-step majority logic decoding, where the partition defines many trivially simple constituent codes.

Given a code $\mathcal{C}$, suppose $\mathbf{P}^m$ is the maximal (i.e. – the best) partition of size m, for each $m = 2, 3, \ldots$. Suppose further, that the associated common-symbol decoding distance is $\delta_{\mathbf{P}^m}$. Then, it is shown in [16] that

$$d_{\min} \geq \delta_{\mathbf{P}^2} \geq \delta_{\mathbf{P}^3} \geq \cdots \geq d_{\text{mld}},$$

where $d_{\text{mld}} = J + 1$, and J is the largest number of orthogonal parity checks one may find for the code. Therefore, in going to a larger partition, the advantages of a simpler decoder implementation may be compromised by a loss in correcting capability. A more detailed consideration of these practical issues may be found in [16].

VI. Search Results

Table 1 lists some cyclic codes for which the common-symbol decoding algorithm corrects/detects more errors than one-step majority-logic decoding. The column entries are as follows:

Col. 1: Serial number –i.e., the code identifier.

Col. 2: $(n, k, d_{\min})$: The code is a linear, cyclic, block-code of block-length n, dimensionality k, and minimum-distance $d_{\min}$.

Col. 3: $g(x)$: the generator polynomial of the code.

Col. 4: d_{mld}: the one-step majority-logic decoding distance of the code – i.e., $d_{\text{mld}} = J + 1$, where J is the maximum number of orthogonal parity checks one can find for the code.

Col. 5: $\mathbf{P}_1$: the first component of a partition of size 2. The other component is given by $\mathbf{P}_2 = \{1, 2, \ldots, n\} - \mathbf{P}_1$. (Here, "$-$" is the set-difference operation.)

Col. 6,7: $(n_1, k_1, \delta_{\mathbf{P}_1})$ and $(n_2, k_2, \delta_{\mathbf{P}_2})$: the component codes are (n_1, k_1) and (n_2, k_2) linear, binary block-codes with $\delta_{\mathbf{P}_1}$ and $\delta_{\mathbf{P}_2}$, the respective symbol-protection distances for the common position.

Col. 8: $\delta_{\mathbf{P}}$: the decoding distance of the code with respect to the partition $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2\}$. Obviously, $\delta_{\mathbf{P}} = \delta_{\mathbf{P}_1} + \delta_{\mathbf{P}_2}$.

VII. Decoder Implementation

Consider again, the $(21, 8, 6)$ linear, cyclic, binary block-code of Example 2. This is a two error-correcting, three error-detecting code. The one-step majority-logic decoding distance of this code is $d_{\text{mld}} = 4$, and so using the one-step majority-logic decoding algorithm, we are able to correct all single errors and detect double errors.

Now, let us consider the common-symbol decoding approach. We partition the code according to the partition $\Theta = \{\Theta_1, \Theta_2\}$, as in Example 2, to get component codes $\mathcal{C}_{\Theta_1}$, and $\mathcal{C}_{\Theta_2}$, with generator matrices $\mathcal{G}_{\Theta_1}$, and $\mathcal{G}_{\Theta_2}$. Both $\mathcal{C}_{\Theta_1}$, and $\mathcal{C}_{\Theta_2}$ are $(11, 8, 2)$ codes. Their parity-check matrices are

$$\mathcal{H}_{\Theta_1} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

and

$$\mathcal{H}_{\Theta_2} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Observe that in each case the smallest number of linearly dependent columns including the first column is three; thus, $\delta_{\Theta_1} = \delta_{\Theta_2} = 3$.

Let $\mathbf{r} = (r_1, r_2, \ldots, r_{24})$ be the received 24-tuple. Then, the received vectors for the component

codes are $(\mathbf{r}_1, \mathbf{r}_2) = \mathcal{B}_{\Theta^2}^{-1}(\mathbf{r})$, where $\mathcal{B}$ is as defined in equation (5.1). Therefore, we have

$$\mathbf{r}_1 = (r_1, r_2, r_4, r_6, r_8, r_{10}, r_{12}, r_{14}, r_{16}, r_{18}, r_{20})$$
$$\text{and} \quad \mathbf{r}_2 = (r_1, r_3, r_5, r_7, r_9, r_{11}, r_{13}, r_{15}, r_{17}, r_{19}, r_{21}).$$

Next, we compute the error-syndromes

$$\mathbf{s}_1 \stackrel{\Delta}{=} \mathbf{r}_1 \mathcal{H}_{\mathbf{P}_1^2}^T = (s_{1,1}, s_{1,2}, s_{1,3})$$
$$\text{and} \quad \mathbf{s}_2 \stackrel{\Delta}{=} \mathbf{r}_2 \mathcal{H}_{\mathbf{P}_2^2}^T = (s_{2,1}, s_{2,2}, s_{2,3}),$$

which gives us the following syndrome generator equations:

$$\begin{aligned}
s_{1,1} &= r_1 \oplus r_2 \oplus r_8 \oplus r_{16} \\
s_{1,2} &= r_2 \oplus r_6 \oplus r_8 \oplus r_{10} \oplus r_{12} \oplus r_{14} \oplus r_{18} \\
s_{1,3} &= r_1 \oplus r_2 \oplus r_4 \oplus r_{10} \oplus r_{12} \oplus r_{14} \oplus r_{20} \\
\text{and} \quad s_{2,1} &= r_1 \oplus r_5 \oplus r_7 \oplus r_9 \oplus r_{11} \oplus r_{13} \oplus r_{17} \\
s_{2,2} &= r_3 \oplus r_7 \oplus r_9 \oplus r_{11} \oplus r_{13} \oplus r_{15} \oplus r_{19} \\
s_{2,3} &= r_1 \oplus r_7 \oplus r_{15} \oplus r_{21}.
\end{aligned}$$

(The $\oplus$ here, represents the *exclusive or* operation.) Next, we will compute the quantities $\hat{\eta}_1$, $\hat{\eta}_2$, $\hat{\tau}_1$, and $\hat{\tau}_2$ as a function of these syndromes. Upon performing a Karnaugh map minimization, the functional relationships are

$$\begin{aligned}
\hat{\eta}_1 &= s_{1,1}\overline{s_{1,2}}s_{1,3} \\
\hat{\tau}_1 &= s_{1,1} + s_{1,2} + s_{1,3} \\
\text{and} \quad \hat{\eta}_2 &= s_{2,1}\overline{s_{2,2}}s_{2,3} \\
\hat{\tau}_2 &= s_{2,1} + s_{2,2} + s_{2,3}.
\end{aligned}$$

Here, the $\hat{\eta}$'s and the $\hat{\tau}$'s are in keeping with the notation of Section 4. The action of the common-symbol decoding algorithm is given in Table 2. Another logic minimization yields

$$\hat{\eta} = \hat{\eta}_1\hat{\tau}_2 + \hat{\eta}_2\hat{\tau}_1.$$

Combining all the parts, a common-symbol decoder for correcting all occurrences of two or fewer errors in a codeword from the $(21, 8, 6)$ cyclic code is presented in Fig. 5. (In the figure, we have shown the circuit to decode one symbol. By cyclic shifts of the input, all other symbols can be corrected.) From the circuit, we see that the decoder is indeed simple.

VIII. Conclusions

This paper outlined a new approach for the distributed decoding of cyclic linear block codes. The approach is a generalization of one-step majority-logic decoding: It involves considering a symbol of a (potentially corrupted) codeword as the one symbol common to $M \geq 2$ less powerful codes; by decoding the less powerful codes and "pooling" the results in a simple fashion we have shown it is possible to reliably estimate that symbol. This approach yields a decoder structure that is suitable for parallel implementation.

The technique developed in this paper suggests a whole spectrum of algorithms for decoding a powerful block code – with one-step majority logic decoding at one extreme and iterative techniques that solve many errors simultaneously (such as Berlekamp-Massey) at the other. The research presented in this paper describes one approach to the "middle ground" between the extremes.

Bibliography

[1] Massey, J.L., *Threshold Decoding,* M.I.T. Press, Cambridge, Mass., 1963.

[2] Lin, S. and Markowsky, G., "On a Class of One-Step Majority-Logic Decodable Cyclic Codes", *IBM J. Res. Dev.,* **24, No. 1**, pp. 56–63, January 1980.

[3] Rudolph, L.D., "Geometric Configuration and Majority Logic Decodable Codes", *M.E.E. thesis,* University of Oklahoma, Norman, Okla., 1964.

[4] Weldon, E.J., Jr., "Difference-Set Cyclic Codes", *Bell Syst. Tech. J.,* **45**, pp. 1045–1055, September 1966.

[5] Bredeson, J.G. and Hakimi, S. L., "Decoding of Graph Theoretic Codes", *I.E.E.E. Trans. Inform. Theory,* **IT-13**, pp. 348–349, April 1967.

[6] Gore, W.C., "The Equivalence of L-Step Orthogonalization and a Reed Decoding Procedure", *I.E.E.E. Trans. Inform. Theory,* **IT-15**, pp. 184–186, January 1969.

[7] Chen, C.-L., "On Majority-Logic Decoding of Finite Geometry codes", *I.E.E.E. Trans. Inform. Theory,* **IT-17, No. 3**, pp. 332–336, May 1971.

[8] Kasami, T. and Lin, S., "On the Construction of a Class of Majority-Logic Decodable Codes", *I.E.E.E. Trans. Inform. Theory,* **IT-17, No. 5**, pp. 600–610, September 1971.

[9] Rudolph, L.D., "A Class of Majority Logic Decodable Codes", *I.E.E.E. Trans. Inform. Theory,* **IT-13**, pp. 305–307, April 1967.

[10] Rudolph, L.D., "Threshold Decoding of Cyclic Codes", *I.E.E.E. Trans. Inform. Theory,* **IT-15, No. 3**, pp. 414–418, May 1969.

[11] Gore, W. C. , "Generalized Threshold Decoding and the Reed-Solomon Codes", *I.E.E.E. Trans. Inform. Theory,* **IT-15, No. 1**, pp. 78–81, January 1969.

[12] Assmus, E.F., Goethals, J.-M., and Mattson, H. F., Jr., "Generalized t-Designs and Majority Decoding of Linear Codes", *Information and Control,* **32**, pp. 43–60, 1976.

[13] Gore, W.C., "Generalized Threshold Decoding of Linear Codes", *I.E.E.E. Trans. Inform. Theory,* **IT–15, No. 5**, pp. 590–592, September 1969.

[14] Fuja, T., Heegard, C., and Goodman, R., "Linear Sum Codes for Random Access Memories", *I.E.E.E. Trans. Computers,* **37, No. 9**, pp. 1030–1042, September 1988.

[15] Masnick, B., and Wolf, J., "On Linear Unequal Error Protection Codes", *I.E.E.E. Trans. Inform. Theory,* **IT–3, No. 4**, pp. 600–607, October 1967.

[16] Murad A.H., "Distributed Decoding of Block Codes Through a Generalization of Majority-Logic Decoding", *M.S. thesis,* University of Maryland, College Park, Mary., 1992.

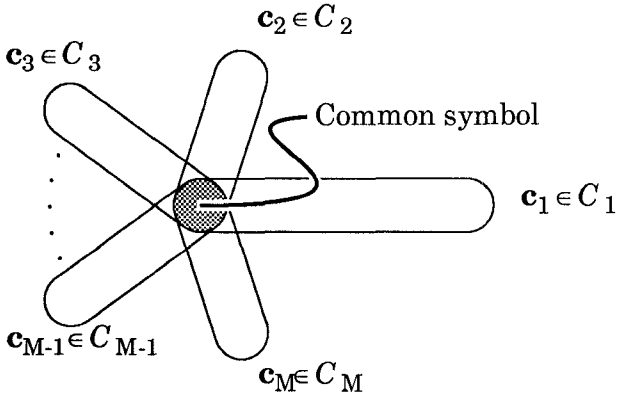


Fig 1. M codewords that share a common symbol.

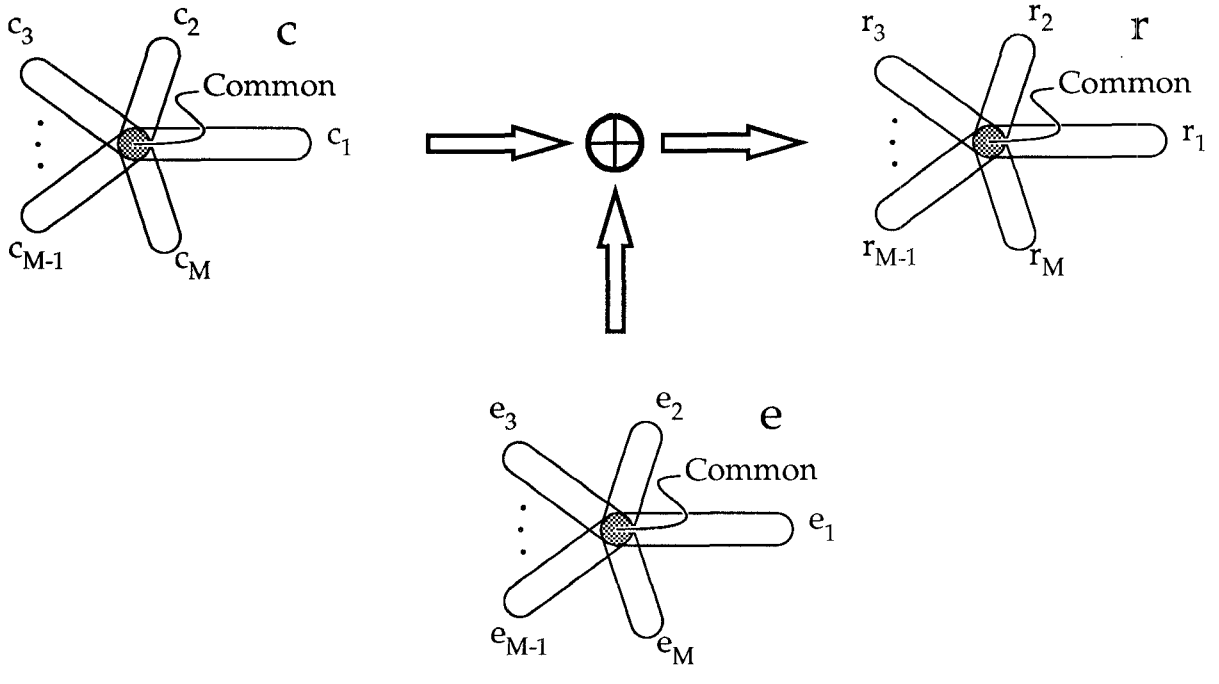


Fig 2. M codewords sharing a common symbol being transmitted over a noisy channel.

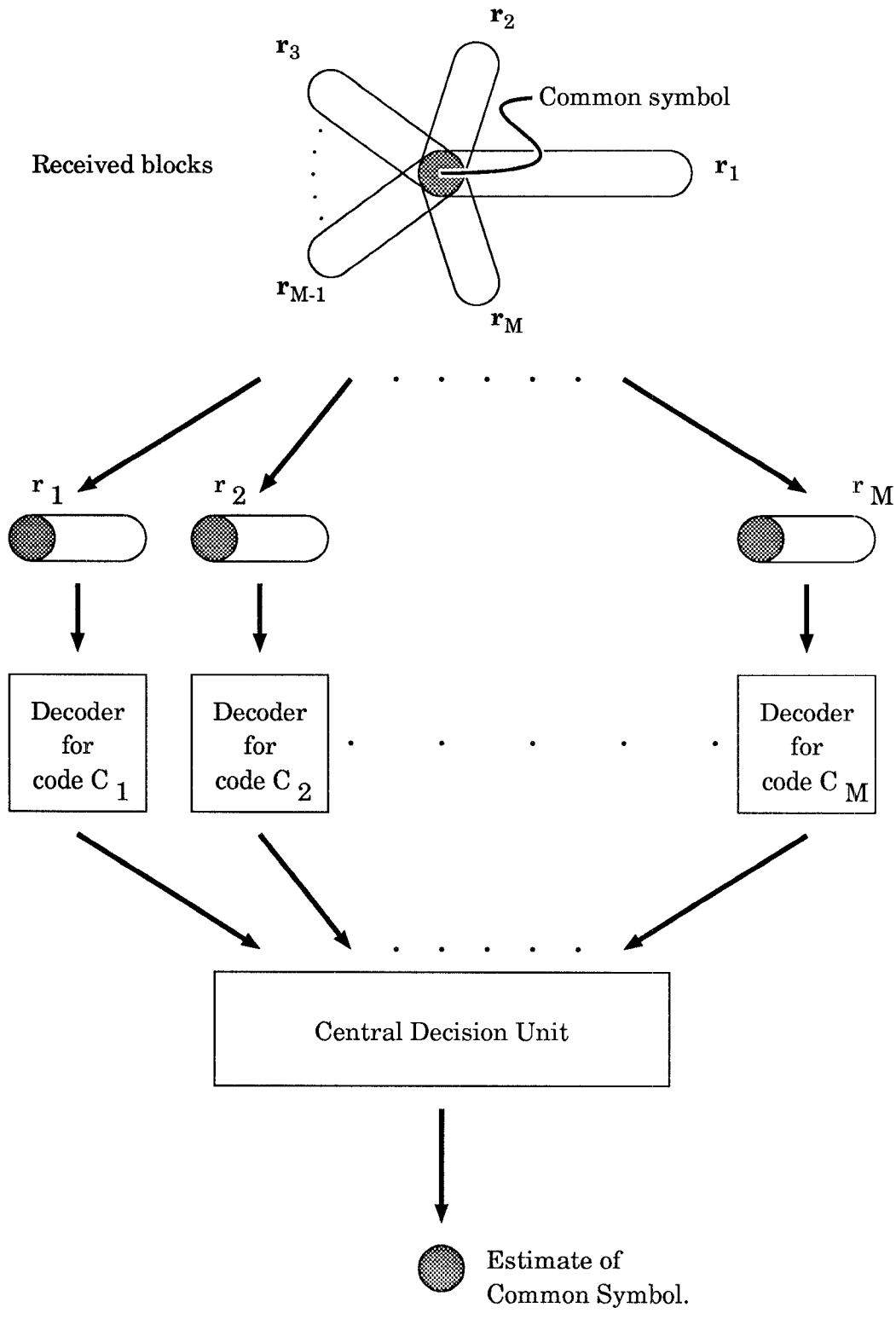


Fig 3. Block diagram of the common-symbol decoding algorithm.

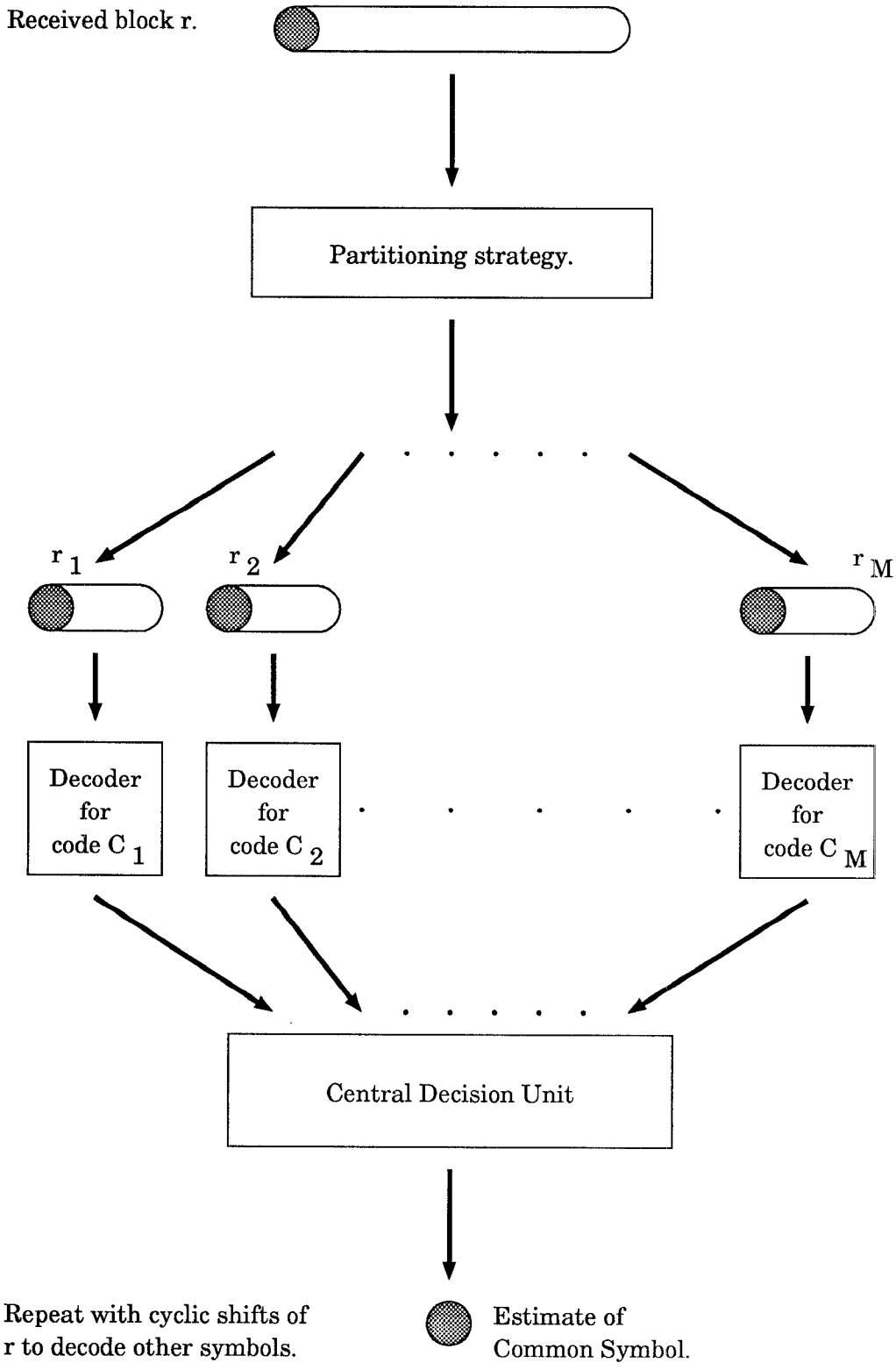


Fig 4. Common-symbol decoder for cyclic codes.

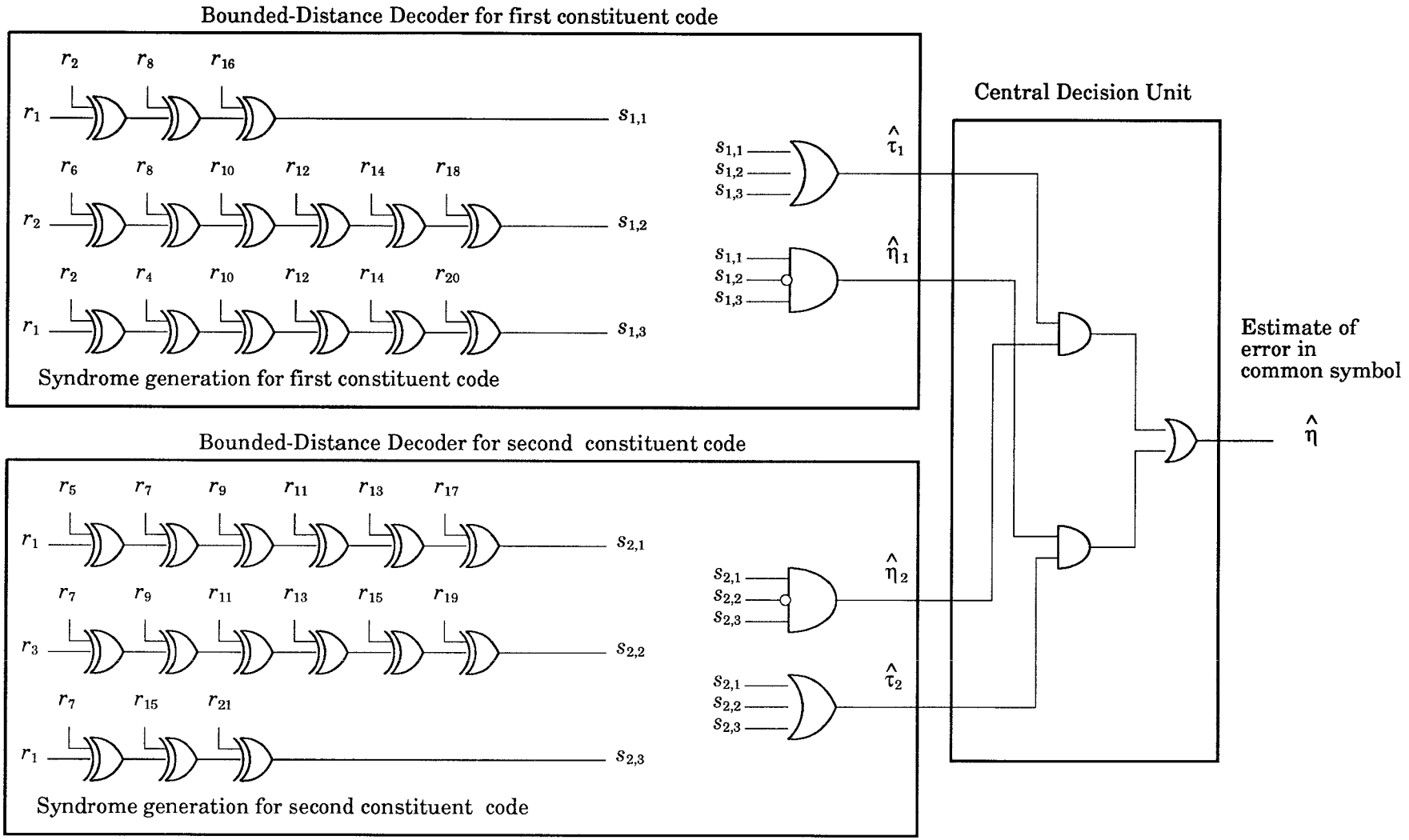


Fig 5. Common-symbol decoder for the (21,8,6) cyclic code.

Table 1. Some cyclic codes for which the common-symbol decoding algorithm corrects/detects more errors than one-step majority logic decoding.

S. No.	$(n, k, d_{\min})$	$g(x)$	d_{mld}	$\mathbf{P}_1$	$(n_1, k_1, \delta_{\mathbf{P}_1})$	$(n_2, k_2, \delta_{\mathbf{P}_2})$	$\delta_{\mathbf{P}}$
1	$(21, 5, 10)$	$1 + x^4 + x^5 + x^8$	8	$\{1, 3, 4, 7, 8, 10, 11, 13, 15, 16, 19\}^*$	$(11, 5, 5)$	$(11, 5, 6)$	10
		$+x^{10} + x^{12} + x^{13}$		$\{1, 2, 3, 6, 11, 12, 14\}^*$	$(7, 4, 4)$	$(15, 5, 7)$	10
		$+x^{14} + x^{15} + x^{16}$		$\{1, 2, 6, 12, 14\}^*$	$(5, 3, 3)$	$(17, 5, 8)$	10
				$\{1, 3, 11\}^*$	$(3, 2, 2)$	$(19, 5, 9)$	10
				$\{1, 3, 4, 7, 10, 11, 12, 13, 14, 16, 19)$	$(11, 5, 5)$	$(11, 5, 5)$	9
				$\{1, 3, 4, 7, 8, 11, 13, 15, 16, 19\}$	$(10, 5, 4)$	$(12, 5, 6)$	9
				$\{1, 4, 7, 8, 10, 13, 15, 16\}$	$(8, 5, 3)$	$(14, 5, 7)$	9
				$\{1, 3, 8, 11, 15\}$	$(5, 4, 2)$	$(17, 5, 8)$	9
2	$(21, 8, 6)$	$1 + x^2 + x^3 + x^5$	4	$\{1, 4, 5, 9, 13, 14, 17\}^*$	$(7, 6, 2)$	$(15, 8, 5)$	6
		$+x^6 + x^7 + x^8$		$\{1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$	$(11, 8, 3)$	$(11, 8, 3)$	5
		$+x^{10} + x^{11} + x^{13}$		$\{1, 3, 4, 6, 9, 12, 14, 16, 19\}$	$(9, 7, 2)$	$(13, 8, 4)$	5
3	$(21, 10, 5)^\dagger$	$1 + x^3 + x^4 + x^9 + x^{11}$	4	$\{1, 2, 6, 10, 14, 16\}^*$	$(6, 5, 2)$	$(16, 10, 4)$	5
4	$(21, 10, 5)$	$1 + x^2 + x^7 + x^8 + x^{11}$	4	$\{1, 6, 9, 13, 15, 16\}^*$	$(6, 5, 2)$	$(16, 10, 4)$	5
5	$(21, 10, 4)$	$1 + x + x^2 + x^3 + x^4$	3	$\{1, 3, 4, 6, 9, 10, 12, 15, 18, 20, 21\}^*$	$(11, 7, 2)$	$(11, 7, 3)$	4
		$+x^5 + x^9 + x^{10} + x^{11}$					
6	$(21, 10, 4)$	$1 + x + x^2 + x^6 + x^7$	3	$\{1, 3, 4, 6, 9, 12, 15, 16, 18, 20, 21\}^*$	$(11, 7, 2)$	$(11, 7, 3)$	4
		$+x^8 + x^9 + x^{10} + x^{11}$					

Continued on next page

Table 1—*Continued*

S. No.	$(n, k, d_{\min})$	$g(x)$	d_{mld}	$\mathbf{P}_1$	$(n_1, k_1, \delta_{\mathbf{P}_1})$	$(n_2, k_2, \delta_{\mathbf{P}_2})$	$\delta_{\mathbf{P}}$
7	(23, 11, 8)	$1 + x + x^2 + x^3$	4	{1, 3, 7, 10, 14, 15, 19}	(7, 6, 2)	(17, 11, 5)	6
		$+x^4 + x^7 + x^{10} + x^{12}$		{1, 2, 6, 9, 11, 13, 17, 20}	(8, 7, 2)	(16, 11, 4)	5
8	(23, 11, 8)	$1 + x^2 + x^5 + x^8$	4	{1, 5, 6, 10, 13, 17, 19}	(7, 6, 2)	(17, 11, 5)	6
		$+x^9 + x^{10} + x^{11} + x^{12}$		{1, 4, 8, 10, 12, 15, 19, 20}	(8, 7, 2)	(16, 11, 4)	5
9	(24, 9, 6)	$1 + x^2 + x^3 + x^5$	4	{1, 4, 6, 8, 11, 15, 18, 21}*	(8, 7, 2)	(17, 9, 5)	6
		$+x^6 + x^9 + x^{10}$		{1, 3, 4, 7, 8, 11, 13, 14, 17, 18, 21, 22}	(12, 9, 3)	(13, 9, 3)	5
		$+x^{12} + x^{13} + x^{15}$		{1, 3, 5, 7, 8, 11, 13, 15, 18, 20, 22}	(11, 9, 2)	(14, 9, 4)	5
10	(28, 6, 12)	$1 + x + x^2 + x^4 + x^6$	≤ 10	{1, 5, 7, 12, 16, 21, 23}*	(7, 5, 3)	(22, 6, 10)	12
		$+x^7 + x^9 + x^{10}$		{1, 4, 10, 19}*	(4, 3, 2)	(25, 6, 11)	12
		$+x^{11} + x^{12} + x^{13}$		{1, 2, 6, 9, 14, 17, 20, 24}	(8, 6, 3)	(21, 6, 9)	11
		$+x^{17} + x^{19} + x^{22}$		{1, 6, 9, 13, 17, 23}	(6, 5, 2)	(23, 6, 10)	11
11	(28, 6, 12)	$1 + x^3 + x^5 + x^9 + x^{10}$	≤ 10	{1, 5, 9, 10, 12, 17, 23}*	(7, 5, 3)	(22, 6, 10)	12
		$+x^{11} + x^{12} + x^{13}$		{1, 10, 16, 19}*	(4, 3, 2)	(25, 6, 11)	12
		$+x^{15} + x^{16} + x^{18}$		{1, 5, 6, 11, 13, 18, 19, 24}	(8, 6, 3)	(21, 6, 9)	11
		$+x^{20} + x^{21} + x^{22}$		{1, 5, 10, 12, 17, 23}	(6, 5, 2)	(23, 6, 10)	11
12	(31, 10, 10)	$1 + x^2 + x^3 + x^4 + x^5 + x^8$	≤ 9	{1, 2, 4, 8, 16}*	(5, 4, 2)	(27, 10, 9)	10
		$+x^{10} + x^{11} + x^{13} + x^{16}$					
		$+x^{17} + x^{18} + x^{19} + x^{21}$					

†Codes 3 and 4 are reciprocal codes; so are 5 and 6, 7 and 8, and 10 and 11.

*This is an optimal partition.

Table 2. Truth table for common-symbol decoding of the $(21, 8, 6)$ cyclic code.

$\hat{\eta}_1$	$\hat{\tau}_1$	$\hat{\eta}_2$	$\hat{\tau}_2$	α	β	$\hat{\eta}$
0	0	0	0	6	6	0
0	0	0	1	4	6	0
0	0	1	1	2	6	0
0	1	0	0	4	6	0
0	1	0	1	2	6	0
0	1	1	1	0	6	1
1	1	0	0	2	6	0
1	1	0	1	0	6	1
1	1	1	1	-2	6	1