*Proceedings of the Second Workshop in Error Correcting Codes*,
IBM Almaden Research Center, September 1987

# "MAPPING VECTOR DECODING"

JOHN T. COFFEY
RODNEY M.F. GOODMAN
MicroSystems Research Laboratory
California Institute of Technology

PATRICK G. FARRELL
Electrical Engineering Laboratories
University of Manchester

This work was supported in part by Pacific Bell.

## Mapping Vector Decoding

An algorithm with the features:

- Adding codewords
- Computing weights
- Making decisions based on values of one bit
- Full Minimum Distance Decoding
- Soft Decision Applications

Motivation: Produce a "simple" algorithm which takes advantage of the weight structure of the code to reduce complexity.

Sufficient to describe:

— Continued Division (Farrell)
— Permutation Decoding (MacWilliams)
— Information Set Decoding (Prange)
— Covering Polynomials (Kasami)
— Zero Neighbors Algorithm (Levitin & Hartmann)

Features:

(i) Add *deterministic* code vectors to received vector

(ii) Add code vectors to the operand word if weight reduction can be achieved

Description in terms of 'Standard' Array:
Rows are cosets; we have *increasing weight across a row.*

'Standard' Array for (6,3) Single-Error-Correcting Code

000000 100101 110010 001011 011100 010111 101110 111001
100000 000101 010010 001110 011001 101011 111100 110111
010000 100010 001100 000111 101001 110101 011011 111110
001000 000011 010100 100110 110001 101101 111010 011111
000100 100001 011000 010011 101010 110110 001111 111101
000010 110000 001001 010101 101100 100111 011110 111011
000001 100100 001010 010110 111000 110011 011101 101111
101000 000110 010001 001101 011010 100011 110100 111111

- Feature (i) adds codewords till we have 101 in the information positions (mapping to shaded words)
- Feature (ii) seeks to find a mapping from the operand word to a word of lower weight in the coset.

## Zero Neighbors Algorithm

— A word $\mathbf{r}$ is in the *domain* of a codeword ($\mathbf{r} \in \mathbf{D(c)}$) iff $d(\mathbf{r}, \mathbf{c}) \leq d(\mathbf{r}, \mathbf{c}_i)$ for all codewords $\mathbf{c}_i$.

— A word $\mathbf{r}$ is in the *domain frame* of a codeword ($\mathbf{r} \in \mathbf{B(c)}$) iff it is not in the domain of the codeword, but is distance one from a word which is in the domain of the codeword.

**Definition.** The set of *zero neighbors* is the minimum set of codewords for which every word in the domain frame of the zero codeword is in the domain of at least one of the zero neighbors.

Suppose $\mathbf{r} \notin \mathbf{D(0)}$
Form a chain of descendants $\mathbf{r}, \ldots, \mathbf{x}_{i+1}, \mathbf{x}_i, \ldots, 0$.
Let $\mathbf{x}_i$ be the first word in the chain in $\mathbf{D(0)}$.

$$\begin{aligned} &\Rightarrow \mathbf{x}_{i+1} \in \mathbf{B(0)} \\ &\Rightarrow \mathbf{x}_{i+1} \in \mathbf{D(n)} \text{ for some } \mathbf{n} \in \mathbf{N}_0 \\ \Rightarrow d(\mathbf{r}, \mathbf{n}) &\leq d(\mathbf{r}, \mathbf{x}_{i+1}) + d(\mathbf{x}_{i+1}, \mathbf{n}) \\ &< d(\mathbf{r}, \mathbf{x}_{i+1}) + d(\mathbf{x}_{i+1}, 0) \\ &= d(\mathbf{r}, 0) \end{aligned}$$

The zero neighbors comprise a set of *low weight* codewords.
For the Golay code, the zero neighbors are the minimum weight words.

Comparison with general model:

- ZNA makes extensive use of the second feature
- No use is made of first feature
- Minimum weight codewords are necessary

## Covering Polynomials

An extension of error trapping in which we guess parts of the error pattern in such a way that the unguessed part becomes trappable.

The set of covering polynomials is the minimum set such that there is at least one polynomial in the set which agrees with $E(x)$ or a cyclic shift of $E(x)$ in the $k$ information positions for all coset leaders $E(x)$.

*Example:* For the Golay code, every untrappable error pattern is a cyclic shift of a pattern which has only a single 1 in the first $k$ positions, in either the sixth or seventh bit. Thus the covering polynomials are $x^{16}$ and $x^{17}$.

Equivalently, we do the following:

- Compute syndrome
- Check if $s(x) + p_i(x)$ is a coset leader for all $p_i(x)$
- Compute 'shifted syndrome'
- Repeat second step using shifted $p_i(x)$'s

Comparison with general model:

- First feature is used (computing syndromes)
- Insist on *immediate* reduction in weight to coset leader

## Permutation Decoding

Apply a set of code-preserving permutations. Error pattern is permuted, and may then be trapped. For example, let $x \rightarrow Tx = x^2$. If first bit of $r(x^2)$ is a one, subtract $g(x)$. Shift left cyclically and repeat.

Equivalently, leave $r(x)$ as it is. If first bit is a one, subtract $T^{-1}g(x) = g(x^{(n+1)/2})$. Shift left cyclically *(n+1)/2* places and continue.

In general, for any permutation $\pi$, we subtract the codewords $\pi^{-1}g(x), \pi^{-1}xg(x), \ldots, \pi^{-1}x^{n-1}g(x)$ at the appropriate locations.

*Example*: For the Golay code, the permutations $x \rightarrow x, x \rightarrow x^2, x \rightarrow x^4$, and $x \rightarrow x^{12}$ are sufficient (with cyclic shifts) to trap all error patterns of weight $\leq 3$. Thus the codewords $g(x), g(x^{2^{-1}}), g(x^{4^{-1}})$, and $g(x^{12^{-1}})$, with their cyclic shifts, are sufficient to decode the Golay code.

Comparison with general model:

- Applies first feature extensively
- Does not use second feature

## Continued Division

```
10100110111) 01111011110111000000000
             001010001010111
             000000010001011
             0000000010110111
             00000000000010001011
             000000000000000000101101111
                          etc.
```

Motivation:

- Very simple implementation
- Possible 'data compression' features
  - Generate required words on-line
  - Generate required words only

*Questions:*

For how long do we divide before we get repetition?

Under what conditions do we get the coset leader?

Is it possible to decode by picking one dividing word?

Clearly, continued division by $g(x)$ corresponds to error trapping, with a few extra 'bonus' patterns detected.

Suppose we divide by $c(x) = i(x)g(x)$, where $c(x)|x^n - 1$. Then $c(x)$ is the generator of a cyclic subcode of $\mathbf{C}$. Continued division by $c(x)$ corresponds to error trapping in the subcode $\mathrm{C}'$. Error trapping detects error patterns of burst length $\leq n - k$. If the transmitted codeword is in the subcode, dividing by $c(x)$ will produce error patterns of burst length $\leq n - k + \deg i(x)$.

*Weight Reduction Mechanism*

We receive

$$\begin{aligned} r(x) &= C(x) + E(x) \\ &= a(x)c(x) + b(x) + E(x) \\ &= a(x)c(x) + E'(x) \end{aligned}$$

Three situations are possible when we begin division by $c(x)$:

- $r(x)$ is a coset leader in $C'$
- $r(x)$ is not a coset leader in $C'$ but no word of lower weight in the same coset has burst length $\leq n - k + \deg i(x)$
- $r(x)$ is not a coset leader in $C'$ and there are words of lower weight in the same coset with burst length $\leq n-k+\deg i(x)$

*Strategy:*

Take a number of words $c_i(x)$ which all divide $x^n - 1$. Divide $r(x)$ by $c_i(x)$ for one cycle after the syndrome. On getting a weight reduction, treat the new word as $r(x)$ and repeat. Finally, store the codewords necessary to map from the output of the first step to the coset leader.

The stored mapping codewords must include the minimum weight words not contained in any of the subcodes. It seems that most of the other required words will be of low weight.

*Division by a 'non-cyclic' word*

We divide by $c(x)$ and assume that deg $E(x) <$ deg $c(x)$. We produce the (subcode) syndrome and continue dividing. After one cycle, we will have added a word (a codeword in the main code) to the syndrome, and during continuing cycles we will have added other codewords.

We have

$$S_1(x) = x^n S_0(x) \bmod c(x)$$

and after $i$ cycles, $S_i(x) = x^{in} S_0(x) \bmod c(x)$

$$\Rightarrow S_0(x) + \beta_i(x) = x^{in} S_0(x) - \alpha(x)c(x)$$

$$\Rightarrow \beta_i(x) = (x^{in} - 1)S_0(x) - \alpha(x)c(x)$$

$$\Rightarrow \beta_i(x) = (x^{in} - 1)S_0(x) \bmod c(x)$$

We get repetition when $\beta_i(x) = 0$. If $(S_0(x), c(x)) = 1$, this means that $c(x) | x^{in} - 1$, and then

$$i = \frac{\text{ord } c(x)}{(\text{ord } c(x), n)}$$

*Example* In the (23,12) Golay Code, continuous division by $i(x)g(x)$, where $i(x)$ is any primitive polynomial of degree 10, will eventually produce the coset leader if neither the received word nor the error pattern is a multiple of $i(x)$.

*Proof:* We get the syndrome $S_{0,0}(x)$. The next $n-1$ words are $S_{0,i} = S_{0,0}(x) \bmod c(x)$, and all these must be relatively prime to $i(x)$ also. In at least one of these words (actually, most of them) deg $E(x) < n-2$. By our analysis above, the number of cycles taken to get a repetition is ord $c(x)/(\text{ord } c(x), n) = 1023/(1023, 23) = 1023$. But this is the number of codewords of degree $< c(x)$ which are not multiples of $i(x)$ so at some stage we have added $S_{i,j}(x) + E(x)$ to get $E(x)$.

*Corollary:* Continuous division of the 'real' syndrome of a received sequence by $(x^{10} + x^3 + 1)g(x)$ or $(x^{10} + x^7 + 1)g(x)$ will always produce the coset leader unless $E(x) = x^j i(x)$ for $1 \leq j \leq 12$.

Comparison of Methods:

| Method | Description | Word Additions |
|---|---|---|
| Permutation Decoding | Mapping vectors are various permutations of the generator | 92 (40) |
| Covering Polynomials | Mapping vectors comprise minimal set of differences between coset leaders and shifted syndromes | 69 |
| Zero Neighbors Algorithm | Mapping vectors comprise minimal covering of domain frame of zero codeword | 253 |
| Continued Division | Mapping vectors are differences between subcode coset leaders and coset leaders | N/A |
| Mapping Vector Decoding | Generalized method | $\leq 40$ |

**Conclusions:**

- Implementationally simple algorithm has been presented
- Sufficient to describe many other methods
- A new decoding algorithm (in Continued Division) has been outlined

**Future Research**

- Investigate Continued Division algorithm further
- Produce enhanced combined algorithms for Quadratic Residue codes
- Include soft decision applications