Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Codificadores Bit-Geometricamente Uniformes para Sistemas com Concatenação Serial

Autor: Manish Sharma
Orientador: Prof. Dr. Jaime Portugheis

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração Engenharia de Telecomunicações.

Banca Examinadora

Prof. Dr. Jaime Portugheis (Orientador), FEEC/UNICAMP
Prof. Dra. Sueli Irene Rodrigues Costa, IMECC/UNICAMP
Prof. Dr. Renato Baldini Filho, FEEC/UNICAMP

Campinas, SP
20 de Fevereiro de 2006

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA- BAE - UNICAMP

Sharma, Manish
Sh23c Codificadores bit-geometricamente uniformes para sistemas com concatenação serial. / Manish Sharma. – Campinas, SP: [s.n.], 2006.

Orientadores: Jaime Portugheis.
Tese (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas de telecomunicação. 2. Códigos de controle de erros (Teoria da informação). 3. Álgebra abstrata. I. Portugheis, Jaime. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título

Titulo em Inglês: Bit-geometrically uniform encoders for serially concatenated systems
Palavras-chave em Inglês: Serial concatenation, Convolutional codes, BGU encoders
Área de concentração: Engenharia de Telecomunicações
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Sueli Irene Rodrigues Costa e Renato Baldini Filho
Data da defesa: 20/02/2006

Resumo

Nesta dissertação abordamos o problema de como construir codificadores bit-geometricamente uniformes (BGU) para a utilização como codificadores internos em sistemas com concatenação serial de códigos. A utilização destes codificadores implica na facilidade de determinação de parâmetros necessários para a análise do desempenho dos sistemas. Há um grande controle sobre estes parâmetros no projeto destes codificadores utilizando o método descrito neste trabalho, o que sugere que bons codificadores e conseqüentemente bons sistemas podem ser obtidos desta maneira. Além disso, os códigos gerados por estes codificadores possuem a propriedade de uniformidade de erro de bit, o que facilita bastante sua análise.

Palavras-chave: Concatenação Serial, Códigos Convolucionais, Codificadores BGU.

Abstract

This thesis approaches the problem of building bit-geometrically uniform (BGU) encoders to be used as inner encoders in systems with serially concatenated codes. By using this type of encoders, certain parameters that are used to analyze the system's performance are easily determined. There is a great control over these parameters when building encoders using the method described in this work, suggesting that good encoders and subsequently good systems can be obtained. Besides, the codes generated by these encoders posses the uniform bit error property, that greatly facilitates their analysis.

Keywords: Serial Concatenation, Convolutional Codes, BGU Encoders.

Agradecimentos

Agradeço ao meu orientador Prof. Jaime Portugheis pela sua orientação.

Agradeço aos membros da banca examinadora pelas sugestões.

Agradeço aos meus pais e aos meus amigos pela motivação e suporte.

Agradeço ao CNPq e FAPESP pelo apoio financeiro.

Sumário

Lista de Figuras **xi**

Lista de Tabelas **xiii**

Glossário **xv**

Lista de Símbolos **xvii**

Trabalhos Publicados Pelo Autor **xix**

1 Introdução **1**
1.1 Motivação 2
1.2 Conceitos básicos 4
1.2.1 Canal de transmissão com ruído branco Gaussiano aditivo 4
1.2.2 Álgebra abstrata 5
1.3 Estrutura 8

2 Rotulamentos e Codificadores Bit-Geometricamente Uniformes **11**
2.1 Constelações Geometricamente Uniformes 11
2.2 Rotulamentos Isométricos e a Uniformidade de Erro de Bit 14
2.3 Rotulamentos BGU 15
2.3.1 Simetrias do Espaço de Hamming 16
2.4 Códigos Geometricamente Uniformes 17
2.5 Codificadores BGU 18
2.6 Rotulamentos não BGU com a UBEP 21
2.7 Síntese 24

3 Sistemas com Concatenação Serial **25**
3.1 Sistemas com Concatenação de Códigos 25
3.2 Limitantes de União para a Probabilidade de Erro de Bit em Sistemas com Concatenação Serial 27
3.2.1 Termos Dominantes do Limitante de União 32
3.3 Diretrizes de Projeto 38
3.4 Considerações sobre o Limitante de União 39
3.5 Síntese 40

4 Método para gerar codificadores internos BGU **41**
4.1 Algoritmo . . . 41
4.2 Considerações sobre o algoritmo . . . 52
4.3 Exemplo . . . 53
4.4 Síntese . . . 59

5 Análise de Resultados **61**
5.1 Apresentação de codificadores . . . 61
5.1.1 Formato de Apresentação . . . 61
5.1.2 Codificadores encontrados . . . 62
5.2 Limitantes de desempenho para os sistemas encontrados . . . 69
5.2.1 Cálculo do limitante de união . . . 70
5.2.2 Resultados . . . 73
5.3 Análise . . . 77
5.3.1 Influência da quantidade de memórias do código externo . . . 77
5.3.2 Influência de termos individuais no limitante de união . . . 78
5.3.3 Influência da taxa e quantidade de memórias do código interno . . . 80
5.3.4 Eficiência do Algoritmo . . . 80
5.4 Síntese . . . 80

6 Conclusões **81**

Referências bibliográficas **83**

Lista de Figuras

1.1 Sistema de comunicação básico. 1
1.2 Constelação 8-PSK (a) e 4-PSK (b). 3

2.1 Constelações GU: a-)$8PSK$ e b-)Treliça infinita regular 12
2.2 Regiões de decisão de 8-PSK 13
2.3 Possível Rotulamento BGU para a constelação 8-PSK 16
2.4 Possível rotulamento não BGU com UBEP para a constelação 8-PSK 22
2.5 Grupos geradores da constelação 8-PSK e geradores associados: a-)Z_8, b-)D_4 23

3.1 Sistema de Concatenação Paralela 26
3.2 Sistema de Concatenação Serial 26
3.3 Variação de $\gamma(1 - d_{e,l})$ 35

4.1 Grupo gerador da constelação 8 PSK 55
4.2 Diagrama de transições 55

5.1 Convergência de truncamento. 71
5.2 Limitantes para sistemas com o codificador C_{51}. 73
5.3 Limitantes para sistemas com o codificador C_{52}. 74
5.4 Limitantes para sistemas com o codificador C_{53}. 74
5.5 Limitantes para sistemas com o codificador C_{63}. 75
5.6 Limitantes para sistemas com o codificador C_{72}. 75
5.7 Limitantes para sistemas com o codificador C_{73}. 76
5.8 Limitantes para sistemas com o codificador C_{81}. 76
5.9 Limitantes para sistemas com o codificador C_{82}. 77
5.10 Influência de diversos termos no limitante de união para o sistema $C^e_{72} - C_{72}$, $N = 840$. 79
5.11 Influência de diversos termos no limitante de união para o sistema $C^e_{73} - C_{72}$, $N = 840$. 79

Lista de Tabelas

1.1 Tabela de operações para o grupo Z_8, grupo aditivo módulo 8 8
1.2 Tabela de operações para o grupo D_4, grupo diedral com 8 elementos 8

4.1 Geradores de H_5 54
4.2 Transições paralelas 54
4.3 Partições do espaço de Hamming 55
4.4 Geradores de 32 elementos em $2 \times 8PSK$ 56
4.5 Associação de geradores 56
4.6 Partições do espaço de Hamming que retornam ao estado inicial 57
4.7 Vetores de distâncias obtidos para os candidatos ao grupo A 58
4.8 Classificação 59
4.9 Grupo A final. 59

5.1 Formato de apresentação de codificadores 62
5.2 Codificadores encontrados e suas características. 63
5.3 Codificador com uma memória C_{51}, taxa 5/6 64
5.4 Codificador com duas memória C_{52}, taxa 5/6 65
5.5 Codificador com três memória C_{53}, taxa 5/6 65
5.6 Codificador com três memória C_{63}, taxa 6/9 66
5.7 Codificador com duas memória C_{72}, taxa 7/9 67
5.8 Codificador com três memória C_{73}, taxa 7/9 67
5.9 Codificador com uma memória C_{81}, taxa 8/9 68
5.10 Codificador com duas memórias C_{82}, taxa 8/9 69
5.11 Códigos externos utilizados. 70

Glossário

AWGN Do inglês *Additive White Gaussian Noise*, ruído branco aditivo com densidade espectral constante.

BGU Bit-Geometricamente Uniforme

CCP Códigos com Concatenação Paralela

CCS Códigos com Concatenação Serial

CIOWEF do inglês *Conditional Input Output Weight Enumerating Function*, função enumeradora dos pesos de entrada e saída condicional.

FDD Função de Distribuição de Distâncias

GU Geometricamente Uniforme

IOWEF do inglês *Input Output Weight Enumerating Function*, função enumeradora dos pesos de entrada e saída.

MV Máxima Verossimilhança

PAM Do inglês *Pulse Amplitude modulation*, modulação por amplitude de pulso

PSK Do inglês *Phase Shift Key*, modulação por deslocamento de fase.

QAM Do inglês *Quadrature Amplitude Modulation*, modulação por amplitude em quadratura.

RSR Relação Sinal Ruído

TCM Do inglês *Trellis Coded Modulation*, modulação codificada por treliça.

UBEP Do inglês *Uniform Bit Error Property*, propriedade de uniformidade de erro de bit

Lista de Símbolos

η	-	Ruído aditivo branco com média 0.
Γ_S	-	Grupo com todas as simetrias de S.
ε	-	Rotulamento de uma constelação por um espaço de Hamming.
Σ_m	-	Conjunto ou grupo de estados de um codificador com m memórias
σ_e	-	Estado identidade.
$\mathbf{0}$	-	Seqüência binária que contém somente zeros.
$A^C_{w,d}$	-	Número de seqüências do código C com peso de entrada w e peso de saída d.
$A^C(W, D)$	-	Função de distribuição de pesos do código C.
$A^C(w, D)$	-	Função de distribuição de pesos do código C para seqüências com peso de entrada w.
$b_{i,k}$	-	i-ésimo bit de uma palavra de entrada no instante k
C_{H_k}	-	Conjunto dos geradores do grupo do espaço de Hamming com k bits, H_k.
C_G	-	Conjunto dos geradores do grupo G.
C_i	-	Código interno.
C_e	-	Código externo.
d_n	-	Distância Euclidiana de saída entre seqüências de entrada com distância de Hamming n.
D_4	-	Grupo diedral com 8 elementos, uma rotação de ordem 4 e uma simetria de ordem 2.
$d_{i,l}$	-	Distância livre Euclidiana do código interno.
$d_{e,l}$	-	Distância livre de Hamming do código externo.
d_{ef}	-	Distância livre efetiva.
$d_h(x, y)$	-	Distância de Hamming entre x e y.
$d_e(x, y)$	-	Distância Euclidiana entre x e y.
d_{min,α_M}	-	Menor distância Euclidiana associada ao maior expoente de N no limitante de união.
$e_{i,k}$	-	Estado da i-ésima memória binária no instante k, vale 0 ou 1.
E_0	-	Mapeamento entre F_0 e o espaço de Hamming correspondente.
F_0	-	Subgrupo de T com os ramos que saem do estado identidade.
F_{00}	-	Subgrupo de T com os ramos que saem do estado identidade e retornam ao mesmo.
G_S	-	Grupo gerador da constelação S.
G_{H_k}	-	Grupo gerador de H_k.
H_k	-	Espaço de Hamming com k bits.
m_e	-	Quantidade de memórias do código externo.
N	-	Tamanho do entrelaçador, em bits.
$P_b(e)$	-	Probabilidade de erro de bit.
R_C	-	Taxa do código C.
$\Re$	-	Espaço dos números reais.
T	-	Um dos possíveis grupos que representa uma seção de treliça de um código TCM.
W_T	-	Banda de transmissão
w	-	Peso de Hamming da seqüência de entrada do sistema.
$w_h(x)$	-	Peso de Hamming de x.
$w_e(x)$	-	Peso Euclidiano de y.
Z_8	-	Grupo aditivo módulo 8.

Trabalhos Publicados Pelo Autor

1. SHARMA, M. ; PORTUGHEIS, J.. *Procura de Codificadores BGU para utilização em Códigos com Concatenação Serial*. XXII SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES, 2005, Campinas. Anais do Simpósio Brasileiro de Telecomunicações, 2005, pp. 529-534.

Capítulo 1

Introdução

Um sistema de comunicação tem como objetivo reproduzir em alguma localidade uma mensagem que se encontra em outro lugar. Um transmissor no local de origem emite uma representação desta mensagem que, através de um canal de comunicação, chega ao receptor que se encontra no local onde se deseja ter esta mensagem, como pode ser visto na figura 1.1. O transmissor tem como função tornar a mensagem apropriada para ser transmitida através do canal utilizado, escolhendo um símbolo (ou seqüência de símbolos) de um conjunto apropriado. O receptor tem como função tentar reproduzir com a maior fidelidade possível a mensagem transmitida a partir do símbolo recebido. O receptor sabe da relação feita entre a mensagem e o símbolo escolhido pelo transmissor, mas não sabe exatamente qual foi o símbolo transmitido pois o canal o modifica, de modo que o que foi recebido é diferente do que foi transmitido. O símbolo transmitido pode ser modificado a ponto de ser confundido com outro, resultando até na escolha de uma outra mensagem por parte do receptor. Nestas situações, ocorre um erro de transmissão.

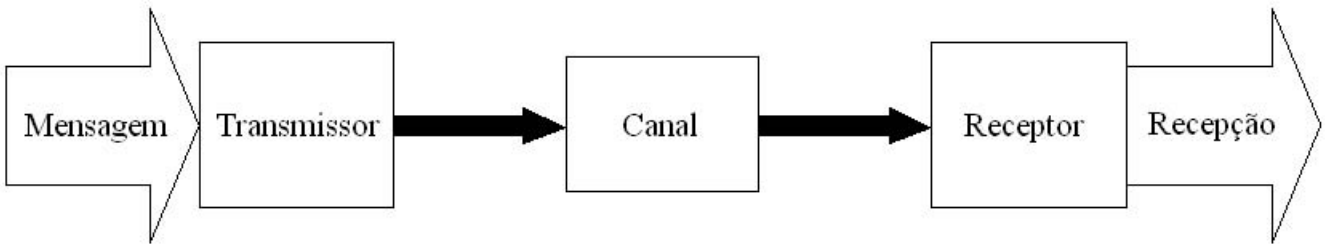

Fig. 1.1: Sistema de comunicação básico.

Nesta dissertação abordamos o problema do transmissor, que é o de como escolher da melhor maneira possível os símbolos a serem transmitidos através do canal para que a chance de se ocorrer um erro de transmissão seja a menor possível.

Um símbolo pode ser representado através de um vetor com n componentes, onde cada elemento

corresponde a algum valor que o símbolo assume quando é transmitido pelo canal. Podemos representar estes vetores como pontos num espaço Euclidiano n-dimensional. O conjunto de pontos obtidos através desta representação vetorial é chamado de constelação. Se utilizamos somente um subconjunto destes símbolos, estamos utilizando um código. A vantagem de se utilizar um código é que ele pode diminuir bastante a probabilidade de se ocorrer um erro de transmissão, melhorando assim o desempenho do sistema. A desvantagem é que, ao utilizarmos somente um subconjunto de todos os símbolos possíveis, perdemos um pouco da quantidade de informação que podemos transmitir. Além disso, precisamos saber quais símbolos utilizar e como associar estes símbolos com a mensagem (como codificador). A quantidade de informação, em bits, que a recepção de um dos símbolos de um conjunto pode transmitir, sendo todos os seus elementos equiprováveis, é $\log_2 a$, onde a é o número de elementos do conjunto. Quanto menor o valor de a, menos informação podemos transmitir.

Algumas constelações são geometricamente uniformes, conceito desenvolvido em [1] e utilizado em [2] para gerar códigos geometricamente uniformes [1]. Estes códigos tem a propriedade de que a probabilidade de se trocar um símbolo(ou seqüência) por outro independe do símbolo transmitido.

Codificadores Bit-Geometricamente Uniformes (BGU) vão um passo além, garantido, além da uniformidade no espaço Euclidiano, uma uniformidade também no domínio do espaço de Hamming utilizado. Isso resulta que, além dos símbolos, a probabilidade de erro de bit também independe da mensagem transmitida. Codificadores BGU também fornecem uma relação de distância de Hamming entre mensagens de entrada do transmissor(e genericamente, a distância do espaço de mensagens) e distância Euclidiana dos símbolos associados a estas, o que é útil ao se projetar sistemas com concatenação serial.

1.1 Motivação

Nas últimas décadas tem se estudado bastante a concatenação de códigos, principalmente depois que Berrou et al [3] apresentaram os códigos "Turbo". Estes códigos tem desempenho muito bom e próximos à capacidade do canal, com uma complexidade computacional razoavelmente baixa. Eles tem como característica principal a concatenação paralela de dois ou mais códigos menores através de entrelaçadores, que entralaçam os bits que entram em cada codificador. Os símbolos enviados ao canal são a combinação multiplexada dos símbolos individuais de cada código. Em [4], o desempenho dos códigos "Turbo"foi caracterizado através dos limitantes de união para a probabilidade de erro de bit, considerando um entrelaçador uniforme.

Em [5], a concatenação serial de códigos foi analisada. Um sistema com concatenação serial tem

[1] Este conceito será melhor definido no capítulo 2

no mínimo dois códigos: um código externo que recebe as informações e um código interno que é a interface para o canal. Os símbolos transmitidos pelo canal são os símbolos do código interno. A conclusão deste artigo foi que, em algumas situações, a concatenação serial pode ter desempenho melhor do que os códigos "Turbo", com uma complexidade semelhante, sendo este um bom motivo para o seu estudo. Este artigo também determinou quais parâmetros dos códigos interno e externo influenciam o desempenho do sistema [2]. Um dos problemas para se projetar códigos internos é o de garantir bons valores para estes parâmetros importantes, com a expectativa de que assim o desempenho do código seja bom. Uma maneira de se garantir isso é através de rotulamentos isométricos, onde a distância de Hamming (número de bits diferentes) associada a dois símbolos é função da distância Euclidiana entre os símbolos.

Uma constelação comum é a constelação $8-PSK$, mostrada na figura 1.2-a. Ela tem a vantagem de que, quando codificada de modo a transmitir 2 bits por símbolo, ter desempenho melhor do que um sistema não codificado com 4 sinais e com a mesma energia, que é a constelação $4-PSK$ mostrada na figura 1.2-b [6]. Estas constelações são geometricamente uniformes, e podem ser bem representadas através de grupos de álgebra abstrata.

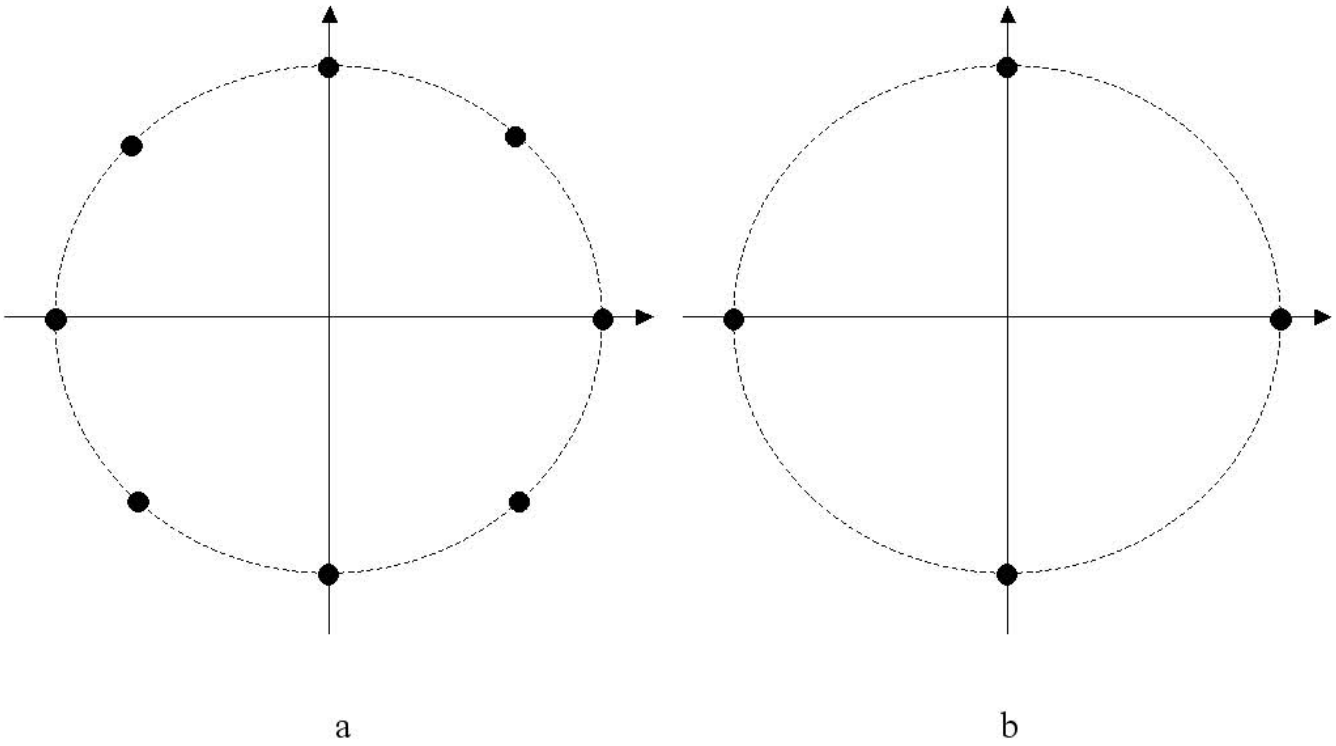

Fig. 1.2: Constelação 8-PSK (a) e 4-PSK (b).

A constelação $8-PSK$ não admite rotulamentos isométricos quando desejamos rotulá-la com 3

[2]Estes parâmetros serão mostrados no capítulo 3

bits. Uma solução foi proposta em [7], onde o conceito de uniformidade binária e geométrica (BGU, do inglês *Bit Geometrically Uniform*) foi apresentado. Neste rotulamento há uma relação entre o grupo associado à constelação e a grupo do espaço de Hamming utilizado para rotular os pontos. Este conceito foi utilizado com sucesso no mesmo artigo para projetar um codificador interno para um sistema com concatenação serial. O resultado forneceu um ganho de desempenho comparado com casos anteriores. Entretanto, uma maneira prática de se obter estes codificadores não foi proposta.

Nesta dissertação, abordamos o problema de como construir codificadores BGU a serem utilizados como códigos internos num sistema de concatenação serial. O trabalho desenvolvido aqui dá continuidade ao trabalho feito em [7], obtendo através do método desenvolvido mais codificadores BGU. Para isso, precisamos saber quais são as propriedades de codificadores BGU e quais são os parâmetros importantes dos codificadores internos. Com estas informações, desenvolvemos um algoritmo que nos permite obter codificadores internos bons dada uma constelação geometricamente uniforme e alguns parâmetros do código externo.

1.2 Conceitos básicos

1.2.1 Canal de transmissão com ruído branco Gaussiano aditivo

Seja $s_t[n]$ um vetor transmitido através de um canal discreto e sem memória que, na recepção, faz com que o sinal recebido $s_r[n]$ seja:

$$s_r[n] = s_t[n] + \eta[n] \tag{1.1}$$

onde $\eta[n]$ é um vetor aleatório. Este canal então é um canal com ruído aditivo, pois ele adiciona um valor aleatório a $s_t[n]$.

Se a distribuição de $\eta[n]$ for Gaussiana com densidade espectral constante e igual em todas as suas dimensões, então este canal será chamado de canal AWGN (do inglês *Additive White Gaussian Noise*), ou canal com ruído branco Gaussiano aditivo. Neste caso, a projeção do ruído em qualquer direção terá o mesmo comportamento do ruído numa única dimensão.

O ruído branco unidimensional com média 0 tem a seguinte função de densidade de probabilidade[8]:

$$p(\eta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\eta^2}{2\sigma^2}} \tag{1.2}$$

onde σ^2 é a variância da variável aleatória $\eta[n]$ e corresponde também à energia do ruído por intervalo de tempo, ou seja, a potência.

A função acumulativa da distribuição Gaussiana é:

$$F(\eta) = \int_{-\infty}^{\eta} p(x)dx \tag{1.3}$$

Se η tem média zero e variância unitária, podemos definir $Q(\eta)$, a função distribuição complementar:

$$Q(\eta) = 1 - F(\eta) = \frac{1}{\sqrt{2\pi}} \int_{\eta}^{\infty} e^{-r^2/2} dr \tag{1.4}$$

Se η tem média μ e variância σ^2, podemos escrever a seguinte equação:

$$P[\eta > \eta^*] = Q\left(\frac{\eta^* - \mu}{\sigma}\right) \tag{1.5}$$

onde $P[\eta > \eta*]$ é a probabilidade de uma variável η assumir um valor maior do que η^*.

Como a integral para o cálculo de $Q(x)$ não é facilmente calculada, podemos utilizar a aproximação 1.6, que é válida para grandes valores de x.

$$Q(x) \leq e^{-x^2/2} \tag{1.6}$$

Por causa desta aproximação, é conveniente utilizar a distância Euclidiana quadrática entre pontos, definida pela equação 1.7 para um espaço Euclidiano n-dimensional.

$$d_e(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} (x_i - y_i)^2 = \|\mathbf{x} - \mathbf{y}\|^2 \tag{1.7}$$

onde x_i é o i-ésimo componente de $\mathbf{x}$. O peso Euclidiano quadrático de $\mathbf{x}$, $w_e(x)$ é igual a $d_e(\mathbf{x}, \mathbf{0})$, onde $\mathbf{0}$ é o vetor com todos os seus componentes iguais a zero.

Por ser conveniente para o cálculo da probabilidade de erro de bit, todas as distâncias e pesos Euclidianos utilizadas nesta dissertação são distâncias Euclidianas quadráticas, a não ser quando especificado o contrário.

A aproximação 1.6 é útil para calcularmos a probabilidade de se trocar um símbolo por outro no receptor quando há presença de ruído aditivo branco.

1.2.2 Álgebra abstrata

Os conceitos de álgebra apresentados nesta seção são suficientes para o desenvolvimento desta dissertação. As definições de grupo, geradores e homomorfismos foram retirados de [9], com algumas modificações para simplificar a apresentação.

Grupos

Seja um conjunto S de elementos. Seja uma operação $\cdot$ definida como um mapeamento:

$$S \cdot S \rightarrow S$$

Este conjunto e esta operação formam um grupo G somente se as seguintes condições forem satisfeitas:

- Se a e b pertencem a S, o elemento $a \cdot b$ também pertence a S, onde $a \cdot b$ é o elemento obtido pela aplicação do operador sobre a e b, nesta ordem.
- A operação $\cdot$ é associativa, isto é, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- Existe um elemento e pertencente a S tal que $a \cdot e = e \cdot a = a$, para qualquer elemento a pertencente a S. O elemento e é chamado de elemento identidade ou simplesmente identidade.
- Para qualquer elemento a pertencente a S, existe um único elemento b tal que $a \cdot b = e$. Nestes casos, é conveniente utilizar a notação $b = a^{-1}$, e b é o inverso de a.

Não é necessário que $a \cdot b = b \cdot a$. Se isto for verdade para todos os elementos de G, então o grupo é abeliano ou comutativo.

Para simplicidade, omitimos o operador $\cdot$, denotando o produto $a \cdot b$ simplesmente por ab. Também podemos utilizar a notação $g \cdot H$ ou gH quando queremos multiplicar todos os elementos do subconjunto H pelo elemento g, seja pela direita ou pela esquerda, dependendo da notação utilizada.

Um subgrupo H de G é qualquer subconjunto de G que satisfaz as condições de um grupo. Se H é subgrupo de G, escrevemos $H \subseteq G$. Todo subgrupo deve conter o elemento identidade e. Um subgrupo H de G é um subgrupo normal se, para qualquer elemento g pertencente a G,

$$g \cdot H \cdot g^{-1} = H$$

Todos os subgrupos de grupos abelianos são normais.

Dado que H é um subgrupo de G, os cosets de H são subconjuntos de G obtidos através do produto de todos os componentes de H por um elemento de G, isto é, $x \cdot H$, onde $x \in G$.

O grupo quociente G/H é formado pelo conjunto de cosets de H. Os elementos de G/H são escritos na forma $N \times a$ e formam um grupo tal que $(N \cdot a) \cdot (N \cdot b) = N \cdot (a \cdot b)$.

Geradores

Se qualquer elemento do grupo G pode ser obtido através de repetidas aplicações do operador em cima de um subconjunto $(g_1, g_2, ..., g_n)$ de elementos de G, então este subconjunto contém os geradores de G. Se nenhum dos elementos deste subconjunto podem ser obtidos através da aplicação repetida do operador em cima dos outros elementos, então este subconjunto é o menor subconjunto que gera G. Qualquer elemento do grupo pode ser descrito através de uma combinação destes geradores. A notação $\langle g_1, g_2, ..., g_n \rangle$ denota o espaço gerado por $(g_1, g_2, ...g_n)$, isto é, todos os elementos de G que podem ser obtidos através de combinações dos elementos do subconjunto. O espaço gerado pelos geradores de um grupo é o próprio grupo. O conjunto $(g_1, g_2, ..., g_n)$ gera G se e somente se o menor subgrupo de G que contém $(g_1, g_2, ..., g_n)$ é o próprio grupo G.

A ordem de um gerador g é o menor número p tal $g^p = e$.

Homomorfismos

Sejam G e H dois grupos, cada um com uma operação. Uma função $f : G \rightarrow H$ é um homomorfismo se:

$$f(g_1 \cdot g_2) = f(g_1) \cdot f(g_2), \forall g_1, g_2 \in G \tag{1.8}$$

onde a operação $g_1 \cdot g_2$ é feita seguindo a regra do grupo G e a operação $f(g_1) \cdot f(g_2)$ é feita seguindo a regra do grupo H.

O elemento identidade de G deve ser mapeado por f para a identidade de H.

A composição de homomorfismos é um homomorfismo. Um homomorfismo $f : G \rightarrow H$ é um isomorfismo de existe um homomorfismo $g : H \rightarrow G$ tal que a função $f \circ g$ e $g \circ f$ são funções identidade. O homorfismo f é um isomorfismo se e somente se ele for bijetivo. Um isomorfismo de G em si mesmo é chamado de automorfismo.

Alguns grupos importantes

Como utilizaremos a constelação $8 - PSK$ com mais freqüência, é importante apresentarmos os grupos Z_8 e D_4, o grupo cíclico módulo 8 e o grupo diedral com 8 elementos, respectivamente. As tabelas indicam a operação do elemento da linha pelo elemento da coluna. O grupo Z_8 é abeliano, e não há problema em se trocar a ordem de operação. O grupo diedral, entretanto, não é abeliano, e a ordem dos elementos é importante.

O grupo aditivo módulo 8 tem a estrutura dada pela tabela 1.1. Ele possui um gerador de ordem 8. Como gerador podemos ter qualquer um dentre os seguintes elementos:1,3,5 ou 7.

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

Tab. 1.1: Tabela de operações para o grupo Z_8, grupo aditivo módulo 8

O grupo diedral tem a estrutura dada pela tabela 1.2. Ele possui dois geradores. Um deles pode ser um dentre qualquer um dos seguintes:1,3,5 ou 7. Este gerador terá ordem 2. O outro gerador deve ser ou 2 ou 6, e terá ordem 4.

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	7	4	1	6	3	0	5
3	3	6	5	0	7	2	1	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	3	0	5	2	7	4	1
7	7	2	1	6	3	6	5	0

Tab. 1.2: Tabela de operações para o grupo D_4, grupo diedral com 8 elementos

Ambos os grupos mencionados acima tem como subgrupos os grupos Z_4 e Z_2, os grupos aditivos módulo 4 e 2, respectivamente. A tabela de operações destes grupos pode ser extraída das tabelas acima. O grupo D_4 admite também o subgrupo Z_2^2, o produto cartesiano $Z_2 \times Z_2$. O grupo Z_8 não admite este subgrupo.

1.3 Estrutura

Esta dissertação esta dividida em 6 capítulos. Este é o primeiro, onde apresentamos o problema que será abordado e introduzimos alguns conceitos básicos. Os conceitos de constelações geometricamente uniformes e rotulamentos bit-geometricamente uniformes são apresentados no capítulo 2, onde também mostramos algumas propriedades provenientes desta uniformidade. No capítulo 3,

apresentamos a estrutura dos sistemas com concatenação serial. Desenvolvemos nele também um limitante de probabilidade de erro de bit, do qual extraímos diretrizes para guiar o projeto destes sistemas. Estas diretrizes são utilizadas, juntamente com as propriedades de rotulamentos e codificadores bit-geometricamente uniformes, no capítulo 4, onde apresentamos um algoritmo para projetar alguns componentes dos sistemas com concatenação serial. Mostramos que o algoritmo de fato gera codificadores BGU. Os resultados obtidos são apresentados no capítulo 5, onde mostramos também os limitantes de união para sistemas que utilizam os componentes encontrados. As conclusões se encontram no capítulo 6.

Capítulo 2

Rotulamentos e Codificadores Bit-Geometricamente Uniformes

Neste capítulo revisaremos o conceito de uniformidade geométrica. Revisaremos também os rotulamentos isométricos e os rotulamentos bit-geometricamente uniformes, enfatizando as qualidades que a sua uniformidade propiciam tais como a uniformidade na probabilidade de erro de bit e a relação entre distâncias e pesos Euclidianos e de Hamming. Indicaremos como rotulamentos e codificadores (rotulamentos multi-dimensionais) podem possuir esta característica. Concluímos mostrando que há casos nos quais se pode obter rotulamentos com a uniformidade de erro de bit que não são binariamente uniformes.

2.1 Constelações Geometricamente Uniformes

Seja um conjunto de pontos no espaço Euclidiano N-dimensional $\Re^N$, que chamaremos de constelação S. Uma isometria μ de S é uma transformação que preserva as relações de distância entre os elementos de S:

$$\begin{gathered} \mu : \Re^N \rightarrow \Re^N \\ s \rightarrow \mu(s), s \in S \\ \|\mu(s_i) - \mu(s_j)\|^2 = \|s_i - s_j\|^2; s_i, s_j \in S \end{gathered} \tag{2.1}$$

Translações, rotações e reflexões são isometrias.

A imagem de S obtida através da aplicação de μ a todos os elementos de S é representada por $\mu(S)$. Se $\mu(S) = S$, a isometria μ é uma simetria de S. Dizemos que S é geometricamente uniforme (GU) se, para quaisquer $s_i, s_j \in S$, existe uma simetria μ_{s_i,s_j} tal que:

$$\mu_{s_i,s_j}(s_i) = s_j, \tag{2.2}$$

Seja Γ_S o conjunto de todas as simetrias de S. Seja G_S um subgrupo de Γ_S. A operação $G_S(s)$ equivale a aplicar todos os elementos de G_S a s. Se $G_S(s) = S, \forall s \in S$, G_S é um grupo gerador de S. Se o número de elementos de G_S for igual ao número de elementos de S, G_S é um grupo gerador transitivo. Só consideraremos casos no qual o grupo gerador é transitivo. Pode haver mais de uma possibilidade de G_S dada uma constelação. A constelação 8-PSK, por exemplo, admite como grupo gerador tanto o grupo diedral D_4 como o grupo aditivo módulo 8 Z_8. Ao tomarmos um dos pontos s_0 de S como referência, há uma correspondência um para um entre os elementos de G_S e de S. Assim, utilizaremos os elementos de G_S para representar os pontos de S, substituindo $g_i(s_0), g_i \in G_S$ por simplesmente g_i.

Exemplos de constelações geometricamente uniformes são as constelações M-PSK e uma treliça infinita cujo padrão se repete, como mostra a figura 2.1. Para a constelação 8-PSK mostrada, qualquer rotação com eixo no centro do círculo e múltipla de 45^o gera simetrias que levam um ponto a ocupar o local ocupado por outro, mantendo a forma da figura. Reflexões em relação aos eixos $22.5^o + k \cdot 45^o$, k inteiro, também são simetrias válidas. Para a treliça infinita com um ponto na origem, qualquer translação cujo vetor pode ser descrito por um ponto da treliça é uma simetria que mantém a forma da figura. Para ambos os casos, as equações 2.1,2.2 são satisfeitas. Outras constelações geralmente utilizadas como PAM e QAM também são geometricamente uniformes se ignorarmos os efeitos das bordas da constelação.

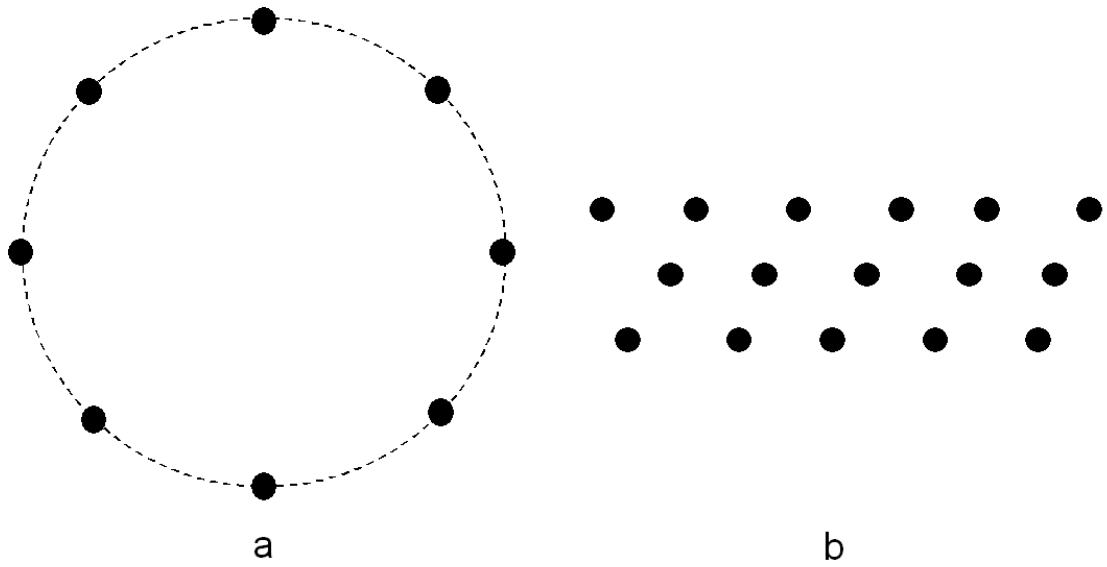

Fig. 2.1: Constelações GU: a-)$8PSK$ e b-)Treliça infinita regular

Num canal de transmissão com ruído aditivo, a transmissão de um ponto dentre aqueles que pertencem a uma constelação resulta na recepção de um ponto distinto daquele transmitido, devido ao ruído. Se o ruído for branco e com mesma energia em todas as suas dimensões, a melhor decisão que podemos fazer nesta situação é assumir que o sinal transmitido foi aquele que é o mais próximo do sinal recebido. O conjunto de pontos que é mais próximo a $s_i \in S$ do que qualquer outro ponto $s_j \in S$ é uma região de decisão por s_i. A decisão na recepção depende então da região onde o sinal recebido se encontra. Esta é a escolha por máxima verossimilhança. Em constelações do tipo M-PSK, estas regiões são congruentes, como pode ser visto na figura 2.2. O critério para decisão neste tipo de constelação, para canais com ruído aditivo, pode ser a determinação da fase do sinal.

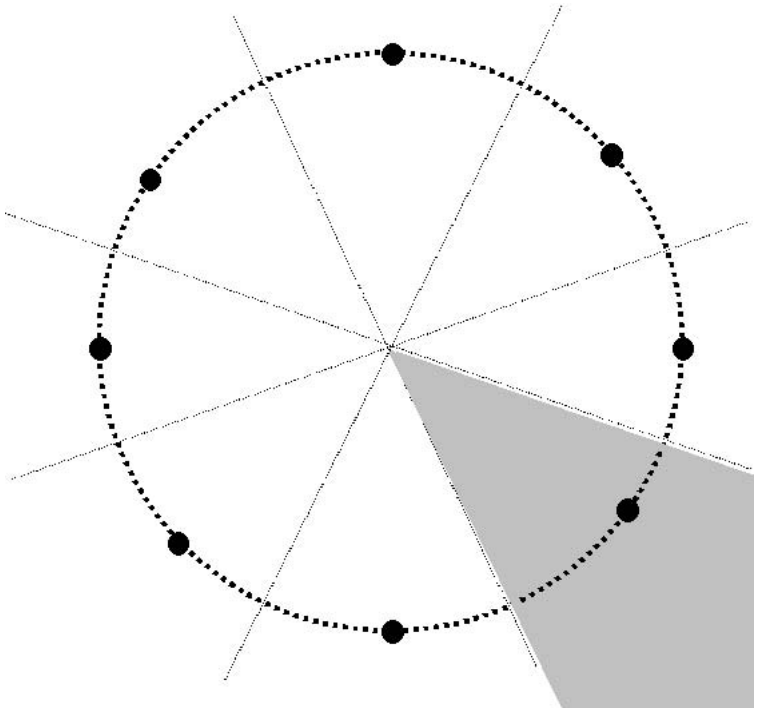

Fig. 2.2: Regiões de decisão de 8-PSK

Numa constelação GU, os pontos da constelação são todos idênticos no que se refere ao número de vizinhos e à distância deles. Como as regiões de decisão são semelhantes e congruente, a probabilidade de se errar ao se escolher o sinal independe do sinal transmitido.

Para uma descrição mais detalhada, sugerimos a leitura de [2].

2.2 Rotulamentos Isométricos e a Uniformidade de Erro de Bit

O rotulamento ε de uma constelação S com grupo gerador G por um rótulo de H_k(espaço de Hamming com k bits) é uma função $\varepsilon : S \rightarrow H_k$, que associa a cada ponto da constelação um valor do espaço de Hamming. A distância de Hamming entre dois elementos de um espaço de Hamming é o número de bits diferentes entre os elementos. Quando, para quaisquer dois elementos de S com distância Euclidiana d, a distância de Hamming dos rótulos associados a estes pontos for a mesma, dizemos que o rotulamento é isométrico. Matematicamente, um rotulamento ε que satisfaz:

$$d_h(\varepsilon(g_i), \varepsilon(g_j)) = f(d_e(g_i, g_j)), \forall g_i, g_j \in G \tag{2.3}$$

onde $d_e(g_i, g_j)$ é a distância Euclidiana entre g_i e g_j, $d_h(\varepsilon(g_i), \varepsilon(g_j))$ é a distância de Hamming entre $\varepsilon(g_i)$ e $\varepsilon(g_j)$, e f é uma função injetora que retorna um valor inteiro de acordo com o argumento, é um rotulamento isométrico. A função f nos diz que a distância de Hamming entre rótulos de dois pontos depende exclusivamente da distância Euclidiana entre os pontos.

A probabilidade de erro de bit ao se transmitir o símbolo g_i é definida pela seguinte equação:

$$\begin{aligned} P_b(e|\mathbf{t} = g_i) &= \sum_{j \neq i} P_b(e, \mathbf{r} = g_j | \mathbf{t} = g_i) \\ &= \sum_{j \neq i} \frac{d_h(\varepsilon(g_i), \varepsilon(g_j))}{k} P[\mathbf{r} = g_j | \mathbf{t} = g_i] \\ &= \sum_{j \neq i} \frac{w_h(\varepsilon(g_i) \oplus \varepsilon(g_j))}{k} P[\mathbf{r} = g_j | \mathbf{t} = g_i] \end{aligned} \tag{2.4}$$

onde g_i, g_j representam símbolos que pertencem à constelação, o símbolo $\oplus$ representa a soma módulo 2 e $w_h(a) = d_h(a, \mathbf{0})$ retorna o peso de Hamming de a, igual ao número de bits iguais a 1 que a possui. Quando a probabilidade de erro é a mesma para todos os símbolos, então o rotulamento possui a propriedade de uniformidade de erro de bit (UBEP do inglês *Uniform Bit Error Property*).

Constelações GU com rotulamentos isométricos possuem UBEP. Isto quer dizer que a probabilidade de erro de bit independe da seqüência transmitida. Como na situação de constelações geometricamente uniformes, isso garante que não há um pior caso. Isto não significa, entretanto, que temos o melhor rotulamento possível, pois podemos ter uma uniformidade de possibilidades ruins. Se vizinhos próximos tiverem uma grande distância de Hamming no seu rótulo, há uma boa chance de se errar muitos bits. Se vizinhos próximos tiverem uma pequena distância de Hamming, apenas alguns bits serão recebidos erroneamente.

Nem todas as constelações podem ser rotuladas isometricamente por rótulos com k bits. Um exemplo clássico é o rotulamento da constelação 8-PSK por H_3. Para tais situações, os rotulamentos Bit-Geometricamente Uniformes (BGU) são apropriados [7].

2.3 Rotulamentos BGU

O seguinte teorema, apresentado em [7], define o que são rotulamentos BGU:

Teorema 2.3.1. *Seja uma constelação geometricamente uniforme (GU) S, onde as regiões de decisão são congruentes, com grupo gerador G_S, tal que $G_S(s_0) = S$. Seja um rotulamento $\varepsilon : S \leftrightarrow H_k$, e H_k o espaço de Hamming com k bits. Se o rotulamento ε satisfaz a seguinte propriedade:*

$$d_h(\varepsilon(g_i), \varepsilon(g_j)) = w_h(\varepsilon(g_i^{-1} \cdot g_j)) \; \forall g_i, g_j \in G_S \tag{2.5}$$

ele possui a UBEP, i.e., a probabilidade de erro de bit independe da seqüencia enviada.

Prova O sinal recebido é $\mathbf{r}$ e o sinal transmitido é $\mathbf{t}$. A função $P_b(e|\mathbf{t} = g_i)$ é a probabilidade de erro de bit dado que o sinal transmitido foi g_i, e a função $P(\mathbf{r} = a|\mathbf{t} = b)$ é a probabilidade de se decidir por a quando b foi transmitido.

$$\begin{aligned} P_b(e|\mathbf{t} = g_i) &= \sum_{j \neq i} P_b(e, \mathbf{r} = g_j|\mathbf{t} = g_i) \\ &= \sum_{j \neq i} \frac{d_h(\varepsilon(g_i), \varepsilon(g_j))}{k} P[\mathbf{r} = g_j|\mathbf{t} = g_i] \\ &= \sum_{j \neq i} \frac{w_h(\varepsilon(g_i^{-1} \cdot g_j))}{k} P(\mathbf{r} = g_i^{-1} \cdot g_j|\mathbf{t} = e) \\ &= \sum_{g_l \neq e} \frac{w_h(\varepsilon(g_l))}{k} P(\mathbf{r} = g_l|\mathbf{t} = e) \\ &= P_b(e|\mathbf{t} = e) \end{aligned} \tag{2.6}$$

Como o resultado independe de g_i, a probabilidade de erro independe do sinal transmitido, e a UBEP se confirma

□

Assim sendo, a uniformidade geométrica da constelação se estende à uniformidade de bit, e o rotulamento ε é BGU. A UBEP nos permite analisar a probabilidade de bit observando somente um dos sinais transmitidos, como por exemplo o sinal rotulado pela palavra toda nula. Um exemplo de rotulamento BGU para a constelação 8-PSK pode ser visto na figura 2.3, utilizando o grupo D_4 como gerador.

Há condições a serem satisfeitas para que uma constelação admita um rotulamento BGU. O segundo teorema de [7] indica que uma constelação S só admite um rotulamento BGU pelos espaço de Hamming H_k se e somente se um de seus grupos geradores for isomorfo a um dos grupos geradores de H_k, subgrupo de Γ_{H_k}, o grupo de simetrias de H_k. Simetrias do espaço de Hamming serão definidas na próxima seção.

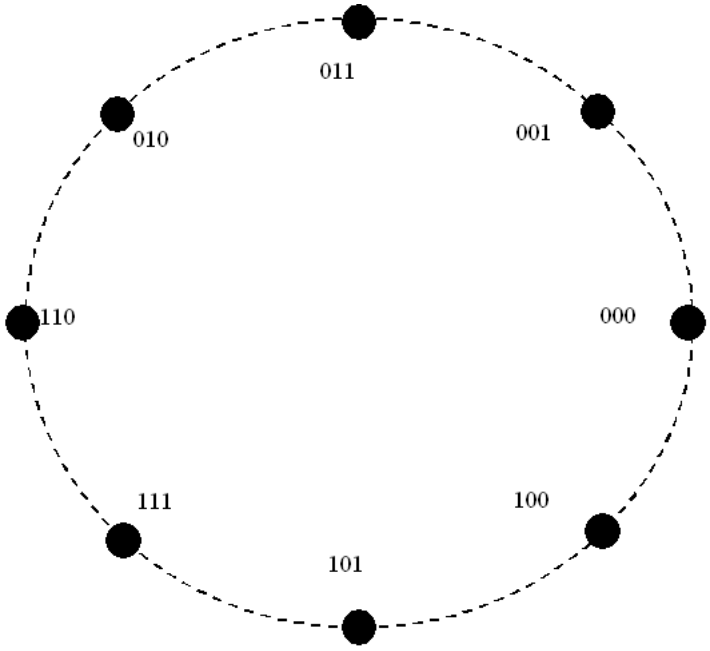

Fig. 2.3: Possível Rotulamento BGU para a constelação 8-PSK

2.3.1 Simetrias do Espaço de Hamming

Precisamos nos aprofundar mais sobre as simetrias do espaço de Hamming para podermos completamente desenvolver o nosso método. A ligação entre o espaço de Hamming e o espaço Euclidiano torna-se evidente ao percebermos que, quando os dois conjuntos tem grupos geradores isomorfos, eles são estruturalmente os mesmos.

No espaço Euclidiano, uma simetria de S é uma isometria que mantém a forma da constelação S. No espaço de Hamming, analogamente, uma simetria é uma transformação $\rho : H_k \rightarrow H_k$ que mantém a distância de Hamming entre os elementos e contém todos os elementos anteriores à transformação. As seguintes transformações geram simetrias de H_k:

- Isometrias de H_k obtidas através da soma de qualquer elemento pertencente a H_k a todos os outros elementos.
- Isometrias de H_k obtidas através da intercâmbio de coordenadas (bits ou dimensões).
- Combinações de isometrias acima.

O conjunto de todas as simetrias de H_k forma o grupo Γ_{H_k}. Um grupo $G_{H_k} \subseteq \Gamma_{H_k}$ que satisfaz $G_{H_k}(h_i) = H_k$ (gera os elementos de H_k a partir de um elemento inicial h_i) é chamado grupo gerador

de H_k. Se o número de elementos de G_{H_k} for igual ao número de elementos de H_k, então G_{H_k} é um grupo transitivo. Consideraremos somente os casos no qual G_{H_k} é transitivo.

Quando um grupo algébrico pode ser gerado utilizando somente alguns dos seus elementos, chamamos estes de geradores. Grupos cíclicos, por exemplo, são gerados por um único termo r , onde $r^n = e$, com n a ordem do grupo e e sendo a identidade.

O espaço de Hamming também pode ser gerado através de sum grupo de simetrias. O espaço H_2, por exemplo, pode ser gerado através da simetria $\langle (21)_p (01)_\oplus \rangle$. O primeiro elemento $(21)_p$ indica o intercâmbio de coordenadas, onde cada número indica qual bit deve a posição que ocupa. O segundo elemento $(01)_\oplus$ indica o termo que deve ser somado módulo 2 ao termo inicial. O bit mais a esquerda é o primeiro. Assim, obtemos:

$$\begin{aligned} 00 \cdot (21)_p (01)_\oplus &= 01 \\ 01 \cdot (21)_p (01)_\oplus &= 11 \\ 11 \cdot (21)_p (01)_\oplus &= 10 \\ 10 \cdot (21)_p (01)_\oplus &= 00 \end{aligned} \tag{2.7}$$

Há muitas possibilidades de grupos geradores para H_k. Uma maneira simples no sentido funcional mas possivelmente complexa no sentido computacional seria de gerar todas as possíveis combinações de permutações e somas, analisar a ordem de cada um dos possíveis geradores, e testar a combinação de candidatos, ficado com aqueles que de fato geram H_k. Para não testar combinações com excesso de candidatos podemos limitar as combinações àquelas cuja multiplicação da ordem dos candidatos seja igual a 2^k. Além disso, precisaríamos retirar todos os grupos geradores equivalentes entre si.

2.4 Códigos Geometricamente Uniformes

Um código é um conjunto $C \subseteq W$, onde W é o conjunto de todos os possíveis símbolos ou valores de um espaço. Se W for uma seqüencia de n símbolos, e em cada instante (ou dimensão) k o símbolo pode assumir valores pertencentes a um alfabeto A_k, então W é o produto cartesiano $W = \prod_{k=1,2,\ldots,n} A_k$. Se todos os alfabetos são iguais a um alfabeto A, então $W = A^n$.

Se cada alfabeto A tem um grupo gerador G_A, $W = A^n$ também é um grupo, sendo que em cada dimensão de W as operações do grupo são definidas pelo grupo G_A. O grupo que gera W é o produto cartesiano dos grupos geradores de cada instante, representado por G_A^n.

Seja $c \in C$ um vetor com n símbolos, um para cada instante ou dimensão de uma seqüência de C. Cada elemento de c é um elemento do grupo gerador do alfabeto A. Assim, podemos realizar operações entre seqüências respeitando as operações definidas pelo grupo G_A. Podemos escrever $c \cdot C$ como sendo o conjunto obtido multiplicando-se todos os elementos de C por c.

Um código C é geometricamente uniforme se, para quaisquer $c_i, c_j \in C$ existe uma operação c tal que :

$$\begin{aligned} c \cdot (c_i) &= c_j, \\ c \cdot (C) &= C. \end{aligned} \tag{2.8}$$

A definição acima é similar à utilizada para definir constelações GU. A simetria é substituída por uma operação de grupo. Necessariamente c pertence à C. O código C é um subgrupo de W.

Um código de grupo pode ser apropriadamente representado por uma treliça, onde os pontos representam estados e os ramos representam transições correspondentes ao símbolo das seqüências. Os estados limitam a possibilidade de escolha do próximo elemento do alfabeto A a ser transmitido. Esta restrição surge naturalmente a partir do código C, que define o espaço de estados Σ_m do código. Uma seção de treliça corresponde à representação do código C entre os instantes $k-1$ e k. Se o código for invariante no tempo, ou quasi-invariante no caso de ser finito ou semi-finito, a seção de treliça define o código completamente, bastando para isso concatenar seções. Estes códigos podem ser chamados de TCM, do inglês *Trellis Coded Modulation*, enfatizando a sua característica de ser um código definido sobre uma treliça. Recomendamos ao leitor que queira se aprofundar mais neste assunto a leitura de [10].

2.5 Codificadores BGU

Um codificador é diferente de um código. O código, como visto na seção anterior, é um conjunto de símbolos ou seqüências. Um codificador associa, para cada seqüência ou símbolo pertencente ao código, um símbolo ou seqüência de valores de um alfabeto de entrada. Sendo um código e um codificador invariantes ou quasi-invariantes no tempo, um codificador que associa para cada seção de treliça um símbolo de um alfabeto de entrada a um ramo da treliça define completamente a associação entre as seqüências de entrada e as seqüências do código. Podemos neste caso considerar o espaço do símbolo de saída e do símbolo do alfabeto de entrada correspondente a uma seção de treliça, em vez de considerar o espaço de toda a seqüência de entrada e saída. Assim, podemos analisar seqüências muito grandes observando somente uma parte bem menor e constante.

Um rotulamento BGU é uma função entre um espaço de Hamming e símbolos de um espaço Euclidiano, que chamamos de constelação. Um código pode ser visto como um conjunto de pontos numa dimensão muito grande. Se por exemplo um código C é um subconjunto de todas as seqüências com n elementos $S \in \Re^m$, $C \subseteq \Re^{n+m}$. Um codificador seria um rotulamento para estes pontos. Seria muito complicado tentar trabalhar com todos os pontos do código, pois a seqüência seria muito grande.

Procuramos codificadores que façam o rotulamento entre um espaço de entrada H_k^n e C levando em consideração a estrutura de grupo e treliça que C possui por ser um código geometricamente uniforme. O codificador será definido para uma seção da treliça, um TCM. A treliça possui uma estrutura de grupo, e podemos realizar operações sobre os seus elementos dado o grupo associado. Pode haver mais de uma possibilidade de associação de grupo a uma treliça.

A vantagem de se ter um codificador BGU é que não precisamos testar as distâncias entre todas as seqüências para determinarmos relações entre distâncias Euclidianas e distâncias de Hamming, fator relevante no projeto de sistemas com concatenação serial. Como a relação de distâncias é a mesma para qualquer seqüência, podemos considerar que uma das seqüências é a seqüência rotulada pela palavra toda nula. Assim, as distâncias se tornam pesos, e a analise do codificador é simplificada.

O seguinte teorema apresentado em [7] define o que é um codificador que possui a UBEP:

Teorema 2.5.1. *Seja S uma constelação GU e $C \subseteq S^n$ um TCM GU invariante no tempo sobre S com grupo gerador G_C. Um codificador binário $E : C \rightarrow H_k$ que satisfaz:*

$$d_h(E(c_i), E(c_j)) = w_h(E(\cdot c_i^{-1} c_j)) \; \forall c_i, c_j \in G_C \tag{2.9}$$

possui a UBEP.

Os codificadores que satisfazem ao teorema 2.5.1 serão chamados de codificadores bit-geometricamente uniformes (BGU). A prova do teorema 2.5.1 é uma extensão da prova do teorema 2.3.1 para vetores n-dimensionais.

O teorema acima não é útil para testar se códigos com seqüências muito grandes são BGU, pois comparar todas as seqüências do código duas a duas é inviável. O ideal seria se pudéssemos aproveitar o fato de que a treliça representa o código para testar se o codificador é BGU. De fato, o seguinte teorema, também de [7], permite o teste:

Teorema 2.5.2. *Seja S uma constelação GU, $C \subseteq S^n$ um TCM GU invariante no tempo sobre S e T um dos grupos de uma seção de treliça deste código. Um codificador binário $E : T \rightarrow H_k$ que satisfaz:*

$$d_h(E(t_i), E(t_j)) = w_h(E(\cdot t_i^{-1} t_j)) \; \forall t_i, t_j \in T \tag{2.10}$$

possui a UBEP.

Prova A prova do teorema 2.5.2 é simples. Toda seqüência c é descrita através de uma seqüência única de ramos, cada ramo representado pela tríplice (σ_i, g, σ_f). G é o grupo gerador de S e Σ é o espaço de estados de T. Temos que $\sigma_i \in \Sigma$ é o estado inicial, $\sigma_f \in \Sigma$ é o estado final e $g \in G$

representa o sinal transmitido, onde G é grupo gerador de S e Σ é o espaço de estados. Assim, dados $c_1, c_2 \in T^n$, temos:

$$c_1 = ..., (\sigma_{1,i}, g_1, \sigma_{1,g}), ...$$
$$c_2 = ..., (\sigma_{2,i}, g_2, \sigma_{2,g}), ...$$
$$c_1^{-1}c_2 = ..., (\sigma_{1,i}^{-1}\sigma_{2,i}, g_1^{-2}g_2, \sigma_{1,g}^{-1}\sigma_{2,g}), ...$$

Como o codificador atua a cada seção de treliça, temos que, se para cada par de ramos a condição 2.10 for satisfeita, o codificador será BGU como um todo. □

Como conseqüência do teorema anterior, chegamos ao seguinte teorema, também apresentado em [7]:

Teorema 2.5.3. *Seja S uma constelação GU, e $C \subseteq S^Z$ um TCM GU invariante no tempo sobre S e T um dos possíveis grupos que representa uma seção de treliça. Seja A um subconjunto de T que contém todos os ramos que são rotulados pela palavra toda nula. Seja S_0 um subconjunto de S dos sinais associados aos ramos que partem do estado identidade. Seja F_0 o subconjunto de T que contém todos os ramos que partem do estado identidade. Um codificador binário $E[C, k]$ é BGU se e somente se as seguintes condições são satisfeitas para pelo menos um dos possíveis T's:*

1. *$E_0(S_0, k)$ é um rotulamenteo BGU para F_0;*
2. *$E_j(f \cdot a) = E_0(f)$, para todo $f \in F_0$, e para todo $a_j \in A$;*
3. *A é um subgrupo de T;*
4. *Para todo $a_j \in A$, $w_h(E_0(f)) = w_h(E_0(f'))$, com $f' = a_j^{-1} \cdot f \cdot a_j$.*

onde E_j é o rotulamento dos ramos que saem do j-ésimo estado e $w_h(x)$ indica o peso de Hamming de x. Quando o grupo A é um subgrupo normal, podemos obter o grupo T através do produto $T = F_0 \cdot A$.

Prova Se um codificador é BGU, então a primeira condição é satisfeita.

Por definição, $w_h(a_i) = 0$. Como conseqüência,

$$d_h(E_0(f), E_j(f \cdot a_i)) = w_h(E(f^{-1} \cdot f \cdot a_i)) = w_h(E(a_i)) = 0 \tag{2.11}$$

o que satisfaz a segunda condição.

O ramo identidade e faz parte do conjunto A, pois:

$$0 = d_h(E(f), E(f))) = w_h(E(f^{-1} \cdot f) = w_h(E(e)) = 0 \tag{2.12}$$

Logo:

$$d_h(E(a_i), E(e)) = 0 = w_h(E(a_i^{-1})) \tag{2.13}$$

de onde concluímos que o elemento inverso de a_i pertence ao conjunto A, formando assim um grupo, satisfazendo a terceira condição.

Finalmente:

$$\begin{aligned} w_h(E_0(f)) &= d_h(E_0(e), E_0(f)) \\ &= d_h(E_j(a_j), E_j(f \cdot a_j)) \\ &= w_h(E_0(a_j^{-1} \cdot f \cdot a_j)) \end{aligned} \tag{2.14}$$

Como F_0 é um subgrupo normal de T, e além disso $T = F_0 \bullet A$,(onde $\bullet$ é o produto semi-direto) o elemento $(a_j^{-1} \cdot f \cdot a_j) \in F_0$.

Provaremos agora que se as condições são satisfeitas, o codificador é BGU.

Se as condições 1-4 forem satisfeitas, então:

$$\begin{aligned} d_h(E(t_i), E(t_j)) &= d_h(E(f_i \cdot a_i), E(f_j \cdot a_j)) \\ &= d_h(E_0(f_i), E_0(fj)) \\ &= w_h(E_0(f_i^{-1} \cdot f_j)) \\ &= w_h(E_0(a_i^{-1} \cdot f_i^{-1} \cdot f_j \cdot a_i)) \\ d_h(E(t_i), E(t_j)) &= w_h(E(a_i^{-1} \cdot f_i^{-1} \cdot f_j \cdot a_i \cdot (a_i^{-1} \cdot a_j)))) \\ &= w_h(E_0(a_i^{-1} \cdot f_i^{-1} \cdot f_j \cdot a_j))) \\ &= w_h(E(t_i^{-1} \cdot t_j)) \end{aligned} \tag{2.15}$$

Nesta prova, utilizamos do fato que $a_i^{-1} \cdot f_i^{-1} \cdot f_j \cdot a_i \in F_0$, e que $E_k(f \cdot a_k) = E_0(f)$, onde $a_k = a_i^{-1} \cdot a_j$ e $f = a_i^{-1} \cdot f_i^{-1} \cdot f_j \cdot a_i$. □

Sendo assim, a procura de um codificador BGU consiste em procurar F_0,E_0 e A, e associar a cada elemento de A um elemento do espaço de estados do código Σ. Encontrar estes elementos é suficiente para definir um codificador BGU.

2.6 Rotulamentos não BGU com a UBEP

Terminamos este capítulo mostrando que existem rótulos que possuem a UBEP sem serem BGU. Seja uma constelação S com 2^k pontos, rotulada através de $E : S \rightarrow H_k$. Além disso, a constelação S

é GU, com regiões de decisão congruentes. Assim, a probabilidade de se trocar um ponto pelo outro, considerando todos os pontos da constelação, depende da distância entre os pontos. Seja $F^i_{DD}(h,d)$ uma função que, para o i-ésimo elemento de S, retorna o número de elementos com distância de Hamming h e distância Euclidiana d do ponto i. Se, para qualquer $i,j \in S$, for verdadeiro que $F^i_{DD}(h,d) = F^j_{DD}(h,d), 0 \leq h \leq k, \forall d \in \Re$, onde $\Re$ é o conjunto dos números reais, então a probabilidade de erro de bit independe do sinal transmitido, e o rotulamento possui a UBEP.

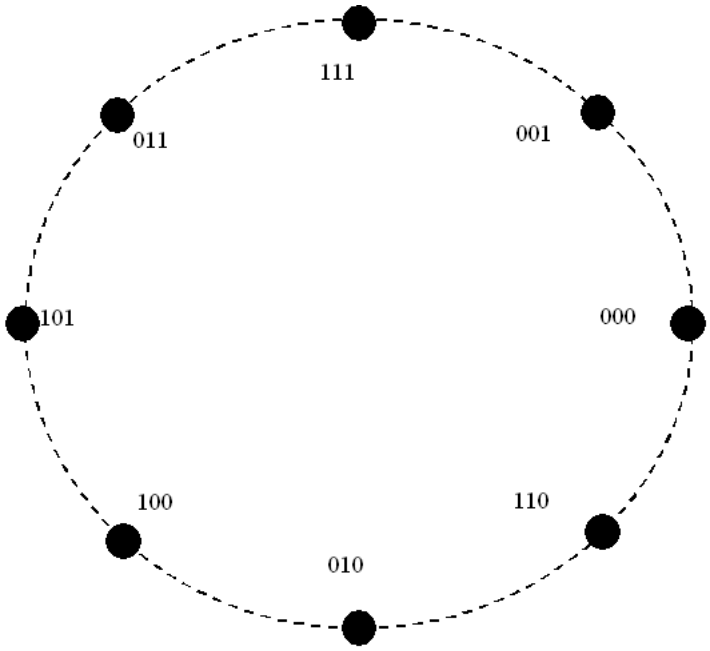

Fig. 2.4: Possível rotulamento não BGU com UBEP para a constelação 8-PSK

A função F^i_{DD} lembra muito a IOWEF condicional, onde IOWEF é a função de distribuição de pesos de um código [1](do inglês *Input Output Weight Enumerating Function*). Podemos definir uma função assim para uma constelação, onde os termos do polinômio indicam a quantidade de pontos com uma certa distância de Hamming e uma certa distância Euclidiana. Chamaremos esta função de função de distribuição de distâncias (FDD). Assim, para a figura 2.4 teríamos a seguinte FDD^{000}:

$$FDD^{000}(H,E) = 1 + (H^1 + H^2)E^{0.58} + (H^3 + H^1)E^2 + (H^1 + H^2)E^{3.42} + H^2E^4 \qquad (2.16)$$

Esta função é definida para cada ponto da constelação. O termo $(H^1 + H^2)E^{0.58}$, por exemplo, nos diz que há uma palavra com distância de Hamming 1 (o ponto 001) e outra com distância de Hamming 2 (o ponto 110) que distam de 0.58 (distância Euclidiana quadrática para constelação com

[1]Vide equação 3.8 do capítulo 3

raio unitário) do ponto 000. As distâncias utilizadas são distâncias quadráticas. Se todos os pontos possuem a mesma FDD e além disso as regiões de decisão forem congruentes, a constelação possui a UBEP.

Pode-se perceber que o rotulamento da figura 2.4 não é BGU. Uma conseqüência do teorema 2.3.1 é:

$$w_h(\varepsilon(g^{-1})) = d_h(\varepsilon(g), \varepsilon(e)) = d_h(\varepsilon(e), \varepsilon(g)) = w_h(\varepsilon(g)), \forall g \in G \tag{2.17}$$

onde e representa a identidade do grupo G gerador de S, e $\varepsilon(e) = 0$

A constelação 8-PSK admite como grupos geradores somente os grupos Z_8 e D_4. Consideraremos o ponto 000 como o ponto inicial. O grupo Z_8 é gerado por um único gerador, que chamaremos de r, como mostra a figura 2.5. A aplicação de r sobre o 000 causa uma rotação, podendo esta ser de 45^o, 135^o, 225^o ou 315^o, para que os 8 pontos sejam gerados. Em qualquer caso, o ponto gerado por r^2 seria o ponto 111, e o ponto gerado por $r^{-2} = r^6$ seria o ponto 010, ou vice-versa. De qualquer forma, $w_h(111) \neq w_h(010)$, e o grupo Z_8 não poderia ser utilizado para gerar o rotulamento indicado.

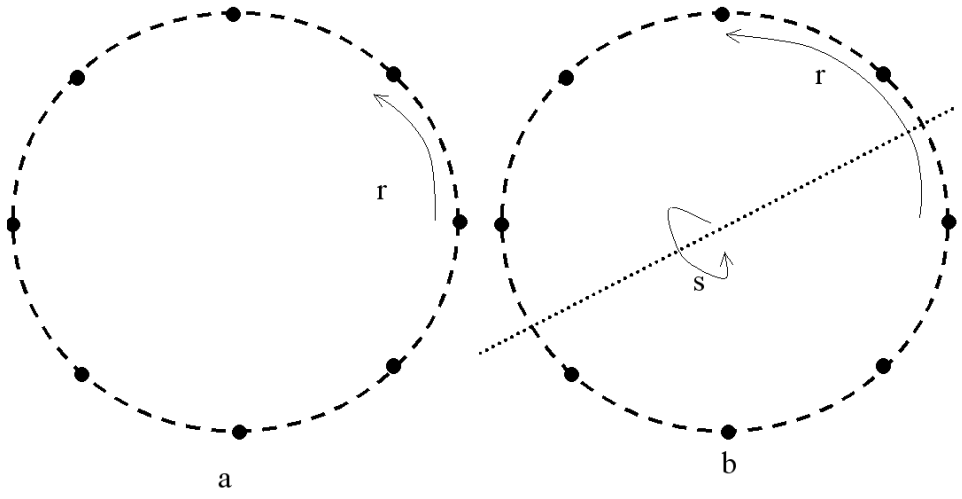

Fig. 2.5: Grupos geradores da constelação 8-PSK e geradores associados: a-)Z_8, b-)D_4

No caso de utilizarmos o grupo D_4 como gerador, os pontos 111,101 e 010 seriam gerados seqüencialmente pelo gerador r, a rotação de ordem 4. Os outros pontos são gerados através do gerador s, simetria de ordem 2, em composição com r. Nesta situação, o ponto 111 é gerado por r, e o ponto 010 é gerado por $r^{-1} = r^3$. A mesma desigualdade do caso anterior esta presente.

Pode-se argumentar que não testamos todas as possibilidades por considerarmos somente um ponto como ponto de origem. Para isso, argumentamos que, em algum sentido, horário ou anti-horário, a distância de Hamming entre vizinhos no caso Z_8 e entre pontos alternados no caso D_4, tem

que ser a mesma, dado que:

$$d_h(\varepsilon(r^n), \varepsilon(r^{n+1})) = w_h(\varepsilon(r)) \tag{2.18}$$

o que também não se comprova em nenhuma situação.

2.7 Síntese

Neste capítulo apresentamos o conceito de uniformidade binária e geométrica, tanto para rotulamentos como para codificadores, em particular para códigos TCM. Para isso apresentamos também uma maneira de gerar espaços de Hamming tendo como base o seu grupo de simetrias obtido através de permutações. Apresentamos a prova de que rotulamentos BGU possuem a uniformidade na probabilidade de erro de bit. Esta prova pode ser estendida para codificadores BGU, que também possuem esta propriedade. Apresentamos 4 condições necessárias e suficientes para que um codificador TCM seja BGU. Mostramos também que existem rotulamentos não BGU que possuem a UBEP, o que significa que ao escolhermos trabalhar com codificadores BGU estamos limitando as nossas alternativas.

Capítulo 3

Sistemas com Concatenação Serial

Neste capítulo apresentaremos a concatenação serial de códigos componentes para gerar um código maior. Obtemos também limitantes de desempenho para estes códigos. Através da análise destes limitantes, obtemos diretrizes que nos orientarão na busca por bons códigos componentes do sistema.

3.1 Sistemas com Concatenação de Códigos

A concatenação de códigos consiste na utilização de mais de um código associados de alguma forma de modo a gerar uma seqüência de saída a partir de uma seqüência de entrada. Chamaremos os códigos menores de códigos-componente, e o código resultante simplesmente de código. Há diversas formas de se realizar esta concatenação. Há concatenações paralelas, seriais, e combinações destas.

Códigos com Concatenação Paralela

Nos códigos com concatenação paralela (CCP), uma seqüência de entrada alimenta dois ou mais códigos componente, como mostra a figura 3.1. A seqüência de entrada é entrelaçada antes de alimentar os outros códigos além do primeiro através do uso de um entrelaçador. Se os códigos forem sistemáticos (os bits de informação aparecem inalterados como parte da saída), podemos retirar os k bits de informação repetidos. A complexidade do código é ligeiramente maior do que a complexidade dos códigos-componente. Estes códigos também são conhecidos por códigos Turbo, e ficaram conhecidos por alcançarem desempenhos próximos dos limites teóricos estabelecidos por Shannon [11]. Eles foram apresentador por Berrou, Glavieux e Thitimajshima em [3]. A sua grande vantagem é que o seu desempenho é muito bom quando decodificado através de um algoritmo iterativo simples.

A taxa do código R_{CCP}, considerando que todos os códigos componente são sistemáticos e que os bits de informação repetidos são retirados, pode ser obtida através da seguinte equação:

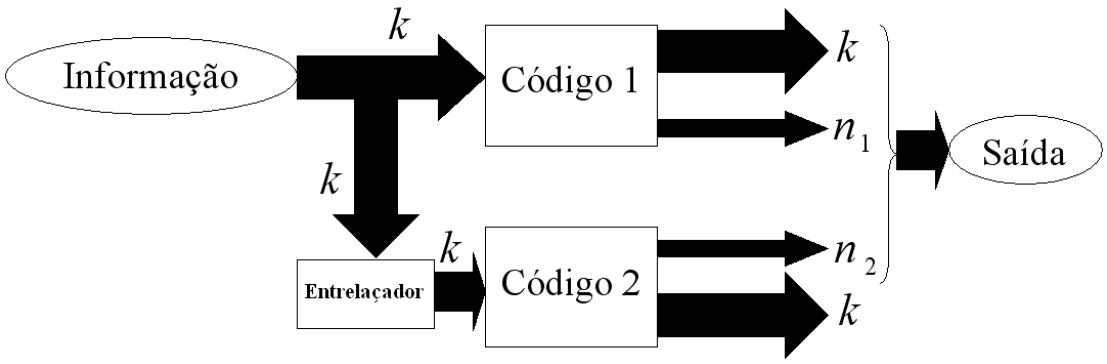

Fig. 3.1: Sistema de Concatenação Paralela

$$R_{CCP} = \frac{\textit{Bits de entrada}}{\textit{Bits de saída}} = \frac{k}{n_1 + n_2 + \cdots + n_m + k} \tag{3.1}$$

onde n_i é o número de bits que o i-ésimo código acrescenta aos k bits de entrada e m é o número de codificadores.

Códigos com Concatenação Serial

Nos códigos com concatenação serial (CCS), a saída de um código-componente é a entrada do segundo, como mostra a figura 3.2. O entrelaçador também esta presente neste sistema, situando-se entre os códigos. Como na concatenação paralela, vários códigos podem ser utilizados conjuntamente para formar um sistema com concatenação serial. Limitaremos a nossa pesquisa ao caso em que dois códigos são utilizados. Isso não é de nenhuma forma uma simplificação drástica do sistema, pois em situações com mais de dois códigos-componente podemos truncar os códigos intermediários e obter o código componente equivalente ou as suas característica mais relevantes para o projeto e análise do sistema. O código que recebe a informação é o código que fica na extremidade do sistema, e por isso é chamado de código externo. O outro código é chamado de código interno.

Fig. 3.2: Sistema de Concatenação Serial

A taxa do código R_{CCS} é dada pela multiplicação das taxas dos códigos intermediários:

$$R_{CCS} = R_1 \cdot R_2 \cdots R_m \tag{3.2}$$

onde R_i é a taxa do i-ésimo código componente.

Não faz sentido falar em sistemas com concatenação serial com codificação sistemática, pois mesmo que ambos os codificadores o fossem, o entrelaçador não permitiria que a informação passasse na sua forma original. Dados dois códigos com a mesma taxa e mesmo número de códigos componente, um CCS e outro CCP, o CCS é ligeiramente mais complexo. O código interno precisa processar, além dos bits de informação, os bits adicionados pelo código externo. O entrelaçador utilizado também é maior.

Estudos comparativos [5] entre CCS's e CCP's indicam que, quando o sistema deve operar próximo à região de corte do canal, os CCP's são mais indicados. Quando o sistema opera numa região da relação sinal/ruído melhor (ruído menor), os CCS's são melhores, pois o seu patamar de erros (error flor) é mais baixo, significando que menos erros ocorrerão.

3.2 Limitantes de União para a Probabilidade de Erro de Bit em Sistemas com Concatenação Serial

A análise do limitante de união que utilizaremos incluída nesta seção foi apresentada em [5].

O limitante de união nos fornece uma estimativa para o desempenho de códigos, analisando a probabilidade de se trocar uma dada seqüência de entrada pela seqüência toda nula e somando estas probabilidades, ponderando cada probabilidade pela percentagem de bits de informação recebidos errados. Esta soma faz com que o valor obtido pelo limitante de união seja superior à probabilidade de erro esperada do sistema. Como estamos somando a probabilidade de eventos distintos ocorrerem sem considerar as ocasiões onde um evento exclui o outro, este valor pode ser maior do que 1 em algumas situações, valor que não corresponde a uma probabilidade. Mesmo assim, o limitante de união é uma analise válida porque a probabilidade de erro real converge para aquela estimada no limitante de união quando a relação sinal/ruído aumenta, o que nos permite analisar assintoticamente o desempenho de códigos.

Aproximamos o sistema pelo código de bloco equivalente. O tamanho do bloco é determinado pelo tamanho do entrelaçador utilizado e pela taxa do código externo. Para os sistemas que estamos estudando, o tamanho do bloco de entrada B é:

$$B = N \cdot R_{Ce}, \tag{3.3}$$

onde R_{Ce} é a taxa do código externo e N é o tamanho do entrelaçador utilizado, em bits.

A probabilidade de se decidir, ignorando todas as outras seqüências, por máxima verossimilhança, por uma seqüência s com peso Euclidiano $w_e(s)$ [1]quando a seqüência toda nula $\mathbf{0}$ foi transmitida através de um canal AWGN é igual à probabilidade de se ter um evento de erro com peso (ou energia) no mínimo igual à metade de w_e na direção de s, pois isso faria com que a seqüência recebida ficasse mais próxima de s do que $\mathbf{0}$. Isto vale mesmo que o espaço Euclidiano seja n-dimensional, se o ruído tiver a mesma energia em todas as dimensões. Matematicamente:

$$P(\mathbf{s}_r = s \mid \mathbf{s}_t = \mathbf{0}) = P(\mathbf{r}_s > \frac{\sqrt{w_e(s)}}{2}) \tag{3.4}$$

onde $\mathbf{s}_t$ é a seqüência transmitida, $\mathbf{s}_r$ é a seqüência recebida, $\mathbf{r}_s$ é o ruído aditivo presente no sinal na direção $s - \mathbf{0}$. A variável $\mathbf{r}_s$ possui apenas uma dimensão. Se o ruído for Gaussiano e tiver a mesma energia (variância) em todas as suas dimensões, a variância de $\mathbf{r}_s$ é igual à variância de $\mathbf{r}$ em uma de suas dimensões. Vamos considerar que esta situação esta sempre presente.

Como indicado na seção 1.2, na equação 1.2, o ruído aditivo do canal Gaussiano possui a seguinte distribuição:

$$\begin{aligned} P(\mathbf{r}_s < r) &= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{r} e^{\frac{-(r-\mu)^2}{2\sigma^2}} dr = 1 - Q\left(\frac{r-\mu}{\sqrt{2}\sigma}\right) \\ P(\mathbf{r}_s > r) &= 1 - P(\mathbf{r}_s < r) = Q\left(\frac{r-\mu}{\sqrt{2}\sigma}\right) \end{aligned} \tag{3.5}$$

onde μ, σ^2 representam respectivamente a média e a variância em uma dimensão da variável $\mathbf{r}$. No caso do ruído aditivo branco, $\mu = 0$ e $\sigma^2 = 2 \cdot W_T \cdot T \cdot N_0$, onde W_T é a banda de transmissão, T é a duração do sinal e N_0 é a densidade espectral do ruído em uma dimensão. Consideraremos o caso em que a densidade é unilateral com valor N_0, o que corresponde a dizer que em cada dimensão o valor é de $N_0/2$. Consideraremos também que $W_T = 1/T$, de modo que a variância a ser utilizada é N_0.

Para facilitar os cálculos, podemos utilizar a seguinte aproximação:

$$Q(x) < e^{-x^2} \tag{3.6}$$

Utilizaremos a aproximação para facilitar a computação do limitante. Esta atitude implica num valor mais relaxado(maior) para o limitante. A probabilidade de se trocar uma seqüência s_1 por outra s_2 a uma distância d, onde d corresponde à distância quadrática entre as condições descritas, é a probabilidade do ruído ser maior do que $\sqrt{d/2}$. Isto vale:

$$P(s_R = s_2 | s_T = s_1) = Q(\frac{\sqrt{d/2}}{\sqrt{2N_0}}) < e^{-\frac{d}{4N_0}} \tag{3.7}$$

[1]Como indicado no capítulo 1, as distâncias e pesos Euclidianos correspondem a distâncias quadráticas

onde s_T e s_R são os sinais transmitido e recebido, respectivamente. Esta probabilidade vale se estivermos considerando somente as duas seqüências envolvidas, e por isso é chamada de probabilidade par a par. Isto não leva em conta a possibilidade da existência de uma terceira seqüência que é mais próxima do sinal recebido do que as duas anteriores. O limitante de união pode então assumir valores maiores do que 1 em algumas situações, valor que não tem significado como uma probabilidade.

A distância entre os sinais depende da energia do sinal transmitido. Como estamos utilizando constelações do tipo $2 \times 8PSK$ e $3 \times 8PSK$, representaremos a distância como sendo $d \cdot E_S$, onde E_s é a energia do símbolo transmitido em cada instante (cada sinal possui 2 ou 3 símbolos, respectivamente) e d é o valor da distância quando cada constelação $8 - PSK$ tem raio unitário.

Precisamos apresentar a função de distribuição de pesos do código, que é um polinômio que contém a informação necessária sobre a relação entre os pesos de entrada e de saída das seqüências pertencentes ao código. Assim, seja o polinômio:

$$A^C(W, D) = \sum_{w=0}^{w_{max}} \sum_{d=0}^{d_{max}} A^C_{w,d} W^w D^d, \tag{3.8}$$

onde o termo $A^C_{w,d}$ representa o número de seqüências com peso de entrada w que geram seqüências com peso de saída d. As variáveis W e D são variáveis de manipulação. Os valores w_{max} e d_{max} correspondem aos maiores valores de entrada e saída do codificador, respectivamente. Como não impusemos nenhuma restrição sobre a natureza de d ou w, estas variáveis podem representar qualquer grandeza. No nosso caso, estamos interessados nas situações onde w representa o peso de Hamming, e d representa ou um peso de Hamming ou uma distância Euclidiana. O polinômio descreve a relação enter o peso de saída e o peso de entrada do código, sendo chamado de função IOWEF (do inglês *Input Output Weight Enumerating Function*). Utilizaremos esta notação para representar também situações na qual a variável d assume uma quantidade finita de valores pertencentes a $\Re$ (o espaço dos números reais), que é o caso quando d representa uma distância Euclidiana. Nestas situações, $d = d_1, ..., d_{max}$ significa que d assume todos os valores possíveis e existentes neste intervalo.

Definimos também a CIOWEF, a distribuição condicional de pesos do código (do inglês *Conditional Input Output Weight Enumerating Function*), que contém os termos da função $A^C(W, D)$ restrita à palavras com peso de entrada igual a w. Assim:

$$A^C(w, D) = \sum_{d=0}^{d_{max}} A^C_{w,d} D^d \tag{3.9}$$

A probabilidade de erro para um código C com decodificação por máxima verossimilhança (MV) e tamanho de bloco de entrada B é limitada superiormente pela seguinte equação:

$$P_b(e) \leq \frac{1}{B} \sum_c w_h(c) e^{-\frac{E_s w_e(c)}{4N_0}}, c \in C \tag{3.10}$$

Na equação anterior, estamos somando todas as probabilidade de se trocar as seqüências do código pela seqüência toda nula, ponderando estas probabilidades pelo percentagem de bits errados $w_h(c)/B$. A energia do sinal é E_s, e a densidade unilateral do ruído aditivo é N_0.

Unindo as equações 3.9 e 3.10, chegamos à seguinte formulação:

$$P_b(e) \leq \sum_w w \cdot A^C(w, D)|_{D=e^{-\frac{E_s}{4N_0}}} \tag{3.11}$$

Não é fácil obter a função $A^C(w, D)$ para um código com concatenação serial, pois o entrelaçador entre os códigos externo e interno dificulta a análise. Por isso utilizamos a idéia do entrelaçador uniforme, como apresentado em [4]. Um entrelaçador uniforme é um dispositivo que mapeia uma seqüência binária de entrada com peso de Hamming l para todas as suas permutações possíveis com a mesma probabilidade. Assim, dada uma seqüência específica de entrada do entrelaçador com peso l e uma seqüência específica de saída com o mesmo peso, a probabilidade da primeira ser mapeada na segunda é $1/\binom{N}{l}$. Conseqüentemente, podemos escrever os valores $A_{w,d}^{CCS}$ e $A_{w,D}^{CCS}$ de uma IOWEF de um CCS da seguinte forma:

$$\begin{aligned} A_{w,d}^{CCS} &\cong \sum_{l=0}^{l_{max}} \frac{A_{w,l}^{C_e} \cdot A_{l,d}^{C_i}}{\binom{N}{l}} \\ A_{w,D}^{CCS} &\cong \sum_{l=0}^{l_{max}} \frac{A_{w,l}^{C_e} \cdot A_{l,D}^{C_i}(l, D)}{\binom{N}{l}} \end{aligned} \tag{3.12}$$

onde N é o tamanho do entrelaçador utilizado e l é o peso de Hamming da seqüência intermediária. As distribuições utilizadas acima correspondem às distribuições do código externo (índice C_e) e do código interno (índice C_i). É muito mais fácil computacionalmente obter as IOWEF e CIOWEF individuais de cada código componente do que do código global, sem usar o entrelaçador uniforme.

Assim, unindo as equações 3.11 e 3.12, podemos escrever o limitante de união para a probabilidade de erro de um sistema com concatenação serial da seguinte forma:

$$P_b(e) \leq \frac{1}{NR_{C_e}} \sum_{w=1}^{NR_{C_e}} \sum_{l=l_{min}}^{l_{max}} w \frac{A_{w,l}^{C_e} \cdot A_{l,D}^{C_i}}{\binom{N}{l}}|_{D=e^{-\frac{E_s}{4N_0}}} \tag{3.13}$$

onde w é o peso de Hamming da palavra de entrada e R_{C_e} é a taxa do código externo.

Considerando que $l_{min} = d_{e,l}$[2], a distância livre do código externo, podemos reescrever a equação 3.13 da seguinte forma:

$$P_b(e) \leq \frac{1}{NR_{C_e}} \sum_{d=d_{min}}^{d_{max}} e^{-\frac{E_s}{N_0}d} \sum_{w=0}^{NR_{C_e}} w \sum_{l=d_{e,l}}^{l_{max}} \frac{A_{w,l}^{C_e} \cdot A_{l,D}^{C_i}}{\begin{pmatrix} N \\ l \end{pmatrix}} \tag{3.14}$$

Desta equação definimos o termo que chamamos de multiplicador de expoente:

$$M_d = \sum_{w=0}^{NR_{C_e}} \frac{w}{NR_{C_e}} \sum_{l=d_{e,l}}^{l_{max}} \frac{A_{w,l}^{C_e} \cdot A_{l,d}^{C_i}}{\begin{pmatrix} N \\ l \end{pmatrix}} \tag{3.15}$$

Para algum valor da relação sinal-ruído, o maior multiplicador do expoente é o fator que domina o valor do limitante de união. Assim, percebemos que, para grandes N's, seqüências que tem grandes l's intermediários não são relevantes para o desempenho do código, pois a sua influência é muito reduzida devido à presença do entrelaçador. Logo, seqüências com baixos valores de d e baixos valores de l são as mais importantes para o desempenho do código nesta situação.

Para altos valores da relação sinal ruído (RSR), a distância mínima do código ainda domina o desempenho, já que a exponencial decai mais devagar quanto menor for o valor de d. Seqüências com altos valores de w normalmente geram seqüências com altos valores de l. Seqüências com l próximos a N normalmente geram seqüências com altos pesos de saída, cuja influência desaparece rapidamente com um aumento da RSR.

A escolha apropriada do código externo influencia no valor do l_{min}, que não é necessariamente igual à distância livre do código externo, assim como na multiplicidade das seqüências intermediárias com baixo peso de Hamming. Uma seqüência com um dado peso de Hamming e um dado peso Euclidiano tem alta multiplicidade se há muitas seqüências com as mesmas características de pesos Euclidiano e de Hamming. A multiplicidade afeta o valor de $A_{w,d}$. Tanto o código externo como o interno devem ser escolhidos de forma a minimizar a multiplicidade das seqüências, principalmente aquelas com peso de Hamming intermediário baixo. Podemos perceber que o entrelaçador é uma grande causa de ganho, distribuindo bem as seqüências intermediárias com peso de Hamming baixo. O entrelaçador uniforme representa o desempenho de um entrelaçador médio, podendo haver casos piores e melhores. Na prática não existem entrelaçadores uniformes, mas é possível que haja alguns entrelaçadores com desempenho igual ou melhor que o entrelaçador uniforme.

[2]Isto nem sempre é verdade. Em alguns casos, $l_{min} > d_{e,l}$, como será visto no fim da seção 3.2.1.

3.2.1 Termos Dominantes do Limitante de União

Para encontrarmos o termo dominante da equação 3.14, precisamos explorar melhor os termos de $A^C_{w,d}$. Desejamos escrever $A^{C_e}_{w,l}$ e $A^{C_i}_{l,d}$ em função do número de vezes em que as seqüências divergem do estado inicial e retornam ao mesmo. Cada seqüência é também um possível evento de erro, e utilizaremos deste fato para reescrever o limitante de união.

Há uma treliça associada aos códigos-componente, e um tamanho em bits da palavra associada a cada passo da treliça. Embora não seja necessário que o tamanho da saída do código externo seja igual ao tamanho da entrada do código interno, é necessário que N seja um múltiplo de ambos.

Seja p o mínimo múltiplo comum do tamanho em bits da entrada do código interno e da saída do código externo. Assim, podemos considerar que as seqüências intermediárias pertencem a uma hiper-treliça rotulada por p bits. Dessa forma, há N/p passo nesta treliça, tanto para o código interno como para o código externo. Nas situações que vamos analisar o tamanho da saída do código externo é igual ao tamanho da entrada do código interno.

Para valores de N muito maiores do que a quantidade de estados de um código de treliça, podemos escrever o seguinte limitante:

$$A^C_{l,d} \leq \sum_{j=1}^{n_M} \begin{pmatrix} N/p \\ j \end{pmatrix} A^C_{l,d,j} \tag{3.16}$$

onde o termo $A^C_{l,d,j}$ indica a quantidade de palavras com peso de entrada l, peso de saída d, obtidas através da concatenação sem intervalos de j caminhos que divergem e retornam ao estado identidade somente uma vez. Um único caminho destes é um possível primeiro evento de erro. Podemos escrever as palavras código em função de primeiros eventos de erro. As permutações destes j eventos de erro, que podem ser iguais ou não, estão inclusas neste número, pois também é um evento de erro que gera uma seqüencia com peso de entrada l, peso de saída d e j eventos de erro. Todas as seqüências com peso de entrada l e peso de saída d são rearranjos destes j eventos de erro com intervalos sem erro, isto é, intervalos no qual a palavra código é idêntica à seqüência da palavra toda nula. Substituindo j por n_e, o número de eventos de erro do código externo, e por n_i, o número de eventos de erro do código interno, podemos escrever a seguinte equação:

$$A^{C_e}_{w,l} \cdot A^{C_i}_{l,D} = \begin{pmatrix} N/p \\ n_e \end{pmatrix} A^{C_e}_{w,l,n_e} \begin{pmatrix} N/p \\ n_i \end{pmatrix} A^{C_i}_{l,d,n_i} \tag{3.17}$$

o que faz com que a equação 3.14 tenha a seguinte forma:

$$P_b(e) \leq \frac{1}{NR_{C_e}} \sum_{d=d_{min}}^{d_{max}} e^{-\frac{E_s}{N_0}d} \sum_{w=0}^{NR_{C_e}} w \sum_{l=d_{e,l}}^{l_{max}} \frac{\begin{pmatrix} N/p \\ n_e \end{pmatrix} \begin{pmatrix} N/p \\ n_i \end{pmatrix}}{\begin{pmatrix} N \\ l \end{pmatrix}} A^{C_e}_{w,l,n_e} A^{C_i}_{l,d,n_i} \tag{3.18}$$

Os seguintes limitantes são válidos, especialmente para situações onde N é grande e $n, l \ll N$:

$$\begin{pmatrix} N \\ n \end{pmatrix} < \frac{N^n}{n!}$$
$$\begin{pmatrix} N \\ l \end{pmatrix} > \frac{(N-l+1)^l}{l!} > \frac{N^l}{l^l l!} \tag{3.19}$$

Assim, chegamos à seguinte formulação:

$$P_b(e) \leq \sum_{d=d_{min}}^{d_{max}} e^{-\frac{E_s}{N_0}d} \sum_{w=0}^{NR_{C_e}} \sum_{l=d_{e,l}}^{l_{max}} \sum_{n_e=1}^{n_e^M} \sum_{n_i=1}^{n_i^M} N^{n_e+n_i-l-1} \cdot \frac{l! l^l}{p^{n_e+n_i-1} n_e! n_i!} \cdot \frac{w}{k} A^{C_e}_{w,l,n_e} A^{C_i}_{l,d,n_i} \tag{3.20}$$

onde n_e^M e n_i^M representam o maior número de eventos de erro do código externo e interno que geram seqüências com peso igual a l e d, respectivamente, e k é o tamanho em bits uma palavra de entrada do código externo que rotula uma seção de treliça. Este termo é derivado de $R_{C_e} = k/p$, a taxa do código externo.

Como N é muito maior do que l, o termo que vai dominar o multiplicador do expoente M_d(vide 3.15) será aquele que tiver o maior expoente de N, dado pela função:

$$\alpha(d) = \max_{w,l} \{n_e + n_i - l - 1\} \tag{3.21}$$

Esta função retorna, para uma dada distância d do código, o maior expoente de N. Desejamos que esta função assuma os menores valores possíveis.

Para grandes valores da RSR, a distância mínima do código domina o desempenho do código, pois com o aumento da RSR os termos com alto d desaparecem rapidamente, com influência pouco significante para o limitante de união. Assim, seria interessante saber:

- Qual é o termo que domina quando $d = d_{min}$, a distância livre do código C, e qual é o valor de l que causa o maior expoente para N;

- Qual é a distância Euclidiana associada ao maior expoente de N. Podemos fazer isso relacionando as variáveis n_e^M e n_i^M com as distâncias livres dos códigos.

A análise dos expoentes depende muito dos códigos-componente utilizados, mas as duas situações descritas podem ser analisadas com apenas algumas considerações sobre os códigos-componente. Em qualquer situação, desejamos que o expoente seja o mais negativo possível, ou seja, o menor possível. Se o expoente α for positivo, o valor de N^α será muito grande e o código sera ruim, pois não teremos ganho. Seria bom se pudéssemos ter situações onde o expoente é sempre negativo, o que indicaria sempre um ganho ao se utilizar o entrelaçador.

Expoente de N para $d = d_{min}$

A distância mínima do código não é necessariamente a distância mínima do código interno, $d_{i,l}$. O maior número de eventos de erro do código interno que causa uma seqüência de saída com peso Euclidiano igual a d_{min} é:

$$n_i^M \leq \left\lfloor \frac{d_{min}}{d_{i,l}} \right\rfloor \tag{3.22}$$

pois qualquer evento de erro aumenta em no mínimo $d_{i,l}$ o peso Euclidiano da saída. A igualdade acontece quando uma seqüência de peso mínimo de saída do código é gerada pela concatenação de n_i^M eventos de erro, cada um gerando um peso de saída exatamente igual a $d_{i,l}$. A função $\lfloor x \rfloor$ retorna o maior inteiro positivo maior menor do que x.

O mesmo pode ser dito sobre o código externo, sendo que desta vez os valores correspondem a pesos de Hamming:

$$n_e^M(l) \leq \left\lfloor \frac{l}{d_{e,l}} \right\rfloor \tag{3.23}$$

O maior valor do expoente de N é:

$$\alpha(d_{min}) \leq \max_l \left\{ \left\lfloor \frac{l}{d_{e,l}} \right\rfloor + \left\lfloor \frac{d_{min}}{d_{i,l}} \right\rfloor - l - 1 \right\} \tag{3.24}$$

Vamos escrever l como sendo igual a $l = \gamma d_{e,l} + \epsilon_\gamma$, onde $0 \leq \epsilon_\gamma < d_{e,l}$. Vamos também dizer que $d_{min} = \beta d_{i,l} + \epsilon_\beta, 0 \leq \epsilon_\beta < d_{i,l}$. Os valores de γ e β são inteiros positivos ou zero. Assim, podemos reescrever a equação 3.24 da seguinte forma:

$$\alpha(d_{min}) \leq \max_l \{\gamma(1 - d_{e,l}) + \beta - \epsilon_\gamma - 1\} \tag{3.25}$$

Podemos concluir da equação acima que, se $d_{min} \gg d_{i,l}$, o expoente de N pode ser até positivo,

dependendo do valor de l, pois o valor de β será grande. Seria interessante se $d_{min} = d_{i,l}$, fazendo com que $\beta = 1$.

O menor valor possível para n_i^M é 1, já que é impossível $d_{min} < d_{i,l}$. O menor valor possível para $\gamma(1 - d_{e,l})$ depende da escolha do código externo. Se $d_{e,l} > 2$, $\gamma(1 - d_{e,l})$ será menor quanto maior for γ. O maior valor possível para ϵ_γ é $d_{e,l} - 1$, pois além disso teríamos um retorno a $\epsilon = 0$ com um aumento no valor de γ. Considerando a pior situação na qual $\epsilon_\gamma = 0$, temos:

$$\begin{aligned} \alpha(d) &\leq \max_{w,l} \{\gamma(1 - d_{e,l}) + \beta - \epsilon_\gamma - 1\} \\ &= \max_{w,l} \{\gamma(1 - d_{e,l}) + \beta - 1\} \end{aligned} \tag{3.26}$$

A função ser minimizada, omitindo o termo β, pode ser visualizada em 3 dimensões como mostra a figura 3.3, com γ e $d_{e,l}$ variando de 0 a 100.

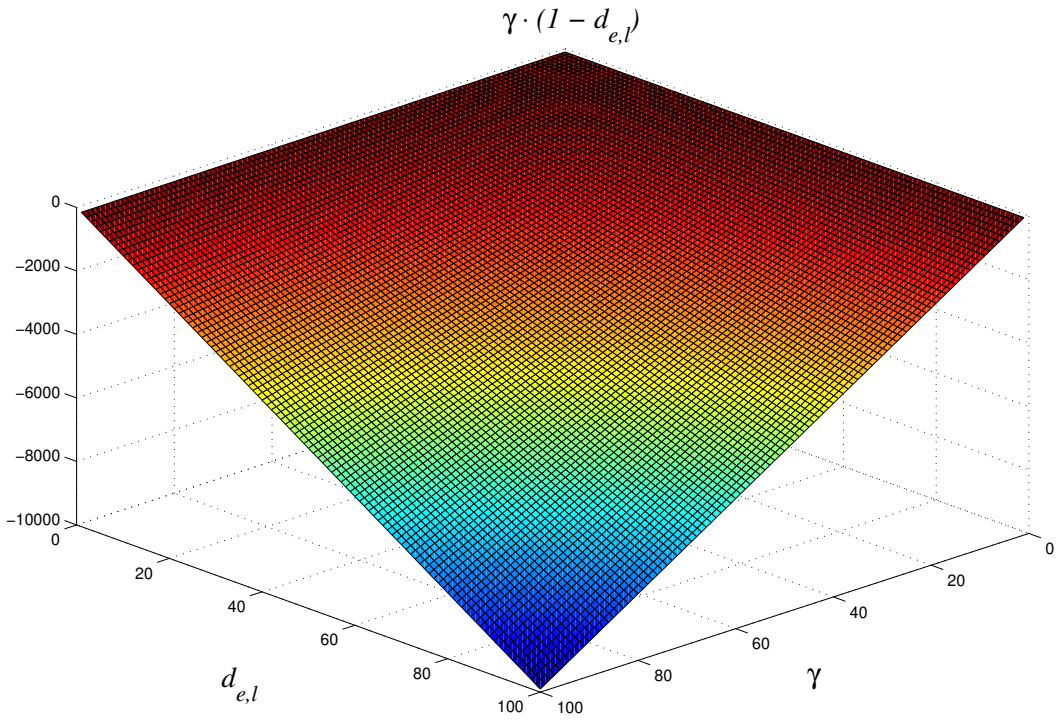

Fig. 3.3: Variação de $\gamma(1 - d_{e,l})$

Aumentar o valor de γ implica em mapear palavras com alto peso de Hamming de saída do código externo em seqüências de saída do código interno com peso $d_{e,l}$, o que involve manipular o codificador interno e possivelmente aumentar o número de memórias para que isso seja feito. Aumentar o valor de $d_{e,l}$ implica em aumentar o número de memórias do código externo. O maior

ganho de desempenho ocorre quando o aumento de γ é acompanhado de um aumento proporcional de $d_{e,l}$. Embora tenhamos ganho ao utilizar um código externo com muitas memórias e um código interno com poucas memórias, possivelmente teríamos um ganho melhor com uma outra distribuição de memórias.

Concluímos que $d_{e,l} > 2$, d_{min} deve ser o mais próximo possível de $d_{i,l}$, e que o menor peso intermediário que gera uma seqüência de saída com peso Euclidiano mínimo deve ser o maior possível, fazendo com que γ seja grande.

Maior Expoente de N

Para encontrar o maior expoente de N, denominado α_M, temos que maximizar o termo $(n_e + n_i - l - 1)$ em função de w,l e d:

$$\alpha_M = \max_d\{\alpha(d)\} = \max_{w,l,d}\{n_e^M + n_i^M - l - 1\} \tag{3.27}$$

Para situações na qual o código interno permite que seqüências com peso de Hamming de entrada 1 geram seqüências finitas de saída, temos que $\max(n_i^M) = l$, pois cada 1 individualmente gera um evento de erro finito em alguma dada seqüência que certamente ocorrerá devido ao entrelaçador uniforme. A equação 3.27 assume então a seguinte forma:

$$\alpha_M = n_e^M - 1 \geq 0 \tag{3.28}$$

de onde podemos concluir que nesta situação o entrelaçador não gera nenhum ganho. Então, precisamos utilizar um código interno recursivo de modo que o menor peso de entrada que causa uma saída e retorno ao estado inicial tenha peso 2. Nestas situações, uma seqüência de entrada do código interno com peso de Hamming l pode gerar no máximo $\lfloor l/2 \rfloor$ eventos de erro, esta situação ocorrendo quando cada 2 bits de entrada gerarem um evento de erro. Novamente, esta seqüência com certeza existira por causa do entrelaçador uniforme.

O maior valor que n_e^M pode assumir depende do peso da seqüência intermediária e da distância livre do código externo. Assim, $\max(n_e^M) \leq \lfloor l/d_{e,l} \rfloor$, a igualdade valendo quando cada $l/d_{e,l}$ bits é gerado por um evento de erro. Considerando que $l = \gamma d_{e,l} + \epsilon_\gamma$, a equação 3.27 fica:

$$\begin{aligned}\alpha_M &= \max_{w,l,d}\{n_e^M + n_i^M - l - 1\} \\ &\leq \max_{l}\{\left\lfloor \frac{l}{d_{e,l}} \right\rfloor + \left\lfloor \frac{l}{2} \right\rfloor - l - 1\} \\ &= \max_{l}\{\left\lfloor \frac{l}{d_{e,l}} \right\rfloor - \left\lfloor \frac{l+1}{2} \right\rfloor - 1\} \\ &= \max_{\gamma,\epsilon_\gamma}\{\gamma - \left\lfloor \frac{\gamma d_{e,l} + \epsilon_\gamma + 1}{2} \right\rfloor - 1\}\end{aligned} \tag{3.29}$$

Se $d_{e,l} \geq 2$, garantido se o código externo for convolucional, o pior valor que γ pode assumir é 1. O pior valor que ϵ_γ pode assumir é o seu menor valor possível, igual a zero. Substituindo estes valores na equação acima, chegamos à conclusão que:

$$\alpha_M \leq - \left\lfloor \frac{d_{e,l} + 1}{2} \right\rfloor \tag{3.30}$$

que é um valor sempre negativo, considerando a hipótese $d_{e,l} \geq 2$.

O valor de d associado ao maior expoente de N depende se o código externo tem distância livre par ou ímpar. Seria interessante se este valor fosse o maior possível, pois assim o expoente $e^{-\frac{E_s}{N_0}d}$ diminuiria mais rapidamente com o aumento da relação sinal-ruído e sua influência no limitante será reduzida.

No caso de $d_{e,l}$ par, e considerando $\gamma = 1, \epsilon_\gamma = 0$, temos que $l = d_{e,l}$. Além disso, o valor de n_i^M deve ser o maior possível, e isso acontece quando os l bits que valem 1 da seqüência intermediária causam $(l/2)$ eventos de erro, cada um causado por 2 dos l bits. Denotamos por d_{ef} a menor distância Euclidiana de saída entre seqüências geradas por seqüências de entrada com distância de Hamming igual a 2. Este parâmetro é uma propriedade do codificador interno. Assim, o menor peso Euclidiano de saída possível d_{min,α_M} para as condições que geram o maior expoente de N acontece quando cada um dos $(d_{e,l}/2)$ eventos de erro aumenta em d_{ef} o peso Euclidiano da seqüência de saída, gerando um peso resultante igual a:

$$d_{min,\alpha_M} \geq \frac{d_{e,l} \cdot d_{ef}}{2} \tag{3.31}$$

No caso de $d_{e,l}$ ímpar, as mesmas considerações sobre γ e ϵ_γ são válidas. Seja d_3 o menor distância Euclidiana de saída entre seqüências de entrada com distância de Hamming igual a 3[3]. Como no caso anterior, n_i^M deve ser máximo. Isso acontece quando temos um único evento de erro com peso de

[3]Este é o menor peso de Hamming ímpar de entrada possível, dado que o codificador externo é recursivo.

entrada com peso 3 seguido de alguns eventos de erros com peso de entrada par, sendo que a entrada se refere ao código interno. Se houvessem somente dois eventos de erros ímpares, o código seria par. Três eventos de erro, cada um com peso de entrada 3, podem ser gerados por um evento de erro com peso de entrada igual a 3 e 3 eventos de erro com peso de entrada igual a 2, aumentando assim o valor de n_e^M. O mesmo raciocínio pode ser aplicado para seqüências com mais erros. Assim, sendo $l = d_{e,l}$ ímpar, temos que o menor valor possível da distância Euclidiana d_{min,α_M} associado ao expoente máximo de N é:

$$d_{min,\alpha_M} \geq \frac{(d_{e,l} - 3) \cdot d_{ef}}{2} + d_3 \tag{3.32}$$

Os valores que encontraremos através das equações 3.31 e 3.32 correspondem a distâncias Euclidianas. Queremos que estes valores sejam os maiores possíveis, pois assim o valor necessário para que o ruído cause um erro destes seria maior, conseqüentemente com uma probabilidade menor de ocorrer. Devemos então tornar os valores $d_{e,l}$,d_{ef} e d_3 os maiores possíveis.

É fácil fazer com que $d_3 = \infty$, pois é simples fazer com que o código interno só permita que seqüências de entrada pares tenham peso de saída finita. Quando, $d_i = \infty$ para qualquer i ímpar e o código é chamado código par. Caso contrário, ele é chamado código ímpar. Quando $d_3 = \infty$ e $d_{e,l} = 3$, $l_{min} = 4$, pois este é o menor valor de Hamming intermediário que causa um caminho que sai e retorna ao estado identidade.

A maior dificuldade é fazer com que d_{ef} seja o maior possível.

3.3 Diretrizes de Projeto

Juntando as informações obtidas na seção anterior, chegamos às seguintes diretrizes para o projeto de um sistema com concatenação serial:

- Tanto o código interno e o externo devem ser recursivos, para que se haja sempre um ganho do entrelaçador. Assim, os valores para n_e^M e n_i^M serão os menores possíveis, contribuindo para um valor mais negativo do maior expoente de N.
- A distância livre efetiva d_{ef} deve ser a maior possível.
- O código externo deve ter a maior distância livre possível. Entretanto, é preferencial ter uma distribuição equilibrada de memórias entre o código interno e o código externo, de modo que $d_{e,l} \sim \gamma$, como mostra a figura 3.3.
- O código externo deve ter distância livre preferencialmente ímpar. Nesta situação, devemos fazer com que $d_3 = \infty$. Sendo estas duas condições satisfeitas, $l_{min} = d_{e,l} + 1$.

- Seqüências de saída que tem peso Euclidiano igual a $d_{i,l}$ devem ter peso de entrada código interno altos. Isto corresponde a ter um alto valor de γ. Se possível, o menor peso de entrada de uma seqüência com peso mínimo deve ter valor próximo a $(n \cdot d_{e,l} - 1)$, onde n é um número inteiro.

3.4 Considerações sobre o Limitante de União

Devemos analisar a validade do limitante de união que foi desenvolvido neste capítulo. Três pontos são de importância.

O primeiro é sobre a decodificação utilizada para receber a informação transmitida. O limitante de união considera a hipótese de decodificação por máxima verossimilhança, o que não é viável em situações onde o tamanho do bloco é grande.

O segundo é sobre o entrelaçador. Um entrelaçador real vai permutar uma dada seqüência de entrada para uma dada seqüência de saída com probabilidade 1. As seqüências de saída do código externo não são uniformemente distribuídas, o que pode fazer com que $l_{min} > d_{e,l}$. Se o número de seqüências de saída do código externo com peso 2 for muito menor do que o número de seqüências de entrada do código interno com peso de saída igual a d_{ef}, é possível que o valor real de d_{ef} seja maior do que o projetado, resultando num valor inferior para a probabilidade de erro. O mesmo pode ser dito sobre a distância mínima do código, que limita o desempenho do sistema para altos valores da relação sinal-ruído.

O limitante de união diverge para relações sinal-ruido próximas à taxa de corte do canal. Assumimos que a probabilidade de dois eventos de erro conjuntos ocorrerem é muito menor do que a probabilidade de um único evento de erro. Isto tem mais validade nas regiões de sinal-ruido médias e altas. Para valores baixos da RSR, a soma das probabilidades sera com muita chance maior do que 1, o que não corresponde a um valor real para uma probabilidade. Então, devemos considerar o limitante válido somente para a região abaixo da taxa de corte do canal.

Entretanto, simplificações para contornar a dificuldade computacional de se calcular o limitante de união para entrelaçadores grandes faz com que este fenômeno não seja percebido. Devido à complexidade de se calcular as relações entre todas as seqüências possíveis, limitamo-nos às seqüências com peso Euclidiano de saída menores do que aproximadamente 15 vezes a distância livre mínima do código interno. Seqüências com peso Euclidiano maior do que este eram ignoradas. Esta limitação restringiu o peso de Hamming das seqüências intermediárias(seqüências entre os codificadores), tornando desnecessário considerar seqüências que tinham peso intermediário maior do que um certo valor encontrado. Esta escolha não influencia muito o valor do limitante de união para valores médios e altos da RSR, pois observamos que os limitantes tendem a convergir, sendo que esta convergência

tende cada vez mais cedo ao valor real do limitante quanto maior for a distância Euclidiana considerada como limite. O limitante obtido será então uma aproximação do limitante real. Este ponto será melhor analisado no capítulo 5.

3.5 Síntese

Neste capítulo apresentamos a concatenação serial de códigos. Desenvolvemos um limitante de união e a partir do limitante obtivemos diretrizes de projeto para os códigos-componente do sistema. Analisamos também a validade do limitante de união. As diretrizes de projeto para sistemas com concatenação serial obtidas neste capítulo, resumidas na seção 3.3, serão utilizadas no capítulo 4 para guiar a procura bons codificadores internos.

Capítulo 4

Método para gerar codificadores internos BGU

Neste capítulo introduziremos um método para gerar codificadores bit-geometricamente uniformes. Provaremos que este método de fato gera codificadores BGU. Este método permite um controle razoável sobre parâmetros dos codificadores gerados, sugerindo assim a sua utilização para gerar codificadores a serem utilizados como codificadores internos em sistemas com concatenação serial.

4.1 Algoritmo

Nesta seção vamos descrever o algoritmo para a construção de um bom codificador interno BGU para ser utilizado num sistema de concatenação serial. Como parâmetros de entrada temos:

- a constelação S a ser utilizada;
- $d_{e,l}$, a distância de Hamming livre do código externo;
- m, o número de memórias binárias utilizadas;
- k, o tamanho em bits da entrada do codificador interno.

Utilizaremos a seguinte nomenclatura:

- E é o codificador que desejamos encontrar
- Σ_m é o espaço de estados de um codificador com m memórias binárias, possuindo então 2^m estados;

- T é um dos grupos que representa uma seção de treliça de um código TCM composto pela tríade (σ_i, s, σ_f), o estado inicial e final $\sigma_i, \sigma_f \in \Sigma_m$ e o sinal associado $s \in S$;
- F_0 o subgrupo de T com os ramos que saem do estado identidade σ_e, i.e, $\sigma_i = \sigma_e$;
- F_{00} o subgrupo de T com os ramos que saem do estado identidade e retornam ao mesmo, i.e., $\sigma_i = \sigma_f = \sigma_e$;
- E_0 é o mapeamento entre F_0 e o espaço de Hamming de entrada H_k;
- E_j é o mapeamento entre os ramos que saem do j-ésimo estado e o espaço de Hamming correspondente;
- S_0 é o subconjunto de S associado aos ramos de F_0;
- A é o subgrupo de T que contém todos os ramos que são rotulados através de E pela palavra toda nula. Este grupo também é o grupo gerador de Σ_m.

O resultado do algoritmo é F_0, E_0, A e uma associação entre A e Σ_m. A partir destes termos podemos definir completamente o codificador.

1^o **passo:**

Escolher grupo gerador G_S de S. ◇

Dependendo da constelação utilizada, há mais de uma possibilidade para o grupo G_S. Como futuramente desejamos que um subgrupo deste grupo gere também o espaço de Hamming que será mapeado em sinais da constelação, demos preferência a grupos diedrais, pois desta forma temos mais geradores com ordens menores e é mais fácil encontrar grupos geradores do espaço de Hamming nestas condições. As constelações utilizadas são o produto cartesiano da constelação $8PSK$. Uma forma fácil de se obter os grupos geradores destas constelações é através do produto cartesiano de L grupos geradores de uma única constelação $8PSK$. Os grupos que geram a constelação $8PSK$ são os grupos Z_8 e D_4. No espaço Euclidiano, os seus elementos correspondem à rotações e reflexões.

2^o **passo:**

Escolher um grupo de ligação L, que seja tanto um subgrupo de Γ_{H_k}, o grupo de simetrias de H_k, como um subgrupo $G \subseteq G_S$. Não faremos mais distinção entre o conjunto de elementos de H_k e o grupo G_{H_k} que gera este conjunto. Escolher geradores de H_k cuja relação entre si seja a mesma da relação dos geradores de L. Os geradores de H_k podem ser descritos através de combinações de permutações e somas. Este conjunto de geradores será chamado de C_H. Escolher geradores de G

que satisfaçam à mesma condição. Este conjunto será chamado de C_G, pois seus elementos geram o subgrupo G. $\diamond$

Como indicado na seção 2.3.1, podemos obter simetrias do espaço de Hamming através de permutações de coordenadas, soma de elementos do espaço a todos os outros elementos, e combinações de ambos. Podemos formar um grupo a partir destes elementos, sendo que um subconjunto destes elementos escolhidos serão os geradores do grupo. Dada a dimensão de um espaço de Hamming, há um número finito de possibilidades de grupos. Se um dos grupos geradores de H_k for isomorfo à um subgrupo $G \subseteq G_S$, podemos prosseguir com o algoritmo.

O grupo L faz com que o grupo H_k e um subgrupo G sejam estruturalmente os mesmos, e o isomorfismo existente entre eles seja um automorfismo trivial, com apenas uma troca de nomes. No nosso caso, G será um dos grupos que gera S_0 e também F_0, a sub-treliça que contém somente os elementos que saem do estado identidade. Homomorfismos de G definirão o subgrupo F_0 da treliça T.

Dado um mapeamento dos elementos de C_{H_k} em C_G, existe no máximo um homomorfismo de H_k em G [9]. Sejam g_i, g_j geradores de G e h_i, h_j os geradores de H_k para os quais os geradores de G são mapeados. Teremos este isomorfismo se:

$$f : G \to H_k,\ f(g_i \cdot g_j) = h_i \cdot h_j; \qquad \forall g_i, g_j \in C_G$$

Com este homomorfismo, garantimos a primeira condição do teorema 2.5.3, como demonstramos a seguir:

Prova Seja $h_i, h_j, h_l \in H_k$, elementos do grupo H_k compostos por somas e permutações. A distância de Hamming entre h_i e h_j é:

$$d_h(h_i, h_j) = \sum_{n=1}^{k} h_{i,n} \oplus h_{j,n} \tag{4.1}$$

onde $\oplus$ é a adição binária. Seja uma permutação $p_l(n) : N_k \to N_k$, definida para $n \in N_k = 1, 2, \cdots, k$, que retorna a posição para qual o bit n de uma palavra binária com k bits deve ser permutado. Se o termo h_l(que representa tanto o termo do conjunto H_k como o elemento do grupo gerador de H_k) for a permutação $p_l(x)$, a sua aplicação sobre h_i e h_j simultaneamente tem o seguinte impacto na distância de Hamming entre ambos:

$$d_h(h_i \cdot h_l, h_j \cdot h_l) = \sum_{n=1}^{k} h_{i,p(n)} \oplus h_{j,p(n)} \tag{4.2}$$

Como a ordem dos fatores não altera o valor da soma, a distância de Hamming é mantida.

Para o caso de h_l ser um termo a ser adicionado, temos a seguinte situação:

$$d_h(h_i \cdot h_l, h_j \cdot h_l) = \sum_{n=1}^{k} h_{i,n} \oplus h_{l,n} \oplus h_{j,n} \oplus h_{l,n} = \sum_{n=1}^{k} h_{i,n} \oplus h_{j,n} \tag{4.3}$$

pois $h_{l,n} \oplus h_{l,n} = 0$, para qualquer valor de $h_{l,n}$.

No caso de h_l ser uma combinação de uma permutação e de uma soma, basta realizar os dois passos (permutação e soma) separadamente para se chegar à conclusão que, sendo H_k um grupo cujos geradores sejam uma composição de permutações e somas:

$$d_h(h_i \cdot h_l, h_j \cdot h_l) = d_h(h_i, h_j) \tag{4.4}$$

Dados $g_i, g_j, g_l \in G$, que geram respectivamente os sinais $s_i, s_j, s_l \in S$ a partir de um ponto identidade $s_0 \in S$. Para grupos que geram constelações em espaços Euclidianos, temos que, definida uma função $d_e(s_i, s_j) : S \to R$, que retorna a distância Euclidiana entre $s_i, s_j \in S$:

$$d_e(g_i \cdot g_l, g_j \cdot g_l) = d_e(g_i, g_j) \tag{4.5}$$

porque g_l é uma isometria que mantém a distância Euclidiana entre os elementos.

A união entre o grupo G e o grupo H_k é feita através do grupo L, que faz com que a aplicação de uma operação sobre um grupo resulte na mesma operação, com outro nome, no outro grupo. Há uma correspondência um para um entre os elementos de G, e L, através de uma função $f_G : G \leftrightarrow L$, e entre L e H_k, através de uma função $f_{H_k} : L \leftrightarrow H_k$. Estas funções são os isomorfismos entre os grupos. A composição destas duas funções, que chamaremos de $f_{G,H_k} : G \leftrightarrow H_k$ também é um isomorfismo. Desta forma, sendo o codificador E_0 o isomorfismo f_{G,H_k}, ele é BGU pois:

$$\begin{aligned} d_h(f_{G,H_k}(g_i), f_{G,H_k}(g_j)) &= d_h(f_{G,H_k}(g_i) \cdot h_l, f_{G,H_k}(g_j) \cdot h_l)) \\ &= d_h(f_{G,H_k}(g_i) \cdot f_{G,H_k}(g_l), f_{G,H_k}(g_j) \cdot f_{G,H_k}(g_l)) \\ &= d_h(f_{G,H_k}(g_i \cdot g_l), f_{G,H_k}(g_j \cdot g_l)) \end{aligned} \tag{4.6}$$

Substituindo g_l por g_j^{-1} e f_{G,H_k} por E_0, chegamos a:

$$\begin{aligned} d_h(E_0(g_i), E_0(g_j)) &= d_h(E_0(g_i \cdot g_j^{-1}), E_0(e)) \\ &= w_h(E_0(g_i, g_j^{-1})) = w_h(E_0(g_i^{-1}, g_j)) \end{aligned} \tag{4.7}$$

Como qualquer termo do grupo pode ser obtido através do produto dos geradores, a equação anterior vale para qualquer elemento dos grupos. Assim, sendo E_0 este homomorfismo, ele é BGU. □

Embora tenhamos um possível E_0, não temos F_0, pois não sabemos o estado final de cada transi-

ção. Sabemos o estado inicial e o sinal associado. Precisamos também encontrar um bom E_0, pois há vários automorfismos possíveis de $H_k \rightarrow H_k$ e de $G \rightarrow G$ que continuam gerando soluções BGU.

3^o **passo:**

Definir o grupo F_{00}, conjunto de ramos que saem e retornam ao estado σ_e. Através deste grupo definimos a estrutura do grupo A, pois as operações entre os estados serão definidas. Fazer com que o grupo A seja o mais simples possível, e de preferência isomorfo ao grupo Z_2^m. $\diamond$

Há várias possibilidades para o grupo F_{00}. O grupo F_{00} será definido através de um subgrupo normal de F_0. Uma vez definido F_{00}, dividimos o grupo F_0 em partições, cada uma correspondendo a um estado final. Quanto mais simples for o grupo F_{00}, mais fácil será encontrar o subgrupo A de T.

O grupo A é composto pela tríade (σ_i, s, σ_f). Os três termos que definem o ramo estão condicionados a obedecer a regra definida pela estrutura do espaço de estados obtida neste passo. Por exemplo, se $a_i \cdot a_j = a_k$ para $a_i, a_j, a_k \in A$, isto deve ser verdade para todos os elementos dos ramos. Isto é, $\sigma_f^i \cdot \sigma_f^j = \sigma_f^k$ e assim por diante. A operação $\cdot$ realizada sobre cada termo é realizada de acordo com o grupo correspondente. Assim, se sabemos a estrutura do grupo do espaço de estados, sabemos a estrutura do grupo A. Ainda não definimos o grupo A porque não sabemos quais serão os outros dois elementos necessários para definir a tríade (σ_i, s, σ_f). Falta determinar, para cada estado inicial, o estado final e o sinal associado ao ramo.

Se possível, A deve ser isomorfo ao grupo Z_2^m, pois isso facilita muito a procura dos seus termos. Isto é possível quando o número de geradores de L for menor ou igual a m. Nestas situações, pode-se em alguns casos determinar o estado final das transições a partir do estado inicial e da palavra de entrada através de equações de paridade.

Sabemos através de F_{00} quais elementos do grupo L causam uma transição paralela do estado identidade para o estado identidade. É razoável considerar que, se menos elementos com peso de entrada de Hamming igual a 2 causarem esta transição, mais fácil será de atribuir altos pesos Euclidianos para estas.

Para fazer com que o codificador seja par precisamos tomar cuidado com os automorfismos do próximo passo que faremos em H_k, pois isto pode modificar os pesos das transições entre estados e também os pesos das seqüências que retornam ao estado identidade.

4^o **passo:**

Procurar mapeamentos do conjunto C_{H_k} que gerem automorfismos em H_k, dando preferência à geradores com baixo peso de Hamming. Através deste automorfismo devemos diminuir ao máximo a quantidade de palavras que causam transições paralelas de σ_e para σ_e com peso de Hamming de entrada igual a 2. $\diamond$

A escolha apropriada dos elementos que compõem C_{H_k} é um dos dois fatores que determina se o código será par com $d_3 = \infty$, pois define as partições a serem utilizadas para realizar as transições paralelas. Transições paralelas são ramos que contém o mesmo estado inicial e final. Para que o código seja par, todas as transições paralelas devem ter o mesmo tipo de peso de Hamming, par ou ímpar. Isto implica inicialmente que, a partir do estado inicial, alguns estados só receberão seqüências com peso ímpar e outros só com peso par. Podemos classificar então os estados como sendo pares ou ímpares. Outras condições para que o codificador seja par são:

- um estado ímpar só recebe transições com peso de Hamming ímpar de estados pares, e só recebe transições com peso de Hamming par de estados ímpares;
- um estado par só recebe transições com peso de Hamming ímpar de estados ímpares, e só recebe transições com peso de Hamming par de estados pares.

O estado identidade é definido como um estado par.

Há vantagens em se minimizar a multiplicidade das palavras de entrada com peso 2. Quanto menos transições deste tipo houver, mais fácil será de fazer com que estas possuam um peso Euclidiano de saída alto, possibilitando assim um valor maior para a distância livre efetiva.

É mais importante aumentar o peso de saída das seqüências que saem e retornam do e a σ_e somente uma vez com peso de entrada igual a 2 do que aquelas que o fazem com peso par maior. Isto acontece porque a concatenação de seqüências com peso de entrada 2 gera seqüências com peso maior, e este efeito aumenta fatorialmente. Por exemplo, se em um passos da treliça temos n_2 seqüências com peso de entrada 2 e que divergem da seqüência toda nula somente uma vez, e n_4 seqüências que o fazem com peso de entrada igual a 4, em p passos teríamos aproximadamente $p \cdot n_4$ seqüências com peso de entrada igual a 4 que divergem somente uma vez, mas teríamos aproximadamente $(p! \cdot n_2^2)/(p-2)!$ seqüências com peso 4 que divergem da seqüência toda nula 2 vezes. Para algum valor finito de p a influência das n_2 seqüências com peso de entrada igual a 2 tem muito mais importância para a multiplicidade das seqüências com peso de entrada igual a 4 do que as seqüências com peso de entrada 4 que divergem somente uma vez. O mesmo raciocínio pode ser aplicado para qualquer seqüência de entrada com peso par. Logo, é melhor fazer com que seqüências com peso de entrada igual a 2 tenham um peso Euclidiano maior, pois seqüências com peso de Hamming maior ocorrem com mais freqüência através da concatenação de várias seqüências menores com peso 2.

O peso de Hamming dos geradores escolhidos influencia nas posições das palavras de entrada que vão gerar seqüências com peso crítico. Por posição entendemos os expoentes dos geradores. Todo termo de L pode ser escrito como uma combinação de geradores, onde cada gerador tem um expoente associado ao número de vezes que ele é utilizado. O expoente é um número inteiro não negativo menor do que a ordem do gerador.

Geradores ímpares sempre geram palavras ímpares se o seu expoente for ímpar e sempre geram palavras pares se o seu expoente for par, se utilizados individualmente. Trocar um gerador ímpar por outro ímpar ou um par por outro par não altera a paridade de uma posição. O peso final de uma palavra qualquer somada com uma palavra com peso ímpar é o contrário da sua paridade inicial. Geradores pares, como aceitáveis neste trabalho, só podem gerar palavras com peso par. A soma de duas palavras com peso de Hamming par gera uma palavra com peso de Hamming par. Assim, é óbvio que necessitamos de pelo menos um gerador ímpar para gerarmos todo o espaço de Hamming de entrada.

Não há necessidade em utilizar geradores com altos pesos de Hamming, pois o mais importante é se o peso dos geradores é par ou ímpar. O resultado obtido através de um automorfismo que mapeia geradores pares em outros geradores pares pode ser obtido através do próximo passo, onde vamos procurar automorfismos do grupo gerador da constelação.

A quantidade de geradores ímpares é um dos fatores mais importantes para a determinação das posições de segmentos críticos. Dado um conjunto de geradores, a troca de um gerador com peso de Hamming ímpar por outro com peso de Hamming ímpar não altera a posição das palavras com peso de Hamming ímpar. A paridade dos geradores utilizados também influencia se o código será par ou ímpar. Como desejamos que $d_3 = \infty$, devemos fazer com que o mapeamento inicial (definido no passo anterior) não permita que palavras com peso de Hamming de entrada ímpares causem uma transição paralela.

5^o **passo:**

Conhecemos o expoente dos geradores que geram segmentos críticos (i.e., todas a transições paralelas, sendo mais importantes aquelas com peso baixo e ramos que saem do estado identidade com peso 1) pois definimos no passo anterior qual será o conjunto C_{H_k} final. Procurar automorfismos do subgrupo gerado por C_G de modo a fornecer altos pesos Euclidianos para estes ramos críticos. Obtemos assim uma função $I_G : G \rightarrow G$. Este automorfismo é completamente determinado pelo mapeamento dos geradores. Este ponto é extremamente importante na otimização do código.

Como critério para a escolha do automorfismo, vamos tentar maximizar o peso Euclidiano das palavras com peso 2 que causam uma transição paralela. Isto nos fornece um limite superior para $d^i_{f,ef}$. Vamos também tentar fazer com que as palavras com peso de entrada 1 tenham no mínimo a metade deste valor. Vamos também tentar fazer com que os sinais com baixo peso Euclidiano tenham altos pesos de Hamming de entrada, de modo que a influência destes seja minimizada devido ao ganho do entrelaçador. $\diamond$

Neste passo determinamos qual é a partição da constelação S que causará transições paralelas de σ_e para σ_e. Como o código é GU, cada conjunto de transições paralelas terá um coset desta partição

associado a ele. As partições obtidas podem ser diferentes das partições normalmente utilizadas na construção de bons códigos GU, pois estas não levam em conta as condições de otimização de um sistema com concatenação serial.

Por exemplo, em [12], a partição ótima com 4 elementos nos subconjuntos da constelação $2 \times 8PSK$ foi gerada pelo subgrupo: {00,44,04,40}. Se tivéssemos que associar um grupo binário {0000, 1100, 1111, 1100}(os elementos de H_k associados ao grupo F_{00} foi determinado no passo anterior) a este subgrupo, as palavras com peso 2 teriam peso Euclidiano 4, e a palavra com peso 4 teria peso 8. Se quiséssemos dar o maior peso possível às palavras com peso 2, e ao mesmo tempo ter um rótulo BGU, poderíamos utilizar o subgrupo: {00,24,40,64}. Associando os elementos na mesma ordem em que aparecem, teríamos que as palavras com peso 2 teriam peso 6, e a palavra com peso 4 teria peso 4. Note que, enquanto no primeiro caso o subgrupo é gerado por dois geradores de ordem 2, {44 e 40} (e o subgrupo é isomorfo a Z_2^2), o segundo é gerado por um único gerador de ordem 4 {24} (e o isomorfismo é com Z_4). Esta variação do subgrupo acontece com a escolha de F_{00}.

6^o **passo:**

Associar os grupos até agora definidos da seguinte forma:

$$I_H(H) \leftarrow H_k \leftarrow L \rightarrow G_S \rightarrow I_G(G) \tag{4.8}$$

Como um isomorfismo de um isomorfismo ainda é um isomorfismo, ainda temos um isomorfismo de H_k em G, o que faz com que o código seja BGU. Através deste mapeamento obtemos E_0. Este mapeamento é completamente determinado pelo mapeamento dos geradores. O estado final de cada ramo é determinado pelas partições obtidas na determinação de F_{00}. Assim, obtemos o subgrupo F_0 de T, pois sabemos o estado inicial (σ_e), o sinal de S associado (através do isomorfismo deste passo) e o estado final (através das partições determinadas no terceiro passo). Definimos também o subgrupo F_{00}, conjunto de ramos que causa transições paralelas de σ_e para σ_e. Todos os outros conjuntos que causam transições paralelas são cosets deste grupo. A determinação de qual coset causa qual transição é importante na hora de escolher os elementos que formarão o grupo A.

Como ainda temos um isomorfismo entre H_k e G, ainda temos um codificador E_0 BGU, determinado pelo isomorfismo obtido neste passo. A prova mostrada no segundo passo ainda vale. O codificador E_0 obtido é agora considerado bom, pois o projetamos de acordo com as diretrizes para o projeto de um codificador interno para sistemas com concatenação serial. Ele não será mais alterado. $\diamond$

7^o passo:

Encontrar candidatos para o grupo A. Dentre todos os possíveis elementos do grupo T, escolher aqueles que satisfazem à condição:

$$w_h(E_0(f)) = w_h(E_0(a \cdot f \cdot a^{-1})); \qquad f \in F_0, a \in T \tag{4.9}$$

Estes candidatos formam o conjunto A'. Todos os elementos do grupo T são obtidos através do produto cartesiano $\Sigma_m \times G_S \times \Sigma_m$. ◇

No caso particular quando o espaço de estados Σ_m for abeliano, podemos ignorar as relações entre os espaços de estado, considerando somente o sinal da constelação g. Através do E_0 já encontrado, há uma relação $G \rightarrow H_k$, e isto é suficiente para testar a condição 4.9 dado que o espaço de estados Σ_m é abeliano. Para qualquer valor σ_i do estado inicial ou final de um ramo $f \in T$, e sendo $a_k = \sigma_{k,i}, g_{a_k}, \sigma k, f$, temos:

$$\begin{aligned} a_k \cdot f \cdot a_k^{-1} &= (\sigma_{k,i} \cdot \sigma_{f,i} \cdot \sigma_{k,i}^{-1}, g_{a,k} \cdot g_f \cdot g_{a,k}^{-1}, \sigma_{k,f} \cdot \sigma_{f,f} \cdot \sigma_{k,f}^{-1}) \\ &= (\sigma_{f,i} \cdot \sigma_{k,i} \cdot \sigma_{k,i}^{-1}, g_{a,k} \cdot g_f \cdot g_{a,k}^{-1}, \sigma_{f,f} \cdot \sigma_{k,f} \cdot \sigma_{k,f}^{-1}) \\ &= (\sigma_{f,i}, g_{a,k} \cdot g_f \cdot g_{a,k}^{-1}, \sigma_{f,f}) \end{aligned} \tag{4.10}$$

onde $\sigma_{f,i}, \sigma_{f,f}$ representam o estado inicial e final do ramo f, respectivamente, g_{a_k} é o sinal associado ao ramo a_k e g_f corresponde ao sinal associado ao ramo f. Basta que $g_{a,k} \cdot g_f \cdot g_{a,k}^{-1} = g_f$ para que $a_k \cdot f \cdot a_k^{-1} = f$. O número de candidatos é a cardinalidade de S.

Se Σ_m não for abeliano, haverá um número maior de elementos a serem testados, pois os elemento a serem testados serão combinações dos elemento de Σ_m com elementos de G_S. Mesmo neste caso, só precisamos nos preocupar com o estado final de $(a_k \cdot f \cdot a_k^{-1})$, pois o estado inicial seria naturalmente igual à identidade já que $\sigma_{f,i} = \sigma_e$

Como vamos escolher os elementos do grupo A entre os candidatos que satisfazem a condição 4 do teorema 2.5.3, esta condição é satisfeita. Qualquer grupo formado com estes candidatos, com cardinalidade igual ao número de estados do código, geraria um codificador BGU juntamente com E_0.

8^o passo:

Para cada estado distinto da identidade, definir qual coset de F_{00} causaria uma transição que faz com que o codificador retorne ao estado identidade. Estas palavras estão associadas, através de E_0, a sinais. Esta é a partição da constelação (e do espaço de Hamming através de E_0) que, ao ser multiplicada por um elemento $a \in A$ apropriado, gera os ramos que retornam ao estado identidade.

Algumas regras tem que ser obedecidas na determinação destas partições:

- Não resultam em auto-laços (caminhos de um estado para ele mesmo) na treliça com peso Euclidiano nulo, pois isso faria com que palavras com alto peso de Hamming teriam baixo peso Euclidiano de saída.

- Devem preferencialmente fazer com que o codificador utilize todos os sinais da constelação.

- um estado ímpar só recebe transições com peso de Hamming ímpar de estados pares, e só recebe transições com peso de Hamming par de estados ímpares;

- um estado par só recebe transições com peso de Hamming ímpar de estados ímpares, e só recebe transições com peso de Hamming par de estados pares.

- Um dado estado não pode receber dois ramos com o mesmo rótulo ou com o mesmo sinal.

◇

As regras sobre a paridade dos estados é a mesma definida no quarto passo. Este é o segundo fator que determina se o código será par. Seguindo estas regras garantimos que $d_3 = \infty$.

9^o **passo:**

Para cada estado e para candidato de A', obter o vetor k-dimensional $d_{ij}(\cdot)$, multiplicando a partição obtida no oitavo passo para o i-ésimo estado pelo j-ésimo elemento de A'. O n-ésimo termo de $d_{ij}(\cdot)$ possui o menor peso Euclidiano de um ramo com peso de Hamming n que causa a transição de σ_i para σ_e, obtido da multiplicação do coset de F_{00} pelo o ramo a'_j. ◇

As transições que causam um retorno ao estado identidade são definidas pelos elementos do conjunto A' e dos cosets de F_{00} para cada estado. É importante para garantir a distância mínima do código que elas tenham um peso Euclidiano alto o suficiente . A informação mais importante dos vetores obtidos é o seu menor valor e o menor valor para as palavras com peso de Hamming igual a 1, pois estas podem gerar seqüências com peso de Hamming 2 com um baixo peso Euclidiano.

10^0 **passo:**

Organizar os vetores $d_{ij}(\cdot)$, em relação a j, para cada estado i. Os melhores são aqueles que resultam no maior valor para a primeira dimensão do vetor. Entre estes os melhores são aqueles que resultam no maior valor para a segunda dimensão, e assim sucessivamente, até a ultima dimensão. Se, para alguma dimensão a distância Euclidiana for zero, o vetor geraria codificadores ruins, pois causaria uma transição que retorna a σ_e com peso Euclidiano nulo, e deve ser colocado no fim da classificação. ◇

Teremos no final $2^m - 1$ listas, uma para cada estado exceto o estado identidade. Em cada lista, o primeiro termo é o candidato que resulta num bom valor para as seqüências com peso de entrada igual a 1. Esta classificação é direcionada a maximizar a distância livre efetiva. Uma outra condição que se pode utilizar é determinar um peso mínimo para o conjunto de transições paralelas, e considerar como ruins qualquer vetor d_{ij} que tenha algum valor menor do que este valor mínimo. Este valor mínimo pode ser determinado através do peso Euclidiano mínimo dos ramos que causam uma transição de σ_e para σ_e ou de um valor estimado para a distância mínima, baseado na distância mínima de codificadores com as mesmas características mas não BGU. Este valor predeterminado deve ser compatível com o que o código permite, e talvez requeira revisão se for muito alto. Este critério não leva em conta a possibilidade de um estado receber uma seqüência que partiu de σ_e e chegou a σ_i através de outro estado, mas tem peso Euclidiano menor do que os ramos que saem de σ_e diretamente para σ_i.

11^o **passo:**

Através da classificação obtida, tentar formar um grupo com o melhor candidato possível para cada estado. A escolha dos candidatos deve ser de tal forma que eles formem um grupo isomorfo ao grupo de estados Σ_m.

Não é útil fazer com que um ramo com peso de entrada 1 tenha peso Euclidiano muito maior do que $d^i_{f,ef}$, pois isso não resultaria em nenhum grande ganho. Os menores pesos Euclidianos devem ficar com sinais com os maiores pesos de Hamming. Devemos também tentar tornar a distância mínima do código alta, pois para altos valores da RSR, é esta distância que determina o desempenho do código.

Os $2^m - 1$ elementos obtidos (um para cada lista), juntamente com o elemento (σ_e, e, σ_e), formam o grupo A.

$\diamond$

As funções E_j são assim definidas tal que $E_j(f \cdot a_j) = E_o(f)$. Assim, a segunda condição é satisfeita. O conjunto A obtido forma um grupo. Como $A \subset T$, A é um subgrupo de T. Além disso, T pode ser obtido pela união dos termos resultantes do produto de F_0 por cada elemento de A. A terceira condição do teorema 2.5.3 esta satisfeita, e o codificador encontrado é portanto BGU.

Seguindo estes passos, obtemos F_0, E_0, A, e uma relação entre os elementos de A e Σ_m. Isto define o codificador, que é BGU por satisfazer os ítens do teorema 2.5.3.

4.2 Considerações sobre o algoritmo

Alguns pontos do algoritmo limitam as soluções. Em alguns casos, precisaremos e conseguimos modificar ligeiramente alguns pontos para obtermos soluções que sabemos que podem ser melhores que as obtidas através deste algoritmo. Em outros casos, o custo computacional é muito grande, e temos que nos restringir ao algoritmo.

Preferencialmente o espaço de estados é isomorfo a Z_2^m. Isto simplifica a procura dos candidatos feita no oitavo passo, bastando testar a relação entre os sinais associados ao ramo testado. Para casos onde o número de geradores de L for menor do que m, não conseguiremos encontrar um isomorfismo entre o grupo G e o grupo Z_2^m. Isso não impede que encontremos um codificador BGU. Seria conveniente nestas situações encontrar o grupo mais simples com o mesmo número de geradores de G. Por exemplo, para o caso em que $m = k$, um grupo razoável para o espaço de estados seria o próprio grupo de ligação entre o espaço de Hamming H_k e a sub-constelação G. Para espaços de estado cujo número de elementos é maior do que a cardinalidade do grupo de ligação, uma boa alternativa seria escolher um grupo que tivesse o grupo de ligação como subgrupo. e A.

A procura do grupo A é extremamente simplificada quando o grupo Σ_m é isomorfo a Z_2^m. Nestas situações, podemos muitas vezes escrever equações de paridade que definem o estado final de uma transição através de combinações lineares binárias do estado inicial e da palavra de entrada. Isto facilita garantir que $d_3 = \infty$, além de facilitar bastante a procura de candidatos e a formação final do grupo A. A partição de F_{00} que causa uma transição que retorna a σ_e também é automaticamente determinada através destas equações de paridade. Entretanto, isso nem sempre é possível, se quisermos codificadores com um número de memórias maior do o número de geradores de L.

Durante a procura de codificadores, escolhemos um número de equações de paridade igual ao número de memórias, associando o resultado de cada equação ao valor final da memória. Além disso, escolhemos geradores g do espaço de Hamming que satisfizessem a seguinte condição:

$$Q(h \cdot g) = Q(h) \cdot p_{H_m} \tag{4.11}$$

A função $Q(h)$ são as equações de paridade, cujo resultado é uma palavra binária de tamanho m. A operação p_{H_m} é uma operação de permutação e soma similar às operações descritas na seção 2.3.1. A equação 4.11 significa que, se uma partição do espaço de Hamming causa uma transição para um dado estado final $Q(h)$, então o produto de qualquer elemento desta partição por um gerador g faz com que o estado final seja o mesmo para todos, sendo também uma partição do espaço de Hamming. Isto facilita a apresentação dos códigos.

Se for possível escolher equações de paridade que definem o estado final a partir da palavra inicial, considerando que o estado inicial é a identidade, podemos descrever as transições entre estados

através de relações entre as memórias, como é usualmente feito em códigos convolucionais. Para garantir que $d_3 = \infty$, podemos simplesmente reservar uma memória para guardar informação sobre a paridade da seqüência transmitida, não permitindo que outras memórias influenciem no seu valor, mas permitindo que o seu valor influencie o valor de outras memórias.

Se testássemos todos os homomorfismos entre o espaço de Hamming H_k e a sub-constelação G, teríamos uma tarefa que é computacionalmente inviável. Há vários automorfismos possíveis, cada um deles com um conjunto de geradores obtidos através de permutações e somas. Como tentamos obter estes automorfismos através do mapeamento dos geradores, precisamos saber a ordem de cada possível gerador do espaço de Hamming. O número de possíveis geradores (que ainda tem que ser combinados entre si para formar o espaço de Hamming) é igual a $(2^k - 1)k!$, pois há $2^k - 1$ possíveis elementos a serem somados (excluímos a palavra toda nula) e há $k!$ permutações possíveis. Esta tarefa é simplificada quando limitamos os geradores àqueles com baixo peso de Hamming. Como visto anteriormente, é mais importante a paridade do que o peso de Hamming dos geradores. Esta escolha é, portanto, computacional.

4.3 Exemplo

Nesta seção desenvolveremos um exemplo de aplicação do algoritmo. Vamos construir um codificador com duas memórias, taxa 5/6, que será mapeado numa constelação $2 \times 8PSK$.

1^o **passo:**

Para representar a constelação, vamos utilizar o grupo D_4^2, que é o produto cartesiano $D_4 \times D_4$.

2^o **passo:**

O grupo de ligação L será o grupo $Dih(Z_4^2)$, que é um subgrupo de D_4^2 e que gera H_5. Este grupo possui três geradores que se relacionam da seguinte forma:

$$\begin{gathered} r_1^4 = r_2^4 = s^2 = (r_1 s)^2 = (r_2 s)^2 = (r_1 r_2 s)^2 = e \\ r_1 r_2 = r_2 r_1 \end{gathered} \tag{4.12}$$

3^o **passo:**

O grupo do espaço de estados será Z_2^2, que é um subgrupo de $Dih(Z_4^2)$. Cada conjunto de transições paralelas terá 8 elementos, pois há 4 estados e 32 entradas. O grupo obtido através de $Dih(Z_4^2)/Z_2^2$ é o grupo que representa um conjunto de transições paralelas. No caso, este grupo é

$Z_4 \times Z_2$. Este subgrupo pode ser gerado a partir de $\langle r_1, r_2^2 \rangle$. Assim, o subgrupo que causa transições paralelas contém os seguintes elementos:

$$\begin{array}{cccc} (e,e) & (r_1,e) & (r_1^2,e) & (r_1^3,e) \\ (e,r_2^2) & (r_1,r_2^2) & (r_1^2,r_2^2) & (r_1^3,r_2^2) \end{array} \tag{4.13}$$

4^o **passo:**

Procuraremos agora automorfismos de H_5 para termos um bom controle sobre a posição das palavras com peso de Hamming igual a 1 ou 2. A tabela 4.1 mostra cinco (mas não todas) possibilidades de se gerar H_5 através das relações entre os geradores definidas anteriormente. Os primeiros 5 termos indicam uma permutação, os últimos 5 termos indicam o que deve ser somado, módulo 2, ao elemento inicial. Parte-se da identidade (00000) para gerar H_5. Estes conjuntos foram escolhidos porque favorecem a visualização da influência do peso dos geradores na posição das seqüências críticas para a treliça.

r_1	r_2	s
$(54321)_p(11000)_\oplus$	$(14325)_p(01100)_\oplus$	$(12345)_p(00111)_\oplus$
$(14325)_p(01100)_\oplus$	$(54321)_p(11000)_\oplus$	$(12345)_p(00111)_\oplus$
$(21345)_p(10000)_\oplus$	$(12354)_p(00001)_\oplus$	$(12345)_p(01110)_\oplus$
$(14325)_p(01100)_\oplus$	$(54321)_p(10000)_\oplus$	$(12345)_p(00011)_\oplus$
$(54321)_p(10000)_\oplus$	$(14325)_p(01100)_\oplus$	$(12345)_p(00011)_\oplus$

Tab. 4.1: Geradores de H_5

Os elementos que causam uma transição paralela estão indicados na tabela 4.2:

e	r_1	r_1^2	r_1^3	r_2^2	$r_1 \cdot r_2^2$	$r_1^2 \cdot r_2^2$	$r_1^3 \cdot r_2^2$
00000	11000	11011	00011	01010	10010	10001	01001
00000	01100	01010	00110	11011	10111	10001	11101
00000	10000	11000	01000	00011	10011	11011	01011
00000	01100	01010	00110	10001	11101	11011	10111
00000	10000	10001	00001	01010	11010	11011	01011

Tab. 4.2: Transições paralelas

onde cada linha apresenta os elementos que causam uma transição paralela gerados pelo mapeamento dos geradores como indicada na linha correspondente da tabela 4.2.

Como podemos ver, não podemos escolher as linhas 3 e 5 porque elas causam uma transição paralela com peso ímpar. Dentre as linhas 1, 2 e 4, as melhores são as linhas 2 e 4, pois possuem

apenas 4 transições com peso 2. Além disso, conseguimos gerar com apenas um dos geradores (no caso, r_1), 3 das 4 palavras com peso 2. Escolhemos a linha 2 pois desta forma geramos todas as palavras com peso par com dois geradores (r_1 e r_2). Definimos também os estados finais de cada transição, sabendo qual é o grupo Σ_m.

Estado Inicial									Estado Final
00	00000	01100	01010	00110	11011	10111	10001	11101	00
	11000	11110	10010	10100	00011	00101	01001	01111	01
	00111	01011	01101	00001	11100	10000	10110	11010	10
	11111	11001	10101	10011	00100	00010	01110	01000	11

Tab. 4.3: Partições do espaço de Hamming

Como queremos que o código seja par, devemos garantir que as transições entre os estados obedeçam ao diagrama 4.2, onde P indica transições pares e I indica transições ímpares.

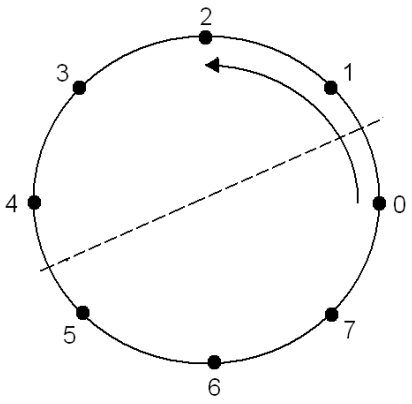

Fig. 4.1: Grupo gerador da constelação 8 PSK

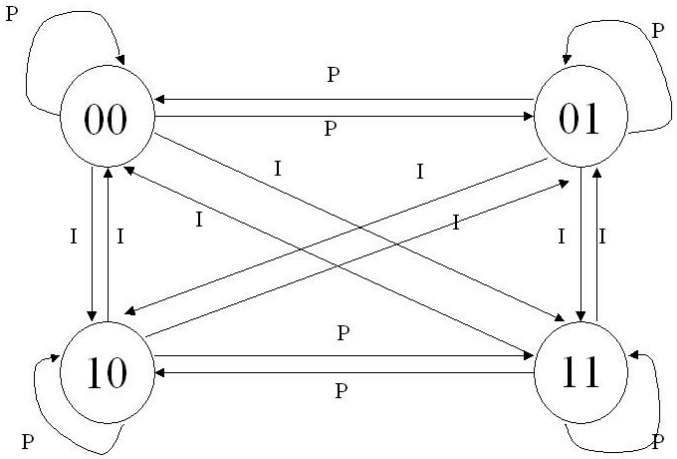

Fig. 4.2: Diagrama de transições

5^o **passo:**

Procuraremos agora por automorfismos do grupo G. A tabela 4.4 mostra três possibilidades de se gerar subgrupos de 32 elementos de $2 \times 8PSK$ utilizando-se a representação da figura 4.1 e geradores que respeitam a regra 4.12, sendo assim isomorfos ao grupo $D(Z_4 \times Z_4)$. A constelação $2 \times 8PSK$ é o produto cartesiano de duas constelações $8PSK$, sendo assim necessários dois elementos para representar um ponto. A linha tracejada indica o eixo de reflexão do grupo diedral, enquanto que

a seta indica o sentido da rotação. Consideraremos que esta constelação tem raio unitário, o que corresponde a dizer que cada sinal tem energia igual 2.

r_1	r_2	s
(2 0)	(0 2)	(1 1)
(2 4)	(0 2)	(3 5)
(2 4)	(2 2)	(1 5)

Tab. 4.4: Geradores de 32 elementos em $2 \times 8PSK$

Gerador de H_5	Gerador de G
$(54321)_p(11000)_{\oplus}$	(2 2)
$(14325)_p(01100)_{\oplus}$	(2 4)
$(12345)_p(00111)_{\oplus}$	(1 5)

Tab. 4.5: Associação de geradores

A última linha da tabela 4.4 fornece bons valores para palavras com peso 2 que causam transições paralelas e também fornece bons pesos Euclidianos para as palavras com peso 1 e 2 que saem do estado identidade, sendo então a nossa escolha.

6^o **passo:**

A associação entre os elementos de H_5 e da constelação $2 \times 8PSK$ é obtida pelo mapeamento dos geradores dada pela tabela 4.5

Desta forma, obtemos F_0 e E_0. Precisamos agora obter o grupo A.

7^o **passo:**

Como A é isomorfo a Z_2^2, podemos testar somente os sinais associados à F_0. Desta forma, os candidatos que satisfazem 4.9 e podem formar o grupo A são:

$$\begin{aligned} A' \;=\; \{&(0,0),(0,3),(0,4),(0,7),\\ &(3,0),(3,3),(3,4),(3,7),\\ &(4,0),(4,3),(4,4),(4,7),\\ &(7,0),(7,3),(7,4),(7,7)\} \end{aligned} \tag{4.14}$$

8^o **passo:**

Agora definiremos as transições entre os estados. Devemos obedecer às restrições impostas sobre a paridade dos estados. Algumas transições são óbvias. Representando os ramos através dos quatro elementos $(\sigma_{m,i}, h, g, \sigma_{m,f})$, ou seja, o espaço inicial, o rótulo de Hamming, o sinal da constelação e o estado final, podemos assumir o seguinte:

$$\begin{gathered}(00, 00000, (0, 0), 00) \\ (01, 00000, ?, 01) \\ (10, 00000, ?, 11) \\ (11, 00000, ?, 10)\end{gathered} \tag{4.15}$$

onde a interrogação indica que ainda não sabemos qual será o símbolo a ser utilizado.

Assim, as palavras que causam um retorno ao estado inicial a partir de cada estado são:

Estado Inicial									Estado Final
00	00000	01100	01010	00110	11011	10111	10001	11101	00
01	11000	11110	10010	10100	00011	00101	01001	01111	
11	11111	11001	10101	10011	00100	00010	01110	01000	
10	00111	01011	01101	00001	11100	10000	10110	11010	

Tab. 4.6: Partições do espaço de Hamming que retornam ao estado inicial

Através de E_0, estas palavras estão associadas a símbolos. Queremos que as palavras que retornam ao estado inicial com peso crítico, nominalmente aquelas que tem peso 1, tenham o maior peso Euclidiano possível. Para o estado (11), por exemplo, os sinais que fazem isso são obtidos através da multiplicação dos sinais associados à estas palavras no estado inicial multiplicado por um elemento a_{11} ainda desconhecido, como mostra 4.16

$$E_0^{-1} \begin{bmatrix} 00100 \\ 00010 \\ 01000 \end{bmatrix} \cdot a_{11} = \begin{bmatrix} (3,7) \\ (1,3) \\ (5,3) \end{bmatrix} \cdot a_{11} \tag{4.16}$$

Assim, dentre os candidatos, devemos escolher aquele que fornece o maior peso Euclidiano para estas seqüências.

9^o **passo:**

A tabela 4.7 apresenta os vetores de distância gerados a partir de cada candidato para cada estado. O número abaixo do estado indica qual é o peso de Hamming da seqüência de entrada. Por exemplo,a quinta coluna do candidato $(0, 7)$ indica que, para o estado 01, o menor peso Euclidiano de uma seqüência com peso de Hamming 4 que causa uma transição do estado 01 para o estado 00 tem peso 4,58. Utilizamos $\cdot$ para denotar quando não há transições com o peso especificado. Organizando os candidatos para cada estado, obtemos a tabela 4.8. As primeiras linhas indicam os melhores candidatos.

Estado inicial	01					10					11				
Candidatos	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
(0, 3)	·	0,58	·	2,58	·	5,42	·	2,58	·	·	0,58	·	3,42	·	0,58
(0, 4)	·	2	·	2	·	4	·	1,16	·	·	1,16	·	4	·	1,16
(0, 7)	·	0,58	·	4,58	·	5,42	·	2,58	·	·	3,42	·	0,58	·	4,58
(3 , 0)	·	2,58	·	2,58	·	0,58	·	2,58	·	·	0,58	·	2,58	·	7,42
(3 , 3)	·	1,16	·	1,16	·	2	·	4	·	·	2	·	0	·	4
(3 , 4)	·	2,58	·	0,58	·	0,58	·	2,58	·	·	2,58	·	0,58	·	4,58
(3 , 7)	·	1,16	·	1,16	·	2	·	4	·	·	0	·	2	·	8
(4 , 0)	·	4	·	2	·	1,16	·	1,16	·	·	1,16	·	1,16	·	6,84
(4 , 3)	·	2,58	·	2,58	·	2,58	·	2,58	·	·	0,58	·	0,58	·	3,42
(4 , 4)	·	4	·	2	·	1,16	·	1,16	·	·	1,16	·	1,16	·	4
(4 , 7)	·	2,58	·	0,58	·	2,58	·	2,58	·	·	0,58	·	0,58	·	7,42
(7 , 0)	·	2,58	·	5,42	·	4,58	·	0,58	·	·	4,58	·	0,58	·	3,42
(7 , 3)	·	1,16	·	4	·	6	·	2	·	·	2	·	4	·	0
(7 , 4)	·	2,58	·	3,42	·	4,58	·	0,58	·	·	2,58	·	3,42	·	0,58
(7 , 7)	·	1,16	·	4	·	6	·	2	·	·	4	·	0	·	4

Tab. 4.7: Vetores de distâncias obtidos para os candidatos ao grupo A

10^o **e** 11^o **passos:**

Não podemos escolher nem $(4, 4)$ e nem $(4, 0)$ para o estado 01 porque isso causaria um autolaço de peso Euclidiano nulo neste estado. Também não podemos escolher para o estado 11 os sinais $(7, 7)$, $(7, 3)$ ou $(3, 3)$ porque há uma transição com peso Euclidiano nulo, o que seria péssimo para a distância mínima. Como as seqüências que saem do estado 00 com peso de Hamming 1 já tem no mínimo peso Euclidiano igual a 4(o valor de d_{ef} até o instante), podemos relaxar o critério de dar o maior peso possível para seqüências com peso de Hamming 1 que retornam ao estado identidade, pois isto não vai afetar o valor de d_{ef}. Assim, fazemos a escolha mostrada na tabela 4.9. Como resultado, obtemos um codificador cuja distância livre efetiva é igual a 4. O menor peso que gera uma seqüência com peso igual à distância mínima do código é 6. o valor de γ será então igual a $\lceil 6/d_{e,l} \rceil$. Teríamos um bom sistema se $d_{e,l} = 3$, como podemos ver analisando a equação 3.26. O código é BGU, como pode ser testado.

Assim, o codificador é completamente definido pelas tabelas 4.3, 4.5 e 4.9.

01	10	11
(4,4)	(7,7)	(7,0)
(4,0)	(7,3)	(7,7)
(7,0)	(0,7)	(0,7)
(7,4)	(0,3)	(7,4)
(4,3)	(7,4)	(3,4)
(3,0)	(7,0)	(7,3)
(4,7)	(0,4)	(3,3)
(3,4)	(4,7)	(0,4)
(0,4)	(4,3)	(4,0)
(7,7)	(3,7)	(4,4)
(7,3)	(3,3)	(0,3)
(3,7)	(4,4)	(3,0)
(3,3)	(4,0)	(4,7)
(0,7)	(3,4)	(4,3)
(0,3)	(3,0)	(3,7)

Tab. 4.8: Classificação

Estado	Elemento correspondente
00	(00,(0,0),00)
01	(01,(7,0),01)
10	(10,(7,7),11)
11	(11,(0,7),10)

Tab. 4.9: Grupo A final.

4.4 Síntese

Neste capítulo apresentamos um algoritmo para encontrar codificadores bit-geometricamente uniformes de uma forma que podemos atribuir um bom peso Euclidiano para seqüências com peso de Hamming de entrada igual a 2. Analisamos os pontos computacionalmente complicados, e também indicamos casos particulares para os quais é mais fácil obter a resposta desejada. Desenvolvemos também um exemplo para mostrar como se deve proceder com os passos mostrados.

Capítulo 5

Análise de Resultados

Neste capítulo mostramos alguns dos codificadores gerador a partir do método descrito no capítulo 4. Os limitantes analíticos de desempenho para sistemas de concatenação serial baseados nestes codificadores serão mostrados e discutidos.

5.1 Apresentação de codificadores

5.1.1 Formato de Apresentação

Um codificador E associa à ramos de uma treliça T um rótulo, que no nosso caso pertence a um espaço de Hamming com k bits H_k. Cada elemento da treliça T contém três informações: estado inicial, sinal s pertencente à constelação utilizada S, e estado final. Para definirmos completamente o codificador E precisamos também associar uma quarta informação, o rótulo de cada ramo de T.

Para um codificador BGU, é suficiente definir as seguintes estruturas para definirmos completamente o codificador E:

- F_0, os subgrupo de T com os ramos que saem do estado identidade
- E_0, o mapeamento entre F_0 e o H_k
- A, um subgrupo de T que contém todos os ramos que são rotulados pela palavra toda nula.

No capítulo 4, definimos E_0 através de um isomorfismo entre o grupo gerador do espaço de Hamming e o grupo gerador da sub-constelação S_0 associada a F_0. Garantimos que F_0 é um grupo através de outros homomorfismos do grupo gerador de H_k. Assim, definimos completamente o codificador E declarando o isomorfismo entre os grupos geradores de S_0 e H_k e declarando o homomorfismo entre o grupo gerador de H_k e A.

O isomorfismo entre os grupos geradores de S_0 e H_k pode ser definido através do mapeamento dos geradores, dado que as operações sobre cada espaço sejam definidas. Escolhemos isomorfismos entre H_k e o grupo A de modo a ser possível escrever esta relação através de equações de paridade, como descrito na seção 4.2. Por causa disso, definimos as transições entre estados através de equações binárias. Portanto, podemos definir o estado final de qualquer ramo em função do estado inicial e do rótulo de entrada. A associação entre os estados e os elementos do grupo A nos permite obter os sinais associados a todos os ramos. Assim, podemos obter todos os ramos de T e os rótulos associados.

Como exemplo, o codificador obtido na seção 4.3 pode ser completamente definido pelas tabelas 4.3, 4.5 e 4.9. Substituiremos a tabela 4.3 por equações binárias, a segunda do conjunto 5.1.

Apresentaremos então os codificadores através das seguintes tabelas:

Gerador de H_k	Gerador de G
h_1	g_1
h_2	g_2
$\cdots$	$\cdots$
h_n	g_n

Memória	Equação
$e_{1,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus \cdots \oplus e_{1,k} \oplus e_{2,k} \oplus \cdots$
$e_{2,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus \cdots \oplus e_{1,k} \oplus e_{2,k} \oplus \cdots$
$\cdots$	$\cdots$
$e_{m,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus \cdots \oplus e_{1,k} \oplus e_{2,k} \oplus \cdots$

Estado inicial	Elemento correspondente	Estado final
Estado inicial 1	Sinal 1	Estado final 1
Estado inicial 2	Sinal 2	Estado final 1
Estado inicial 3	Sinal 3	Estado final 1
$\cdots$	$\cdots$	$\cdots$
Estado inicial 2^m	Sinal 2^m	Estado final 1

Tab. 5.1: Formato de apresentação de codificadores

Na tabela acima, o espaço de Hamming com k bits H_K possui n geradores. Cada um deles esta associado a um dos n geradores da constelação como a primeira tabela indicar. Será explicitada qual é a constelação utilizada. O estado final é determinado pelo valor de cada uma das m memórias, resultando num total de 2^m possibilidades. Cada memória e_i tem o seu valor determinado pelas equações da segunda tabela. Todas as somas são módulo 2. A terceira tabela determina qual é o símbolo associado a cada estado para formar o grupo A. Estas três tabelas são suficientes para definir completamente os codificadores encontrados.

5.1.2 Codificadores encontrados

Procuramos por alguns códigos BGU para a constelação $L \times 8 - PSK$. Para efeitos comparativos, geramos a partir do algoritmo um codificador semelhante àquele encontrado em [7].

Limitamos a nossa pesquisa com quatro critérios. O primeiro limitava a quantidade de memórias em no máximo 3. A justificativa é que, com o aumento do número de memórias, a quantidade de elementos que compõem o grupo A aumenta exponencialmente, complicando assim a sua procura. Limitamos também o número de ramos em 2^{10}. Isto é, o número de entradas mais o número de memórias deveria ser inferior a 10. A terceira limitação era sobre a dimensão da constelação, limitada a 6 dimensões. Como utilizamos a constelação $8-PSK$ que possui duas dimensões, isto quer dizer que a maior constelação utilizada foi a constelação $3 \times 8 - PSK$. O aumento em dimensões aumenta o número de geradores de da constelação complicando assim a procura dos isomorfismos entre o espaço de Hamming utilizado H_k e o subgrupo da constelação G. Como buscamos codificadores que tenham uma boa eficiência espectral (alta taxa de bits), utilizamos a menor constelação possível que permitisse o envio de k bits codificados, onde k é o tamanho do espaço de Hamming de entrada.

Os codificadores encontrados tem as suas características resumidas na tabela 5.2.

Codificador	Bits	Memórias	Ramos	Constelação utilizada	Taxa	d_{ef}	d_{min}	$h_{d_{min}}$
C_{51}	5	1	64	$2 \times 8 - PSK$	5/6	3.757	1.757	4
C_{52}	5	2	128	$2 \times 8 - PSK$	5/6	4.000	1.757	6
C_{53}	5	3	256	$2 \times 8 - PSK$	5/6	4.000	1.757	8
C_{63}	6	3	512	$3 \times 8 - PSK$	6/9	8.000	3.757	6
C_{72}	7	2	512	$3 \times 8 - PSK$	7/9	4.000	2.000	4
C_{73}	7	3	1024	$3 \times 8 - PSK$	7/9	4.000	2.929	10
C_{81}	8	1	512	$3 \times 8 - PSK$	8/9	4.000	1.171	8
C_{82}	8	2	1024	$3 \times 8 - PSK$	8/9	4.000	1.757	6

Tab. 5.2: Codificadores encontrados e suas características.

A tabela acima informa, para cada codificador, o tamanho da sua entrada em bits, a quantidade de memórias, a quantidade de ramos que cada seção da treliça possui, a constelação utilizada, a taxa do codificador, a distância livre efetiva d_{ef}, a distância mínima d_{min} e o menor peso de Hamming de entrada que gera uma saída com peso Euclidiano mínimo $h_{d_{min}}$.

Para cada codificador, além das tabelas, informaremos também a distância livre efetiva e a distância mínima. Para isto consideramos que cada constelação $8-PSK$ tem energia unitária. As distâncias correspondem a distâncias quadráticas, como indicadas no capítulo 1. Os geradores da constelação e as operações entre eles, assim como os geradores do espaço de Hamming, serão representados utilizando a mesma notação do capítulo 4. O gerador da constelação terá um elemento para cada constelação $8-PSK$ utilizada. Assim, para a constelação $3 \times 8 - PSK$, um gerador terá 3 elementos. Informamos também qual foi o grupo de ligação L utilizado.

Codificador de H_5 em $2 \times 8 - PSK$, uma memória

Este codificador é semelhante ao encontrado em [7] no que se refere à distância livre efetiva e à distância Euclidiana mínima entre seqüências com um dado peso de Hamming. A constelação utilizada é $2 \times 8 - PSK$. A distância livre efetiva é de $3,757$, e a distância mínima é de $1,757$. O menor peso de Hamming que gera um caminho com distância mínima é 4. O grupo de ligação é o grupo $Dih(Z_4 \times Z_4)$, o mesmo descrito para o exemplo do capítulo 4.

O codificador esta descrito na tabela 5.3.

Gerador de H_5	Gerador de G
$(54321)_p(11000)_{\oplus}$	(66)
$(14325)_p(01100)_{\oplus}$	(24)
$(12345)_p(00111)_{\oplus}$	(11)

Memória	Equação
$e_{1,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus\ b_{3,k} \oplus b_{4,k} \oplus b_{5,k} \oplus e_{1,k}$

Estado inicial	Elemento correspondente	Estado final
0	(0 0)	0
1	(7 4)	1

Tab. 5.3: Codificador com uma memória C_{51}, taxa 5/6

Codificador de H_5 em $2 \times 8 - PSK$, duas memórias

Este codificador é o mesmo do exemplo do capítulo 4. A constelação utilizada é $2 \times 8 - PSK$. A distância livre efetiva é de $4,000$ e a distância mínima ainda é $1,757$. O menor peso de Hamming que gera um caminho com distância mínima é 6. O grupo de ligação é o grupo $Dih(Z_4 \times Z_4)$. Como há mais ramos do que sinais, os sinais são reaproveitados, sendo associados a mais de um ramo da treliça.

O codificador esta descrito na tabela 5.4.

Codificador de H_5 em $2 \times 8 - PSK$, três memórias

A distância livre efetiva é de $4,000$ e a distância mínima é $1,757$. O menor peso de Hamming que gera um caminho com distância livre é 8. Esta aumento causará ganho para altos valores da relação sinal-ruído. O grupo de ligação é o grupo $Dih(Z_4 \times Z_4)$.

O codificador esta descrito na tabela 5.5.

Gerador de H_5	Gerador de G
$(54321)_p(11000)_{\oplus}$	(22)
$(14325)_p(01100)_{\oplus}$	(24)
$(12345)_p(00111)_{\oplus}$	(15)

Memória	Equação
$e_{1,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus \ b_{3,k} \oplus b_{4,k} \oplus b_{5,k} \oplus e_{1,k}$
$e_{2,k+1}$	$b_{2,k} \oplus \ b_{3,k} \oplus b_{4,k} \oplus e_{1,k} \oplus e_{2,k}$

Estado inicial	Elemento correspondente	Estado final
00	(0 0)	00
10	(7 0)	11
01	(7 4)	01
11	(0 4)	10

Tab. 5.4: Codificador com duas memória C_{52}, taxa 5/6

Gerador de H_5	Gerador de G
$(54321)_p(11000)_{\oplus}$	(22)
$(14325)_p(01100)_{\oplus}$	(02)
$(12345)_p(00111)_{\oplus}$	(11)

Memória	Equação
$e_{1,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus \ b_{3,k} \oplus b_{4,k} \oplus b_{5,k} \oplus e_{1,k}$
$e_{2,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus b_{5,k} \oplus e_{1,k} \oplus e_{3,k}$
$e_{3,k+1}$	$b_{1,k} \oplus b_{4,k} \oplus b_{5,k} \oplus e_{2,k}$

Estado inicial	Elemento correspondente	Estado final
000	(0 0)	000
100	(3 0)	110
010	(3 7)	001
110	(0 7)	111
001	(4 0)	010
101	(7 0)	100
011	(7 7)	011
111	(4 0)	101

Tab. 5.5: Codificador com três memória C_{53}, taxa 5/6

Codificador de H_6 em $3 \times 8 - PSK$, três memórias

Para utilizarmos todos os 2^9 sinais desta constelação, precisamos de no mínimo 2^9 ramos. Como a entrada tem 6 bits, precisamos então de no mínimo 3 memórias. Embora haja codificadores BGU que utilizam todos os sinais, o melhor codificador encontrado para sistemas com concatenação serial não o faz. A distância livre efetiva é de $8,000$ e a distância mínima é $3,757$. O menor peso de Hamming que gera um caminho com distância livre é 6. O grupo de ligação é o grupo $Z_4 \times Z_4 \times Z_4$.

O codificador esta descrito na tabela 5.6.

Gerador de H_6	Gerador de G
$(213456)_p(100000)_{\oplus}$	(224)
$(124356)_p(001000)_{\oplus}$	(242)
$(123465)_p(000010)_{\oplus}$	(222)

Memória	Equação
$e_{1,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus b_{3,k} \oplus b_{4,k}$ $\oplus b_{5,k} \oplus b_{6,k} \oplus e_{1,k}$
$e_{2,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus e_{1,k} \oplus e_{3,k}$
$e_{3,k+1}$	$b_{3,k} \oplus b_{4,k} \oplus e_{2,k}$

Estado inicial	Elemento correspondente	Estado final
000	(0 0 0)	000
100	(0 4 0)	110
010	(1 1 3)	001
110	(1 5 3)	111
001	(0 4 4)	010
101	(0 0 4)	100
011	(1 5 7)	011
111	(1 1 7)	101

Tab. 5.6: Codificador com três memória C_{63}, taxa 6/9

Codificador de H_7 em $3 \times 8 - PSK$, duas memórias

Pelo mesmo motivo do codificador anterior, um código com 7 bits de entrada mapeado na constelação $3 \times 8 - PSK$ precisa de no mínimo 2 memórias para que todos os 2^9 sinais sejam utilizados. Espera-se um decrescimento nos valores da distância mínima e da distância livre efetiva, pois temos mais informação para a mesma quantidade de espaço do caso anterior. De fato, a distância livre efetiva é de $4,000$ e a distância mínima é $2,000$. O menor peso de Hamming que gera um caminho com distância livre é 4. O grupo de ligação é o grupo $Dih(Z_4 \times Z_4 \times Z_4)$. Este grupo tem quatro geradores que se relacionam como indicado por 5.1. O codificador esta descrito na tabela 5.7.

$$\begin{aligned} g_1^4 = g_2^4 = g_3^4 = g_4^2 = e \\ g_i \cdot g_j = g_j \cdot g_i \qquad i,j = 1,2,3 \\ (g_i \cdot g_4)^2 = e \qquad i = 1,2,3 \end{aligned} \tag{5.1}$$

Codificador de H_7 em $3 \times 8 - PSK$, três memórias

A distância livre efetiva é de $4,000$ e a distância mínima é $2,929$. Embora não haja uma melhora na distância livre efetiva, a multiplicidade (como definida no capítulo 3) diminui. O menor peso de Hamming que gera um caminho com distância livre é 10. Esta aumento em relação ao caso anterior causará ganho para altos valores da relação sinal-ruído. O grupo de ligação é o grupo $Dih(Z_4 \times Z_4 \times Z_4)$. O codificador esta descrito na tabela 5.8.

Gerador de H_7	Gerador de G
$(7654321)_p(1001000)_{\oplus}$	(200)
$(1654327)_p(0101000)_{\oplus}$	(020)
$(1254367)_p(0011000)_{\oplus}$	(042)
$(1234567)_p(0000111)_{\oplus}$	(111)

Memória	Equação
$e_{1,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus b_{3,k} \oplus b_{4,k}$ $\oplus b_{5,k} \oplus b_{6,k} \oplus b_{7,k} \oplus e_{1,k}$
$e_{2,k+1}$	$b_{3,k} \oplus b_{4,k} \oplus e_{5,k} \oplus e_{1,k} \oplus e_{2,k}$

Estado inicial	Elemento correspondente	Estado final
00	(0 0 0)	00
10	(7 1 4)	11
01	(0 5 4)	01
11	(7 4 0)	10

Tab. 5.7: Codificador com duas memória C_{72}, taxa 7/9

Gerador de H_7	Gerador de G
$(7654321)_p(1001000)_{\oplus}$	(200)
$(1654327)_p(0101000)_{\oplus}$	(020)
$(1254367)_p(0011000)_{\oplus}$	(022)
$(1234567)_p(0000111)_{\oplus}$	(111)

Memória	Equação
$e_{1,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus b_{3,k} \oplus b_{4,k}$ $\oplus b_{5,k} \oplus b_{6,k} \oplus b_{7,k} \oplus e_{1,k}$
$e_{2,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus e_{6,k} \oplus b_{7,k} \oplus e_{1,k} \oplus e_{3,k}$
$e_{3,k+1}$	$b_{1,k} \oplus b_{4,k} \oplus b_{7,k} \oplus e_{2,k}$

Estado inicial	Elemento correspondente	Estado final
000	(0 0 0)	000
100	(7 4 0)	110
010	(4 5 3)	001
110	(3 1 3)	111
001	(7 1 3)	010
101	(0 5 3)	100
011	(3 4 0)	011
111	(4 0 0)	101

Tab. 5.8: Codificador com três memória C_{73}, taxa 7/9

Codificador de H_8 em $3 \times 8 - PSK$, uma memória

Uma memória é suficiente pra utilizar todos os sinais da constelação $3 \times 8 - PSK$ quando o tamanho da entrada é 8 bits. A distância livre efetiva é de $4,000$ e a distância mínima é $1,171$. O menor peso de Hamming que gera um caminho com distância livre é 8. O grupo de ligação não pode ser descrito como uma combinação de grupos diedrais e abelianos. Ele possui 5 geradores que obedecem às regras do conjunto de equações 5.2, que define completamente o grupo.

$$\begin{aligned} g_1^4 = g_2^4 = g_3^4 = g_4^2 = g_5^2 = e \\ g_i \cdot g_j = g_j \cdot g_i; \qquad i,j = 1,2,3 \\ (g_i \cdot g_4)^2 = e; \qquad i = 1,2,3 \\ (g_i \cdot g_5)^4 = e; \qquad i = 1,2,4 \\ (g_3 \cdot g_5)^2 = e \end{aligned} \tag{5.2}$$

O codificador esta descrito na tabela 5.9.

Gerador de H_8	Gerador de G
$(15432678)_p(01100000)_\oplus$	(022)
$(12435678)_p(00100010)_\oplus$	(046)
$(62345178)_p(10000010)_\oplus$	(204)
$(12345678)_p(00011101)_\oplus$	(333)
$(12345678)_p(00011001)_\oplus$	(110)

Memória	Equação
$e_{1,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus b_{3,k} \oplus b_{4,k} \oplus b_{5,k}$ $\oplus b_{6,k} \oplus b_{7,k} \oplus b_{8,k} \oplus e_{1,k}$

Estado inicial	Elemento correspondente	Estado final
0	(0 0 0)	0
1	(1 0 0)	1

Tab. 5.9: Codificador com uma memória C_{81}, taxa 8/9

Codificador de H_8 em $3 \times 8 - PSK$, duas memórias

A distância livre efetiva é de $4,000$ e a distância mínima é $1,757$. O menor peso de Hamming que gera um caminho com distância livre é 8. O grupo de ligação não pode ser descrito como uma combinação de grupos diedrais e abelianos, mas é mais simples do que o anterior. Ele possui 5 geradores que obedecem às regras do conjunto de equações 5.3, que define completamente o grupo.

$$\begin{aligned} g_1^4 = g_2^4 = g_3^4 = g_4^2 = g_5^2 = e \\ g_i \cdot g_j = g_j \cdot g_i; \qquad i,j = 1,2,3 \\ (g_i \cdot g_4)^2 = e \qquad i = 1,2 \\ g_4 \cdot g_i = g_i; \cdot g_4 \qquad i = 3,5 \\ g_5 \cdot g_i = g_i; \cdot g_5 \qquad i = 1,2,4 \\ (g_3 \cdot g_5)^2 = e \end{aligned} \tag{5.3}$$

O codificador esta descrito na tabela 5.10.

Gerador de H_8	Gerador de G
$(54321678)_p(11000000)_{\oplus}$	(660)
$(14325678)_p(01100010)_{\oplus}$	(420)
$(12345786)_p(00000110)_{\oplus}$	(042)
$(12345678)_p(00011100)_{\oplus}$	(114)
$(12345678)_p(00000001)_{\oplus}$	(441)

Memória	Equação
$e_{1,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus b_{3,k} \oplus b_{4,k} \oplus b_{5,k}$ $\oplus b_{6,k} \oplus b_{7,k} \oplus b_{8,k} \oplus e_{1,k}$
$e_{2,k+1}$	$b_{1,k} \oplus b_{2,k} \oplus b_{3,k} \oplus b_{4,k}$ $\oplus b_{5,k} \oplus e_{1,k} \oplus e_{2,k}$

Estado inicial	Elemento correspondente	Estado final
00	(0 0 0)	00
10	(0 4 7)	11
01	(0 7 4)	01
11	(0 3 3)	10

Tab. 5.10: Codificador com duas memórias C_{82}, taxa 8/9

5.2 Limitantes de desempenho para os sistemas encontrados

Para termos um código com concatenação serial, precisamos também de um código externo. Para isso, utilizamos os codificadores convolucionais recursivos com alta taxa descritos em [13]. A taxa destes codificadores é $n/n+1$, onde n é o tamanho em bits da entrada. Para cada codificador encontrado, escolhemos codificadores externos cuja saída tivesse o mesmo tamanho da entrada. Isto não é necessário, basta que o tamanho do entrelaçador utilizado seja um múltiplo da saída do código externo e da entrada do código interno. O tamanho do entrelaçador afeta o desempenho do código, por isso ele foi fixado em 840 bits para todos os casos. Este número foi escolhido porque ele é o menor múltiplo comum de 5,6,7 e 8, o tamanho da entrada dos codificadores internos. Para observarmos a importância da distância livre do código externo, variamos a memória deste de 1 a 4, o que resultou numa variação na distância livre de Hamming de 2 a 4. Logo, temos 4 códigos com concatenação para cada codificador interno encontrado. Os codificadores externos estão listados na tabela 5.11. A quantidade de memórias de cada codificador esta indicado na coluna m_e, a distância mínima na coluna $d_{e,l}$, e a taxa na coluna r_e.

Os codificadores internos encontrados são todos pares. Por isso, quando a distância livre do código externo for igual a 3, o menor peso de Hamming de uma seqüência de entrada do codificador interno que causa um retorno ao estado identidade tem peso 4. Embora não aumentemos a distância mínima quando aumentamos a quantidade de memórias de 1 para 2, diminuímos significamente a quantidade de palavras geram seqüências com distância mínima.

Código	m_e	$d_{e,l}$	r_e
C^e_{51}	1	2	4/5
C^e_{51}	1	2	4/5
C^e_{52}	2	2	4/5
C^e_{53}	3	3	4/5
C^e_{54}	4	4	4/5
C^e_{61}	1	2	5/6
C^e_{62}	2	2	5/6
C^e_{63}	3	3	5/6
C^e_{64}	4	4	5/6
C^e_{71}	1	2	6/7
C^e_{72}	2	2	6/7
C^e_{73}	3	3	6/7
C^e_{74}	4	4	6/7
C^e_{81}	1	2	7/8
C^e_{82}	2	2	7/8
C^e_{83}	3	3	7/8
C^e_{84}	4	4	7/8

Tab. 5.11: Códigos externos utilizados.

5.2.1 Cálculo do limitante de união

Para calcular o limitante de união, precisamos das funções de distribuição de pesos IOWEF dos codificadores internos e externos. Estamos utilizando a treliça terminada (*zero tail*), isto é, estamos interessados somente nos caminhos que saem do estado identidade e terminam no mesmo. Esta tarefa pode ser computacionalmente muito complicada, pois a IOWEF é um polinômio com muitos termos. Analisando a equação 3.20, percebemos que podemos ignorar seqüências com altos valores de d, o peso Euclidiano, pois a exponencial diminui muito rapidamente com um pequeno aumento da relação sinal-ruído. Podemos determinar um peso de corte, e calcular somente as seqüências com peso Euclidiano de saída menores do que este valor. Desta forma, também impomos um peso de Hamming de corte para as seqüências intermediárias l, que é o maior peso de Hamming de uma seqüência intermediária que gera uma saída com o peso Euclidiano de corte. Este limite, por sua vez, restringe da mesma forma o peso das seqüências de entrada do código, pelo mesmo argumento do caso anterior. Como estamos truncando o limitante de união somando menos termos, é de se esperar que o valor do limitante de erro seja menor do que o esperado sem truncamento. O valor encontrado passa então a ser uma aproximação do limitante de erro.

Este truncamento não modifica o limitante de união para a região de interesse, como pode ser visto na figura 5.1. Ela mostra o limitante para a probabilidade de erro para o caso do sistema composto

pelo codificador externo C_{51}^{e} com o codificador interno C_{51}, com um entrelaçador de tamanho 100 bits. Por causa do pequeno tamanho do entrelaçador, podemos calcular todos os caminhos desejados, e assim gerar a função IOWEF exata para este código. Aplicando o truncamento (com vários valores de l_{max}, o maior peso de Hamming das seqüências intermediárias considerado), obtivemos as curvas indicadas na figura 5.1. Percebemos que as curvas convergem para o mesmo valor para uma relação sinal-ruído um pouco maior do que a menor relação sinal-ruído para o qual o limitante é válido. Isto é, as curvas tem o mesmo comportamento na região de validade do limitante.[1]

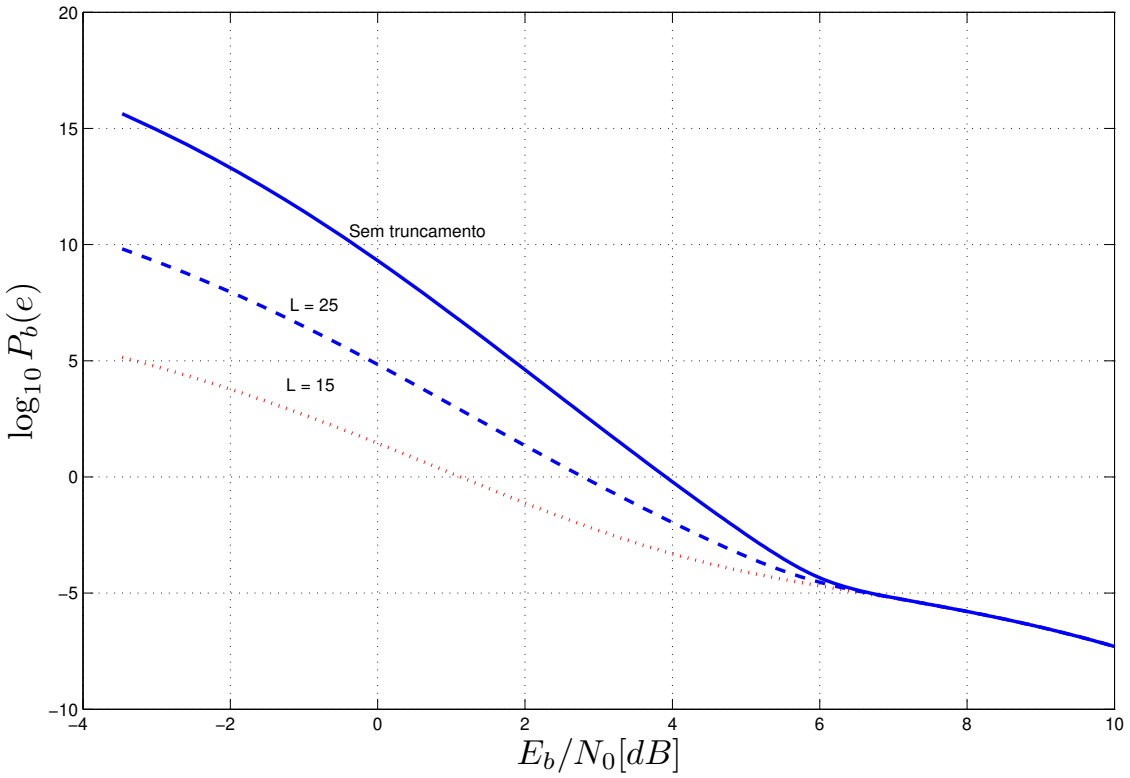

Fig. 5.1: Convergência de truncamento.

Aspectos Computacionais

Para calcular as IOWEF de cada codificador, utilizamos uma pequena modificação do método proposto em [14]. Para uma treliça com 2^m estados, podemos definir uma matriz **A** 2^m por 2^m. Cada elemento a^{xy} de **A** é um polinômio que descreve o conjunto de ramos que saem do estado x para o

[1] Vide seção 3.4.

estado y, como indicado pela equação 5.4,

$$A^{xy} = \sum_{w=1}^{k} \sum_{d=0}^{dmax} a_{w,d}^{xy} W^w D^d \tag{5.4}$$

onde $A_{w,d}^{xy}$ é o número de ramos com peso de entrada w e peso de saída d que causam uma transição do estado x para o estado y. Se algum A^{xy} é igual a zero, não há nenhuma transição do estado x para y. A matriz pode ser facilmente montada sabendo o codificador.

Os caminhos que chegam ao estado y, saindo do estado x passando pelo estado z, podem ser caracterizados através da multiplicação de $A^{xz} \cdot A^{zy}$. Assim, se quisermos obter a distribuição de todos os caminhos entre todos os estados depois de i passos, basta calcular $\mathbf{A}^i$. Podemos simplificar esta conta como mostrado em [15], pois estamos interessados somente no elemento a_{11} da matriz $\mathbf{A}^i$. Assim, seja o vetor $\mathbf{b}$ igual à primeira linha de $\mathbf{A}$. Seja também:

$$\mathbf{g}_i = \mathbf{b} \cdot \mathbf{A}^{i-1} \tag{5.5}$$

obtido recursivamente;

$$\begin{aligned} \mathbf{g}_1 &= \mathbf{b} \\ \mathbf{g}_{i+1} &= \mathbf{g}_i \cdot \mathbf{A} \end{aligned} \tag{5.6}$$

O termo desejado é igual ao primeiro elemento de $\mathbf{g}_i$. A vantagem de realizar o cálculo desta forma é que, na primeira situação, para cada termo da matriz $\mathbf{A}^i$ obtido a partir de $\mathbf{A}^{i-1} \cdot \mathbf{A}$, precisávamos realizar 2^m multiplicações e $2^m - 1$ somas de polinômios, totalizando assim 2^{3m} multiplicações e $2^{3m} - 2^{2m}$ somas de polinômios; no segundo caso, precisamos realizar o mesmo número de operações para cada termo, mas há somente 2^m termos, e não 2^{2m}. Assim, há um grande ganho computacional.

Devemos abordar também a complexidade de multiplicação dos polinômios. Quanto maiores os polinômios, mais tempo para realizar o calculo de multiplicação. Caso não tivéssemos limitado a maior valor do peso de saída (a ordem de D no polinômio), os termos de $\mathbf{g}_l$ seriam polinômios de ordem cada vez maior com o aumento de l. Isto acarretaria num aumento progressivo no tempo para o cálculo de $\mathbf{g}_{l+1}$. Além disso, este vetor ocuparia um espaço maior na memória do computador. O processo de limitação implementado era realizado a cada iteração, eliminando de cada polinômio os termos cuja grau de D fossem maiores do que o limite estabelecido. Assim, depois de um certo número de iterações, o polinômio atingia uma certa ordem e permanecia nesta, de modo que o tempo para o cálculo de cada iteração não aumentava mais pois a quantidade de operações era mantida. O que se alterava com o aumento das iterações era o valor de $a_{w,d}^{xy}$.

Limitamos os valores de d em 11 vezes a maior distância mínima de todos os codificadores internos. Para cada codificador, isto resultou numa limitação do peso de Hamming de uma seqüência

intermediária para cada codificador. Como um mesmo codificador externo era combinado com vários codificadores internos, escolhemos o maior valor dentre os limites dos codificadores internos que seriam combinados com ele. Assim, havia uma limitação sobre o peso de entrada do código. Desta forma, obtivemos as funções $A^C(W, D)$, para todos os codificadores envolvidos e os termos $A^C_{w,d}$, possibilitando assim calcular o valor aproximado para o limitante de união.

5.2.2 Resultados

Apresentamos a seguir os limitantes de erro para os sistemas encontrados. Para cada gráfico, estão indicados qual é o codificador interno. Mostramos em cada gráfico as curvas para as 4 combinações que fizemos por codificador interno, uma para cada codificador externo. Fizemos as combinações somente nos casos onde o tamanho em bits da saída por seção da treliça de um codificador externo era igual ao tamanho de entrada do codificador interno. O valor m_e indica qual é a quantidade de memórias. O tamanho do entrelaçador é de 840 em todos os casos. A eficiência espectral muda de acordo com a taxa do código.

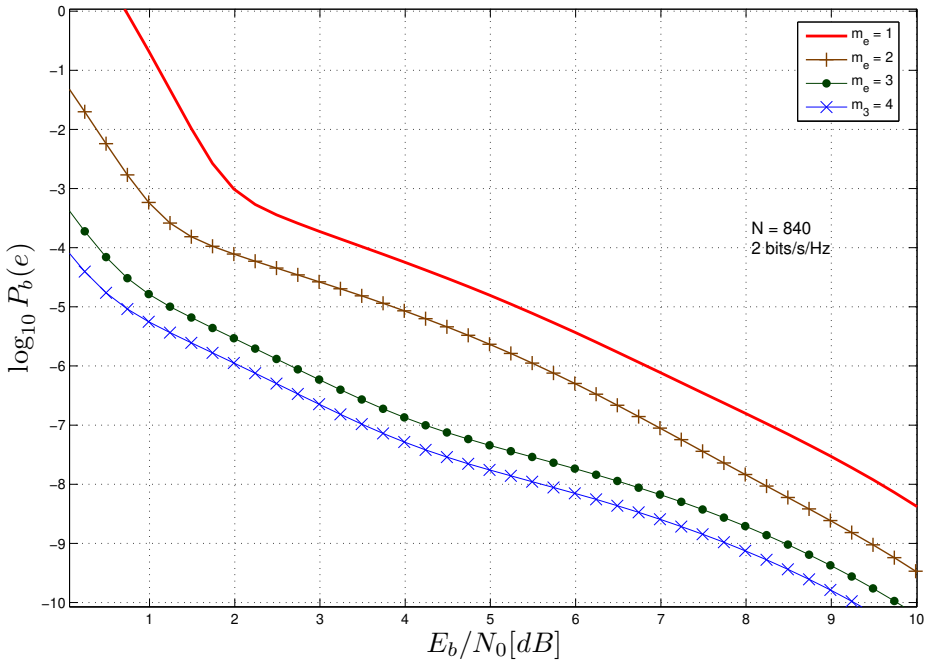

Fig. 5.2: Limitantes para sistemas com o codificador C_{51}.

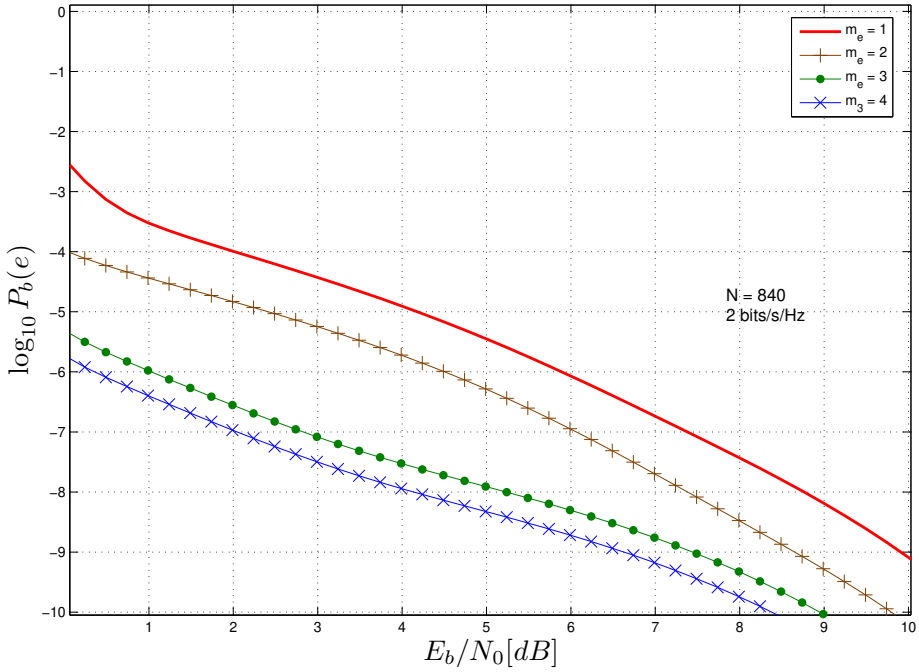

Fig. 5.3: Limitantes para sistemas com o codificador C_{52}.

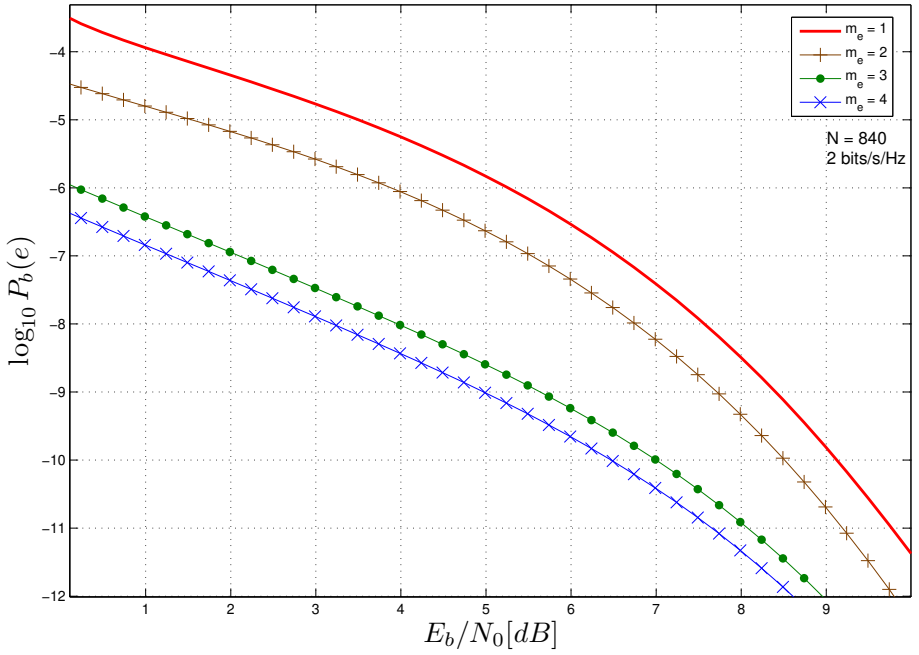

Fig. 5.4: Limitantes para sistemas com o codificador C_{53}.

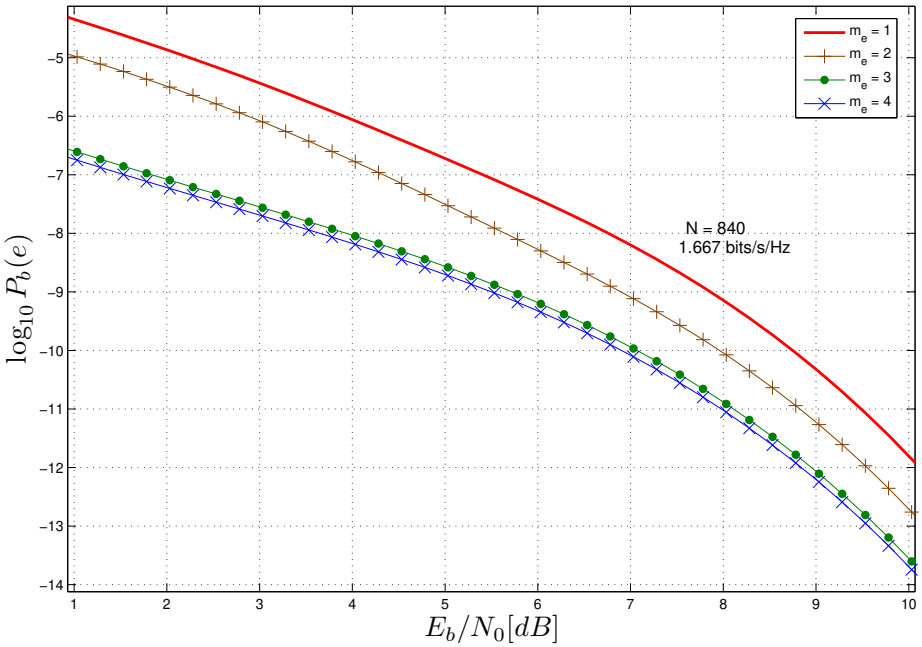

Fig. 5.5: Limitantes para sistemas com o codificador C_{63}.

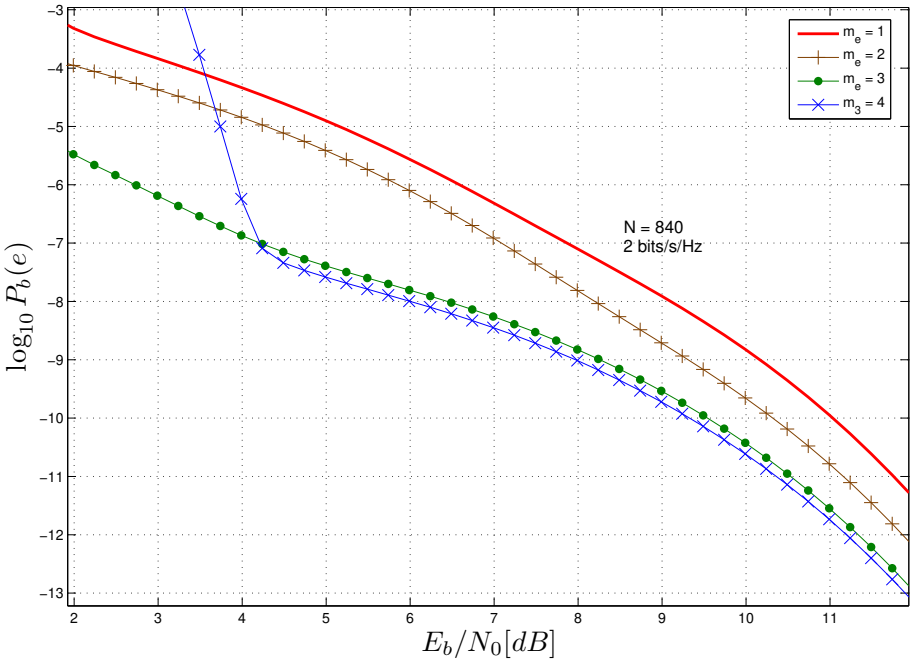

Fig. 5.6: Limitantes para sistemas com o codificador C_{72}.

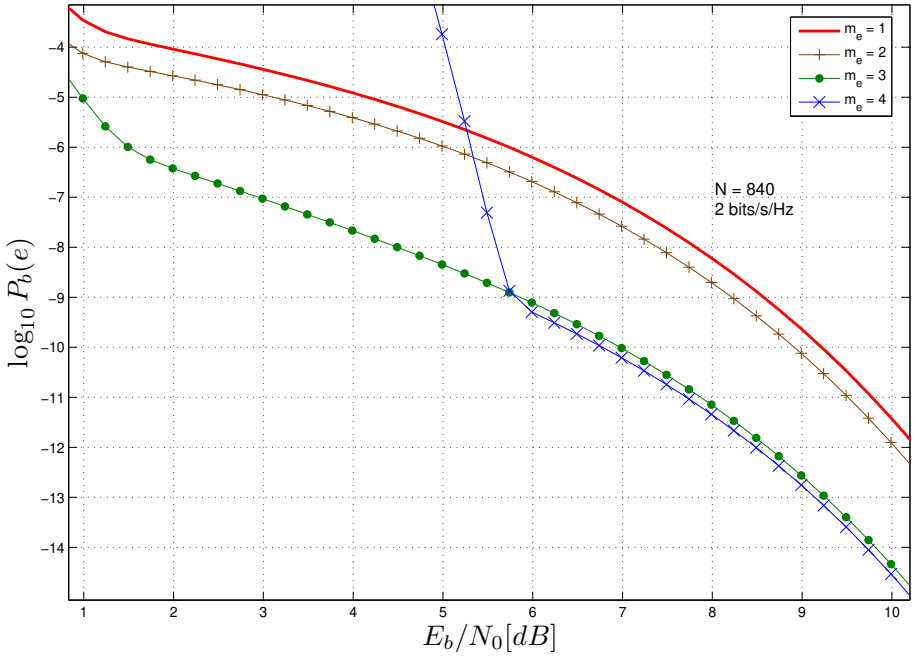

Fig. 5.7: Limitantes para sistemas com o codificador C_{73}.

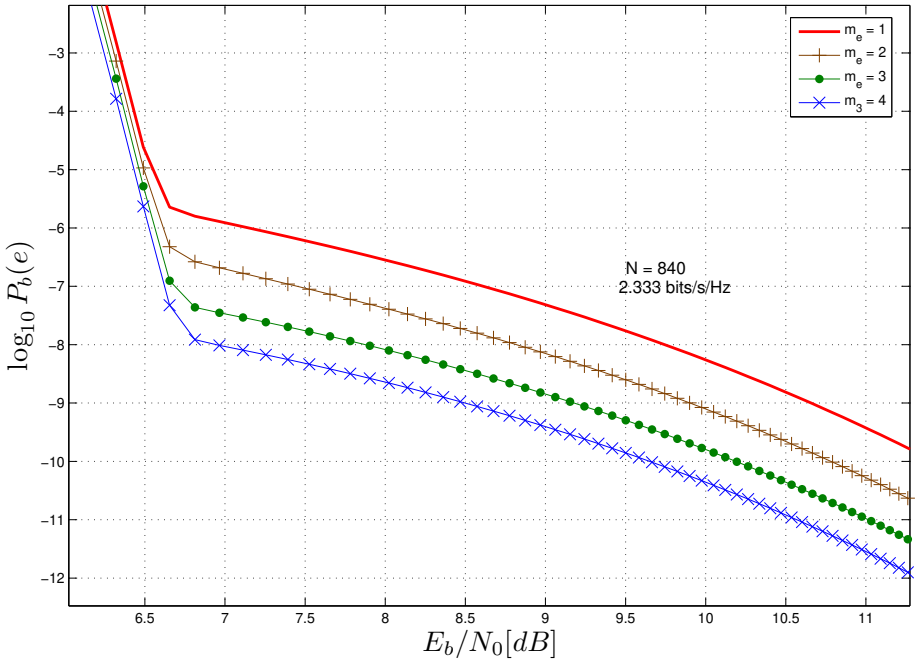

Fig. 5.8: Limitantes para sistemas com o codificador C_{81}.

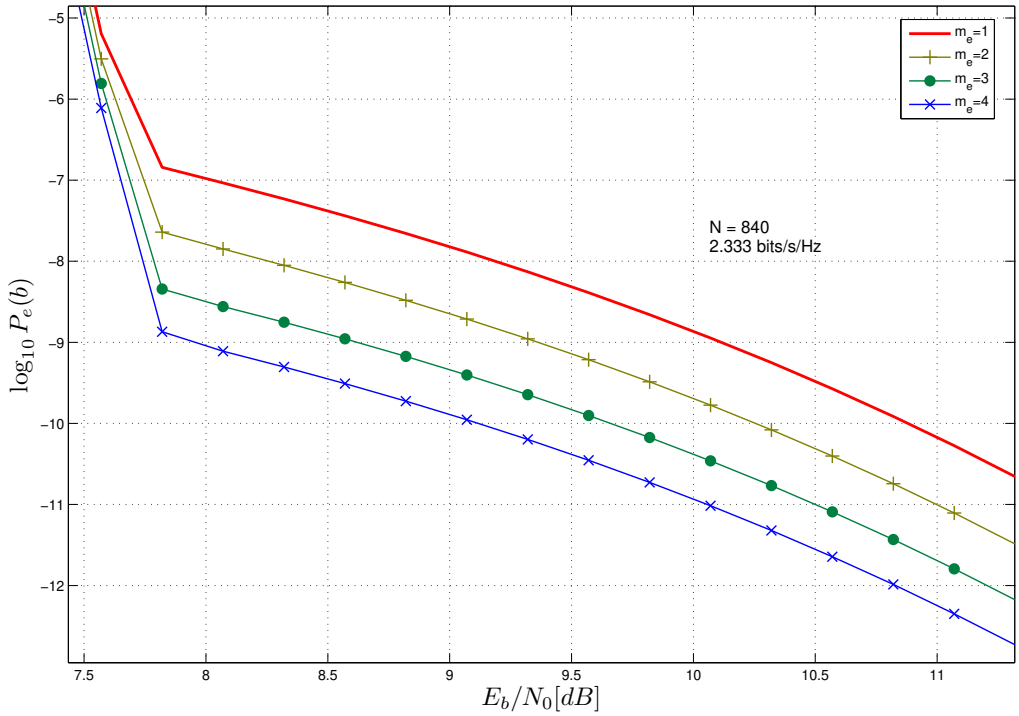

Fig. 5.9: Limitantes para sistemas com o codificador C_{82}.

5.3 Análise

Vamos analisar a influência dos seguintes elementos:

- A quantidade de memórias do código externo,
- A influência de d_{min} e d_{ef},
- A influência da taxa e quantidade de memórias do código interno.

5.3.1 Influência da quantidade de memórias do código externo

Com o aumento da quantidade de memórias do código externo, a sua distância livre aumenta. Podemos observar das figuras 5.3 até 5.8 que há um ganho maior quando a distância livre aumenta de 2 para 3 do que quando ela aumenta de 3 para 4. Isto acontece porque $d_3 = \infty$ para os codificadores internos, e eles são assim pares. Nestas situações, o valor de l_{min} da equação 3.13 assume o próximo valor viável, que no caso é 4. Por isso, quando aumentamos a distância livre de 3 para 4, não estamos

efetivamente alterando o valor de l_{min}. Nos casos 5.6 a 5.8, o ganho obtido ao se passar de 3 para quatro memórias no código externo é tão pequeno que não justifica o acréscimo de uma memória (e da complexidade) no código exterior.

Aumentar a quantidade de memórias do código externo altera também os termos $A_{w,l}^{C_e}$ da IOWEF do código externo. Quanto maior a quantidade de memórias, menor este número, o que reduz a probabilidade de erro de bit. Por exemplo, para um entrelaçador de tamanho 840, há um total de 672 seqüências de saída do código externo com uma memória e taxa 4/5 que tem peso de Hamming igual a 2. Destas, 336 tem peso de entrada igual a 1 e 336 tem peso de entrada igual a 2. Ao adicionarmos uma memória, este número cai para 168, sendo que todas elas tem peso de entrada igual a 1. É uma redução de 75%.

O efeito desta redução é mais aparente nos casos mostrados nos gráficos 5.8 e 5.9. Nestes casos, a redução no termo $A_{w,l}^{C_e}$ é tão influente quanto a variação de l_{min}.

5.3.2 Influência de termos individuais no limitante de união

Não podemos analisar separadamente a influência de d_{min} e d_{ef} no limitante de união. A distância mínima domina quando a relação sinal-ruído é alta. Quanto maior for o valor de d_{ef} menor será o valor da energia para que isto aconteça. Na figura 5.10, desenhamos a influência de alguns dos termos do limitante de união para o sistema composto pelo codificador externo C_{72}^{e} e pelo codificador interno C_{72}. Neste caso, $d_{e,l} = 2$, $d_{min} = 2.000$(distância Euclidiana) e $d_{ef} = 4$. Como $d_{e,l} = 2$, $d_{min,\alpha_{max}} = d_{ef}$. As curvas foram obtidas avaliando a função 3.14, para somente um valor de d.

É vantajoso que a participação de d_{ef} desapareça o mais rápido possível, pois assim chegaríamos à participação da distância mínima mais rápido. Por um bom intervalo, o desempenho do código depende muito da distância Euclidiana efetiva. Este intervalo explica porque em algumas curvas temos alguma variação na velocidade de diminuição da probabilidade de erro de bit. Observamos também porque em alguns casos é visível a "explosão"do limitante. Isto acontece por causa do maior termo considerado. Como ele tem um valor de d muito alto, ele cresce muito rápido com o aumento da RSR. Se por acaso palavras com baixo peso de Hamming de entrada causarem saídas com este peso, o multiplicador M_d será alto, pois l será relativamente pequeno.

Para sistemas onde $d_{min,\alpha_M} \neq d_{ef}$, o termo que domina o limitante é d_{min,α_M}, como podemos ver na figura 5.11. Podemos ver que a influência do termo relativo à d_{ef} não domina diretamente nunca o limitante união. Ele domina indiretamente através de d_{min,α_M}, que neste caso vale $2d_{ef}$.

Em nenhum dos casos a distância livre do código externo foi maior do que o maior peso de Hamming de uma seqüência de entrada do codificador interno que gera uma saída com peso igual a distância mínima. Se isto acontecesse, a distância mínima de fato seria maior do que distância mínima do código interno.

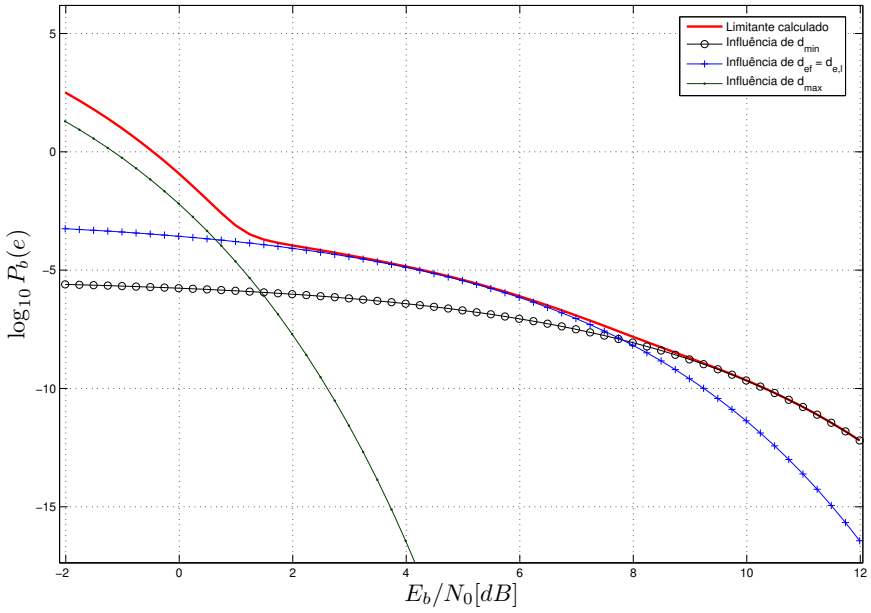

Fig. 5.10: Influência de diversos termos no limitante de união para o sistema $C_{72}^{e} - C_{72}$, $N = 840$.

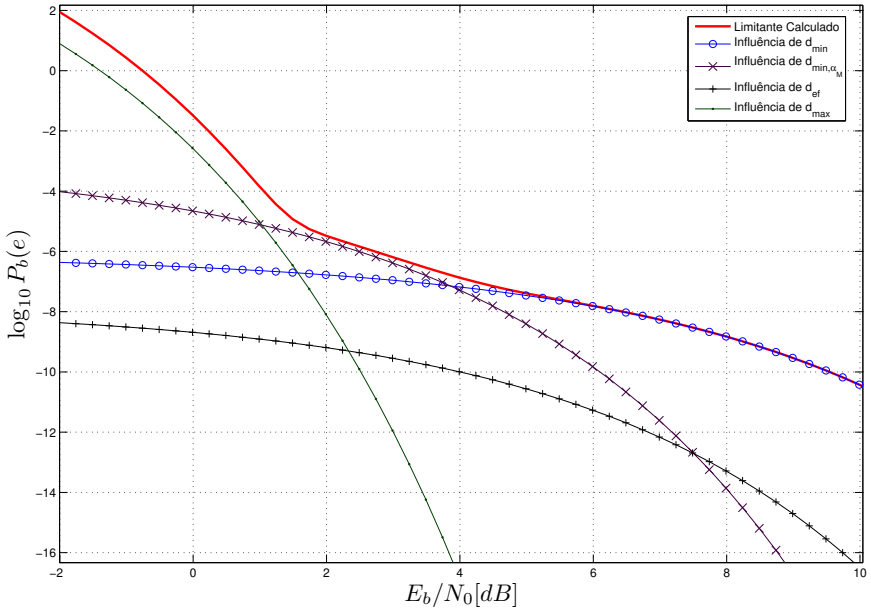

Fig. 5.11: Influência de diversos termos no limitante de união para o sistema $C_{73}^{e} - C_{72}$, $N = 840$.

5.3.3 Influência da taxa e quantidade de memórias do código interno

A taxa do código interno, junto com a sua quantidade de memórias, influencia nos valores de d_{min} e d_{ef}. Como podemos ver na tabela 5.2, códigos com alta taxa e poucas memórias tem valores menores para d_{min} e d_{ef}. O algoritmo também não fornece altos valores para d_{ef} quando há várias transições paralelas do estado identidade para o estado identidade. Quanto maior a quantidade destas transições, maior a quantidade de palavras com peso de Hamming 2 que o fazem. Como há um número limitado de sinais, fica mais difícil fornecer altos pesos Euclidianos de saída para estes sinais de forma BGU.

Aumentando a quantidade de memórias diminuímos a quantidade de transições paralelas, tendo efeito similar à diminuição da taxa do codificador.

5.3.4 Eficiência do Algoritmo

Os resultados apresentados pelo algoritmo são satisfatórios. Comparando com o codificador apresentado em [7] (codificador C_{51}), o ganho obtido com o acréscimo de memórias é significativo. Comparando o sistema $C_{51}^{e} - C_{51}$ com o sistema $C_{71}^{e} - C_{72}$, temos um ganho de aproximadamente 0.5dB para $P_b(e) = 10^{-6}$. Ambos os codificadores tem a mesma taxa de 2 bits de informação por uso de canal, e ambos tem a mesma quantidade de memórias por uso de canal: $C_{51}^{e} - C_{51}$ possui duas memórias para a constelação $2 \times 8PSK$ e $C_{71}^{e} - C_{72}$ possui 3 memórias para a constelação $2 \times 8PSK$.

O codificador proposto em [7] foi obtido através de uma pesquisa extensa (mas não exaustiva). Os codificadores propostos aqui foram encontrados através de um método. Podemos afirmar que o algoritmo é bom pois resulta em bons codificadores BGU, pois comparados com os resultados de uma procura extensa, retornaram desempenhos similares.

5.4 Síntese

Neste capítulo apresentamos os codificadores BGU obtidos através da aplicação do algoritmo para vários casos. A partir destes codificadores montamos alguns sistemas com concatenação serial e traçamos limitantes de desempenho para eles. Os limitantes foram obtidos através de uma aproximação que é válida. Explicamos graficamente a influência de alguns parâmetros nos limitantes de união obtidos para alguns exemplos, e justificamos qualitativamente esta influência. Comparando com resultados anteriores de outros autores, concluímos que os codificadores encontrados através do algoritmo proposto são bons.

Capítulo 6

Conclusões

A principal contribuição desta dissertação foi a de propor um guia para o projeto de codificadores internos BGU para uso em sistemas com concatenação serial. Não demos muita ênfase às dificuldades de cada passo do algoritmo pois elas são contornáveis com alguma experiência. Não há nenhuma garantia de que os algoritmos encontrados são ótimos no sentido de que fornecem a maior distância livre efetiva possível, ou que fornecem o melhor desempenho quando utilizados como codificadores internos em sistemas com concatenação serial. Entretanto, comparado com resultados anteriores de outros autores, os codificadores encontrados podem ser considerados bons.

Obtivemos codificadores somente para o caso binário, mas o método não se restringe à esta limitação. Além disso, aplicamos o método considerando que era importante obter uma boa distância Euclidiana de saída para entradas com distância de Hamming 2. O método pode ser aplicado utilizando outras métricas, realizando algumas modificações nas funções que avaliam distâncias, etc.

Seria interessante analisar os codificadores BGU sob o aspecto de otimalidade. Como há codificadores BGU ruins, uma pergunta melhor seria saber se um codificador ótimo seria BGU. Para isto, precisaríamos obter limitantes para os valores da distância mínima e da distância livre efetiva para codificadores, e mostrar que codificadores BGU atendem às condições necessárias para se atingir estes limitantes. Assim como constelações geometricamente uniformes são ótimas no que se refere à distâncias entre os seus pontos, rotulamentos BGU também distribuem uniformemente os bits. Intuitivamente, eles devem ser bons.

Uma questão levantada no capítulo 2 é a de como tratar rotulamentos e codificadores que possuem a propriedade de uniformidade de erro de bit mas que não possuem nenhum grupo associado. Precisamos de uma estrutura para podermos manipular estes rotulamentos, que possuem a mesma vantagem de codificadores BGU: distribuem os rótulos de alguma maneira uniforme.

Os codificadores projetados neste trabalho levavam em conta que o canal de transmissão era Gaussiano e que o método de decodificação era por máxima verossimilhança. Em vários canais reais, ne-

nhuma destas duas hipóteses é verdadeira. Há muitos canais não Gaussianos onde seria interessante utilizar sistemas com concatenação serial para se obter bons desempenhos. Para se ter bons desempenhos, utiliza-se seqüências com grande tamanho, o que inviabiliza a decodificação por máxima verossimilhança. Atualmente, há uma grande pesquisa para a decodificação iterativa sobre grafos com ciclos, que pode muito bem ser utilizada para a decodificação de sistemas com concatenação serial.

Um caminho futuro para esta pesquisa seria a de considerar a influência da distância livre efetiva e da distância mínima no processo de decodificação iterativo, e posteriormente para casos onde os canais são não-Gaussianos. Este estudo provavelmente levaria a outras diretrizes de projeto. A qualidade de codificadores BGU para estes casos dependeria de como eles atendem a estas novas diretrizes.

Referências Bibliográficas

[1] D. Slepian, "Group codes for the gaussian channel," *Bell Syst. Tech. Journal*, vol. 47, pp. 575–602, Abril 1968.

[2] G. D. Forney Jr., "Geometrically uniform codes," *IEEE Transactions on Information Theory*, vol. 37, pp. 1241–1260, Setembro 1991.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," *Proc. ICC*, pp. 1064–1070, Maio 1993.

[4] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, vol. 42, pp. 409–428, Março 1996.

[5] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Transactions on Information Theory*, vol. 44, pp. 909–926, Maio 1998.

[6] E. Biglieri, D. Divsalar, P. J. McLance, and M. K. Simon, *Introduction do Trellis Coded Modulation*. Macmillian Publishing Company, 1991.

[7] R. Garello, G. Montorsi, S. Benedetto, D. Divsalar, and F. Pollara, "Labelings and encoders with the uniform bit error property with applications to serially concatenated trellis codes," *IEEE Transactions on Information Theory*, vol. 48, pp. 123–136, Janeiro 2002.

[8] E. A. Lee and D. G. Messerschmitt, *Digital Communication*. Kluwer Academic Press, 1994.

[9] S. Lang, *Algebra*. Addison Wesley, 1965.

[10] G. D. Forney Jr. and M. D. Trott, "The dynamics of group codes: State spaces, trellis diagrams, and canonical encoders," *IEEE Transactions on Information Theory*, vol. 39, pp. 1491–1513, Setembro 1993.

[11] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423,623–656, Julho 1948.

[12] S. Benedetto, R. Garello, M. Mondin, and G. Montorsi, "Geometrically uniform partitions of L x MPSK constellations and related binary trellis codes," *IEEE Transactions on Information Theory*, vol. 39, pp. 1773–1798, Novembro 1993.

[13] A. Graell i Amat, G. Montorsi, and S. Benedetto, "Design and decoding of optimal high-rate convolutional codes," *IEEE Transactions on Information Theory*, vol. 50, pp. 867–881, Maio 2004.

[14] J. K. Wolf and A. J. Viterbi, "On the weight distribution of linear block codes formed from convolutional codes," *IEEE Transactions on Communications*, vol. 44, pp. 1049–1051, Setembro 1996.

[15] G. A. de Deus Jr., *Sistemas FFH-CDMA Codificados*. PhD thesis, UNICAMP, Agosto 2002.