UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

DEPARTAMENTO DE COMUNICAÇÕES

# DECODIFICAÇÃO DE CÓDIGOS CONCATENADOS COM DECISÃO SUAVE E SAÍDA PONDERADA

Jônio Roberto de Hollanda Cavalcanti

Orientador : Dalton Soares Arantes †

Tese apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de *Mestre em Engenharia Elétrica.*



Julho 1992

*A*

*Manoel, Eliane,*

*Magda, Márcia e Roseane*

*Pelo carinho e apoio.*

# ÍNDICE

# RESUMO

Os códigos corretores de erros são hoje largamente utilizados em diversos sistemas de armazenamento, processamento ou transmissão de informação digitalizada. Entretanto, existem ainda hoje vários problemas de implementação física de códigos eficientes. A grande dificuldade reside na implementação de decodificadores para códigos longos ou para certos códigos multiníveis.

Uma técnica muito eficiente de utilização de códigos consiste na concatenação de dois ou mais códigos que em geral são decodificados isoladamente. Recentemente, todavia, têm surgido técnicas onde um decodificador recebe informação de confiabilidade do decodificador anterior. Dessa forma, ganhos adicionais podem ser conseguidos sem aumento significativo da complexidade total.

O objetivo deste trabalho é estudar estas técnicas recentes e propor alterações que podem levar a uma maior eficiência computacional. Para isso, são estudados os sistemas concatenados com um código interno e um código externo. Neste último, utiliza-se um código de Reed-Solomon onde uma decodificação com decisão suave pode fornecer ganhos da ordem de 2 dB em relação à decodificação com decisão abrupta.

# AGRADECIMENTOS

A realização de um trabalho nem sempre se resume ao seu aspecto material. É preciso considerar o ambiente social e as relações pessoais desenvolvidas ao longo do processo. Na realidade, pode-se pensar no aspecto humano como sendo a lembrança mais agradável que se pode ter ao final da etapa cumprida.

É neste sentido que gostaria de deixar meu agradecimento a todos que direta ou indiretamente contribuíram para a conclusão deste trabalho. Em especial ao Prof. Dalton pela confiança e pelas importantes sugestões. Aos professores da Faculdade de Engenharia Elétrica pelos cursos indispensáveis e finalmente a todos os amigos pela presença fundamental à continuidade do mesmo. Não podendo esquecer do trabalho datilográfico pacientemente realizado pelas secretárias da Faculdade de Engenharia Elétrica.

# NOTAÇÃO

| | |
|---|---|
| $V$ | - espaço vetorial formado por todas as ênuplas sobre um corpo de Galois. |
| $GF(p)$ | - corpo de Galois de p elementos. |
| $p$ | - número primo diferente da unidade. |
| $\mathcal{C}$ | - código de bloco linear. |
| $(n,k)$ | - parâmetros de um código $\mathcal{C}$. |
| $n$ | - comprimento de uma palavra código. |
| $k$ | - número de símbolos de informação em uma palavra código. |
| $\mathbf{c}$ | - palavra código. |
| $\mathbf{G}$ | - matriz geradora de um código $\mathcal{C}$. |
| $\mathcal{C}'$ | - código dual a $\mathcal{C}$. |
| $d_{min}$ | - distância mínima de um código $\mathcal{C}$. |
| $\mathbf{r}$ | - vetor recebido. |
| AWGN | - Aditive White Gaussian Noise. |
| $\omega(\mathbf{c})$ | - peso de Hamming de $\mathbf{c}$. |
| $\nu$ | - comprimento de memória de um código convolucional. |
| $K$ | - restrição de comprimento de um código convolucional. |
| $\mathbf{v}$ | - vetor saída. |
| $\mathbf{u}$ | - vetor entrada. |
| $d_{FREE}$ | - distância livre de um código convolucional. |
| $d_i$ | - distância mínima para um comprimento i de um código convolucional. |
| $\oplus$ | - soma módulo 2. |
| $(n_2,k_2)$ | - parâmetros de um código interno em um sistema de dois códigos concatenados. |
| $(n_1,k_1)$ | - parâmetros de um código externo em um sistema de dois códigos concatenados. |
| $q$ | - potência de um número primo. |
| $GF(p^m)$ | - corpo de Galois de extensão m, m inteiro não nulo, sobre GF(p). |
| $X^c$ | - monômio associado à palavra código $\mathbf{c}$. |
| $\mathbf{I}_{n-k}$ | - matriz identidade (n-k)x(n-k). |
| $\mathbf{I}_k$ | - matriz identidade kxk. |
| $\bar{\mathbf{r}}$ | - vetor obtido por decisão abrupta. |
| $S_t$ | - t-ésimo estado de um decodificador de Viterbi. |

# INTRODUÇÃO

Neste trabalho iniciamos com um revisão das técnicas de decodificacão de códigos corretores de erros, começando com os decodificadores que utilizam decodificação abrupta e concluindo com os que utilizam decodificaçao suave. A seguir analizamos a utilização destas técnicas de decodificação em estruturas que utilizam códigos concatenados. Neste momento surge o algoritmo de Battail que tenta solucionar os problemas decorrentes da utilização da decodificação suave nos estágios intermediários do decodificador para códigos concatenados.

Ocorre que a utilização do algoritmo de Battail necessita de uma forma não usual de representação dos códigos corretores de erros. São então estudadas as estruturas de treliça que facilitam a implementação do algoritmo. Para resolver o problema da transferência de confiabilidade entre os estágios de decodificação utilizamos o decodificador de Hartmann e Rudolph que possibilita esta tranferência.

Assim, da união do algoritmo de Battail com o algoritmo de Hartmann e Rudolph na decodificação de códigos concatenados, apresentamos o Algoritmo de Battail Modificado para decodificação de códigos concatenados (ABM).

Em seguida, apresentamos uma forma adicional do ABM que deve ser utilizada principalmente na decodificação de códigos longos. Finalmente, avaliamos o desempenho destes decodificadores, medido em termos de probabilidade de erro de bit, através de simulação em computador.

# CAPÍTULO 1

# CONCEITOS BÁSICOS

## 1.1. INTRODUÇÃO

Neste capítulo apresentaremos uma breve descrição dos conceitos a serem utilizados ao longo deste trabalho. Iniciaremos com os conceitos de códigos lineares sobre um corpo finito, códigos cíclicos, códigos de Reed-Solomon e códigos convolucionais. A seguir descreveremos os códigos lineares concatenados e finalmente a decodificação de códigos com saída ponderada como instrumento na decodificação de códigos lineares concatenados. Utilizaremos, para tanto, alguns conceitos de álgebra moderna [13] que serão descritos nos Apêndices.

## 1.2. CÓDIGOS DE BLOCO LINEARES

Considere $V_n$ o espaço vetorial formado por todas as ênuplas sobre um corpo de Galois GF(q), onde q é uma potência de um número primo. Assim, definimos $\mathcal{C}$ como sendo um código linear de parâmetros (n,k) sobre o corpo base GF(p) se e somente se $\mathcal{C} \subset V_n$ é um subespaço vetorial, onde k é a dimensão do código $\mathcal{C}$.

Um parâmetro importante a ser considerado é a distância mínima de Hamming de $\mathcal{C}$. Por definição,

$$d_{min} = MIN \left\{ d_H(c,c') : c, c' \in \mathcal{C} \; ; \; c \neq c' \right\} \tag{1.1}$$

onde $d_H(c,c')$, $c$, $c' \in \mathcal{C}$, é a distância de Hamming entre $c$ e $c'$, dada pelo número de símbolos distintos, posição a posição entre $c$ e $c'$, também chamado de peso de Hamming de $c-c'$.

Assim, para um código linear

$$d_{min} = MIN \left\{ \omega(c) \; , \; c \in \mathcal{C} \; ; \; c \neq 0 \right\} \tag{1.2}$$

onde $\omega(c)$ é o peso de Hamming de $c$ [13].

Este parâmetro é largamente utilizado quando da avaliação do desempenho de códigos lineares. Contudo, não é o único indicador para um bom código. Definimos ainda capacidade de correção de um código como sendo

$$t = \left\lfloor \frac{d - 1}{2} \right\rfloor, \tag{1.3}$$

onde $d$ é a distância mínima do código e $\lfloor \cdot \rfloor$ representa a parte inteira contida no argumento.

### 1.2.1. Descrição dos Códigos

Desde que um código é um conjunto de ênuplas, a sua enumeração seria suficiente para descrevê-lo. Contudo, isso se torna impraticável quando se está considerando códigos de parâmetros $q$ ou $k$ elevados, desde que $\|\mathcal{C}\| = q^k$. Recorremos, então, à característica linear do código para descrevê-lo através de matrizes.

Seja $\mathcal{C}$ um código linear sobre GF(q) de parâmetros (n,k). Seja uma matriz **G**, constituída de k linhas e n colunas, que tenha como conjunto de vetores linha uma base para $\mathcal{C}$, podemos mostrar que $\mathcal{C}$ é o espaço linha de **G**. Esta matriz é chamada matriz geradora de $\mathcal{C}$. Uma outra maneira de descrever $\mathcal{C}$ é através da matriz de paridade **H**. Essa matriz é definida como sendo a matriz geradora do código $\mathcal{C}'$ que é o código dual a $\mathcal{C}$. Por sua vez, $\mathcal{C}'$ é definido como sendo o complemento ortogonal do subespaço vetorial $\mathcal{C}$ em $V_n$. Desta forma, podemos descrever $\mathcal{C}$ tanto através da matriz **H** quanto da matriz **G**.

Um código $\mathcal{C}$ de parâmetros (n,k) está em forma sistemática se cada palavra código $\mathbf{c} \in \mathcal{C}$ possui em suas k primeiras posições os coeficientes da combinação linear das k linhas de **G**, matriz geradora de $\mathcal{C}$, que produziu **c**.

### 1.2.2. Códigos Cíclicos [13]

A estrutura dos códigos lineares é regida pelos conceitos de estruturas algébricas como corpos, classes laterais, etc. Ou seja, por uma álgebra elementar. Assim como os códigos cíclicos consistem de uma extensão dos códigos lineares, as estruturas que regem os códigos cíclicos correspondem a uma extensão daquelas utilizadas em códigos lineares.

Considere um código linear $\mathcal{C}$ de paramentos (n,k). Seja $\mathbf{v} \in \mathcal{C}$, $\mathbf{v} = (v_1,...,v_n)$, $v_i \in GF(p)$ $\forall i$. Chamamos de deslocamento cíclico de **v** o vetor $\mathbf{v}' = (v_n, v_1, v_2, ..., v_{n-1})$. O código $\mathcal{C}$ será dito cíclico se sempre que $\mathbf{v} \in \mathcal{C}$ então $\mathbf{v}' \in \mathcal{C}$.

Dizemos que um código cíclico sobre GF(q) é um ideal em um anel de polinômios sobre GF(q), assim como um código linear é um subespaço vetorial de

um espaço vetorial sobre GF(q). Podemos ainda dizer que um código cíclico sobre GF(q) é um subespaço vetorial cíclico de um espaço vetorial em GF(p).

Uma outra maneira de se caracterizar um código cíclico é através das raízes de um polinômio chamado polinômio gerador do código cíclico. Assim, seja {f(x)} uma classe de resíduos para um anel de polinômios sobre GF(q) e sejam $\alpha_1$, ...,$\alpha_r$ elementos de algum corpo de extensão de GF(q). Então {f(x)} está no código cíclico $\mathcal{C}$ se e somente se $\alpha_1$, ...,$\alpha_r$ forem raízes de f(x). Neste caso, g(x), o polinômio gerador do código cíclico $\mathcal{C}$, é dado por g(x)= MMC($m_1(x)$,....., $m_r(x)$), sendo $m_i(x)$ a função mínima de $\alpha_i$, i=1, ..., r. Em geral, supõe-se que não há raízes múltiplas no conjunto $\{\alpha_i\}$.

Dentre os códigos cíclicos, existe uma classe de códigos de interesse em nosso trabalho, os códigos de Reed-Solomon (RS). Os códigos RS são largamente utilizados em estruturas concatenadas por sua capacidade de correção de blocos de erros e por serem ótimos segundo o critério da distância mínima de Hamming, já que são códigos de máxima separação, ou seja, para os códigos de Reed-Solomon $d_{min} = n - k + 1$.

Se $\mathcal{C}$ é um código de Reed-Solomon, então {f(x)} está no código $\mathcal{C}$ se e somente se $\alpha$, $\alpha^2$, ..., $\alpha^{d-1}$ são raízes de f(x), onde $\alpha \in GF(q)$, f(x) é um polinômio sobre GF(q) e q uma potência de um número primo. Neste caso, $\mathcal{C}$ é um código de parâmetros n = ordem ($\alpha$) e d = n-k+1, onde $d = d_{min}$.

Existe ainda a possibilidade de se representar um código cíclico através de matrizes. Sendo o código cíclico um ideal para uma álgebra de polinômios sobre um dado corpo base, o ideal deve possuir um polinômio gerador que se for mônico (Apêndice B) deverá ser único. Este polinômio gerador pode ser usado na construção da matriz G geradora do código.

Seja $\mathcal{C}$ um código cíclico sobre GF(q) de parâmetros (n,k) e seja

g(x) seu polinômio gerador. Então, a matriz geradora de $\mathcal{C}$ é dada por

$$G = \begin{bmatrix} g_o & g_1 & \cdots & g_r & & & 0 \\ 0 & g_o & \cdots & g_{r-1} & g_r & & 0 \\ \vdots & \vdots & & \vdots & & & \\ 0 & 0 & \cdots & 0 & 0 & g_o \cdots & g_r \end{bmatrix}$$

sendo $g(x) = g_o + g_1 x + \ldots + g_r x^r$ e $r=k$.

### 1.2.3. Decodificação dos Códigos

Em geral, dado um vetor recebido $\mathbf{r}$ e um código $\mathcal{C}$ sobre um corpo GF(q), a decodificação por decisão abrupta é feita escolhendo-se $\mathbf{c} \in \mathcal{C}$ tal que

$$d_H(\bar{\mathbf{r}},\mathbf{c}) = \left\{ \omega(\mathbf{c}-\bar{\mathbf{r}}) \ , \ \mathbf{c} \in \mathcal{C} \right\} \qquad (1.4)$$

é mínima, onde $\bar{\mathbf{r}}$ é obtido através da decisão abrupta sobre cada componente do vetor recebido $\mathbf{r}$ e $\omega(\mathbf{c}-\bar{\mathbf{r}})$ é o peso de Hamming de $\mathbf{c}-\bar{\mathbf{r}}$.

Este tipo de decodificação é denominada decodificação de mínima distância por decisão abrupta. Observe que $\bar{\mathbf{r}}$ deve ter componentes em GF(q) e que a decodificação descrita acima é tal que a distância de Hamming seja minimizada. Outros tipos de métrica, tais como a distância Euclideana, podem também ser utilizadas.

Um exemplo para este tipo de decodificação é o arranjo-padrão onde o

espaço $V_n$ é decomposto em classes laterais (cosets) que têm como líderes as palavras de menor peso dentro da classe.

O critério de mínima distância é equivalente ao critério da Máxima Verossimilhança e minimiza a probabilidade de erro de palavra quando as palavras código são equiprováveis. Neste caso, ele coincide também com o critério do Máximo a Posteriori (MAP).

O decodificador para o canal gaussiano seleciona palavras código de maneira a minimizar a distância Euclidiana em vez da distância de Hamming. Neste caso, dizemos que o decodificador efetua uma decodificação por decisão suave, pois as amostras ou símbolos recebidos são quantizados em níveis (geralmente estes símbolos são quantizados em $2^s$ níveis com s = 3,4 ou 5).

Pode-se mostrar [4] que a decodificação suave apresenta um ganho assintótico da ordem de 3 dB sobre a decodificação abrupta, desde que se utilize uma modulação antipodal para transmissão dos símbolos e que esta transmissão ocorra em um canal gaussiano.

## 1.3. CÓDIGOS CONVOLUCIONAIS [14]

Os códigos convolucionais diferem dos códigos de bloco por possuírem um codificador com memória. Ou seja, as n saídas codificadas em um dado instante de tempo dependem não só das k entradas naquele instante de tempo mas também de $\nu$ entradas anteriores, isso para um código de parâmetros $(n,k,\nu)$.

Um código convolucional pode ser implementado através de um circuito linear discreto. Se os parâmetros do código são $(n,k,\nu)$, então n será o número

de saídas, k será o número de entradas e $\nu$ será o tamanho da memória utilizada pelo circuito linear discreto.

Consideremos um exemplo simples como ilustração. Seja um código convolucional binário de parâmetros (2,1,3) que pode ser implementado como mostra a Figura 1.1, onde o vetor **u** é a entrada e os vetores $\mathbf{v}^{(1)}$ e $\mathbf{v}^{(2)}$ formam a saída.

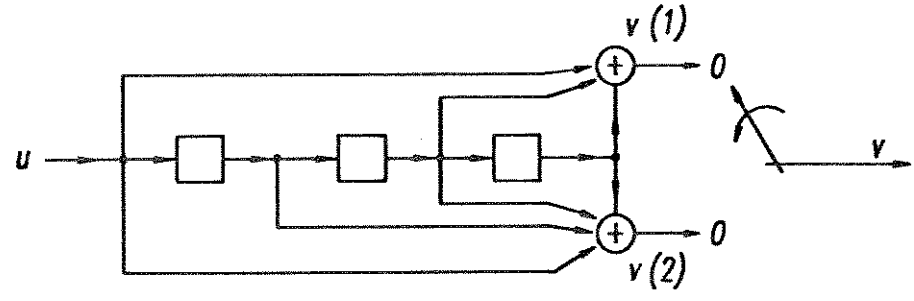


Fig. 1.1. Um codificador binário convolucional (2,1,3)

Considerando adição em GF(2), a seqüência de informação $\mathbf{u} = (u_o, u_1, \ldots)$ entra no codificador bit a bit e as seqüências codificadas $\mathbf{v}^{(1)} = (v_o^{(1)}, v_1^{(1)}, \ldots)$ e $\mathbf{v}^{(2)} = (v_o^{(2)}, v_1^{(2)}, \ldots)$ podem ser obtidas de

$$\mathbf{v}^{(1)} = \mathbf{u} * \mathbf{g}^{(1)} \tag{1.5}$$

$$\mathbf{v}^{(2)} = \mathbf{u} * \mathbf{g}^{(2)} \tag{1.6}$$

ou seja, cada $\mathbf{v}^{(i)}$ pode ser obtida através da convolução da entrada **u** com a respectiva resposta impulsional $\mathbf{g}^{(i)}$, i = 1,2. Para o codificador da Fig. 1.1, $\mathbf{g}^{(1)}$ = (1 0 1 1) e $\mathbf{g}^{(2)}$ = (1 1 1 1).

Resta ainda multiplexar as duas saídas $\mathbf{v}^{(1)}$ e $\mathbf{v}^{(2)}$ de maneira que $\mathbf{v} = (v_o^{(1)} v_o^{(2)}, v_1^{(1)} v_1^{(2)}, \ldots)$ seja a saída codificada.

### 1.3.1. Descrição dos Códigos Convolucionais

Pode-se definir uma matriz geradora para o código convolucional a partir de suas respostas impulsivas, também denominadas seqüências geradoras. No caso da Fig. 1.1, temos

$$\mathbf{G} = \begin{bmatrix} g_o^{(1)}g_o^{(2)} & g_1^{(1)}g_1^{(2)} & \cdots & g_\nu^{(1)}g_\nu^{(2)} & & \\ & g_o^{(1)}g_o^{(2)} & \cdots & g_{\nu-1}^{(1)}\,g_{\nu-1}^{(2)} & g_\nu^{(1)}g_\nu^{(2)} & \\ & & \ddots & & \ddots & \\ & & & \ddots & & \ddots \end{bmatrix}$$

onde os vazios correspondem a zeros. Podemos, então, escrever

$$\mathbf{v} = \mathbf{u}\,\mathbf{G} \qquad (1,7)$$

onde as operações são em GF(2). A matriz **G** é semi-infinita porque o comprimento de **u** é semi-infinito.

Para um código convolucional $(n,k,\nu)$ codificado por um circuito linear de k entradas, n saídas e de memória $\nu$, podemos definir a matriz geradora da seguinte forma

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_o & \mathbf{G}_1 & \mathbf{G}_2 & \cdots & \mathbf{G}_\nu & & \\ & \mathbf{G}_o & \mathbf{G}_1 & \cdots & \mathbf{G}_{\nu-1} & \mathbf{G}_\nu & \\ & & \ddots & & \ddots & & \ddots \end{bmatrix}$$

onde

$$G_1 = \begin{bmatrix} g_{1,1}^{(1)} & g_{1,1}^{(2)} & \cdots & g_{1,1}^{(n)} \\ g_{2,1}^{(1)} & g_{2,1}^{(2)} & \cdots & g_{2,1}^{(n)} \\ \vdots & \vdots & & \vdots \\ g_{k,1}^{(1)} & g_{k,1}^{(2)} & \cdots & g_{k,1}^{(n)} \end{bmatrix}$$

sendo $g_{i,j}^{(t)}$ a j-ésima componente da seqüência geradora associada à entrada i e saída t.

Com a definição de G torna-se mais fácil definir o parâmetro $\nu$ do código. Seja $k_i$ a memória associada à i-ésima entrada, com i = 1, ..., k. A memória do código é então definida por

$$\nu = \max_i k_i \tag{1.8}$$

A restrição de comprimento K (constraint length) de um código convolucional é definida por [14]

$$K \triangleq n(\nu + 1) \tag{1.9}$$

que pode ser interpretada como sendo o máximo número de saídas codificadas que podem ser afetadas por um bit de informação.

Assim como os códigos de bloco, os códigos convolucionais possuem critérios de distância importantes na análise de seu desempenho. Desta forma, definindo

$$[v]_i = (v_o^{(1)} v_o^{(2)} \cdots v_o^{(n)},\ v_1^{(1)} v_1^{(2)} \cdots v_1^{(n)},\ \ldots,\ v_i^{(1)} v_i^{(2)} \cdots v_i^{(n)})$$

$$[u]_i = (u_o^{(1)} u_o^{(2)} \cdots u_o^{(k)},\ u_1^{(1)} u_1^{(2)} \cdots u_1^{(k)},\ \ldots,\ u_i^{(1)} u_i^{(2)} \cdots u_i^{(k)})$$

a distância mínima do código é definida por

$$d_i \triangleq \min \{d_H([v']_i, [v'']_i) : [u']_o \neq [u'']_o\}$$
$$= \min \{w(v)_i : [u]_o \neq 0\} \qquad (1.10)$$

onde **v** é a palavra código associada a **u** e o índice i é um inteiro não nulo. Um outro parâmetro bastante utilizado na avaliação de códigos convolucionais é a distância livre $d_{FREE}$ do código, dada por

$$d_{FREE} = \lim_{i=\infty} d_i \qquad (1.11)$$

### 1.3.2. Decodificação de Códigos Convolucionais

Existem várias técnicas de decodificação de códigos convolucionais. Normalmente, cada uma dessas técnicas é descrita através de um algoritmo, que surge como uma forma de implementação da mesma. O algoritmo de Viterbi (AV) corresponde à técnica mais largamente empregada na decodificação dos códigos convolucionais, apesar de sua complexidade computacional crescer exponencialmente com o comprimento e número de estados do código. Em geral, para um K muito elevado utiliza-se decodificação seqüêncial [14], algoritmo de Fano por exemplo, em vez do algoritmo de Viterbi na decodificação de códigos convolucionais.

No sentido de se compreender o algoritmo de Viterbi, é preciso introduzir o conceito de diagrama de treliça. Este diagrama é obtido considerando-se cada combinação possível nas posições de memória do código e

das posições de entrada e obtendo-se transições de estados através do codificador. Por exemplo, considere o código (3,1,2) dado pelo codificador da Fig. 1.2, onde **u** corresponde ao vetor de entrada do codificador e $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ e $\mathbf{v}^{(3)}$ formam a saída codificada pelo codificador.

A treliça para este código e a tragetória correspondente a uma seqüência de informação de comprimento L = 5 é dada na Fig. 1.3.

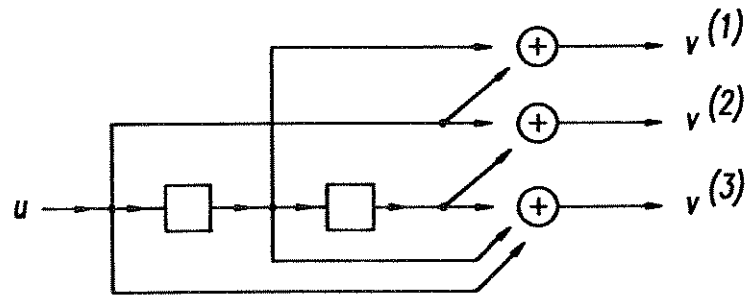


Fig. 1.2. Codificador para o código convolucional (3,1,2)

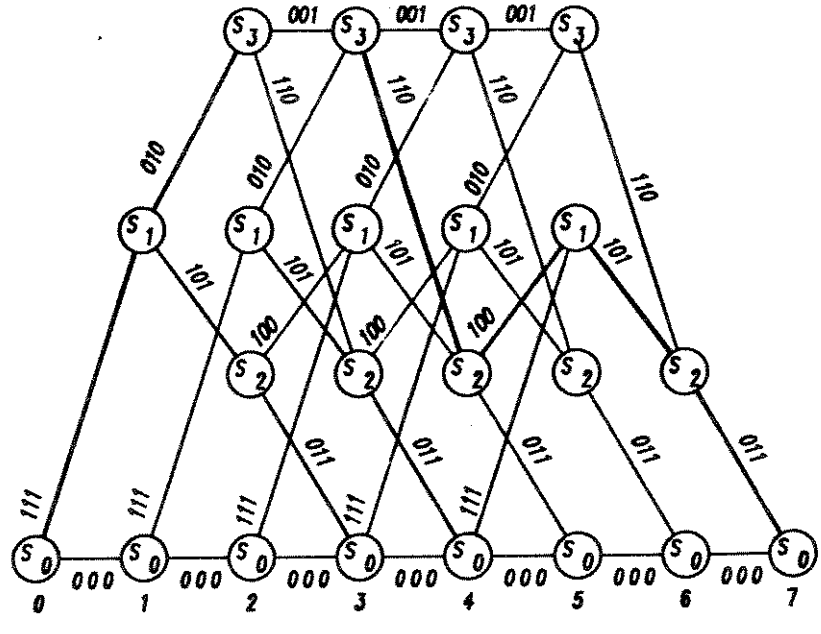


Fig. 1.3. Treliça para o código (3,1,2)

É interessante agora definir uma métrica para esta treliça. Se utilizarmos o critério da Máxima Verossimilhança uma métrica ótima seria

$$M([r/v]) = \sum_{i=o}^{L-1} M(r_i/v_i) \tag{1.12}$$

onde

$$M(r_i/v_i) \triangleq \log P(r_i/v_i) \tag{1.13}$$

sendo

$P(r_i/v_i) \triangleq$ probabilidade de ocorrência da componente $r_i$ dado o bit $v_i$

$r = (r_o, \ldots, r_{L-1})$ o vetor recebido

$v = (v_o, \ldots, v_{L-1})$ o caminho de comprimento L na treliça

O algoritmo de Viterbi produz o caminho de máxima Verossimilhança quando utiliza a expressão (1.12) como métrica e será apresentado no Capítulo 2. Existem ainda algoritmos sequenciais utilizados na decodificação dessa classe de códigos, como por exemplo o algoritmo de Fano e o algoritmo stack. Tais algoritmos surgem no sentido de superar algumas limitações do algoritmo de Viterbi, mas apresentam desvantagens que às vezes não podem ser toleradas. Os algoritmos seqüenciais não serão estudados neste trabalho.

## 1.4. CONCATENAÇÃO DE CÓDIGOS LINEARES

O conceito de concatenação de códigos foi inicialmente introduzido por Forney (1966) e surge como um recurso para se obter códigos longos a partir da combinação de códigos curtos. Um possível esquema consiste em codificar a saída da fonte através de um código sobre GF(q), onde q é uma potência de um número primo, denominado código externo $(n_1,k_1)$. A seguir, cada símbolo do código externo, que são elementos de GF(q), é codificado através de um código $(n_2,k_2)$ denominado código interno.

Após a saída do canal, o decodificador interno recupera cada um dos símbolos até formar uma seqüência de $n_1$ símbolos sobre GF(q). A partir desta seqüência o decodificador externo deve recuper a informação transmitida.

A codificação concatenada pode ser representada como na figura abaixo

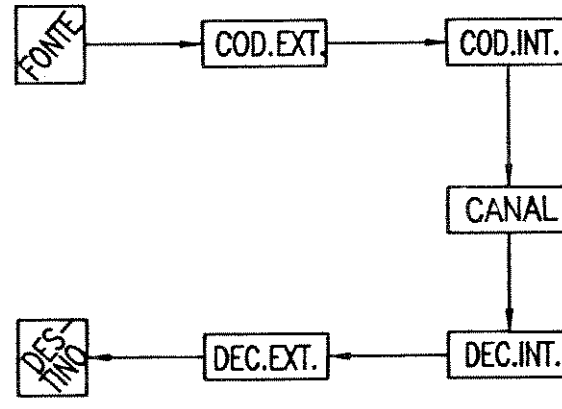


Fig. 1.4 - Esquema de códigos concatenados

O conjunto formado pelo codificador interno (cod. int.), canal e decodificador interno (dec. int.), figura 1.4, é denominado supercanal, que pode ser considerado como sendo um canal digital. Uma vez que o decodificador

externo corrige surtos de erros de tamanho limitado, haverá problemas caso ocorram surtos de erros muito longos na saída do supercanal. Quando isto ocorre, utiliza-se um dispositivo denominado par embaralhador/desembaralhador que tem por função promover a independência entre os símbolos adjacentes decodificados. Além disso, este dispositivo pode também ser utilizado para promover o casamento entre os codificadores interno e externo, quando não ocorre uma atuação direta do código interno sobre os símbolos do código externo, ou seja, quando $m \neq k_2$ onde m é um inteiro não nulo tal que o código externo seja definido sobre $GF(p^m)$. Assim, quando $m = k_2$ dizemos que os códigos interno e externo estão casados [14].

Dentre os esquemas de decodificação conhecidos, existe uma classe de decodificadores que utiliza uma informação sobre a qualidade do símbolo decodificado, não só o seu valor discreto. Como exemplo, temos o Algoritmo de Viterbi que atua sobre os valores reais dos símbolos recebidos ou sobre valores suavemente quantizados. Tal classe de decodificadores efetua uma decodificação com decisão suave.

Observe, na figura 1.4, que o decodificador interno recebe os símbolos transmitidos a partir da saída do canal. Desta forma, o decodificador interno recebe toda a informação disponível sobre cada símbolo recebido, podendo sempre efetuar uma decodificação suave. Por sua vez, o decodificador externo recebe os símbolos decodificados pelo decodificador interno. Conseqüentemente, uma decodificação suave torna-se impossível caso o decodificador interno não entregue também alguma informação a respeito da decodificação de cada símbolo. Quando o decodificador interno entrega ao decodificador externo alguma informação a respeito da decodificação de cada símbolo, não só o resultado da mesma, a decodificação é dita decodificação com

saída ponderada. Caso contrário, dizemos que a decodificação é abrupta.

A decodificação dos símbolos pelo decodificador interno deve ser tal que a atuação do mesmo entregue ao decodificador externo a informação a respeito do símbolo, ou parte dela, recebida do canal, por exemplo, o decodificador interno pode informar qual a chance de cada um dos símbolos ter sido codificado corretamente através de uma medida de probabilidade. Quanto "maior" for a quantidade de informação entregue pelo decodificador interno, melhor será o processo de decodificação suave efetuado pelo decodificador externo, no caso podemos dizer que quanto mais precisa for a medida de probabilidade entregue pelo decodificador interno melhor será o processo de decodificação com saída ponderada efetuado pelo decodificador externo.

Caso o decodificador interno produza símbolos independentes, podemos afirmar que o ganho assintótico da decodificação suave com saída ponderada sobre a decodificação abrupta é dado por [4]

$$G = 10 \log \left[ \frac{d}{t + 1} \right] \qquad (1.14)$$

onde d é a distância mínima e t é a capacidade de correção do código externo. Logo, um código externo que atue sobre símbolos transmitidos atarvés de um canal AWGN, com modulação antipodal e independentemente decodificados pelo código interno poderá fornecer um ganho assintótico de aproximadamente 3dB sobre a decodificação abrupta, caso seja decodificado com saída ponderada [4].

# CAPÍTULO 2

# ALGORÍTMOS DE DECODIFICAÇÃO DE CÓDIGOS CONCATENADOS

## 2.1. INTRODUÇÃO

A concatenação de códigos é uma técnica introduzida por Forney (1966) como uma maneira de se obter uma taxa de erro de decodificação exponencialmente decrescente, em função do comprimento do código ou tamanho da memória utilizada, sem incorrer em um crescimento exponencial da complexidade do decodificador [8]. Battail [7], [11] e Hagenauer et al [3] apresentaram resultados sobre a decodificação de códigos concatenados com transmissão de informação adicional entre estágios de decodificação, ou seja, decodificação de códigos concatenados com saída ponderada (ver cap. 1, item 1.4). Tais resultados, juntamente com conceitos já conhecidos na literatura, foram utilizados no sentido de estender o que já fôra apresentado em Hagenauer [3] e Battail [7]. Para tanto, foi necessária uma representação em treliça para códigos de bloco, tal como desenvolvido em Battail [1].

A introdução da decisão suave em um esquema de decodificação usual pode acarretar um ganho da ordem de 2 a 3 dB [4]. É este o objetivo dos esquemas que ora estudamos, lembrando que estamos considerando o canal linear com ruído gaussiano, aditivo e branco (AWGN) e modulação antipodal.

No que diz respeito à complexidade. não nos parece necessário

discuti-la uma vêz que, no esquema ora estudado, o decodificador interno utiliza técnicas de decodificação amplamente conhecidas (decodificação palavra a palavra via algoritmo de Hartmann e Rudolph [13] e decodificação por correlação completa [13]) no que diz respeito à complexidade e ao desempenho. Além disso, como poderá ser observado a seguir, o esquema estudado é tal que não se apresenta iteração entre os processos de decodificação interno e externo a ponto de um processo interferir no outro. Ou seja, o decodificador externo utiliza a informação entregue pelo decodificador interno sem interferir no processo utilizado para obtê-la.

## 2.2. DIAGRAMA DE TRELIÇA PARA CÓDIGOS DE BLOCO [1]

Um diagrama de treliça para um código de bloco nada mais é senão uma representação compacta para um código de bloco. Através desta representação é possível implementar algoritmos decodificação que visam à obtenção de um caminho ótimo segundo determinado critério.

### 2.2.1. Alfabeto

Serão considerados códigos definidos sobre corpos finitos GF(q), onde q é uma potência de número primo positivo.

### 2.2.2. Representação em Treliça para uma Palavra Código

Sejam $\mathcal{C}$ um código de bloco linear de parâmetros (n,k) sobre um corpo base GF(q) e $\mathbf{c} = [c_1, c_2, \ldots, c_n]$, tal que, $\mathbf{c} \in \mathcal{C}$.

O monômio associado a **c** é dado por

$$X^{\mathbf{c}} \triangleq X_1^{c_1} X_2^{c_2} \ldots X_n^{c_n}$$

que é função de $c_1, \ldots, c_n$. Assumindo que o código está em sua forma sistemática, podemos afirmar que os primeiros k símbolos de **c** corresponderão aos símbolos de informação e os n-k últimos aos símbolos de redundância.

Ocorre que para representação em treliça de cada palavra código de $\mathcal{C}$ só serão necessários os $X_i^{c_i}$ que correspondam aos símbolos reduntantes, ou seja, os símbolos que não sejam de informação. Como se trata de um código em sua forma sistemática só serão necessários os $X_i^{c_i}$ com $i = k+1, \ldots, n$. Finalmente, definimos

$$S = [0 \;\vdots\; I_{n-k}] \quad , \qquad \text{onde } I_{n-k}$$

que é uma matriz nxn e que será utilizada para obter-se o fator de $X^{\mathbf{c}}$ necessário á representação de **c** na treliça. Assim definimos,

$$X_s^{\mathbf{c}S^T} = X_{\mathbf{c}}^{\mathbf{c}S^T} \tag{2.3}$$

onde s indica que só os coeficientes de **c** correspodentes aos símbolos de redundância serão considerados.

### 2.2.3. Representação do Código

Para representação do código $\mathcal{C}$, necessitaremos de algumas definições adicionais. Seja inicialmente

$$T_C = \left[\begin{array}{c:c} I_k & \\ & -H^T \\ 0 & \end{array}\right] \qquad (2.1)$$

uma matriz que será denominada matriz $T_C$ do código $\mathcal{C}$, onde **H** é a matriz de paridade do código, que assumimos estar na sua forma sistemática.

A partir de $T_C$ podemos definir o polinômio descritor da treliça de acordo com a expressão

$$\mathcal{C}_{c,i} = \prod_{j=o}^{i} \left[ \sum_{m \in GF(q)} (y_j \, X_s^{t_j S^T})^m \right] \quad , \; i = 0, 1, \ldots, n \qquad (2.2)$$

onde $t_j$ é a j-ésima linha da matriz $T_C$ e $y_j$ é uma variável que corresponde aos ganhos de ramo em uma transição de estados e cujo significado ficará mais claro adiante. Observe que o produto $t_j S^T$ produz os símbolos de redundância de **c**.

A construção da treliça é feita a partir do polinômio descritor como segue [1]

1 - Construa a treliça colocando na vertical todos os possíveis símbolos em grupos de n-k símbolos e na horizontal todos os níveis da profundidade 0 até a profundidade n. A coluna vertical será denominada estados da treliça e a linha horizontal os níveis de nós da treliça.

Esta treliça terá o mesmo formato da treliça que é construída para

os códigos convolucionais, com a diferença de que agora trata-se de uma treliça de comprimento finito, ou seja, tem começo e fim. Além disso, é possível decodificar-se à medida que se vai avançando em direção ao final, através da expressão (2.2), sem a necessidade de se armazenar toda a treliça. Em um dado ponto, a expressão (2.2) especifica o próximo estado a ser atingido e a decodificação funciona como no caso de códigos convolucionais. Pode-se, para o caso de códigos curtos, armazenar toda a treliça e utilizar a sua estrutura completa para a decodificação.

2 - Expanda o produtório, do nível de nó i=0 até o nível de nó i=n.

O estado atual é determinado pelo coeficiente de $X_s$ em $\mathcal{C}_{c,i}$, para um dado i, e os próximos estados são determinados pelos coeficientes de $X_s$ no produto de $\mathcal{C}_{c,i}$ por $\sum_{m\in GF(q)} \left( y_{i+1} \; X_s^{t_{i+1} S^T} \right)^m$. Isto pode ser melhor entendido utilizando-se o raciocínio a seguir.

Considere um termo genérico do primeiro fator da expressão (2.1) definida acima, ou seja, $y_1^m \; X_{k+1}^{v_1} \; X_{k+2}^{v_2} \; \ldots \; X_n^{v_{n-k}}$ onde $\mathbf{v} = (v_1, \ldots, v_{n-k}) = m t_1 S^T$ com $t_1$, m e S já definidos anteriormente. O produto efetuado por este termo e um termo genérico do segundo fator da expressão (2.1) será dado por $y_1^m y_2^{m'} X_{k+1}^{v_1+v'_1} X_{k+2}^{v_2+v'_2} \ldots X_n^{v_{n-k}+v'_{n-k}}$, onde $\mathbf{v'} = (v'_1,\ldots,v'_{n-k}) = m' t_2 S^T$, com $t_2$, m' e S já definidos acima e sendo a soma a mesma operação de soma do corpo GF(p). Este termo indica o símbolo associado à transição do estado **0** para o estado **v**, através da potência de $y_1$, e o símbolo associado à transição do estado **v** para o estado **v** + **v'**, através do símbolo m'. Se efetuarmos o produto, como definido acima, de cada um dos q termos do primeiro fator da expressão (2.1) com cada um dos q termos do segundo fator da expressão (2.1) obteremos $q^2$ termos da forma $y_1^m y_2^{m'} X_{k+1}^{v_1+v'_1} X_{k+2}^{v_2+v'_2} \ldots X_n^{v_{n-k}+v'_{n-k}}$, o que corresponde a avançar dois níveis

na treliça, com os símbolos associados a cada transição determinados pelas potências de $y_1$ e $y_2$.

3 - Repita as operações definidas no item anterior, considerando agora os produtos entre cada um dos $q^2$ termos obtidos e cada um dos q termos do terceiro fator da expressão (2.1). Esta operação prossegue até que todos os $q^{n-k}$ estados sejam atingidos.

4 - A partir de cada um dos $q^{n-k}$ estados atingidos, considere os produtos de cada um destes $q^{n-k}$ termos obtidos e os termos restantes da expressão (2.2).

Estes produtos representarão todo o espaço sobre o qual o código está "mergulhado", ou seja, representam o espaço $V_n$. Desta representação, somente os caminhos que retornam para o estado nulo, após efetuados todos os produtos restantes da expressão (2.1), corresponderão a palavras código do código a ser representado [1].

O exemplo (2.1) ilustra um código (4,2) representado através desta estrutura de treliça, o que deixará mais clara a maneira de se construi-la [1].

Exemplo 2.1

Considere representação para o corpo de Galois de 4 elementos, GF(4), dada por $\{0,1,\alpha,\beta\}$, onde $\alpha + \alpha = 0$, $\beta + \beta = 0$ e $\alpha + \beta = 1$ e seja $\mathcal{C}$ um código (4,2) sobre GF(4) cuja matriz H é dada por

$$H^T = \left[\begin{array}{c:c} \beta & \alpha \\ \alpha & \beta \\ 1 & 0 \\ 0 & 1 \end{array}\right] \qquad (2.4)$$

Então, a matriz $T_C$ para este código é dada por

$$T_C = \begin{bmatrix} 1 & 0 & \beta & \alpha \\ 0 & 1 & \alpha & \beta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (2.5)$$

Substituindo a matriz $T_C$ no polinômio descritor do código obtemos a seguinte expressão

$$\begin{aligned} \mathcal{C} = {} & (1 + y_1 x_3^{\beta} x_4^{\alpha} + y_1^{\alpha} x_3 x_4^{\beta} + y_1^{\beta} x_3^{\alpha} x_4) \; x \\ & (1 + y_2 x_3^{\alpha} x_4^{\beta} + y_2^{\alpha} x_3^{\beta} x_4 + y_2^{\beta} x_3 x_4^{\alpha}) \; x \\ & (1 + y_3 x_3 + y_3^{\alpha} x_3^{\alpha} + y_3^{\beta} x_3^{\beta}) \; x \\ & (1 + y_4 x_4 + y_4^{\alpha} x_4^{\alpha} + y_4^{\beta} x_4^{\beta}). \end{aligned} \qquad (2.6)$$

Observe que os elementos de GF(4) são tais que $\alpha = -\alpha$ e $\beta = -\beta$. Para obter a treliça deste código utilizaremos as regras previamente apresentadas. Com i = 0 iniciaremos com o estado 0 (ver fig. 1.2), isto corresponde a considerar o primeiro termo do primeiro fator da expressão (2.6) (1). Se efetuarmos o produto destetermo com o primeiro termo do segundo fator de (2.6) (1) obteremos o valor 1, isto corresponde a avançar dois níveis na treliça (fig. 2.1). De outra forma, dizemos que partimos do nó representado pelo termo $y_1^0 x_3^0 x_4^0$, efetuamos o produto deste último termo por $y_2^0 x_3^0 x_4^0$ e obtivemos $y_1^0 y_2^0 x_3^0 x_4^0$, que representa o nó obtido ao final desta transição. Observe que as potências de $y_1$ e $y_2$ correspondem aos símbolos associados à transição do nível de nó 0 ao nível de nó 1 no estado (0,0) e do nível de nó 1 ao nível de nó 2 ainda no estado (0,0) (ver fig. 2.1).

Continuando a construção considere agora o nó representado pelo termo $y_1^0 y_2^0 x_3^0 x_4^0$, que representa um nó no estado (0,0) e nível de nó 2. Seja o produto deste termo pelo primero termo do terceiro fator de (2.6), ou seja, o produto de $y_1^0 y_2^0 x_3^0 x_4^0$ por $y_3 x_3^0 x_4^0$ o que produz $y_1^0 y_2^0 y_3^0 x_3^{0+0} x_4^{0+0} = y_1^0 y_2^0 y_3^0 x_3^0 x_4^0$. O que corresponde a avançar mais um nó na treliça partindo do nó de nível 2 no estado (0,0) para o nó de nível 3 no estado (0,0). Finalmente, considere o produto de $y_1^0 y_2^0 y_3^0 x_3^0 x_4^0$, que representa um nó de nível 3 no estado (0,0), por $y_4^0 x_3^0 x_4^0$, que é o primeiro termo do quarto fator de (2.6). Do resultado deste produto obtemos $y_1^0 y_2^0 y_3^0 y_4^0 x_3^0 x_4^0$ que representa o nó de nível 4 no estado (0,0). Como este nó é um nó final e está no estado (0,0), a seqüência obtida pelas potência dos $y_i$'s corresponde à palavra código (0,0,0,0) ( ver fig. 2.1).

Como exemplo final, suponha que após dois avanços na direção do final da treliça obtivemos o monômio $y_1^\beta y_2^1 x_3^0 x_4^\alpha$, que representa o nó de nível 2 no estado $(0,\alpha)$. Se efetuamos o produto deste monômio pelo monômio $y_3^1 x_2^1 x_3^0$ obtemos $y_1^\beta y_2^\alpha y_3^1 x_3^{1+0} x_4^{0+\alpha} = y_1^\beta y_2^\alpha y_3^1 x_3^1 x_4^\alpha$ que representa o nó de nível 3 no estado $(1,\alpha)$ e finalmente efetuando o produto deste último termo pelo monômio $y_4^1 x_3^0 x_4^\beta$ obtemos $y_1^\beta y_2^\alpha y_3^1 y_4^1 \; x_3^{1+0} x_4^{\beta+\alpha} = y_1^\beta y_2^\alpha y_3^1 y_4^1 \; x_3^1 x_4^1$. Observe que neste caso o estado final (0,0) (potências de $x_3 x_4$) não foi atingido, logo as potência dos $y_i$' $(\beta,\alpha,1,1)$ não formam uma palavra código (ver fig. 2.1). Concluimos que nem todos os produtos obtidos da expressão (2.6) correspondem a caminhos que convergem para o estado (0,0) no nível de nó 4, não sendo portanto correspondentes a palavras código de $\mathcal{C}$.

A representação de todas as outras palavras código pode ser obtida similarmente. Resta ainda observar que o estado obtido ao se avançar dois níveis de nó na treliça define os dois últimos símbolos do caminho que passa por este estado e converge ao estado nulo, caminho este que correspode a uma

palavra código. Por exemplo, o monômio $y_1^{\beta} y_2^{1} x_3^{0} x_4^{\alpha}$ corresponde ao nó de nível 2 no estado $(0,\alpha)$ logo a palavra código associada é $(\beta,1,0,\alpha)$, onde os dois primeiros símbolos foram obtidos das potências de $y_1$ e $y_2$ e os dois últimos do estado atingido no nível de nó 2, desde que somente os caminhos que convergem ao estado (0,0) correspondem à palavras código e $\alpha = -\alpha$.

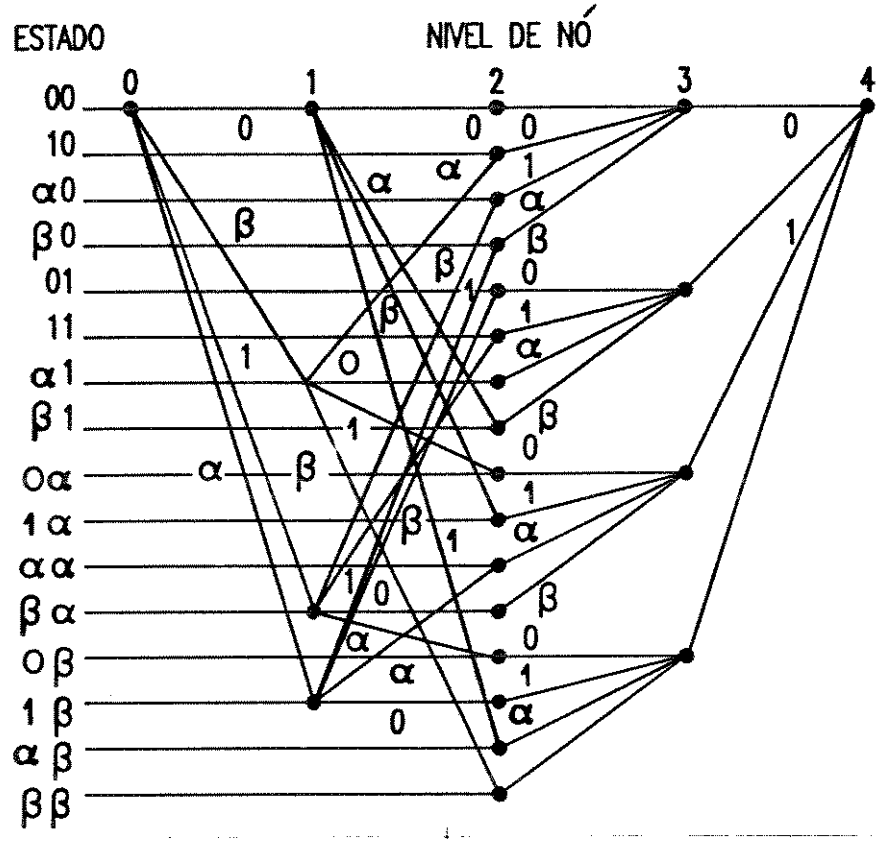


Figura 2.1 - Treliça para o código do exemplo 2.1

Observe que, realmente não é necessário nenhum atraso para a convergência do caminho decodificado, visto que as treliças para códigos de bloco têm comprimento finito.

## 2.3 - DECODIFICAÇÃO DE BATTAIL [1]

Neste esquema de decodificação de códigos concatenados, a medida de confiabilidade a ser transmitida entre estágios de decodificação é dada pela probabilidade de decodificação correta de cada símbolo em cada posição da palavra código. Assim, seja $P_i(c_i)$ a probabilidade a priori (em relação ao decodificador externo) do i-ésimo símbolo transmitido tomar o valor $c_i \in GF(q)$, logo,

$$\sum_{c_i \in GF(q)} P_i(c_i) = 1, \ \forall \ i = 1, \ ..., \ n_1. \tag{2.7}$$

Neste ponto devemos considerar um algoritmo que percorra a treliça de modo a determinar uma palavra código ótima segundo determinado critério.

Seja $\mathbf{c} = (c_1, c_2, \ ... \ c_{n_1})$, $c_i \in GF(q)$. Ignorando-se as restrições do código, temos que $P(\mathbf{c}) = \prod_{i=1}^{n_1} P_i(c_i)$. Por outro lado, quando as restrições do código são consideradas, temos

$$P(\mathbf{c} | \mathbf{c} \in \mathcal{C}(n_1,k_1)) = \frac{P(\mathbf{c}; \mathbf{c} \in \mathcal{C}(n_1,k_1))}{P(\mathbf{c} \in \mathcal{C}(n_1,k_1))} = \prod_{i=1}^{n_1} P_i(c_i) \Big/ \sum_{\mathbf{c}' \in \mathcal{C}(n_1,k_1)} \prod_{i=1}^{n_1} P_i(c'_i) \tag{2.8}$$

onde o produtório do numerador é tomado somente sobre palavras código de $\mathcal{C}$. Assim, o critério do Máximo a Posteriori (MAP) ou, equivalentemente, o critério da Máxima Verossimilhança, aqui utilizado, consiste em maximizar o numerador da expressão (2.8).

Para tanto, considere o termo $y_i^m$ na expressão (2.2). Esta variável deve ser substituída por $P_i(m)$, supostamente conhecido, onde $m \in GF(q)$. Como

$y_i^m$ corresponde ao ganho de ramo quando se passa de um nó no nível i-1 para um nó no nível i, vê-se que cada caminho na treliça corresponde a um valor de P(c), desde que $y_i^m$ seja associado a $P_i(m)$.

Vemos, então, que um algoritmo que percorra a treliça maximizando P(c) deverá realmente efetuar uma decodificação de máxima verossimilhança.

### 2.3.1. Determinação de $P_i(c_i)$

A determinação das probabilidades de ocorrência para cada símbolo pode ser feita utilizando-se a equação

$$P_k(c_k) = [p^{-n-1}/p(r)]\ A_k(c_k) \tag{2.9}$$

onde

$$A_k(c_k) = \sum_{t=o}^{p-1} \omega^{-c_k t} \prod_{j=1}^{n-k} \left[ \prod_{l=0}^{n-1} \sum_{i=0}^{p-1} \omega^{-i(c'_{jl} - t\delta_{kl})}\ p(r_l \mid i) \right] \tag{2.10}$$

$\omega$ ≡ $\exp(2\pi\sqrt{-1}/p)$

$c'_{jl}$ ≡ l-ésimo símbolo da j-ésima palavra código do código dual a $\mathcal{C}$.

$\delta_{kl}$ = 1 se k=l , $\delta_{kl}$=0 se k≠l

p(r) ≡ densidade do vetor recebido r.

$p(r_l \mid i)$ ≡ densidade condicional da componente $r_l$ de r dado o o símbolo i.

p = número primo não nulo e diferente de 1.

Esta equação, devida a C.R.P. Hartmann e L.D. Rudolph [2], é largamente utilizada na decodificação símbolo a símbolo de um código $\mathcal{C}(n,k)$ e é aqui considerada em sua forma completa.

No próximo capítulo apresentaremos uma versão mais conveniente da equação (2.3).

### 2.3.2. Ponderação

A eficiênca do algoritmo de Battail se baseia numa ponderação efetuada sobre as probabilidades associadas a cada símbolo em cada posição da palavra código. Assim, para efeito de decodificação, devemos substituir $P_j(m)$ por $Z_j(m)$ na equação (2.2) e associar $Z_j(m)$ a cada $y_j^m$ na expressão (2,8), onde

$$Z_j(m) = \log_e \lambda_j P_j(m) \tag{2.11}$$

e

$$Z_k(\mathbf{u}) = \sum_{j=1}^{k} Z_j(u_j) \tag{2.12}$$

sendo

$$\lambda_j = \max_m P_j(m).$$

$$\mathbf{u} = (u_1, u_2, \ldots, u_{n_1})$$

É óbvio que maximizar P(c) corresponde a maximizar

$$Z(\mathbf{c}) = \sum_{j=1}^{n_1} Z_j(c_j) \tag{2.13}$$

onde

$$c_j \in GF(q).$$

Observe que estamos considerando símbolos em GF(q), que é o corpo base para o código externo. Logo, este método se aplica quando cada símbolo para o código externo, em GF(q), é codificado por um código interno em cada posição. Assim, é possível projetar um código diferente para cada símbolo em cada posição em uma palavra código do código externo. Contudo, neste trabalho utilizaremos um único código interno, ou seja, não será utilizada a codificação concatenada com proteção desigual.

### 2.3.3. A Decodificação de Battail

A decodificação é efetuada através de um algoritmo que procura sobre a treliça o caminho de métrica máxima, ou seja, maximiza P(c) e obtém a palavra código de máxima verossimilhança. Tal algoritmo, denominado algoritmo de Battail, poderia ser encarado como uma versão do algoritmo de Viterbi (AV) para treliças finitas, supondo uma métrica conveniente. Na realidade, Battail [1] afirma que este algoritmo é computacionalmente mais eficiente que o AV para o caso de códigos de bloco representados por este tipo de treliça.

O algoritmo de Battail pode sofrer algumas simplificações, em detrimento da otimalidade, a fim de que seu desempenho computacional seja melhorado. Pode-se ainda utilizá-lo para decodificação abrupta no estágio posterior, supondo que o decodificador anterior não efetuou uma decodificação com saída ponderada. Neste caso a decodificação externa resulta em uma decodificação por distância mínima de Hamming.

### 2.3.4. O Algoritmo de Battail

1 - Determine $\mathbf{u}$ tal que $Z_{k_1}(\mathbf{u}) = 0$, onde $Z_{k_1}(\mathbf{u}) = \sum_{i=1}^{k_1} Z_i(u_i)$, $u_i \in GF(q)$.

Este passo determina a palavra código, através de seus símbolos de informação, que servirá de candidato inicial no processo de decodificação, ou seja, poderá ser descartada e substituída por uma palavra de métrica maior (na realidade mais próxima de zero) dada por $Z_{k_1}(\mathbf{u})$, se esta existir. Neste momento, estamos selecionando o caminho candidato na treliça que deve ser armazenado inicialmente.

2 - Determine a métrica associada ao caminho determinado em 1, $Z_{n_1}(\mathbf{u})$, e faça deste valor o limiar inicial, onde $n_1$ é o comprimento de uma palavra código do código externo.

Nos passos 1 e 2 estamos inicializando o algoritmo com a palavra código que supostamente não possui nenhum símbolo de informação errado. Obviamente, esta é uma condição inicial conveniente em muitos casos. Observe que não existe palavra código com métrica maior que zero.

3 - Determine $Z_{k_1}(\mathbf{u}^1)$ para um outro vetor de informação $\mathbf{u}^1$.

Este outro vetor deve ser escolhido segundo uma lei que minimize o tempo de execução do algoritmo. Battail [1] propõe que $Z_{k_1}(\mathbf{u}^1)$ seja considerado dentro de um conjunto de números reais ordenados decrescentemente. Caso se faça uma seleção decrescente em $P_j(m)$ para cada j, este fato ocorrerá naturalmente. Um algoritmo com uma seleção adequada será apresentado no Capítulo 4.

4 - Verifique se existe vetor de informação $\mathbf{u}^1$ tal que $Z_{k_1}(\mathbf{u}^1) > Z_{n_1}(\mathbf{u})$.

Caso positivo determine $Z_{n_1}(\mathbf{u}^1)$.

Caso negativo, **u** é a seqüência selecionada e fim.

Este passo corresponde ao momento de comparação entre os valores parciais de métrica e o limiar. É um passo importante no que diz respeito à eliminação de alguns caminhos sem a necessidade de cálculo de seus valores de métrica final. Pode ser comparado no AV ao momento em que este elimina caminhos para cada estado considerado à procura de um único sobrevivente.

5 - Verifique se $Z_{n_1}(\mathbf{u}^1) > Z_{n_1}(\mathbf{u})$

Caso negativo volte ao passo 3

Caso positivo substitua o limiar por $Z_{n_1}(\mathbf{u}^1)$ e **u** por $\mathbf{u}^1$ e volte ao passo 3.

Este passo corresponde à alteração do limiar. Garante que a a palavra código selecionada é a de maior valor de métrica sem a necessidade de uma comparação duas a duas entre todas as métricas possíveis.

## 2.4. DECODIFICAÇÃO DE HAGENAUER [3]

Este esquema de decodificação de códigos concatenados, alternativo ao de Battail, corresponde a uma aplicação pura e simples do algoritmo de Viterbi (AV) à treliça construída para um código convolucional. Uma simples modificação permitirá a transferência de confiabilidade entre estágios de decodificação. Este processo deve ser utilizado quando se está considerando códigos externos, de bloco ou convolucionais, concatenados a códigos convolucionais internos que tenham como corpo base GF(2).

Uma vez que o processo de decodificação externa só depende do resultado da decodificação interna e não do tipo de decodificação interna, pode-se usar como código interno um código convolucional.

O cálculo da confiabilidade de cada dígito binário pode ainda ser efetuado através de uma decodificação interna dígito a dígito. Contudo, seria interessante utilizar uma técnica que fizesse uso do algoritmo de Viterbi por sua reduzida complexidade em comparação com o decodificador dígito a dígito de códigos convolucionais.

Assim, podemos utilizar os resultados de J. Hagenauer e P. Höher [3] na determinação da confiabilidade de cada dígito binário decodificado pelo decodificador interno.

### 2.4.1. O Algoritmo

Seja $\mathbf{u} = (u_1, \ldots, u_{k_2})$, $u_i \in GF(2)$, $i = 1, \ldots, n_2$, um vetor de informação. Suponha que $\mathbf{u}$ seja codificado e enviado através de um canal com ruído gaussiano, branco, de média nula e variância $\sigma^2$. Assuma que o algoritmo de Viterbi efetua uma decisão final com retardo $\delta$. Seja $\nu$ a memória do código $\mathcal{C}(n_2,k_2,\nu)$ utilizado em $\mathbf{u}$. Seja $S_t$ um estado, entre os possíveis $S = 2^{\nu}$ estados, em um instante t. Então, o algoritmo de Viterbi Modificado pode ser descrito pelos seguintes passos

(i) Armazenamentos

t (índice de tempo módulo $\delta$+1)

$\hat{\mathbf{u}} = \{\hat{u}_{t-\delta}(S_t), \ldots, \hat{u}_t(S_t)\}$, (valores da decisão abrupta $\hat{u} \in \{\pm 1\}$)

$\hat{\mathbf{L}}(S_t) = (\hat{L}_{t-\delta}(S_t), \ldots, \hat{L}_t(S_t))$, (confiabilidades $0 \leq \hat{\mathbf{L}} \leq \infty$)

$\Gamma(S_t)$, $0 \leq S_t \leq S-1$ (valores da métrica acumulada)

(ii) Algoritmo

Para cada estado $S_t$

a) Determine

$$\Gamma(S_{t-1}, S_t) = \Gamma(S_{t-1}) + \frac{E_s}{No} \sum_{j=1}^{N} (y_{tj} - x_{tj}^{(i)})^2 \qquad (2.15)$$

para cada uma das possíveis transições $(S_{t-1}, S_t)$, onde

$\frac{E_s}{N_o} \triangleq$ razão sinal-ruído à saída do canal

$y_{tj} \triangleq$ j-ésimo símbolo recebido associado ao ramo $(S_{t-1}, S_t)$

$x_{tj} \triangleq$ j-ésimo símbolo associado ao ganho do ramo $(S_{t-1}, S_t)$

$i \triangleq$ possível transição $(S_{t-1}, S_t)$

$N$ = número de dígitos binários em um ramo $(S_{t-1}, S_t)$.

b) Encontre $\Gamma(S_t) = \text{MIN } \Gamma(S_{t-1}, S_t)$

c) Armazene $\Gamma(S_t)$ e o correspondente $\hat{u}_t(S_t)$

Estes passos correspondem ao algoritmo de Viterbi clássico, onde ocorreu o necessário retardo que garante a convergência do algoritmo.

A seguir, uma comparação entre caminhos convergindo em cada estado $S_k$ produzirá a medida de confiabilidade que necessitamos.

d) Armazene $\Delta = \text{Max } \Gamma(S_{t-1}, S_t) - \text{Min } \Gamma(S_{t-1}, S_t)$

e) Faça $\hat{L}_t(S_t) = +\infty$

f) Para $j=t-\nu$ até $\delta_m$ compare os símbolos entre os caminhos convergindo em cada estado $S_t$, onde $\delta_m$ é o comprimento dos caminhos até eles convergirem em $S_t$.

Se houver símbolos distintos entre o caminho de máxima verossimilhança e os outros possíveis caminhos, então,

$$\hat{L}_j \leftarrow f(\Delta,\hat{L}_j) , \qquad \text{para j onde os símbolos são distintos}$$

A função $f(\Delta,\hat{L}_j)$ é dada por

$$f(\Delta,\hat{L}_j) = \frac{1}{\alpha} \log \frac{1 + e^{(\alpha L_j + \Delta)}}{e^{\Delta} + e^{\alpha \hat{L}_j}} \tag{2.16}$$

$$\alpha \triangleq d_{free} \frac{E_s}{N_o}$$

onde $\alpha$ garante $E[\hat{L}_j]$ assintoticamente unitário.

Uma boa aproximação para a equação (2.2) é

$$f(\hat{L}_j,\Delta) = \text{Min } (\hat{L}_j,\Delta/\alpha) \tag{2.17}$$

Esta função pode ainda ser tabulada em $\hat{L}_j$ e $\Delta$.

A aplicação deste algoritmo só deverá ser considerada para o caso binário, ou seja, o número de símbolos do corpo base para o código é 2 (dois).

## 2.5. CONCLUSÃO

Se compararmos a técnica de decodificação de Battail à técnica de decodificação de Hagenauer, verificaremos que ambas as técnicas permitem a extensão para concatenação com proteção desigual, visto que cada símbolo é decodificado em separado. O segundo método, decodifi-cação de Hagenauer, permite a concatenação de mais de dois códigos em seqüên-cia, necessitando-se para tanto simplesmente de uma métrica adequada a partir do segundo estágio.

Observe que na decodificação de Hagenauer [3] é preciso decodificar uma palavra código do código interno inteira para que se possa determinar a confiabilidade de cada dígito desta palavra código. O decodificador de Battail [1] determina dígitos e confiabilidades individualmente, fato que simplificará o decodificador externo, como veremos a seguir. Além do mais, o decodificador de Hagenauer necessita de uma estrutura, denominada embaralhador/desembaralha-dor, que elimine os surtos de erro produzidos pelo decodificador interno.

# CAPÍTULO 3

# ALGORITMOS MODIFICADOS

## 3.1. INTRODUÇÃO

Os algoritmos apresentados no capítulo anterior encontram-se em sua forma básica. A sua aplicação na decodificação de códigos de bloco concatenados, ou mesmo de um código convolucional concatenado a um código de bloco, necessita de algumas modificações no sentido de promover uma interface adequada entre os códigos. Como afirmado antes, apresentaremos um esquema de correção de erros baseado no uso de um código externo concatenado a um outro código interno.

No contexto deste trabalho, uma interface entre os códigos deve ser entendida como sendo o processo de transferência de confiabilidade do código interno para o código externo. Esta interface se dá através da transferência dos dígitos constituintes de cada símbolo em cada posição da palavra código do código externo juntamente com suas probabilidades de ocorrência [3], [7].

O cálculo da probabilidade de ocorrência dos dígitos constituintes de cada símbolo pode ser feita de duas maneiras. Na primeira, considera-se cada símbolo constituinte de uma palavra código como um todo, o que pode ser obtido através de uma decodificação palavra a palavra. Na segunda, consideram-se os dígitos constituintes de cada símbolo separadamente, o que

pode ser obtido através de uma decodificação interna dígito a dígito para cada símbolo. Mostraremos que a decodificação palavra a palavra para o código interno levará a uma decodificação por correlação completa para o código concatenado.

## 3.2. DECODIFICADOR INTERNO

### 3.2.1. Código de Bloco

Considere como código interno um código de bloco $\mathcal{C}(n_2,k_2)$. O casamento entre este código e o código externo, definido sobre um corpo base $GF(p^m)$, será conseguido quando $k_2 = m$. Isto feito, decodificar uma palavra do código interno corresponderá a decodificar um símbolo do código externo.

Desta forma, para todo vetor recebido $\mathbf{r}$ e toda palavra $\mathbf{c}_t$ de $\mathcal{C}(n_2,k_2)$, temos

$$P(\mathbf{c}_t/\mathbf{r}) = \frac{p(\mathbf{r}/\mathbf{c}_t)\, P(\mathbf{c}_t)}{p(\mathbf{r})} \tag{3.1}$$

onde $P(\cdot)$ representa uma função de distribuição de probabilidade discreta e $p(\cdot)$ representa uma função de densidade de probabilidade.

Seja $\mathbf{c}_t^*$ tal que $P(\mathbf{c}_t^*/\mathbf{r})$ é máximo. Assim, supondo fonte equiprovável ($P(\mathbf{c}_t) = P(\mathbf{c}_t^*)\ \forall\ \mathbf{c}_t$), temos

$$\frac{P(\mathbf{c}_t/\mathbf{r})}{P(\mathbf{c}_t^*/\mathbf{r})} = \frac{p(\mathbf{r}/\mathbf{c}_t)}{p(\mathbf{r}/\mathbf{c}_t^*)} \leq 1 \tag{3.2}$$

mas, desde que estamos supondo canal com ruído gaussiano, branco, de média nula e variância $\sigma^2$, temos, para $c_t = (c_1, c_2, \ldots, c_{n_2})$,

$$p(r/c_t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\bar{\sigma}^2 \sum_i (c_i - r_i)^2/2} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\bar{\sigma}^2 \sum_i r_i^2/2} \; e^{-\bar{\sigma}^2 \sum_i c_i^2/2} \; e^{\bar{\sigma}^2 \sum_i c_i r_i} \tag{3.3}$$

Logo,

$$\frac{P(c_t/r)}{P(c_t^*/r)} = e^{\bar{\sigma}^2 \Sigma c_i r_i} \; e^{-\bar{\sigma}^2 \Sigma c_i^* r_i} \tag{3.4}$$

e

$$\ln \frac{P(c/r)}{P(c_t^*/r)} = \bar{\sigma}^2 \sum_i c_i r_i - \bar{\sigma}^2 \sum_i c_i^* r_i$$

$$= \sum_i \bar{\sigma}^2 \, (c_i - c_i^*) \, r_i \leq 0. \tag{3.5}$$

Daí, a métrica para cada símbolo do código externo será dada por

$$\ln \frac{P(c_t/r)}{P(c_t^*/r)} = z_{cc}(c_t) \tag{3.6}$$

onde o índice (cc) indica que esta é a métrica de símbolo quando decodificador interno efetua uma decodificação palavra a palavra utilizando para tal a correlação completa.

Agora, considere um código externo $\mathcal{C}(n_1, k_1)$ concatenado a um código interno $\mathcal{C}(n_2, k_2)$. Assim, se o decodificador externo utiliza o algoritmo de Battail [1], a métrica para uma palavra código c do código externo será dada por

$$Z_{cc}(c) = \sum_{i=1}^{n_1} z_{cc}(c_i) = \sum_{i=1}^{n_1} \left\{ \sum_{j=1}^{n_2} \bar{\sigma}^2 \, (c_{ij} - c^*_{ij}) \, r_{ij} \right\} \tag{3.7}$$

onde $c \in \mathcal{C}(n_1,k_1)$, $c_i \in GF(p^m)$ , $r = (r_1, \ldots, r_{n_1})$ e $r_i = (r_{i1}, \ldots, r_{in_2})$, $i = 1, \ldots, n_1$, são os vetores recebidos correspondentes à cada símbolo transmitido.

Vê-se, pela equação (3.7), que para uma decodificação interna do tipo palavra a palavra, obtemos uma decodificação por correlação completa para o código concatenado $(n_1n_2,k_1k_2)$.

Suponha, agora, que os símbolos da palavra código $c \in \mathcal{C}(n_1,k_1)$ sejam independentemente decodificados, ou seja,

$$P(c/r) = \prod_{i=1}^{n_1} P(c_i/r_i) = \prod_{i=1}^{n_1} \prod_{j=1}^{n_2} P(c_{ij}/r_i) \tag{3.8}$$

com $r_i$ e $c_i$ já definidos e $c_{ij} \in GF(p)$, $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$.

A partir do esquema de decodificação símbolo a símbolo proposto por Hartmann e Rudolph [2], podemos obter as probabilidades de ocorrência de cada dígito decodificado pelo decodificador interno.

De fato, seja $c = (c_1, \ldots, c_{n_1})$, $c_i \in GF(p^m)$, uma palavra código do código externo $\mathcal{C}(n_1,k_1)$. Conforme já vimos, o código interno, $\mathcal{C}(n_2,k_2)$, atua sobre os símbolos $c_i$ das palavras código do código externo. Assim, $\mathcal{C}(n_2,k_2)$ é um código sobre GF(p).

Suponha que o decodificador interno receba à saída do canal um vetor $r_i = (r_{i1}, \ldots, r_{in_2})$, $i = 1, \ldots, n_1$ com componentes reais $r_{ij}$. Então,

$$P(c_{ij}/r_i) = \left\{ p^{n_2 - 1} / P(r_i) \right\} A(c_{ij}) \quad , \quad c_{ij} \in GF(p) \tag{3.9}$$

onde

$$A(c_{ij}) = \sum_{t=o}^{p-1} w^{-c_{ij}t} \sum_{k=1}^{p^{n_2-k_2}} \left[ \prod_{l=1}^{n_2} \sum_{s=o}^{p-1} w^{-s(c'_{kl} - t\delta_{jl})} \; p(r_{il}/s) \right] \tag{3.10}$$

e

$w \triangleq \exp(2\pi\sqrt{-1}/p)$

$c'_{kl} \triangleq$ l-ésimo dígito da k-ésima palavra do código dual a $\mathcal{C}(n_2,k_2)$

$\delta_{jl} = 1$ se $j = l$ e $\delta_{jl} = 0$ caso contrário

p = número de elementos de GF(p)

Seja $c^*_{ij}$ tal que $P(c^*_{ij}/r_i)$ seja máximo em j, para cada i. Seja também

$$\frac{P(c_i/r_i)}{P(c^*_i/r_i)} = \prod_{j=1}^{n_2} \frac{A(c_{ij})}{A(c^*_{ij})} \tag{3.11}$$

ou

$$\ln \frac{P(c_i/r_i)}{P(c^*_i/r_i)} = \sum_{j=1}^{n_2} \ln \left\{ \frac{A(c_{ij})}{A(c^*_{ij})} \right\}$$

Para p = 2 (caso binário), obtém-se

$$A(c_{ij}) = \sum_{t=o}^{1} (-1)^{-tc_{ij}} \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \sum_{s=o}^{1} (-1)^{-s(c'_{kl}-t\delta_{jl})} \; p(r_{il}/s) \tag{3.12}$$

$$= \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \left[ p(r_{il}/0) + (-1)^{c'_{kl}} \; p(r_{il}/1) \right] \tag{3.13}$$

$$+ (-1)^{-c_{ij}} \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \left[ p(r_{il}/0) + (-1)^{c'_{kl} \oplus \delta_{jl}} \; p(r_{il}/1) \right] \tag{3.14}$$

onde $\oplus \triangleq$ soma em GF(2).

Definindo

$$\alpha_i = \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \left[ p(r_{il}/0) + (-1)^{c'_{kl}} p(r_{il}/1) \right] \tag{3.15}$$

temos

$$A(c_{ij}) = \alpha_i + (-1)^{-c_{ij}} \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \left[ p(r_{il}/0) + (-1)^{c'_{kl} \oplus \delta_{jl}} p(r_{il}/1) \right] \tag{3.16}$$

Podemos ainda reescrever a expressão (3.16) em termos da razão de verossimilhança $\phi_{il} = p(r_{il}/1)/p(r_{il}/0)$ da seguinte forma

$$\alpha_i = \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \left[ p(r_{il}/0) + (-1)^{c'_{kl}} p(r_{il}/1) \right]$$

$$= \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} p(r_{il}/0) \prod_{l=1}^{n_2} \left[ 1 + (-1)^{c'_{kl}} \phi_{il} \right] =$$

$$= \prod_{l=1}^{n_2} p(r_{il}/0) \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \left[ 1 + (-1)^{C'_{kl}} \phi_{il} \right] =$$

$$= \prod_{l=1}^{n_2} p(r_{il}/0) \; \beta_i \tag{3.17}$$

Analogamente,

$$(-1)^{-c_{ij}} \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \cdot \left[ p(r_{il}/0) + (-1)^{c'_{kl} \oplus \delta_{jl}} p(r_{il}/1) \right]$$

$$= (-1)^{-c_{ij}} \prod_{l=1}^{n_2} p(r_{il}/0) \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \left[ 1 + (-1)^{c'_{kl} \oplus \delta_{jl}} \phi_{il} \right]$$

$$= (-1)^{-c_{ij}} \prod_{l=1}^{n_2} p(r_{il}/0) \; \beta_{ij} \qquad (3.18)$$

Assim,

$$\frac{A(c_{ij})}{A(c^*_{ij})} = \frac{\beta_i + (-1)^{c_{ij}} \beta_{ij}}{\beta_i + (-1)^{c^*_{ij}} \beta_{ij}} \qquad (3.19)$$

onde

$$\beta_i = \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \left[ 1 + (-1)^{c'_{kl}} \phi_{il} \right] \qquad (3.20)$$

$$\beta_{ij} = \sum_{k=1}^{2^{n_2-k_2}} \prod_{l=1}^{n_2} \left[ 1 + (-1)^{c'_{kl} \oplus \delta_{jl}} \phi_{il} \right] \qquad (3.21)$$

Para o canal com ruído gaussiano de média nula e variância $\sigma^2$

$$\phi_{il} = e^{2\sigma^{-2} r_{il}}. \qquad (3.22)$$

Finalmente,

$$Z_{hr}(c) = \ln \left[ \frac{P(c/r)}{P(c^*/r)} \right] = \sum_{i=1}^{n1} \sum_{j=1}^{n2} \ln \frac{\beta_i + (-1)^{c_{ij}} \beta_{ij}}{\beta_i + (-1)^{c^*_{ij}} \beta_{ij}} \qquad (3.23)$$

onde $\beta_i$ e $\beta_{ij}$ são definidos por (3.2) e (3.3), respectivamente, $c \in \mathcal{C}(n_1,k_1)$ e r é o vetor recebido.

Assim, para este tipo de decodificador interno podemos definir uma métrica $Z_{hr}(\cdot)$ , a partir das expressões de Hartmann e Rudolph (3.26), dada por

$$Z_{hr}(c) = \sum_{i=1}^{n_1} z_{hr}(c_i) \tag{3.24}$$

e $z_{hr}(c_i)$ dado por

$$z_{hr}(c_i) = \sum_{j=1}^{n_2} \ln \frac{\beta_i + (-1)^{c_{ij}} \beta_{ij}}{\beta_i + (-1)^{c^*_{ij}} \beta_{ij}} , \quad i = 1, \ldots, n_1. \tag{3.26}$$

onde o índice (hr) indica que esta é a metrica de símbolo quando o decodificador interno efetua uma decodificação dígito a dígito utilizando para tal as expressões obtidas por Hartmann e Rudolph [2].

Observe que a expressão (3.7), repetida aqui por simples conveniência

$$Z_{cc}(c) = \sum_{i=1}^{n_1} z_{cc}(c_i) = \sum_{i=1}^{n_1} \left\{ \sum_{j=1}^{n_2} \bar{\sigma}^{-2} (c_{ij} - c^*_{ij}) r_{ij} \right\} \tag{3.7}$$

depende dos códigos utilizados no esquema de códigos concatenados e da razão sinal-ruído canal por onde foram enviadas as palavras código. Assim, se fixarmos um código interno binário (6,3) , fizermos $n_1 = 1$ e gerarmos um ruído branco gaussiano para varios valores de razão sinal-ruído poderemos observar o comportamento da métrica da correlação completa quando utilizada em um canal AWGN para vários valores de razão sinal ruído. Isto feito, obtivemos a

primeira curva (curva pontilhada) da figura 3.1.

Por outro lado, se considerarmos a expressão (3.26) com o índice i fixo, fixarmos o mesmo código binário (6,3) e gerarmos novamente um canal AWGN para vários valores de razão sinal-ruído obteremos a segunda curva (curva contínua) da fig. (3.1).

Estes resultados nos sugerem que a métrica de Hartmann e Rudolph seja quase estatisticamente equivalente à métrica da correlação completa, quando se fixa um código e se varia a razão sinal-ruído em um canal com ruído branco e gaussiano. Fato que é confirmado pela pequena diferença de desempenho entre as métricas.

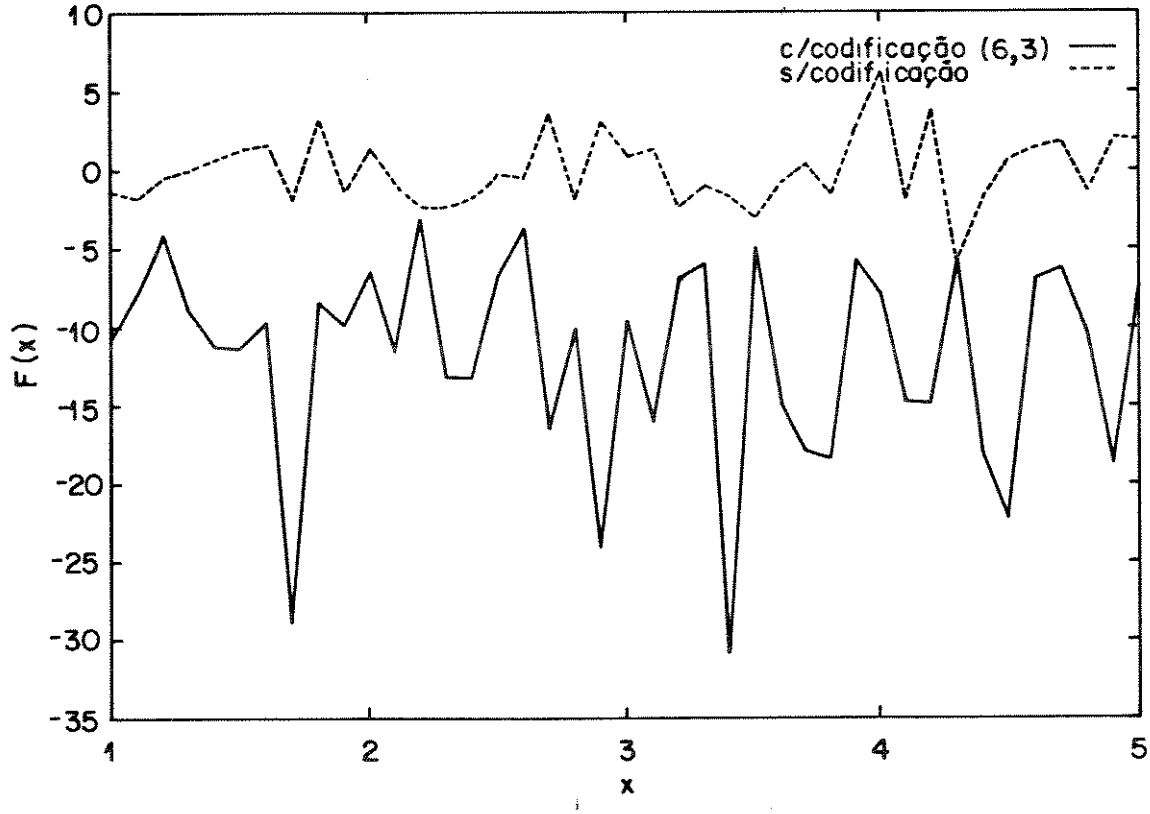


Fig. 3.1 - Simulação de métrica
F(x) é dado pela eq. 3.7 na curva pontilhada e
é dado pela eq. 3.26 na curva contínua
x ≡ razão sinal-ruído medida em dB.

Na realidade, a fig. (3.1) mostra na curva superior uma soma ponderada de amostras de um processo gaussiano e branco obtidas para vários valores de razão sinal-ruído (métrica da correlação completa) e na curva inferior os valores obtidos quando estas amostras são utilizadas como argumento na equação (3.26), também para vários valores de razão sinal-ruído (métrica de Hartmann e Rudolph).

Observe que o esquema que utiliza a métrica de Hartmann e Rudolph tem sua complexidade dependente do código dual ao código usado. Logo, ele deve ser utilizado quando se utilizam códigos internos de taxas elevadas. Por outro lado, o esquema que utiliza a métrica da correlação completa deve ser utilizado no caso de códigos internos de baixas taxas, pois sua complexidade é função do código utilizado. Contudo, como descrito no capitulo anterior, Hartmann e Rudolph mostram uma técnica de decodificação símbolo a símbolo onde a complexidade do decodificador é função do código utilizado e não de seu dual. Assim, podemos sempre usar a métrica de Hartmann e Rudolph para qualquer taxa de codificação.

Observamos que a expressão de $Z_{hr}(\cdot)$ apresenta uma elevada redundância em termos de operações de adição e multiplicação, já que $\beta_i$ e $\beta_{ij}$ só precisam ser determinados $n_1$ vezes e $n_1 n_2$ vezes respectivamente (para um código concatenado $(n_1 n_2, k_1 k_2)$) e podem ser obtidos em paralelo, pois possuem basicamente a mesma estrutura. Assim, o código interno pode ser decodificado da forma usual sem ônus significativo para o tempo de decodificação do mesmo.

Apresentaremos, agora, a equação (3.26) de uma maneira mais conveniente para fins de implementação.

Assim, dividindo $\beta_i$ por $\prod_{l=1}^{n_2} (1 + \phi_{il})$ temos,

$$\frac{\beta_i}{\prod_{l=1}^{n_2}(1+\phi_{il})} = \sum_{k=1}^{2} \prod_{l=1}^{n_2-k_2} \left[1 + (-1)^{c'_{kl}} \phi_{il}\right] \Big/ \prod_{l=1}^{n_2}(1+\phi_{il}) .$$

Usando a identidade

$$\frac{1 + (-1)^{c'_{kl}} \phi_{il}}{1+\phi_{il}} = \left(\frac{1-\phi_{il}}{1+\phi_{il}}\right)^{c'_{kl}} .$$

temos,

$$\frac{\beta_i}{\prod_{l=1}^{n_2}(1+\phi_{il})} = \sum_{k=1}^{2} \prod_{l=1}^{n_2-k_2} \left(\frac{1-\phi_{il}}{1+\phi_{il}}\right)^{c'_{kl}} = \gamma_i . \qquad (3.27)$$

Analogamente,

$$\frac{\beta_{ij}}{\prod_{l=1}^{n_2}(1+\phi_{il})} = \sum_{k=1}^{2} \prod_{l=1}^{n_2-k_2} \left(\frac{1-\phi_{il}}{1+\phi_{il}}\right)^{c'_{kl} \oplus \delta_{jl}} = \gamma_{ij} . \qquad (3.28)$$

Assim,

$$\frac{P(c_{ij}=0/r)}{P(c_{ij}=1/r)} = \frac{\prod_{l=1}^{n_2}(1+\phi_{il})}{\prod_{l=1}^{n_2}(1+\phi_{il})} \cdot \frac{\beta_i + \beta_{ij}}{\beta_i - \beta_{ij}}$$

$$= \frac{\gamma_i + \gamma_{ij}}{\gamma_i - \gamma_{ij}} \quad , \quad \begin{matrix} i = 1, \ldots, n_i \\ j = 1, \ldots, n \end{matrix} \qquad (3.29)$$

Como exemplo, utilizaremos o código de Hamming (7,4) apresentado em [2], para ilustrar este procedimento.

Quando se trata de decisão abrupta, temos

$$c_{ij} = 0 \quad \text{se e só se} \quad \gamma_{ij} > 0$$

podemos calcular os valores de $\gamma_{ij}$ através de um circuito que pode se obtido a partir da expansão da equação (3.27) [2]. Para o caso de um código de Hamming (7,4) este circuito pode ser apresentado da seguinte forma, onde o retângulo superior deve ser utilizado para armazenar cada um dos vetores $r_{ij}$ recebidos,

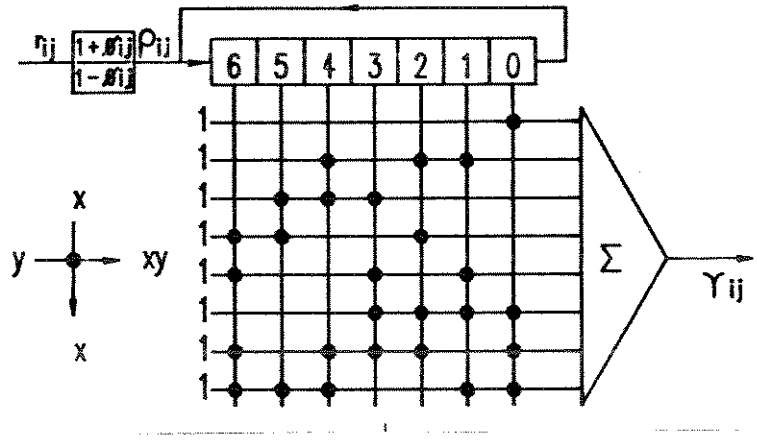


Fig. 3.2 - Circuito decodificador

Quando se trata de decisão suave, é preciso calcular a constante $\gamma_i$. O circuito da fig. (3.3), obtido a partir da equação (3.29), calcula os valores de $\gamma_i$ apropriados. Observe que o circuito da fig. (3.3) pode ser obtido a partir do circuito da fig. (3.4) simplesmente invertendo-se a entrada correspondente ao campo indexado com o número 0.

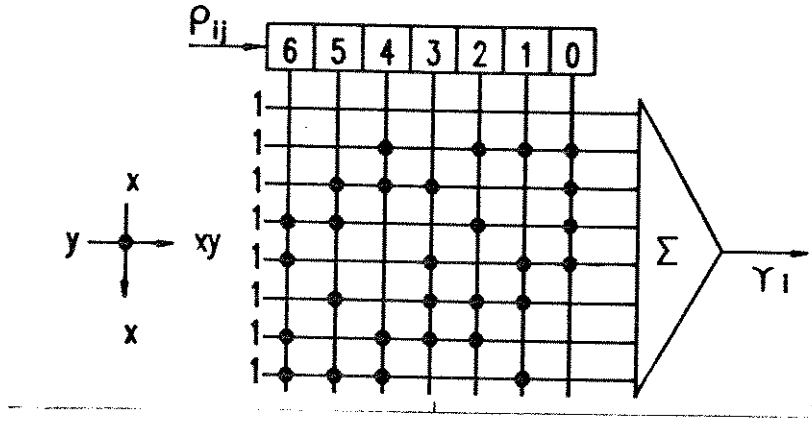


Fig. 3.3 - Circuito adicional

O funcionamento dos circuitos das fig. (3.3) e (3.4) é bastante simples [2]. Cada componente, $r_{ij}$ do vetor recebido **r** é processada e introduzida, como mostram as fig. (3.3) e (3.4), nas posições 6,5,4,3,2,1 e 0. Neste momento o valor $\gamma_{i1}$ está presente à saída do somador e cada deslocamento cíclico, que ocorre como indicado em cada uma das figuras, produz os valores de $\gamma_{i2},\ldots,\gamma_{i7}$ para $i = 1,\ldots,n_1$.

### 3.2.2. Código Convolucional

Considere agora um código convolucional $\mathcal{C}(n_2,k_2,\nu)$ atuando como código interno. Seja $\mathbf{c} \in \mathcal{C}(n_2,k_2,\nu)$. Como afirmado anteriormente, Hagenauer e Höhen [3] mostraram que uma boa medida da confiabilidade para um símbolo binário $c_i$ de **c** é dada por

$$L_i \leftarrow \frac{1}{\alpha} \ln \left[ \frac{1 + e^{(\alpha \hat{L}_i + \Delta)}}{e^{\Delta} + e^{\alpha L_i}} \right] \qquad (3.30)$$

onde

$$\Delta = M_2 - M_1 \geq 0$$

$\alpha = 4d_{free}.E_s/N_o$ (garante $E[L_j] = 1$ assintoticamente)

$$M_t = \frac{E_s}{N_o} \sum_{j=k-\delta}^{k} \sum_{i=1}^{N} (y_{ji} - x_{ji}^{(t)})^2$$

$\frac{E_s}{N_o} \triangleq$ razão sinal-ruído à saída do canal

$y_{ji} \triangleq$ i-ésimo dígito binário recebido, de um bloco de N dígitos binários, no instante j

$x_{ji}^{(t)} \triangleq$ i-ésimo dígito binário modulado (modulação antipodal), de um bloco de N dígitos binários, no instante j referente ao caminho t, t = 1,2

$\delta \triangleq$ retardo necessário à convergência ($\delta \geq k_2$).

Observamos que esta medida de confiabilidade leva em conta possíveis erros de decodificação anteriores num caminho de métrica máxima (que é o caminho de métrica $M_1$, por hipótese), para cada estado.

Assim, cada símbolo para o código externo (em $GF(2^m)$) deve ser decomposto em dígitos binários e codificado via $\mathcal{C}(n_2,k_2,\nu)$. A decodificação deve produzir uma seqüência de $k_2$ dígitos $\{\hat{u}_k\}$ juntamente com uma seqüência de $k_2$ números reais $\{\hat{L}_k\}$, que medem a confiabilidade de cada um dos $k_2$ dígitos, em GF(2), e que formam um símbolo para o código externo.

O decodificador externo deve utilizar o algoritmo de Viterbi sobre a treliça construída para o código externo e deve usar como métrica a seguinte expressão [3]

$$M_t = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} x_{ij}^{(t)} \hat{y}_{ij} L_{ij} \qquad (3.31)$$

onde

$x_{ij}^{(t)} \triangleq$ j-ésimo dígito do i-ésimo símbolo associado ao caminho t, e $x_{ij}^{(t)} \in \{\pm 1\}$.

$\hat{y}_{ij} \triangleq$ j-ésimo dígito do i-ésimo símbolo estimado e $\hat{y}_{ij} \in \{\pm 1\}$.

$L_{ij} \triangleq$ medida de confiabilidade de $\hat{y}_{ij}$

Evidentemente, $\delta$ deve ser suficientemente grande ($\delta \geq k_2$) para que o processo de decodificação interna produza um símbolo para o código externo.

Neste trabalho, mostramos que a decodificação de Viterbi sobre uma treliça para o código externo é equivalente à decodificação externa que utiliza o algoritmo de Battail, no caso binário, quando a métrica utilizada pelo decodificador externo é dada pela equação (3.31).

De fato, o decodificador externo de Viterbi produz,

$$\max_t M_t = \max_t \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} x_{ij}^{(t)} \hat{y}_{ij} L_{ij} \quad , \quad x_{ij}^{(t)} , \hat{y}_{ij} \in \{\pm 1\} \qquad (3.32)$$

onde

$$L_{ij} = \ln \frac{1 - p_{ij}}{p_{ij}}$$

é a confiabilidade do j-ésimo dígito binário do i-ésimo símbolo de uma palavra código do código externo, sendo

$$p_{ij} = P\{y_{ij} \neq \hat{y}_{ij} / r_i\} \qquad (3.33)$$

onde $\mathbf{r}_i$ é o i-ésimo vetor recebido, $y_{ij}$ é o j-ésimo dígito binário do i-ésimo símbolo da palavra código do código externo transmitida e $\hat{y}_{ij}$ é o j-ésimo dígito binário do i-ésimo símbolo de uma palavra código do código externo.

Mas, desde que,

$$\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} x_{ij}^2 = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \hat{y}_{ij}^2 = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} (\pm 1)^2 = n_1 n_2$$

então,

$$\max_t M_t = \max_t \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} x_{ij}^{(t)} \hat{y}_{ij} L_{ij} \iff \min_t \frac{1}{4}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} (x_{ij}^{(t)} - \hat{y}_{ij})^2 L_{ij}$$

e

$$\frac{1}{4}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} (x_{ij}^{(t)} - \hat{y}_{ij})^2 L_{ij} = \frac{1}{4}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \ln\left(\frac{1-p_{ij}}{p_{ij}}\right)^{(x_{ij}^{(t)} - \hat{y}_{ij})^2}.$$

Mas,

$$(x_{ij}^{(t)} - \hat{y}_{ij})^2 / 4 = \begin{cases} 0 & \text{se} \quad x_{ij}^{(t)} = \hat{y}_{ij} \\ 1 & \text{se} \quad x_{ij}^{(t)} \neq \hat{y}_{ij} \end{cases} \tag{3.34}$$

e desde que [3]

$$0 \leq p_{ij} \leq 1/2 \Rightarrow p_{ij} \leq 1 - p_{ij}$$

podemos fazer

$$P(c_{ij}^*) = 1 - p_{ij} \Rightarrow c_{ij}^* = x_{ij}^{(t)}$$

e

$$P(c_{ij}) = p_{ij} \Rightarrow c_{ij} = \hat{y}_{ij}$$

para cada caminho t, que corresponde a uma palavra código do código externo.

O que resulta

$$\frac{P(c^{*}_{ij}/r_i)}{P(c_{ij}/r_i)} = \begin{cases} L_{ij} & \text{se} \quad c_{ij} \neq c^{*}_{ij} \\ 1 & \text{se} \quad c_{ij} = c^{*}_{ij} \end{cases}$$

logo,

$$\max_{t} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} x_{ij}^{(t)} \hat{y}_{ij} L_{ij} \Leftrightarrow \max_{\mathbf{c}} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \ln\left(\frac{P(c_{ij}/r)}{P(c^{*}_{ij}/r)}\right) \qquad (3.35)$$

com $c_{ij}$ e $c^{*}_{ij}$ já definidos anteriormente.

Concluimos que a métrica para o decodificador externo proposta por Hagenauer [3] é equivalente à métrica definida pela equação (3.29) apropriadamente adaptada.

Observe que o esquema que utiliza um decodificador interno via Hartmann e Rudolph nos parece mais complexo quando comparado com o esquema que decodifica via Hagenauer e Höher. Contudo, como veremos a seguir, esta vantagem aparente perde sentido quando se considera o decodificador externo atuando.

## 3.3. DECODIFICADOR EXTERNO

### 3.3.1. Códigos de Bloco Concatenados

Quando se está utilizando código de bloco no codificador interno, o decodificador externo utilizará o algoritmo de Battail [1]. Tal algoritmo, descrito no capítulo anterior, se baseia nas probabilidades de ocorrência de cada símbolo na palavra código do código externo.

Assim, considerando $\mathcal{C}(n_1,k_1)$ um código multinível sobre $GF(p^m)$ , $\mathbf{c} \in \mathcal{C}(n_1,k_1)$ e $\mathbf{r}$ um vetor recebido temos

$$P(\mathbf{c}/\mathbf{r}) = \prod_{i=1}^{n_1} P(c_i/\mathbf{r}) \ , \ c_i \in GF(p^m),$$

que é a probabilidade de ocorrência da palavra código $\mathbf{c}$ a posteriori em relação ao decodificador interno. Note também que

$$\ln P(\mathbf{c}/\mathbf{r}) = \sum_{i=1}^{n_1} \ln P(c_i/\mathbf{r}) \ .$$

Para cada $c_i \in GF(p^m)$, $i = 1, \ldots, n_1$, seja $c_i^*$ tal que $P(c_i^*/\mathbf{r})$ é máximo em $GF(p^m)$, ou seja, para a i-ésima posição encontre o símbolo que possui probabilidade máxima de ocorrência. Então escreva,

$$\ln\left\{P(\mathbf{c}/\mathbf{r}) \ / \ P(\mathbf{c}^*/\mathbf{r})\right\} = \sum_{i=1}^{n_1} \ln\left\{P(c_i/\mathbf{r}) \ / \ P(c_i^*/\mathbf{r})\right\} \leq 0 \ . \qquad (3.36)$$

Evidentemente,

$$\max_{\mathbf{c}} P(\mathbf{c}/\mathbf{r}) \Leftrightarrow \max_{\mathbf{c}} \ln \left(P(\mathbf{c}/\mathbf{r}) \ / \ P(\mathbf{c}^*/\mathbf{r})\right)$$

Mas, já vimos que,

$$\ln(P(c) \,/\, P(c^*)) = \begin{cases} Z_{hr}(c) & \text{, para probabilidades marginais} \\ Z_{cc}(c) & \text{, para probabilidades conjuntas} \end{cases} \qquad (3.37)$$

Vemos, então, que a estrutura do decodificador externo, que maximiza (3.37), poderá utilizar duas métricas distintas. O decodificador que utiliza a métrica de Hartmann e Rudolph assume a hipótese de independência entre os dígitos constituintes de cada palavra do código interno, o que não ocorre para o decodificador interno palavra a palavra. Esta hipótese de independência nos parece razoável para os dígitos de informação. Contudo, não é verdadeira quando se consideram também os dígitos de redundância de cada palavra código do código interno.

Assim, é de se esperar que o desempenho medido pela probabilidade de erro de bit seja inferior para o esquema que utiliza a métrica de Hartmann e Rudolph.

O algoritmo de Battail [1] foi apresentado anteriormente em sua forma básica, ou seja, supondo ser aplicado a um esquema sem concatenação. Contudo, quando aplicado em estruturas concatenadas necessita de algumas modificações no sentido de otimizar seu desempenho.

Como contribuição do nosso trabalho, apresentamos o algoritmo de Battail modificado para o caso binário. Esse algoritmo supõe a utilização de um decodificador dígito a dígito interno no sentido de melhorar seu desempenho.

Desta forma, sejam $c \in \mathcal{C}(n_1,k_1)$, $c = (c_1, \ldots, c_{n_1})$, $c_i \in \mathcal{C}(n_2,k_2)$,

$c_i = (c_{i1}, \ldots, c_{in_2})$ e $Z(c_{ij})$, $i = 1, \ldots, n_1$; $j = 1, \ldots, n_2$, definido pela expressão abaixo

$$Z(c_{ij}) = \ln \frac{P(c_{ij}/r)}{P(c^*_{ij}/r)}$$

e ainda

$$Z_k(c) = \sum_{i=1}^{k} \sum_{l=1}^{n_2} Z(c_{il}) \quad , \quad k = 1, \ldots, n_1 \, . \tag{3.38}$$

onde $c^*_{ij}$ é tal que $P(c_{ij}/r) < P(c^*_{ij}/r)$, $j = 1, \ldots, n_2$, para cada i.

Seja ainda a seqüência de vetores $u^1, \ldots, u^S$, $S = 2^{k_1}$ tal que $Z_{k_1}(u^1) > Z_{k_1}(u^2) > \ldots > Z_{k_1}(u^S)$, com $u^i = (u^i_{11}, \ldots, u^i_{n_1 1})$, $u^i_{jl} \in GF(2)$, $i = 1, \ldots, S$, $j = 1, \ldots, n_1$ e $l = 1, \ldots, n_2$. Então, o decodificador externo utiliza um algoritmo de decodificação composto pelas instruções

1 - Calcule $Z_{n_1}(u^1)$

2 - Fixe limiar $\alpha \leftarrow Z_{n_1}(u^1)$ ; $J \leftarrow 1$

3 - $J \leftarrow J + 1$ ; $K \leftarrow 0$ ; $\beta \leftarrow 0$. Se $J = S$ vá para 9

4 - $K \leftarrow K + 1$ ; $L \leftarrow 0$. Se $K = n_1$ vá para 8

5 - $L \leftarrow L + 1$. Se $L = n_2$ vá para 4

6 - $\beta \leftarrow \beta + Z(u^J_{KL})$

7 - Se $\beta < \alpha$ vá para 3. Senão vá para 5

8 - Fixe limiar $\alpha \leftarrow \beta$ e $u^1 \leftarrow u^J$ .Vá para 3

9 - $u^1$ é a sequência decodificada e fim.

onde S é o número total de caminhos utilizados na treliça.

Observe que este algoritmo apresenta essencialmente o mesmo número

de instruções que o algoritmo básico. Contudo, os passos 5 e 6 permitem que somente as métricas de símbolo que forem utilizadas sejam calculadas.

O algoritmo de Battail modificado pode sofrer alterações no sentido de se melhorar seu desempenho computacional, em detrimento da otimalidade. No capítulo 4 mostraremos uma versão deste algoritmo em que o processo de ordenação se apresenta de maneira natural, de forma a permitir que alguns caminhos sejam eliminados, melhorando-se assim seu desempenho computacional. O exemplo (3.1), a seguir, ilustra o algoritmo de Battail modificado.

Exemplo 3.1

Considere o código de parâmetros (4,2) sobre GF(4) utilizado para construir a treliça da fig. 2.1 (cap. 2). Seja a seguinte representação para os elementos de GF(4)

$0 = 00$
$1 = 01$
$\alpha = 10$
$\beta = 11$

Adicionalmente suponha que a seguinte palavra código foi transmitida com modulação antipodal através de um canal AWGN

$$\mathbf{c} = (0,0,0,0)$$

Após a decodificação interna obtemos os seguintes valores de probabilidade de ocorrência de símbolos

$$P_1(0) = 0.6 \quad P_2(0) = 0.4 \quad P_3(0) = 0.5 \quad P_4(0) = 0.5$$
$$P_1(1) = 0.2 \quad P_2(1) = 0.3 \quad P_3(\beta) = 0.3 \quad P_4(1) = 0.2$$
$$P_1(\alpha) = 0.1 \quad P_2(\alpha) = 0.2 \quad P_3(1) = 0.1 \quad P_4(\alpha) = 0.2$$
$$P_1(\beta) = 0.1 \quad P_2(\beta) = 0.1 \quad P_3(0) = 0.1 \quad P_4(\beta) = 0.1$$

O algoritmo seleciona inicialmente a palavra código **c'** definida pelos dois símbolos mais confiáveis em cada uma das posições de informação, ou seja, caminho candidato inicial é

$$\mathbf{c'} = (0,0,0,0)$$

e sua métrica, que será utilizada como limiar inicial $\lambda$, tem o valor

$$\lambda = 0.6*0.4*0.1*0.5 = 0.012$$

Num próximo passo o algoritmo seleciona uma seqüência de informação candidata **u** alterando o símbolo de informação da posição menos confiável e calcula sua métrica

$$\mathbf{u} = (0,1) \qquad \lambda_u = 0.6*0.3 = 0.18$$

Desde que $\lambda_u > \lambda$ o algoritmo prossegue calculando a métrica $\lambda'_u$ dos 3 primeiros símbolos da palavra código definida por **u**, dados por 0,1 e $\alpha$

$$\lambda'_u = \lambda_u * 0.5 = 0.18*0.5 = 0.09$$

e finalmente desde que $\lambda'_u$ não é maior que $\lambda$ não é necessário calcular a métrica final $\lambda''_u$ da palavra código $\mathbf{c}_u = (0,1,\alpha,\beta)$, definida por **u** e dada por

$$\lambda''_u = \lambda'_u * 0.1 = 0.009.$$

Como $\lambda'_u < \lambda$ a seqüência **u** é substituinda pela nova seqüência candidata, obtida agora considerando o segundo símbolo mais provável na posição menos confiável

$$\mathbf{u} = (0,\alpha) \qquad \lambda_u = 0.6*0.2 = 0.12$$

Se prosseguirmos, como feito para o caso de **u** = (0,0), obteremos uma métrica final e uma palavra código associada ao vetor de informação **u** dados por

$$\mathbf{c}_u = (0,\alpha,\beta,1) \qquad \lambda''_u = 0.6*0.2*0.3*0.2 = 0.0072$$

que é menor que a métrica inicial $\lambda$.

O próximo passo do algoritmo é considerar **u** = $(0,\beta)$, ou seja, o algoritmo seleciona como próximo candidato o símbolo menos provável da posição menos confiável da palavra recebida. Neste caso a métrica é dada por

$$\mathbf{u} = (0,\beta) \qquad \lambda_u = 0.6*0.1 = 0.06$$

Desde que $\lambda_u < \lambda$ não precisamos determinar $\lambda'_u$, associada aos três primeiros símbolos da palavra código definida por **u** = $(0,\beta)$ e dada por

$$\lambda'_u = \lambda_u * 0.1 = 0.06*0.1 = 0.006.$$

Neste ponto o algoritmo elimina **u** = $(0,\beta)$ e seleciona uma nova seqüência de informação, agora considerando símbolos na segunda posição menos confiável da palavra recebida. A seleção continua até que todas as 16 palavras código sejam testadas. Após o que obtemos **c'** = (0,0,0,0) como a palavra código de métrica máxima sendo pois a palavra decodificada. A fig. (3.4) mostra os caminhos que foram testados, juntamente com suas métricas, no processo de decodificação ora descrito. Observe que nem todos os caminhos são pecorridos até o seu final.

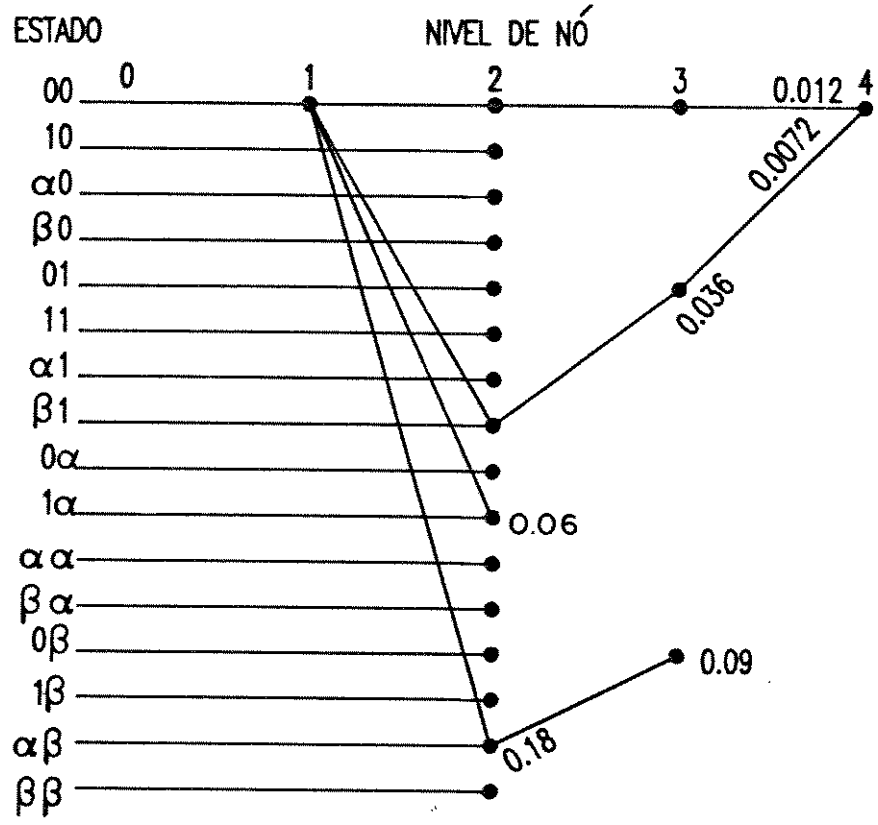


Fig. 3.4 - Decodificacão de uma palavra código do código (4,2) sobre GF(4).

### 3.3.2. Códigos Internos Convolucionais

Quando o código interno for um código convolucional, o decodificador externo utilizará o algoritmo de Viterbi com métrica dada por (3.30). Neste caso o processo de transferência de confiabilidade se dá através da probabilidade de erro para cada dígito em cada símbolo constituinte de uma palavra código do código externo, ou seja,

$$p_{ij} = P\{x_{ij}^{(t)} \neq \hat{x}_k / r_i\}$$

onde $x_{ij}$ é o dígito correto, $\hat{x}_{ij}$ o dígito estimado pelo decodificador interno

e $\mathbf{r}_i$ o vetor recebido. Assim, $p_{ij}$ é a probabilidade de $\hat{x}_{ij}$ ser decodificado incorretamente. A métrica para o decodificador externo é dada por

$$M_t = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} x_{ij}^{(t)} \hat{x}_{ij} L_{ij} \tag{3.39}$$

onde

$$L_{ij} = \ln \left[ \frac{1 - p_{ij}}{p_{ij}} \right]$$

que é ótima para o caso binário, segundo o critério de Máxima Verossimilhança.

Já vimos que este esquema se apresenta bem mais simples que aquele que utiliza a métrica de Hartmann e Rudolph. Já vimos, também, que ambos são equivalentes quando atuam sobre GF(2).

Contudo, observe que (3.7) terá valor nulo quando não houver erro a ser corrigido pelo decodificador externo . Assim, temos que para altas razões sinal-ruído o algoritmo de Battail torna-se muito mais rápido uma vez que somente ocorre decodificação interna, fato que não ocorre quando se utiliza o algoritmo de Viterbi [1], que necessita sempre de ambos os decodificadores. Neste trabalho, todavia, não estamos considerando algoritmos de Viterbi com complexidade reduzida.

Logo, vê-se que embora o esquema que emprega códigos convolucionais no código interno simplifique o decodificador interno, o seu desempenho medido pelo tempo de processamento deve ser inferior, em relação ao algoritmo de Battail, quando ambos atuam num mesmo esquema concatenado.

Existe, contudo, uma limitação do esquema de Hagenauer já que o mesmo necessita que os símbolos binários decodificados sejam descorrelacio-

nados, ou seja, necessita de um par embaralhador/desembaralhador, fato que não ocorre para o esquema que utililiza a métrica de Hartmann e Rudolph.

Finalmente, podemos contornar o problema de aumento do tempo computacional quando se usa o algoritmo de Viterbi no decodificador externo, simplesmente utilizando (3.30) em (3.29) e fazendo uso do algoritmo de Battail para o código externo,

$$Z_{hr}(c) = \sum_{j=1}^{n_1} \sum_{i=1}^{n_2} \ln \frac{P(c_{ij}/r_i)}{P(c^*_{ij}/r_i)} = -\sum_{j=1}^{n_1} \sum_{i=1}^{n_2} \ln \left[\frac{1 - p_{ij}}{p_{ij}}\right] = \sum_{j=1}^{n_1} \sum_{i=1}^{n_2} L_{ij}^{(-1)}$$

onde

$$L_{ij} \leftarrow \frac{1}{\alpha} \ln \left[\frac{1 + e^{(\alpha \hat{L}_{ij} + \Delta)}}{e^{\Delta} + e^{\alpha \hat{L}_{ij}}}\right]$$

$\Delta \stackrel{\Delta}{=}$ diferença de métricas convergindo num estado $S_k$

$\alpha = 4\, d_{free} . E_s/N_o$

Observe que neste caso também podemos utilizar a versão modificada do algoritmo de Battail, visto que os dígitos binários, juntamente com suas probabilidades, são produzidos individualmente.

Assim, mesmo sendo esquema de Hagenauer mais simples, em relação ao cálculo da métrica, ele perde em desempenho computacional devido à dependência dos dígitos binários decodificados, pois necessita de um par embaralhador/desembaralhador, e perde também devido à impossibilidade de se decodificar dígito a dígito.

## 3.4. CONCLUSÃO

Como vimos, existem várias maneiras de se combinar técnicas de transferências de confiabilidade. Somente as técnicas que supõem a independência entre os dígitos decodificados pelo código interno, podem não apresentar resultados coincidentes com o decodificador de correlação completa. O próximo capítulo apresenta resultados de algumas simulações utilizando códigos de bloco multiníveis.

# CAPÍTULO 4

# RESULTADOS DE SIMULAÇÃO

## 4.1 INTRODUÇÃO

Neste capítulo avaliamos o desempenho, medido através da probabilidade de erro de bit, do algorítmo de Battail modificado descrito no Capítulo 3. Para tanto, utilizamos esquemas em que um código externo é concatenado a um outro interno e uma modulação tal que o conjunto de símbolos transmitidos através do canal permite o mapeamento do bit "0" no símbolo -1 e do bit "1" no símbolo +1. Apresentamos uma versão sub-ótima do algoritmo de Battail juntamente com um exemplo que avalia o desempenho do mesmo, medido pela probabilidade de erro de bit, em canal gausiano.

Serão utilizados dois esquemas de códigos contenados. No primeiro será utilizado um código externo de Reed-Solomon (7,3) concatenado a um código interno binário (6,3). No segundo esquema de códigos concatenados utilizaremos o mesmo código externo do primeiro esquema concatendo a um código interno binário (4,3).

O objetivo destas duas simulações é avaliar o desempenho da métrica de Battail quando utilizada na decodificação de códigos concatenados e compará-lo com o desempenho ótimo obtido pela correlação completa. Para tanto utilizaremos a métricas de Battail e da correlação completa na decodificação dos códigos concatendos.

Também será apresentado um terceiro esquema em que concatenamos um código externo de Reed-Solomon (15,12) a um código interno binário de Hamming (7,4) com o intuito de avaliar o desempenho do algoritmo de Battail quando utilizado para decodificação de códigos relativamente longos. Neste esquema utlizaremos, na decodificação, o algoritmo de Battail sub-ótimo, que será convenientemente apresentado no item 4.3.

Em todos os esquemas de códigos concatenados o canal utilizado será um canal AWGN e o método de simulação será o de Monte Carlo.

## 4.2 SIMULAÇÃO DO ALGORITMO DE BATTAIL MODIFICADO

Considere um esquema de códigos concatenados com um o código externo da classe Reed-Solomon e definido pelo polinômio

$$g(x) = x^4 + \alpha^3 x^3 + x^2 + \alpha x + \alpha^3,$$

onde $\alpha$ é um elemento primitivo de $GF(2^3)$, trata-se de um código de parâmetros (7,3) que possui distância mínima 5, e com um código interno binário, de parâmetros de (6,3) e definido pela matriz de paridade **H** dada por

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Utilizando o método de Monte Carlo para determinar a probabilidade

de erro de bit no decodificador externo, ou seja, comparando a saída do decodificador externo com a palavra transmitida e contando o número de bits em erro para um número suficientemente elevado de palavras decodificadas, obtemos as curvas da Fig. (4.1). Observe um ganho de aproximadamente 2.0 dB da decodificação com saída ponderada sobre a decodificação abrupta no segundo estágio para uma probabilidade de erro de bit ( $P_{eb}$ ) de $10^{-4}$. Pode-se notar, nesta figura, uma diferença de desempenho de aproximamamente 0.4 dB entre o esquema que utiliza a métrica de Hartmann e Rudolph e o esquema que utiliza a métrica da correlação completa (onde codificador interno é palavra a palavra) confirmando o que já fora afirmado anteriormente.

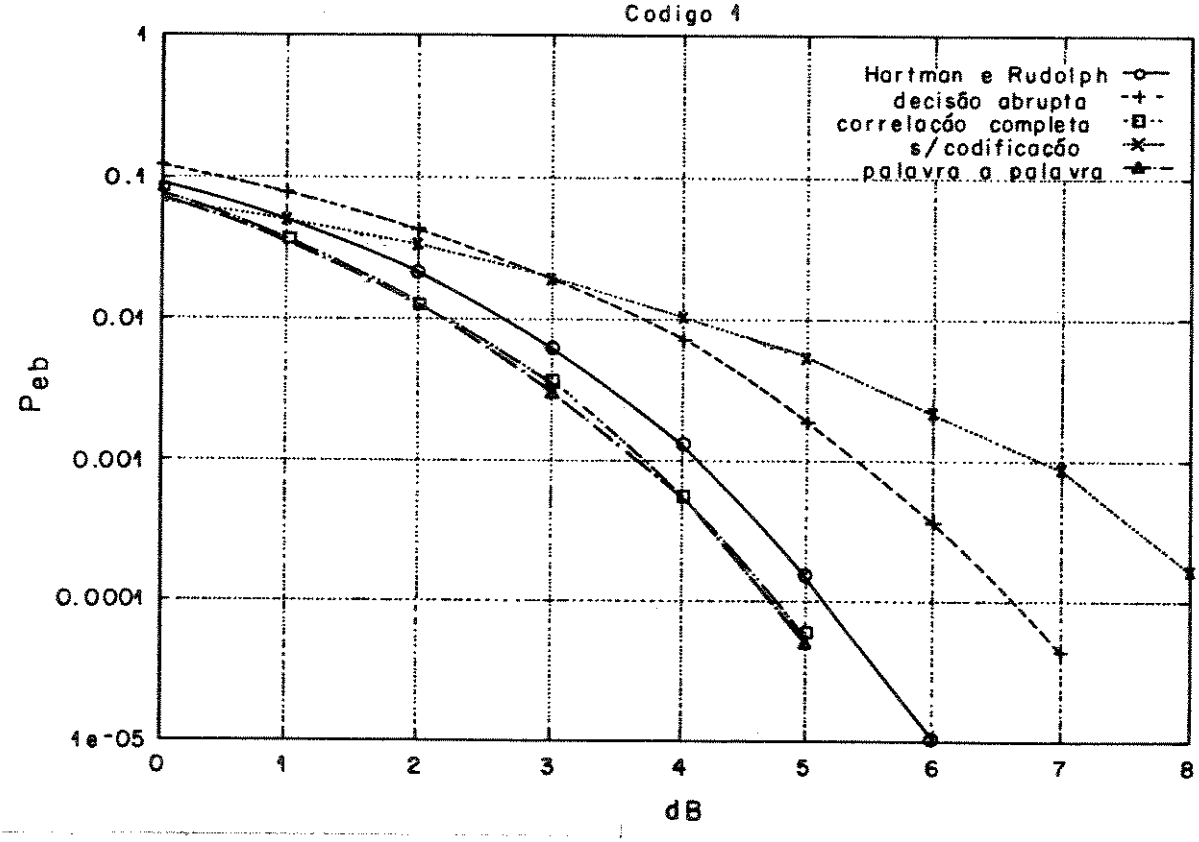


Fig. 4.1 - Desempenho do código RS (7,3,5) concatenado ao código (6,3)

Se, da mesma forma, concatenamos um código de Reed-Solomon (7,3,5) a um código (4,3) binário, com um dígito de verificação de paridade, obtemos as curvas da fig. (4.2). Novamente, observa-se aproximadamente 2.0 dB de ganho do esquema que utiliza decodificação suave no segundo estágio sobre o esquema que utiliza uma decodificação abrupta no segundo estágio. Contudo, a diferença de desempenho entre o esquema que utiliza a métrica de Hartmann e Rudolph e o esquema que utiliza um decodificador interno palavra a palavra é de apenas 0.2 dB, isto porque o código (4,3) possui taxa de codificação superior à do código (6,3). Observe que o aumento da taxa de codificação faz a métrica de Hartmann e Rudolph se aproximar de uma função linear.

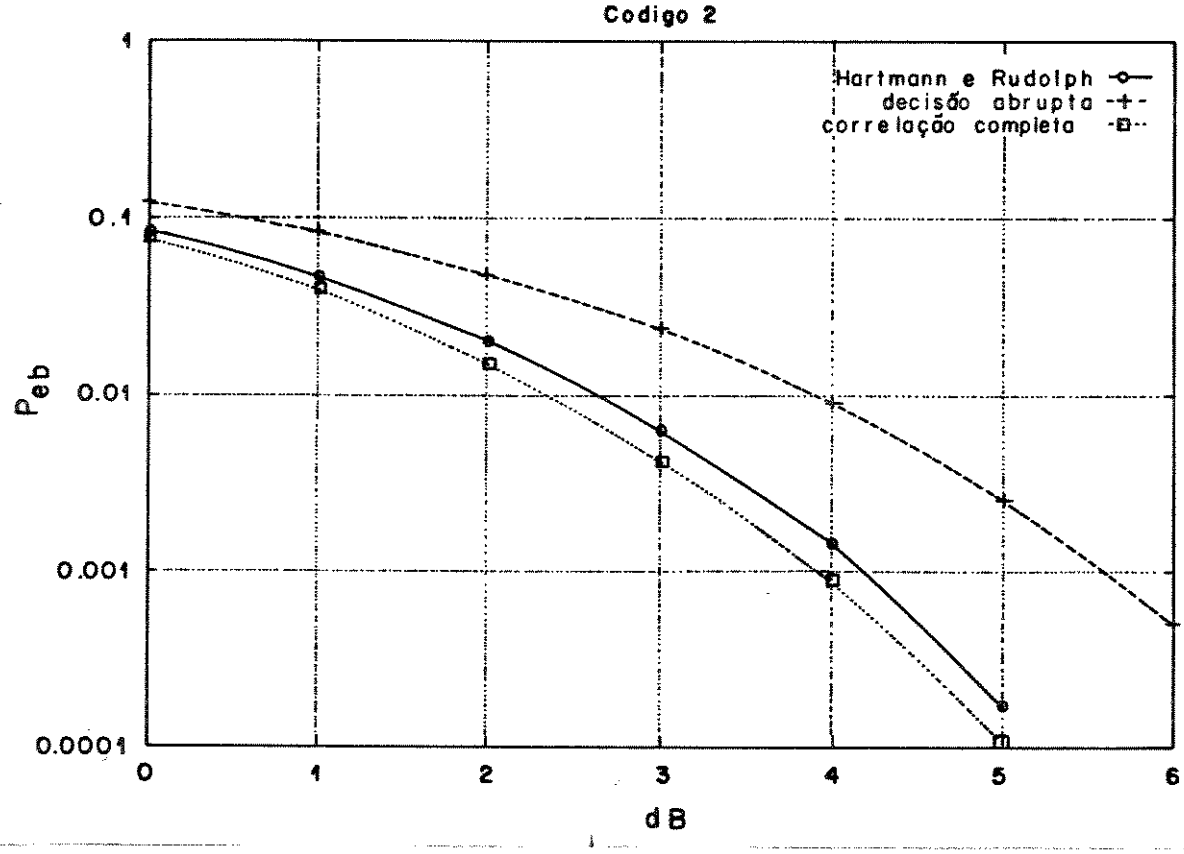


Fig. 4.2 - Desempenho do códgo (7,3,5) concatenado ao código (4,3)

## 4.3 ALGORÍTMO DE BATTAIL MODIFICADO SUB-ÓTIMO

### 4.3.1 Introdução

O algoritmo de Battail apresentado no capítulo anterior (item 3.3) quando aplicado na decodificação de códigos longos apresenta problemas de tempo de processamento, uma vez que a técnica de ordenação utlizada naquele algoritmo supõe que o código externo tenha um número reduzido de palavras códigos. Com o objetivo de solucionar este problema apresentamos um versão do algoritmo de Battail para decodificação de códigos concatenados em que é possível abortar a decodificação sem a necessidade de testar todas as palavras código.

Com esta versão do algoritmo de Battail é possível determinar o número de caminhos na treliça como uma função da probabilidade de erro de bit medida na saída do decodificador externo, de modo que seja possível determinar, através de simulações, o número adequado de caminhos de acordo com o desempenho requerido. Vale a pena lembrar que quanto maior for o número de caminhos na treliça utlizado maior será o tempo de processamento.

### 4.3.2 Algoritmo de Battail sub-ótimo

Seja $P_j(c_{ik})$, $c_{ik} \in GF(2^m)$, $i, \quad j = 1, \ldots, n_1$ e $k = 1, \ldots, 2^m$, m inteiro, as probabilidades de ocorrência de cada um dos dígitos decodificados pelo decodificador interno. Estas probabilidades podem ser obtidas através de um decodificador interno dígito a dígito, como descrito no capítulo anterior.

Considere uma ordenação tal que $M_j = P_j(c_{ij}) >\ldots> P_j(c_{iq})$, $q = 2^m$, m inteiro, e $M_j > M_k$ se $j > k$.

Defina $\lambda_i = M_i^{-1}$

Defina $Z_i(c_i) = \log_e \lambda_i P_i(c_i)$ , $c_i \in GF(2^m)$, e

$$Z_i(c) = \sum_{j=1}^{i} Z_i(c_i) \ , \ i=1, \ ..., \ n_1$$

Um algoritmo sub-ótimo, porém relativamente eficiente, pode ser obtido do algoritmo de Battail modificado. Os passos deste algoritmo podem ser obtidos da seguinte versão do algortimo de Battail modificado

1 - Determine $u = (P_1^{-1}(M_1), P_2^{-1}(M_2), ..., P_{k_1}^{-1}(M_{k_1}))$

2 - Fixe limiar $\alpha = Z_{n_1}(u)$. $J \leftarrow 0$.

3 - Se $\alpha = 0$ então vá para 11

4 - $J \leftarrow J + 1$. $L \rightarrow 0$. Se $J = q^{k_1}$ então vá para 11

5 - $L \leftarrow L + 1$. Se $L = q^J$ então vá para 4

6 - Faça $u^1$ uma nova permutação dos J símbolos menos confiáveis de u

7 - Se $Z_{k_1}(u^1) < \alpha$ então vá para 5

8 - Calcule $Z_{n_1}(u^1)$

9 - Se $Z_{n_1}(u^1) < \alpha$ então vá para 4

10 - Fixe novo limiar $\alpha = Z_{n_1}(u^1)$ e faça $u = u^1$. Vá para 4

11 - u é a seqüência decodificada e fim.

Obviamente, este algoritmo necessita da representação em treliça do código externo para sua implementação. Observamos que a ordenação inicial implica que os valores de métrica sejam decrescentes à medida que a decodificação se desenvolve. Por isso, é possível eliminar alguns caminhos simples-

mente testando no passo 9, se a métrica $\alpha$ é inferior a algum valor dado por uma probabilidade de ocorrência de símbolo, ou seja, podemos eliminar todos os caminhos que apresentam símbolos confiáveis em posições em que a métrica possa ser desprezada.

Por exemplo, observe o resultado de simulação apresentado na fig. (4.3). Nesse gráfico obtivemos o desempenho com canal gaussiano e modulação antipodal, do código (15,12) de Reed-Solomon concatenado ao código (7,4) binário e cíclico. A curva correspondente ao número de caminhos 4096 (indexada por uma cruz) foi obtida considerando-se corretos os 8 símbolos de informação mais confiáveis, o que corresponde a abortar a decodificação sempre que o limiar for inferior probabilidade de ocorrência do 4º símbolo de informação menos confiável.

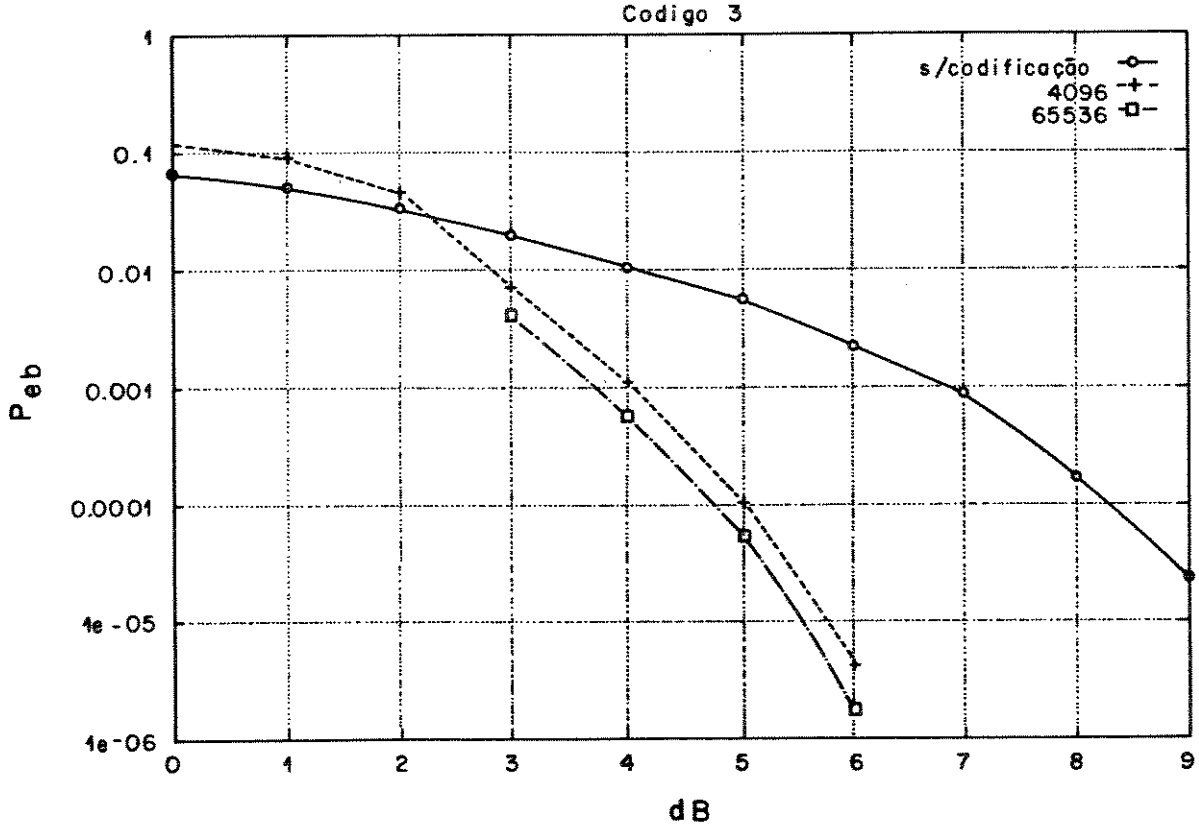


Fig. 4.3 - Desempenho do código concatenado (105,48)

A curva associado ao número 65536 (indexada por um quadradao) foi obtida da mesma forma, só que consideramos os 5 símbolos menos confiáveis e o teste foi efetuado com respeito ao 5º símbolo menos confiável. O passo 7 pode ser decomposto de modo a permitir um teste para cada dígito.

O número de caminhos deve ser determinado de acordo com um compromisso entre o tempo de processamento e o ganho de codificação. Na figura (4.4) temos uma comparação entre os desempenhos, medidos pela probabilidade de erro de bit, dos decodificadores que utilizam 4, 10 e 16 caminhos para decodificar o código de Hamming (7,4).

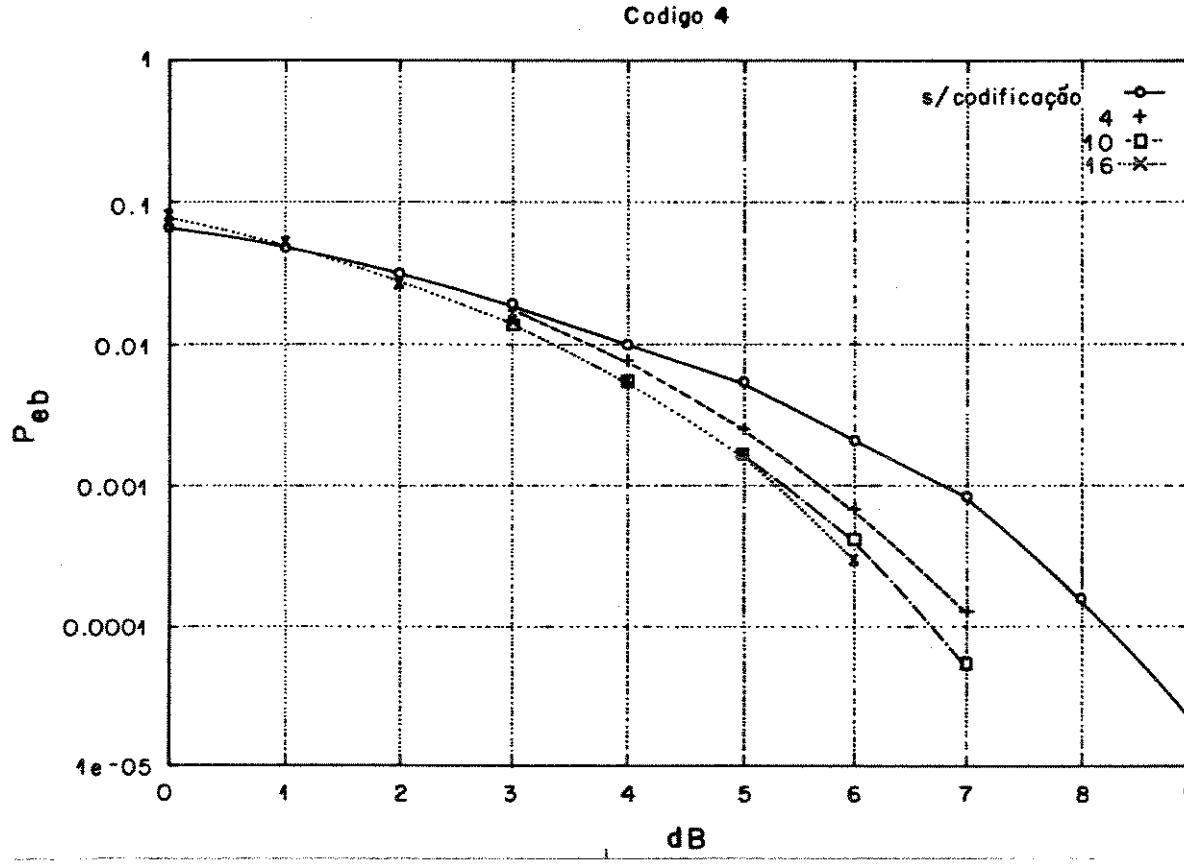


Fig. 4.4 - Decodificação do código (7,4)

## 4.4. PROPOSTAS FUTURAS

O presente trabalho se baseou em uma estrutura constituída por dois códigos concatenados. A extensão para um número maior de códigos pode ser feita simplesmente codificando cada símbolo da palavra código do código externo com um código diferente. Este esquema permite uma proteção desigual para cada símbolo e é possível utilizar-se tanto a métrica de Hartmann e Rudolph quanto a métrica de Hagenauer na decodificação interna.

Pode-se também estudar o comportamento destas estruturas quando se utilizam códigos TCM no codificador interno para um ou mais códigos internos. Também neste caso ambas as métricas pode ser utilizadas.

## 4.5. CONCLUSÃO FINAL

Em Battail [6] são apresentados alguns resultados de simulação quando se utiliza o algoritmo de Battail, onde são observados ganhos de aproximadamente 2,0 dB, e uma completa avaliação do desempenho computacional do mesmo.

Wolf [10] e Battail [1] apresentam duas técnicas de construção de uma treliça para códigos de bloco. Em Wolf [10] o objetivo é implementar o algoritmo de Viterbi e em Battail [7] o objetivo é implementar o algoritmo de Battail.

Em Rocha [9] é apresentada uma nova classe de códigos de máxima separação que, como os códigos de Reed-Solomon, também podem ser utilizados

como códigos externos. Trata-se de códigos de máxima separação com diferente parametrização.

Em Brine [8] é apresentado um esquema de decodificação em que um código de Reed-Solomon (15,9) é concatenado a um código (7,4) binário, e cíclico. A decodificação é suave inter-estágios, porém, o algoritmo só se aplica a código externos cíclicos. Além disso, poucos detalhes são apresentados sobre o algoritmo proposto.

Battail [7] apresenta um outra forma de se obter as probabilidades de ocorrência dos símbolos decodificados por um código interno em um sistema de códigos concatenados. Contudo, os resultados só se aplicam para o caso de códigos internos convolucionais e sofrem das mesmas limitações observadas quando da análise do decodificador que utiliza a métrica de Hagenauer, ou seja, necessita de uma estrutura que descorrelate os dígitos decodificados. Obtém-se um ganho de aproximadamente 1.5 dB quando esta técnica é utilizada em um esquema em que um código convolucional interno é concatenado a um outro código convolucional externo que efetua uma decodificação suave.

Em Hagenauer [3] é apresentado um esquema em que um código convolucional interno é concatenado a um código de bloco externo. É obtido um ganho de aproxidamente 1.5 dB da decodificação com saída ponderada sobre a decodificação abrupta.

Observamos que a técnica apresentada neste trabalho generaliza o que fora apresentado em Battail [7], onde foi sugerido um algoritmo para decodificação com decisão suave e saída ponderada de códigos convolucionais concatenados, sendo que o algoritmo ora proposto além de poder ser usado tanto para decodificação de códigos convolucionais concatenados quanto para decodificação de códigos de bloco concatenados se apresenta num forma que

permite o controle do desempenho computacional em função da otimalidade, ou seja, é possível determinar o desempenho computacional do algoritmo para uma dada probabilidade de erro.

Também sugerimos uma ordenação de métricas que simplifica e generaliza o que fora apresentado em Battail [1], onde somente o caso binário foi estudado. Na realidade, é esta ordenação que permite o controle do desempenho a partir da probabilidade de erro.

Finalmente, vemos que é possível projetar sistemas concatenados em que há transferência de informação entre os estágios de decodificação sem que haja um aumento significativo na complexidade do sistema, uma vez que fora mantida a estrutura básica dos decodificadores de cada um dos estágios de decodificação. Por este motivo, podemos afirmar que a complexidade do esquema de decodificação de códigos concatenados proposto é aproximadamente igual à soma das complexidades dos esquemas de decodificação utilizados em cada estágio. Em outras palavras, não há dependência entre as complexidades dos decodificadores em cada estágio.

# BIBLIOGRAFIA


[1] G. Battail, "Decogage Ponderé Optimal des Codes Lineares en Blocs I-Emploi Simplifié di Diagrams du Trellis", *Annales des Télécomunications*, Tomo 38 nºs 11-12, Novembro-Dezembro 1983.

[2] C.R.P. Hartman e L.D. Rudolph, "An Optimum Symbol-by-Symbol Decoding Rule for Linear Codes", *IEEE Transactions on Information Theory*, vol. IT-22 Nº 6, Setembro 1976.

[3] J. Hagenauer e P. Höher, "A Viterbi Algorithm with soft-Decisions Output and Applications", *Anais Globecom'89*, vol. 27, 30.11.89, Dalas - Texas.

[4] G.C. Clark e J.B. Cain, "Error Correction Coding for Digital Communications *Plenum Press*, New York, 1981

[5] G. Battail, M.C. Decouvelaere e P. Godlewski, "Replication Decoding", *IEEE Transaction on Information Theory*, vol. IT-25, Nº 3, Maio 1979.

[6] G. Battail e J. Fang, "Décogage Ponderé Optimal des Codes Lineares en Blocs II - Analyse et Résults de Simulation", *Annales des Télécomunications*, Tom 41, Nºs 11-12, Novembro-Dezembro 1986.

[7] G. Battail, "Pondérations des Symboles Décodes par L'algorithme de Viterbi" *Annales des Télécomunications*, Tomo 42, Nºs. 1-2, Janeiro-Fevereiro, 1987.

[8] A. Brine e P. Farrel, "Study of Reduced Complexity Concatenated Coding Schemes", artigo final ao ESA/ESTEC, Junho 1988.

[9] V.C. da Rocha, R.M.C. de Souza e P.G. Farrel, "Multilevel Pseudocyclic Codes", *Journal of Information & Optimization Sciences*, vol. 11, nº 1, pp 101-106, 1990.

[10] J.K. Wolf, "Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis", *IEEE Transactions of Information Theory*, vol. IT-24, nº 1 Janeiro 1978.

[11] G. Battail, "Coding for the Gaussian Channel: The Promise of Weighted-Output Decoding", *International Journal of Satellite Communications*, vol 7 (1989).

[12] F. Taleb e P.G. Farrel, "Soft Minimum Weight Decoding of Reed-Solomon Codes", *IEEE International Symposium on Information Theory*, Kobe, Japan, Junho, 1988.

[13] W. Wesley Peterson e E.J. Weldon, Jr, "Error-Correcting Codes", The MIT Press, 1972.

[14] Shu Lin e Daniel J. Costello, Jr., "Error Control Coding: Fundamental and Applications", Prentice-Hall, 1983.

# APÊNDICE A

# ESTRUTURAS ALGÉBRICAS I

## 1. INTRODUÇÃO

Uma revisão da teoria algébrica é apresentada no sentido de tornar o presente trabalho um pouco mais auto-suficiente. Trata-se de conceitos básicos de álgebra moderna largamente utilizados na teoria de codificação [4], [13], [14].

## 2. GRUPOS

Seja G um conjunto qualquer e * uma operação sobre os elementos deste conjunto. Sejam ainda os seguintes axiomas:

(a) Se $g \in G$ e $h \in G$ , então $g * h \in G$

(b) $\forall$ g, h e $\ell \in G$ , então $(g * h) * \ell = g * (h * \ell)$

(c) $\exists$ e $\in G$ tal que $e * g = g \quad \forall g \in G$ .

(d) $\forall g \in G \; \exists h \in G$ , tal que $h * g = e$ .

O conjunto G, juntamente com uma operação * que satisfaça os axiomas (a), (b), (c) e (d) sobre G, é dito um grupo e denotado <G,*>.

Por exemplo, considere o conjunto G = {0, 1} e a operação sobre G

| * | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

Este conjunto G junto com a operação * é um grupo. Se o grupo <G,*> satisfaz ao axioma

(e) $\forall\ g, h \in G,\ g * h = h * g.$

dizemos que <G,*> é abeliano.

## 3. ANÉIS

Seja <G,*> um grupo e defina • uma operação sobre G que satisfaça (a), (b) e (c) e suponha ainda que <G,*> é abeliano. O grupo <G,*> assim definido juntamente com a operação "•" é dito um anel e será representado por <G,*,•>. O anel <G,*,•> é dito comutativo se o axioma (e) for satisfeito para a operação "•".

Por exemplo, o conjunto {0, 1} juntamente com as operações

| * | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| • | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

é um anel.

## 4. CORPOS

Seja <G,*,•> um anel comutativo e suponha que o axioma (d) também seja satisfeito por "•". Então, o anel <G,*,•> torna-se uma estrutura algébrica denominada um corpo.

Por exemplo, o conjunto {0,1} juntamente com as operações

| * | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| • | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

é um corpo.

## 5. SUBGRUPOS E GRUPOS FATORES

Seja H ⊆ G. Se <H,*> for um grupo dizemos que <H,*> é um subgrupo de <G,*>.

Suponha que os elementos de um grupo <G,*> sejam $g_1$, $g_2$, ... e que os elementos do subgrupo <H,*> de <G,*> sejam $h_1$, $h_2$, ... e considere o arranjo definido como segue: a primeira linha é formada pelos elementos de H, com a identidade aparecendo à esquerda e cada elemento aparecendo uma e só uma vez. O primeiro elemento da segunda linha é qualquer elemento que não apareceu na primeira linha e o resto dos elementos são obtidos multiplicando-se cada elemento do subgrupo por este primeiro elemento à esquerda. Similarmente, uma terceira, quarta e quinta linha são formadas e assim por diante, até que todos os elementos do grupo apareçam no arranjo. Assim,

$$
\begin{array}{ccccccc}
h_1 = 1 & & h_2 & h_3 & h_4 & \cdots & h_n \\
g_1 * h_1 = g_1 & & g_1 * h_2 & g_1 * h_3 & g_1 * h_4 & & g_1 * h_n \\
\vdots & & \vdots & \vdots & \vdots & & \vdots \\
g_m * h_1 = g_m & & g_m * h_2 & g_m * h_3 & g_m * h_4 & \cdots & g_m * h_m
\end{array}
$$

O conjunto de elementos formados por uma linha deste arranjo é denominado classe lateral (coset) à esquerda. Os elementos da primeira coluna são os líderes das classes laterais. Classes laterais à direita podem ser similarmente construídas. O arranjo assim obtido é denominado decomposição em classes laterais do grupo G.

Um subgrupo $\langle H,* \rangle$ de $\langle G,* \rangle$ é dito subgrupo normal se $\forall\ g \in G,\ h \in H,\ g^{-1} * h * g \in H$, onde $g^{-1}$ é tal que $g^{-1} * g = e$.

## 6. ESPAÇOS VETORAIS

Um conjunto V de elementos é um espaço vetorial sobre um corpo $\langle F,*,\cdot \rangle$ se satisfaz aos seguintes axiomas:

(A) $\langle V, + \rangle$ é abeliano (elementos do corpo F são escalares e os elementos de V são vetores)

(B) $\forall\ f \in F\ e\ v \in V$ $fv$ é bem definido

(C) $\forall\ u, v \in V\ e\ f \in F,\ f(u + v) = fu + fv$

(D) $\forall\ c, f \in F\ e\ v \in V,\ (c + f)\ v = cv + fv$

(E) $\forall\ c, f \in F\ e\ v \in V,\ (cf)v = c(fv)$ e $1v = v$ , onde 1 é o elemento identidade da operação de multiplicação de $\langle F,+,\cdot \rangle$.

Uma ênupla sobre um corpo é um conjunto ordenado de n elementos do corpo e é denotado por $(a_1, a_2, ..., a_n)$, onde cada $a_i$ pertence ao corpo. Adição de ênuplas é definida por

$$(a_1, a_2, ..., a_n) + (b_1, b_2, ..., b_n) = (a_1 + b_1, a_2 + b_2, ..., a_n + b_n)$$

e multiplicação por um elemento c do corpo é definida por

$$c(a_1, a_2, ..., a_n) = (ca_1, ca_2, ..., ca_n) .$$

Com estas duas definições pode-se facilmente verificar que o conjunto de todas as ênuplas sobre um corpo forma um espaço vetorial.

O elemento identidade do espaço vetorial será denotado por **0**, isto é

$$\mathbf{0} = (0, 0, ..., 0)$$

Seja V um espaço vetorial e $U \subseteq V$. Se U for também um espaço vetorial, então dizemos que U é um subespaço vetorial de V. Finalmente, uma combinação de k vetores $v_1, ..., v_k$ sobre um corpo F é a soma da forma

$$\mathbf{u} = \sum_{i=1}^{k} a_i v_i , \quad a_i \in F \ \forall \ i.$$

Se $\exists\ a_i \neq 0$, $a_i \in F$, $i = 1, ..., n$, tal que $\mathbf{u} = \mathbf{0}$, então $\{v_1, ..., v_k\}$ é dito linearmente dependente, senão, $v_1, ..., v_k$ é dito linearmente independente.

Sejam $v_i \in V$, $i = 1, ..., k$ com $\{v_1, ..., v_k\}$ linearmente independente. Quando

$$\forall\ u \in V\ \exists\ a_i \in F,\ i = 1, \ldots, k, \text{ tais que } \sum_{i=1}^{k} a_i v_i = u$$

então dizemos que $\{v_1, \ldots, v_k\}$ é uma base para V, e que a dimensão de V é k.

# APÊNDICE B

# ESTRUTURAS ALGÉBRICAS II

## 1. INTRODUÇÃO

Alguns conceitos importantes de álgebra moderna são apresentados neste apêndice. Tais conceitos podem ser vistos como extensões dos conceitos apresentados no Apêndice A [13], [14].

## 2. CORPOS DE EXTENSÃO

Seja o conjunto F = {0, 1, ..., p - 1} onde p é um número primo não nulo. Considere a função:

x mod p ≡ resto da divisão de x por p.

Assim, o conjunto F juntamente com as operações

$$x \oplus y \equiv (x + y) \bmod p$$

$$x \odot y \equiv (xy) \bmod p$$

forma o corpo $\langle F, \oplus, \odot \rangle \equiv F$

Pode-se mostrar que qualquer corpo finito é isomorfo a F, ou seja, é estruturalmente equivalente a F.

Agora, seja p(x) um polinômio definido sobre o corpo F. Considere ainda que p(x) é um polinômio irredutível, ou seja, não existe polinômio q(x) de grau maior que zero e menor que m, o grau de p(x), tal que $p(x) = q(x)s(x)$.

Seja F(x) o conjunto de todos os polinômios com coeficientes em F e ainda,

s(x) mod p(x) ≡ Resto da Divisão Polinomial de $s(x) \in F(x)$ por p(x).

Defina ainda, para $s(x), t(x) \in F(x)$,

$$( s(x) + t(x)) \bmod p(x) \equiv s(x) * t(x) \qquad (B.1)$$

$$( s(x) . t(x)) \bmod p(x) \equiv s(x) \times t(x) \qquad (B.2)$$

Assim, o conjunto F(x) munido das operações * e x forma o corpo de Galois $GF(p^m)$ que é um corpo de extensão de grau m (grau de p(x)) sobre GF(p). Por simplicidade faremos * ≡ + e x ≡ . , adição e multiplicação ordinárias respectivamente.

Este processo de construção é denominado adjunção de raízes, isto porque, se p(x) é primitivo em F, ou seja, se $\exists\ \alpha$ tal que $p(\alpha) = 0$ e $\alpha^t = 1$ se $t = m - 1$, $\alpha^t \neq 1$ se $0 < t < m - 1$, com $\alpha^0 \equiv 1$, então podemos representar cada elemento não nulo de F(x) através de uma potência de $\alpha$. Definimos um elemento de F(x) como sendo cada conjunto de polinômios {f(x)} tais que

$$t(x), s(x) \in \{f(x)\} \Leftrightarrow (s(x) - t(x)) \bmod p(x) = 0 \qquad (B.3)$$

Como exemplo vamos construir $GF(2^4)$.

Neste caso, tome $p(x) = x^4 + x + 1$ e seja $\alpha$ tal que $p(\alpha) = 0$. Assim, $m = 4$, e

$$\alpha^0 = 1$$
$$\alpha^1 = x$$
$$\alpha^2 = x^2$$
$$\alpha^3 = x^3$$
$$\alpha^4 = x + 1$$
$$\alpha^5 = x^2 + x$$
$$\alpha^6 = x^3 + x^2$$
$$\alpha^7 = x^3 + x + 1$$
$$\alpha^8 = x^2 + 1$$
$$\alpha^9 = x^3 + x$$
$$\alpha^{10} = x^2 + x + 1$$
$$\alpha^{11} = x^3 + x^2 + x$$
$$\alpha^{12} = x^3 + x^2 + x + 1$$
$$\alpha^{13} = x^3 + x^2 + 1$$
$$\alpha^{14} = x^3 + 1$$
$$\alpha^{15} = x^4 + x = x + 1 + x = 1 .$$

Portanto, vemos que os elementos de $GF(2^4)$ podem ser representados pelo conjunto de todos os polinômios de grau menor ou igual a 3 sobre GF(2).

## 3. IDEAIS DE ANÉIS

Seja A(x) o conjunto de todos os polinômios de grau n sobre um corpo F, que pode ser GF($p^m$), m ≥ 1.

Considere as operações B.1 e B.2 com p(x) = $x^n$ - 1. Assim, estas novas operações juntamente com o conjunto A(x) formam um anel, denominado anel de polinômios módulo p(x), e cada elemento a(x) deste anel é definido como em (B.3).

Considere a fatoração de $x^n$ - 1 = g(x) h(x). O conjunto de todos os múltiplos de g(x) é definido como sendo um ideal de A(x). Podemos equivalentemente definir um ideal especificando as raízes de g(x) em algum corpo de extensão em relação ao corpo base de g(x), ou seja, o corpo onde estão definidos os coeficientes de g(x).

A função mínima de um elemento α é o polinômio mônico de menor grau, sobre determinado corpo, que tem α como raiz, onde α pode ou não ser elemento do corpo base da função mínima.

Um polinômio p(x) é dito mônico quando o coeficiente não nulo associado à maior potência de x em p(x) é 1.