UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

MONOGRAFIA EM ENGENHARIA DE TELEINFORMÁTICA

ANÁLISE DE CÓDIGOS CORRETORES DE ERROS LEXICOGRÁFICOS EM SISTEMAS DE TRANSMISSÃO DIGITAL

RAFAEL COELHO ASSUNÇÃO

FORTALEZA

2009

RAFAEL COELHO ASSUNÇÃO

ANÁLISE DE CÓDIGOS CORRETORES DE ERROS LEXICOGRÁFICOS EM SISTEMAS DE TRANSMISSÃO DIGITAL

Monografia apresentada à disciplina de Projeto de Final de Curso, do curso de graduação em Engenharia de Teleinformática da Universidade Federal do Ceará.

Orientador: Elvio César Giraudo

FORTALEZA

2009

Dedico este trabalho aos meus pais, pelo valioso incentivo e pela confiança.

AGRADECIMENTOS

Meus sinceros agradecimentos...

... a Deus, pois sem Ele, a conclusão deste trabalho teria sido impossível;

... a minha família, pela compreensão e o apoio nos momentos difíceis de dedicação;

... a minha noiva, Viviane, que nos momentos em que pensei em desistir esteve sempre ao meu lado me apoiando e dando forças para seguir em frente;

... aos amigos, pela força e pelos votos sinceros de sucesso neste trabalho;

... a meu orientador, Prof. Dr. Elvio César Giraudo, por aceitar a orientação deste trabalho e conduzi-la com sabedoria e paciência.

"Para se ser feliz até um certo ponto
é preciso ter-se sofrido até esse mesmo ponto.".

Edgar Allan Poe

RESUMO

Este trabalho reúne contribuições da teoria da codificação para a compreensão de um tipo de código corretor de erros, chamado código de bloco lexicográfico ou *lexicode*. Com a grande quantidade de informação que é transmitida atualmente pelos mais variáveis canais de comunicação, é cada vez mais necessária a garantia da integridade dessas informações e isso exige o uso de sofisticados sistemas de códigos corretores de erros. Além disso, a presença de ruído na transmissão torna essa tarefa imprescindível. Os códigos corretores de erros têm como função acrescentar redundância às mensagens que serão transmitidas, fazendo com que estas possam ser corrigidas quando ocorrem erros de transmissão de dados. Neste trabalho, são descritos alguns aspectos básicos da teoria dos códigos de bloco lexicográficos, com a análise das diferentes abordagens que existem na literatura atual e uma discussão sobre as características e a eficiência de correção de erros dos *lexicodes* abordados por diversos autores.

Palavras-chave: *lexicodes*, *BER*, códigos corretores de erro, síndrome, *coset*, códigos de bloco, ordenamento lexicográfico

SUMÁRIO

LISTA DE SÍMBOLOS E SIGLAS

F_2^n	Espaço n-dimensional binário
G	Matriz geradora
H	Matriz de paridade
n	Tamanho do código
k	Dimensão do código
R	Taxa do código
d	Distância de Hamming
wt	Peso de Hamming
dmin	Distância mínima do código
u	Vetor mensagem
$\widehat{u}$	Vetor mensagem estimada
v	Seqüência transmitida
r	Seqüência recebida
e	Vetor erro
s	Síndrome
E_b/N_0[dB]	Energia por *bit* sobre a densidade espectral de potência do ruído
BER	*Bit error rate* - taxa de erro de bit
AWGN	*Additive White Gaussian Noise* - Ruído Branco Aditivo e Gaussiano
σ^2	Variância
E	Energia do sinal
E_b	Energia por bit
E_s	Energia por símbolo

f_0	Freqüência de transmissão
max	Função máximo
Lex	Ordenamento lexicográfico padrão
CLex	Ordenamento lexicográfico classificado
CRev	Ordenamento lexicográfico classificado reverso
SPB	*Sphere packing bound*
SNR	Relação sinal-ruído
C_4^3	Código com $k = 4$ e $d = 3$
ρ	Raio de cobertura

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

A presente pesquisa monográfica foi realizada no contexto da Disciplina de Projeto de Final de Curso, do curso de graduação em Engenharia de Teleinformática da UFC, de agosto a dezembro de 2009.

No âmbito da teoria da informação, que trata dos aspectos quantitativos de armazenamento e transmissão das mensagens, e tem como um de seus objetivos principais garantir a integridade dos dados enviados através de algum tipo de canal, trabalhamos, aqui, os códigos corretores de erros, analisando suas características e utilidades, tendo nosso foco direcionado aos códigos de bloco lexicográficos.

Ao contrário das teorias matemáticas que surgiram nas universidades e que geralmente, após um longo período de tempo, migraram para as aplicações práticas em tecnologia e indústrias, a teoria de códigos corretores de erros surgiu nos laboratórios de empresas de telefonia e posteriormente se transformou em uma teoria matemática completa, com aplicações em várias áreas como, por exemplo, geometria algébrica. Um código corretor de erros visa recuperar informações que no processo de emissão tenham sofrido alteração devido a algum tipo de ruído e/ou desvanecimento.

Pode-se afirmar que hoje praticamente todo sistema de envio de informações possui algum tipo de código corretor de erros. Como exemplos típicos, a telefonia digital, a transmissão de dados via satélite, a comunicação interna em computadores, o armazenamento de dados em discos rígidos e o *Bluetooth*.

Com o aumento da confiabilidade nas comunicações digitais e sendo o computador digital atualmente uma ferramenta essencial, os códigos corretores de erros vêm conquistando uma posição relevante. Existem diversos exemplos que podem ilustrar a praticidade e a importância do seu uso. Tais códigos estão sendo muito utilizados no armazenamento em discos devido ao aumento da densidade destes. Quanto maior for esta, a probabilidade de ocorrência de erros também aumenta. Outro exemplo é seu uso na transmissão de informação pelas naves espaciais, na qual a atividade solar e outras condições atmosféricas, por exemplo, podem introduzir erros em sinais fracos vindos do espaço. Também como exemplo, pode ser citado o áudio digital, que teve o aumento da sua popularidade devido ao desenvolvimento dos códigos corretores de erros, que facilitam o processo de digitalização. Ao inicializar a leitura do CD, o sistema corrige os erros produzidos por marcas de dedos, arranhões e outras imperfeições, para logo em seguida transformá-las em sinais sonoros.

Justificou-se, assim, o estudo deste tema, a partir da crescente utilização desses códigos corretores de erros e a conseqüente necessidade de serem gerados códigos mais robustos e eficientes. Uma classe de códigos bastante usada na correção de erros é a dos códigos de bloco lineares. Como exemplo pode ser citado, segundo Hefez e Vilela (2002), o código de Hamming, o código de Golay e o código de Reed-Muller. Os códigos escolhidos para serem analisados neste trabalho foram os lexicográficos, que são um exemplo de códigos lineares.

Diz-se que o código tem um ordenamento lexicográfico se seu ordenamento respeita uma base dada. Os códigos lexicográficos são construídos a partir dessa base, aplicando um algoritmo *greedy*, formando uma lista lexicográfica. Se o ordenamento de vetores binários de tamanho n para a base dada é lexicográfico, isto é, de acordo com os valores crescentes quando interpretados como números binários, é feita uma busca nesta lista, começando do vetor nulo, selecionando o próximo vetor na lista que tem sua distância de Hamming para cada vetor escolhido anteriormente de pelo menos d, obtendo-se um código lexicográfico com distância mínima d.

1.1 Objetivos

Nossos objetivos principais nessa pesquisa foram os de analisar as características e particularidades dos códigos de blocos lexicográficos e apresentar o estado da arte.

Objetivamos apontar as considerações feitas por diversos autores acerca desses códigos e analisar a capacidade de correção dos principais códigos lexicográficos presentes na literatura, através da implementação de algoritmos geradores dos códigos lexicográficos existentes, além de propormos um ordenamento lexicográfico alternativo baseado nas características desses códigos e a escolha de um método para a geração da matriz G de palavras-código.

Fez parte de nossos objetivos específicos a simulação no *Matlab*®, por meio do método de Monte Carlo, dos códigos lexicográficos em um sistema de comunicações, fazendo uma análise da relação entre a taxa de erro de *bit* (*bit error rate – BER*) e a energia por bit sobre a densidade espectral de potência do ruído, $E_b/N_0[dB]$, para cada um desses códigos gerados.

1.2 Metodologia

Nossa pesquisa teve como metodologia a análise do texto de Brualdi e Pless (1993), que formalizaram uma generalização dos códigos lexicográficos através de algoritmos *greedy*, e a análise de textos de outros autores que contribuíram para o entendimento do tema ao qual nos propusemos a analisar.

Assim, coletamos as referências feitas por estes autores em relação aos *lexicodes* e, para cada autor, desenvolvemos algoritmos específicos para cada código lexicográfico existente na literatura, com o auxílio do programa *Matlab®,* e por fim os comparamos entre si, como definido nos nossos objetivos.

Este trabalho é formado por cinco capítulos, além deste, organizados da seguinte forma: sistema de transmissão digital; codificação de canal; códigos lexicográficos; simulação e resultados; conclusão e trabalhos futuros.

No Capítulo 2, descrevemos a estrutura de um sistema de transmissão digital, explicando como uma mensagem é enviada a partir da fonte e o que ocorre até ela chegar ao seu destino.

No Capítulo 3, tratamos especificamente da codificação de canal, detalhando principalmente os códigos de bloco e suas principais definições, que serão utilizadas no Capítulo 4.

No Capítulo 4, apresentamos a abordagem principal deste trabalho, que é a teoria dos códigos de bloco lexicográficos e as técnicas para a geração dos principais *lexicodes* conhecidos na literatura.

No Capítulo 5, simulamos os códigos gerados no *Matlab®,* comentando e analisando os resultados obtidos.

Na conclusão, ressaltamos as contribuições que auxiliaram a elaboração desse trabalho. E, para finalizar, sugerimos alguns trabalhos futuros relacionados com esta monografia.

2. SISTEMA DE TRANSMISSÃO DIGITAL

O início da teoria da informação e da comunicação se deu quando Shannon (1948) publicou um artigo intitulado *"A mathematical theory of communication"* (Uma teoria matemática da comunicação). Dado um canal de comunicação que pode ter a informação enviada por ele corrompida, ele identificou um valor chamado de capacidade do canal e provou que uma comunicação confiável é possível a uma taxa abaixo da capacidade de canal. O resultado de Shannon garante que os dados podem ser codificados antes da transmissão, tal que os dados alterados podem ser decodificados com um grau de precisão específico (HUFFMAN, PLESS, 2003).

O propósito de um sistema de comunicação é transmitir sinais de banda-base portadores de informação de um lugar para outro através de um canal de comunicação (HAYKIN, 2004). Um problema clássico de transmissão através de um canal ruidoso é mostrado no diagrama de blocos da Figura 2.1, no qual temos uma mensagem *u* que queremos enviar da fonte para o destino.

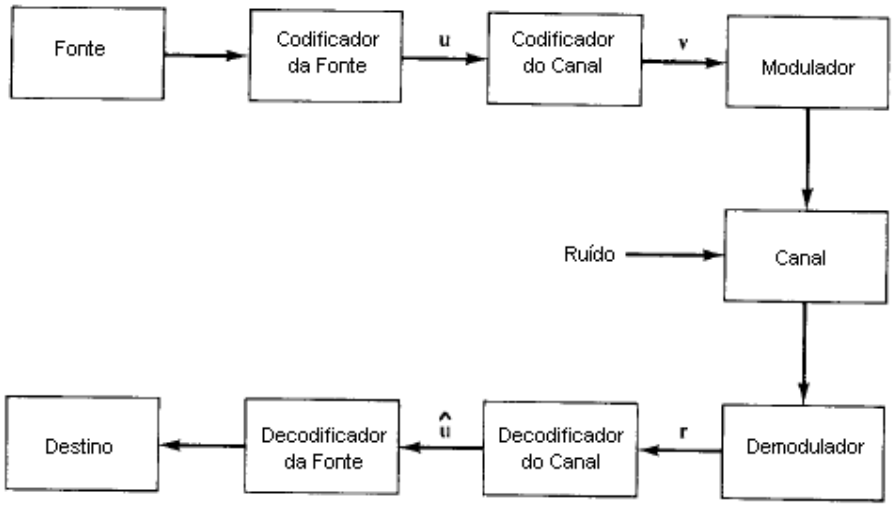


Figura 2.1 - Diagrama de blocos de um sistema de transmissão digital.

A fonte pode ser uma onda contínua ou uma seqüência de valores discretos. A saída dessa fonte é uma seqüência de símbolos discretos.

Segundo Haykin (2004), um problema importante em comunicações é a representação eficiente, tirando a redundância, de dados gerados por uma fonte discreta. O processo pelo qual essa representação é realizada denomina-se codificação de fonte. Uma forma de codificar a fonte é tendo conhecimento da estatística da mesma. Em especial, se alguns símbolos-fonte forem conhecidos como mais prováveis do que outros, então poderemos explorar essa característica na geração de um código da fonte atribuindo palavras-código breves a símbolos da fonte freqüentes e palavras-código longas a símbolos da fonte raros.

O codificador da fonte transforma a saída dela em uma seqüência de dígitos binários, que é a mensagem codificada. As palavras-código produzidas pelo codificador devem estar na forma binária e o código da fonte deve ser singularmente decodificável, de forma que a seqüência da fonte original possa ser reconstruída perfeitamente a partir da seqüência binária codificada.

Para um canal relativamente ruidoso (por exemplo, um canal de comunicação sem fio), a probabilidade de erro pode atingir valores tão elevados quanto 10^{-1}, o que significa que (em média) somente 9 em 10 dos *bits* transmitidos são recebidos corretamente. Para muitas aplicações, este nível de confiabilidade é inaceitável. De fato, uma probabilidade de erro igual a 10^{-6}, ou até mesmo mais baixa, freqüentemente é um requisito necessário. Uma das metas do projeto da codificação de canal é aumentar a resistência de um sistema de comunicação digital a ruídos de canal. Esta resistência é obtida mediante a incorporação de redundância controlada no codificador, que logo é removida no decodificador. Desta forma pode-se reconstruir a seqüência da fonte com a maior precisão possível.

Símbolos discretos não são adequados para transmissão através de um canal físico, por isso necessitamos de um modulador. O modulador em um sistema de comunicação deve selecionar uma onda de duração T segundos, que é adequada para a transmissão, para cada símbolo de saída do codificador. No caso do código binário, o modulador pode gerar um dos dois sinais, $s_o(t)$ para "0" ou $s_1(t)$ para "1".

$$s_0(t) = \sqrt{\frac{2E}{T}} \operatorname{sen}(2\pi f_0 t + \frac{\pi}{2}), \quad 0 \le t \le T$$

$$s_1(t) = \sqrt{\frac{2E}{T}} \operatorname{sen}(2\pi f_0 t - \frac{\pi}{2}), \quad 0 \le t \le T, \tag{2.1}$$

onde f_0 é a freqüência de transmissão, que é um múltiplo de $\frac{1}{T}$ e E é a energia de cada sinal. Chamamos essa modulação de *binary-phase-shift-keyed* (*BPSK*), já que o sinal transmitido é um pulso de onda senoidal no qual a fase é $+\frac{\pi}{2}$ ou $-\frac{\pi}{2}$, dependendo do símbolo do codificador.

O ruído consiste numa alteração de alguma das características do sinal transmitido por efeito de um outro sinal exterior ao sistema de transmissão ou gerado pelo próprio sistema. Estes sinais indesejados são de natureza aleatória, não sendo possível prever o seu valor num instante de tempo futuro. O ruído pode ser aditivo (soma-se ao sinal) ou multiplicativo (o sinal resultante é o produto do sinal transmitido pelo ruído).

Uma vez que o ruído é um processo aleatório, este deve ser descrito e tratado com recurso a métodos estatísticos. O ruído diz-se branco quando a sua densidade espectral de potência média é constante a todas as frequências; diz-se colorido no caso contrário. As características do ruído são ainda descritas através da função densidade de probabilidade da sua amplitude. Diz-se então que o ruído segue uma distribuição Normal (Gaussiana), de Poisson, etc. Uma das formas de ruído mais utilizadas para modelar este aspecto de um sistema de transmissão é o AWGN.

Dada a natureza aleatória do ruído, não é possível eliminá-lo completamente num sistema de transmissão. Os efeitos do ruído fazem-se sentir pela introdução de erros nos sistemas de transmissão digital. Nestes, o desempenho é medido por meio da probabilidade de ocorrerem erros, frequentemente erros de *bit*. Chamamos essa probabilidade de probabilidade de erro de *bit* ou taxa de erro de *bit* (*BER – Bit Error Rate*).

Após a inserção de ruído, o demodulador processa cada onda recebida de duração *T* segundos e produz uma saída chamada de seqüência recebida $r = v + e$, onde *e* é algum vetor randômico de erro que foi adicionado e *v* é a seqüência transmitida.

O decodificador de canal transforma a seqüência *r* em uma seqüência binária $\hat{u}$ chamada de mensagem estimada. O objetivo da decodificação é determinar o vetor de erro *e* e corrigir a seqüência recebida. A decodificação da fonte transforma a mensagem prevista em uma estimativa da saída da fonte e entrega essa estimação para seu destino. (SHU LIN, COSTELO, 1983).

A tarefa com que se defronta o projetista de um sistema de comunicação digital é a de prover uma instalação eficiente, em termos de custo, capaz de transmitir informações de uma extremidade do sistema, a uma taxa e em um nível de confiabilidade e qualidade que sejam aceitáveis, a um usuário na outra extremidade. Os dois parâmetros de sistema fundamentais à disposição do projetista são a potência do sinal transmitido e a largura de banda do canal. Esses dois parâmetros, juntamente com a densidade espectral de potência de ruído do receptor, determinam a relação energia por *bit* sobre a densidade espectral de potência do ruído E_b/N_0[dB]. Para uma E_b/N_0[dB] fixa, a única opção prática disponível para modificar a qualidade dos dados de problemática aceitável é usarmos a codificação para controle de erros (HAYKIN, 2004).

Outra motivação prática para o uso de codificação é reduzirmos a E_b/N_0[dB] necessária para uma taxa de erro fixa. Essa redução, por sua vez, pode ser explorada para reduzir a potência transmitida necessária ou para reduzir os custos de *hardware*, exigindo um menor tamanho de antena no caso de comunicações de rádio.

Uma das formas de controle de erro para obtenção de integridade de dados pode ser exercida, por exemplo, por meio de uma correção direta de erros (*FEC - forward error correction*), ou seja, tenta-se corrigir os erros da palavra após seu recebimento. Como já dito anteriormente, a fonte gera informações e o codificador de canal no transmissor aceita *bits* de mensagem e adiciona redundância de acordo com uma regra predefinida, produzindo assim dados codificados a uma taxa de *bits* mais elevada. O decodificador de canal no receptor então explora a redundância para decidir quais *bits* de mensagem foram de fato transmitidos.

A meta conjunta do codificador e decodificador de canal é minimizar o efeito de ruído de canal, ou seja, o número de erros entre a entrada do codificador de canal e a saída do decodificador de canal é minimizado.

Para um esquema de modulação fixo, a adição de redundância às mensagens codificadas implica a necessidade de grande largura de banda de transmissão. Além disso, o

uso de codificação para controle de erros adiciona complexidade ao sistema, especialmente para a implementação de operações de decodificação no receptor. Dessa forma, as compensações de projeto no uso da codificação para controle de erros para obter um desempenho de erro aceitável incluem considerações quanto à largura de banda e complexidade do sistema. (HAYKIN, 2004)

A detecção de erro pode ser simplificada respondendo a seguinte questão: a mensagem recebida é uma palavra-código ou não? Se a resposta é sim, então assumimos que não ocorreram erros. A probabilidade de um erro não ter sido detectado é então a probabilidade de suficientes erros terem ocorrido, transformando a palavra-código realmente transmitida em outra, aparentemente correta, mas na realidade sendo falsa. Se, por outro lado, um erro é detectado, ele pode ser corrigido a princípio por alguns métodos. Neste trabalho iremos nos ater à decodificação de síndrome para a decodificação de canal, que será explicada no próximo capítulo.

3. CODIFICAÇÃO DE CANAL

Há muitos códigos de correção de erros diferentes que podemos usar. Historicamente, esses códigos foram classificados como códigos de bloco e códigos convolucionais. Neste trabalho iremos nos ater aos códigos de bloco, particularmente aos códigos de bloco chamados de lexicográficos.

3.1 Códigos de bloco

Para gerar um código de bloco *(n,k)*, o codificador de canal aceita informações em blocos sucessivos de *k bits*; para cada bloco, ele adiciona *n – k bits* redundantes que se relacionam algebricamente com os *k bits* de mensagem, produzindo assim um bloco codificado geral de *n bits*, onde *n > k.* O bloco de *n bits* denomina-se palavra-código e *n*, tamanho de bloco do código. O codificador de canal produz *bits* a uma taxa

$$R_0 = \left(\frac{n}{k}\right) R_s , \tag{3.1}$$

onde R_s é a taxa de *bits* da fonte de informação. A relação adimensional

$$R = \frac{k}{n} , \tag{3.2}$$

denomina-se taxa de código onde *0 < R < 1*. A taxa de *bits* R_0 que sai do codificador denomina-se taxa de dados do canal. Dessa forma, a taxa de código é uma relação adimensional, ao passo que a taxa de dados produzida pela fonte e a taxa de dados de canal são medidas em *bits* por segundo. (HAYKIN, 2004)

3.2 Definições fundamentais

Consideremos F_2^n um espaço vetorial *n*-dimensional linear de todas as *n*-uplas binárias sobre o campo de *Galois GF(2)*. Um código de bloco binário *C* é qualquer

subconjunto *C* de *M* vetores em F_2^n. Identifica-se esse código como *(n,M),* para marcar que ele é definido sobre o espaço *n*-dimensional F_2^n e que o tamanho deste é de *M* vetores. Referimo-nos aos vetores do código *C* como palavras-código v_i, como apresentado anteriormente. Similarmente, podemos definir códigos de bloco ternários sobre o campo F_3, códigos de bloco quaternários sobre o campo F_4 e assim sucessivamente.

Códigos de bloco lineares são uma classe muito importante dos códigos de bloco binários. O código de bloco linear *C* é qualquer subespaço *k*-dimensional de F_2^n. Nós escrevemos *[n,k]* para identificar um código linear C no espaço *n*-dimensional F_2^n, que consiste de *k* palavras-código linearmente independentes, que podem gerar um subespaço inteiro de 2^k palavras-código. O parâmetro *k* é chamado dimensão, então algumas vezes nos referiremos ao código *[n,k]* como código de dimensão *k* e tamanho *n*. Essas *k* palavras-código independentes são organizadas em uma matriz *k x n* chamada de matriz geradora G para o código C, onde

$$v = uG. \tag{3.3}$$

Dizemos que a matriz geradora está na forma sistemática se ela é escrita como

$$G = [I_k \mid P], \tag{3.4}$$

onde I_k é a matriz identidade *k x k.* Estando G na forma sistemática, as primeiras *k* posições da palavra-código *v* são os *bits* de informação *u*; os restantes *n – k bits* são os *bits* de paridade, ou seja, a redundância adicionada a *u* com a finalidade de recuperar *u* se erros ocorrerem. (HUFFMAN, PLESS, 2003).

Outra matriz importante para os códigos lineares é a matriz de paridade *H*, usada principalmente na decodificação de canal, tal que

$$v.H^T = 0, \forall v \in C, \tag{3.5}$$

onde *v* é uma palavra-código que pertence ao código *C* e *0* é a matriz nula *k x n – k.* Em outras palavras, *H* consiste na escolha de *n – k* vetores linearmente independentes tal que os vetores do subespaço de H sejam ortogonais aos vetores do subespaço de *G*, ou seja:

$$GH^T = 0. \tag{3.6}$$

A matriz de paridade é uma matriz de dimensão *(n - k) x n*. Dizemos que a matriz de paridade está na forma sistemática se ela é escrita como se segue:

$$H = [P^T \mid I_{n-k}], \tag{3.7}$$

onde I_{n-k} é a matriz identidade *(n – k) x (n – k)*.

A distância de *Hamming d(x,y)* entre dois vetores $x, y \in F_2^n$ é definida como o número de coordenadas nas quais os dois vetores *x* e *y* diferem. A distância de *Hamming* é uma boa métrica, já que satisfaz a inequação triangular e tem um importante papel na análise dos códigos. Exemplo: d(1100,1000) = 1.

O peso de *Hamming wt(x)* de um vetor $x \in F_2^n$ é o número das suas coordenadas diferentes de zero. Exemplo: wt(1001) = 2.

A distância mínima *d* de um código *C* é a menor distância de *Hamming* entre quaisquer duas palavras-código diferentes. A distância mínima de um código *C* é também definida como o peso da palavra-código de menor peso. Se essa distância é conhecida, usamos a notação *[n,k,d]* para indicar um código linear de tamanho *n*, dimensão *k* e distância mínima *d*.

3.3 *Cosets* e decodificação de síndrome

Os termos elaborados nessa seção são usados para explicar um algoritmo geral para a decodificação dos códigos de bloco lineares, chamada de decodificação de síndrome.

Na decodificação de síndrome os *cosets* do código *C* são muito importantes. Podemos definir *coset* da seguinte forma: dado *C*, que é representado pelas palavras-código, e um vetor *a*, tal que $a \in F_2^n$, o conjunto

$$a + C = \{a + x \mid x \in C\} \tag{3.8}$$

é chamado de *coset* de C. Dois vetores *a* e *b* estão no mesmo *coset* se $a + b \in C$. Existem 2^k elementos em cada *coset* e $2^{(n-k)}$ *cosets* que dividem todo o espaço vetorial *n*-dimensional F_2^n. O código C é também um *coset*, presente na primeira linha da tabela de *cosets*.

Com os *cosets*, formaremos um arranjo padrão. A finalidade é prover um modo de decodificar os códigos de bloco. Se a palavra recebida *r* pertence ao subconjunto associado à palavra-código v_i, dizemos que v_i foi transmitida. Um arranjo padrão é construído da seguinte maneira: na primeira linha são colocadas as 2^k palavras-código, sendo que na primeira posição (coluna) é colocado o vetor nulo. Sob o vetor nulo é colocado um vetor padrão de erro e_2. Os restantes dos elementos da segunda linha são obtidos fazendo-se a soma de e_2 com v_i, que são os elementos da primeira linha. Sob o vetor e_2 é colocado um vetor e_3, ainda não usado, e o restante dos elementos são obtidos fazendo $e_3 + v_i$ e assim por diante. O processo finaliza quando tivermos usado 2^{n-k} vetores possíveis, conforme o arranjo abaixo. (GIRAUDO, 2001)

Tabela 3.1: Arranjo padrão dos *cosets* (GIRAUDO, 2001).

$v_1 = 0$	v_2	.....	v_2^k
e_2	$e_2 + v_2$	.....	$e_2 + v_2^k$
e_3	$e_3 + v_2$	.....	$e_3 + v_2^k$
...	...	.....	...
e_2^{n-k}	$e_2^{n-k} + v_2$	.....	$e_2^{n-k} + v_2^k$

No lado do receptor é recebido o vetor $r = v + e$, onde r e $e \in$ a F_2^n. De acordo com a definição de *coset*, podemos concluir que tanto *e* quanto *r* pertencem ao mesmo *coset*. A estratégia da decodificação de síndrome é encontrar o vetor de menor peso *e,* que está no mesmo *coset* com o vetor *r,* e fazer a decodificação. O vetor de menor peso em um *coset* é chamado de líder de *coset*, que será único se o peso do *coset* for

$$wt(e) \leq \lfloor (d-1)/2 \rfloor . \tag{3.9}$$

A síndrome *s* de um vetor $r \in F_2^n$ com respeito ao código *[n,k,d]* é um vetor no subespaço F_2^{n-k}, definida como

$$s = r.H^T . \tag{3.10}$$

Se o vetor *r* é uma palavra-código, então a sua síndrome é zero. Pode-se então afirmar que todo vetor no *coset* do código *C* tem síndrome igual a zero. Se dois vetores *e* e *r* estão no mesmo *coset*, então $r = v + e$, onde *v* é uma palavra-código. Usando a definição anterior,

$$s = r.H^T \rightarrow s = (v+e).H^T \rightarrow s = v.H^T + e.H^T \text{. Como } v.H^T = 0 \text{, então } s = e.H^T \quad (3.11)$$

Podemos afirmar que cada *coset* é caracterizado por três parâmetros: líder de *coset*, peso do *coset* e síndrome. O algoritmo de decodificação, portanto, pode ser definido usando o arranjo padrão ou a tabela de síndrome. Depois que um vetor *r* é recebido, computa-se, por exemplo, a síndrome para esse vetor, procura-o na tabela de *coset* e encontra-se o vetor de erro *e*. Adiciona-o ao vetor *r* para proceder à correção e encontra-se o vetor *v_estimado*, comparando-o com a palavra-código *v* enviada, para saber se houve algum erro na transmissão.

Seja t um inteiro positivo tal que

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor, \quad (3.12)$$

um código de bloco linear corrige todos os padrões até *t* erros (GIRAUDO, 2001).

4. CÓDIGOS LEXICOGRÁFICOS

Existem diversas maneiras de gerar um código de bloco. No nosso trabalho iremos abordar os códigos de bloco lexicográficos, que têm nos códigos *greedy* sua generalização. Os códigos *greedy* têm sido estudados há anos. Levenshtein (1960) provou que os códigos *greedy* produzidos de um ordenamento lexicográfico de vetores binários são lineares (o ordenamento lexicográfico padrão do conjunto de vetores binários de tamanho n é o ordenamento natural, no qual x aparece antes de w no ordenamento quando $x_i = 0$ e $w_i = 1$, onde i é a primeira posição na qual x e w diferem). Sua prova também é válida para um tipo de ordenamento chamado de ordenamento *B*, estudado primeiramente por Brualdi e Pless (1993), que é relacionado ao ordenamento lexicográfico.

Independentemente, Conway e Sloane (1986) provaram que os códigos *greedy* produzidos do ordenamento lexicográfico de vetores sobre um campo de ordem 2^{2^a}, onde a é um inteiro positivo, também são lineares. Sua abordagem teórica sobre alguns jogos é completamente diferente da abordagem de Levenshtein (1960), e fornece uma ligação entre os códigos *greedy* lexicográficos e alguns tipos de jogos, como os jogos de *Grundy*.

Brualdi e Pless (1993) generalizaram esses resultados no caso binário para um ordenamento mais geral que o ordenamento lexicográfico padrão, ao qual chamaram de ordenamento B. A prova de Levenshtein da linearidade dos códigos *greedy* ainda não era conhecida por eles, então Brualdi e Pless provaram essa linearidade como um resultado imediato do teorema do homomorfismo.

Tem grande importância o estudo dos algoritmos *greedy*, generalização dos algoritmos lexicográficos, pois eles também produzem códigos muito bons, sendo esses códigos lineares. O algoritmo *greedy* é usado para problemas de otimização e tipicamente executa uma seqüência de passos, com um conjunto de escolhas a cada etapa. Ele sempre escolhe a melhor opção para o momento, tomando uma solução ótima local e esperando que esta o leve a uma solução ótima global. A escolha feita por esse algoritmo pode depender das escolhas anteriores, mas não depende das futuras.

Os códigos *greedy* são códigos gerados pela aplicação de um algoritmo *greedy* em vetores arranjados em um ordenamento chamado de ordenamento *B*. Para cada vetor é

assinalado um único inteiro não nulo chamado de valor-g. Todos os possíveis vetores com um dado tamanho são organizados em uma tabela de acordo com seus valores-g. A aplicação dos algoritmos *greedy* requer certo tipo de ordenamento nas escolhas, sendo que aplicaremos o algoritmo na escolha de arranjos de acordo com o ordenamento *B*.

Quando o primeiro vetor em uma lista tendo uma dada propriedade é selecionado, o algoritmo continua recursivamente, até que a lista termine. O problema é maximizar o número de vetores em um código. O uso do algoritmo *greedy* para gerar códigos é especialmente atrativo devido à sua forma natural de produzir códigos com uma dada distância mínima. Os vetores são arranjados em uma ordem e cada vetor é analisado de forma a ser aceito ou rejeitado de acordo com sua distância aos outros vetores já escolhidos. O ordenamento escolhido para a geração dos códigos *greedy* é muito importante. É fácil construir ordenamentos que não produzam códigos lineares ou que produzam códigos relativamente ruins. Um tipo de ordenamento que sempre produz códigos lineares é exatamente o ordenamento B.

Os códigos *greedy* em Conway e Sloane (1986) têm as *n*-uplas binárias listadas em uma ordem lexicográfica. Em particular os *lexicodes* são códigos lineares. Para

$$n = 2^r - 1, \tag{4.1}$$

onde r é o número de *bits* redundantes e $d = 3$ os *lexicodes* são os códigos de *Hamming* e para $n = 23$ e $d = 7$ são os códigos de *Golay* binários. Para o desenvolvimento do algoritmo *greedy* usaremos algumas notações. Usamos $\oplus$ para designar a adição de vetores binários. Então, para os inteiros *a* e *b*, a $\oplus$ b é a soma entre eles como vetores binários. Por exemplo, 12 $\oplus$ 5 = 9, já que (1,1,0,0) $\oplus$ (0,1,0,1) = (1,0,0,1). Note-se que para os inteiros *a* e *b*, $a < b$ é equivalente à declaração de que *a* vem antes de *b* na ordem lexicográfica de seus valores na base 2.

4.1 Ordenamento lexicográfico padrão

Seja B uma base ordenada $b_1,b_2,...,b_n$ de F_2^n. B induz um ordenamento de vetores de F_2^n definido recursivamente como se segue: seja $V_0 = \{(0,0,...,0)\}$ e $V_i = \langle b_1,...,b_i \rangle$, onde i = 1,2,...,n, o subespaço de F_2^n relacionado aos vetores $\{ b_1,b_2,...,b_i\}$. Supondo que os vetores em V_{i-1} foram ordenados como $x_1,x_2,...,x_m$ ($m = 2^{i-1}$), então temos a distribuição

$$V_i = V_{i-1} \bigcup (b_i \oplus V_{i-1}), \tag{4.2}$$

e ordenamos os vetores em V_i colocando os vetores $x_1,x_2,...,x_m$ seguidos pelos vetores $b_i \oplus x_1$, $b_i \oplus x_2$,..., $b_i \oplus x_m$.

Já que $V_n = F_2^n$, isso define um ordenamento para os vetores de F_2^n que se chama ordenamento induzido por B, ou, como já definimos, ordenamento B de F_2^n (BRUALDI, PLESS, 1993). Para n = 3, o ordenamento B de F_2^3 é

$$0, b_1, b_2, b_2 \oplus b_1, b_3, b_3 \oplus b_1, b_3 \oplus b_2, b_3 \oplus b_2 \oplus b_1, \tag{4.3}$$

onde b_1, b_2, b_3 fazem parte da base ordenada escolhida. Neste trabalho utilizaremos três bases ordenadas diferentes, que geram três tipos de códigos lexicográficos também diferentes: a base padrão, a base *Gray* e a base complementar (BRUALDI, PLESS, 1993). Uma quarta base definida por nós será também proposta, a qual chamaremos de base alternada.

Trabalharemos primeiramente com a base ordenada padrão, que é definida como b_1 = (0,...,0,1), b_2 = (0,...,1,0),..., b_n = (1,0,...,0). Nesse caso, o ordenamento B de F^n é chamado de ordem lexicográfica padrão das *n*-uplas binárias. Seja B uma base ordenada de F_2^n e *d* um inteiro tal que $0 \le d \le n$, aplicando o algoritmo *greedy* para um ordenamento B de F_2^n nós obtemos um código $C = C(B,d)$ que tem distância mínima pelo menos *d*. Esse código é o código *greedy* lexicográfico padrão de tamanho *n* e distância *d,* ou também chamado *lexicode* padrão. O código lexicográfico padrão é, portanto, um caso especial dos códigos *greedy* B, formado por uma base ordenada lexicográfica padrão. Tomando outras bases ordenadas diferentes, podemos obter outros tipos de código, que podem ter características lexicográficas ou não.

Supondo o código de tamanho $n = 5$ e distância $d = 3$, a base ordenada lexicográfica padrão será, portanto, $b_1 = (0,0,0,0,1)$, $b_2 = (0,0,0,1,0)$, $b_3 = (0,0,1,0,0)$, $b_4 = (0,1,0,0,0)$, $b_5 = (1,0,0,0,0)$. Para gerarmos o ordenamento B de F_2^5 e chegarmos às palavras-código de *C* devemos seguir o seguinte algoritmo:

i) Escolhe-se o primeiro vetor da ordem B como o vetor nulo;

ii) Os próximos serão $b_1, b_2, b_1 \oplus b_2, b_3, b_3 \oplus b_1, b_3 \oplus b_2, b_3 \oplus b_2 \oplus b_1, b_{4,\ldots\ldots}$ em que os primeiros 2^{i-1} vetores da ordem são gerados usando os elementos da base B, $b_1, b_2, ..., b_{i-1}$ e os próximos 2^{i-1} vetores são gerados adicionando b_i àqueles vetores já produzidos em ordem. (AHMAD, 2005);

iii) Dada uma distância mínima *d*, escolhe-se um conjunto de vetores com o vetor zero sendo o primeiro. Então, procura-se através dos vetores no ordenamento *B* e escolhe-se o próximo que tem distância *d* ou mais para todos os vetores já escolhidos anteriormente;

iv) Repete-se iii) até encontrar todas as palavras-código.

Códigos construídos por esse algoritmo se mostram ótimos ou quase ótimos, pois têm dimensões *k* sendo as maiores possíveis para um dado *n* e *dmin*, ou entre as maiores possíveis. Outra característica é que esses códigos satisfazem o limite de *Varshamove-Gilbert* (ACOSTA, VIEIRA, MARTINEZ, 2001), que diz que dado um *n* e um *d*, existe um código linear C de tamanho *n* e distância maior que ou igual a *d* tal que

$$|C| \geq \frac{2^{n-1}}{\binom{n-1}{0} + \binom{n-1}{1} + \ldots + \binom{n-1}{d-2}}. \tag{4.4}$$

Uma forma mais dinâmica de organizar os vetores do ordenamento *B* e, conseqüentemente, encontrar as palavras-código de *C* é usar o que chamamos de valor-g. Associamos a cada vetor do ordenamento *B* um valor-g, que é um inteiro não negativo. Eles podem também ser escritos como vetores binários representando esses inteiros. Usaremos a seguinte forma recursiva para acharmos os valores-g: cada vetor é considerado em ordem e o primeiro vetor, que é o vetor nulo, tem assinalado 0 como seu valor-g, ou seja, $g(0) = 0$. Então, se z_n é um vetor em consideração e *x* o conjunto de vetores no ordenamento *B*, então o valor-g de z_n é o menor inteiro *t* tal que $d(z_n,x) \geq d$ para todos os vetores *x* que satisfazem

$g(x) = t$. Se não existe tal t, é definido $g(z_n)$ como o menor inteiro que não esteja em $g(z_1,z_2,...,z_{n-1})$ (BRUALDI, PLESS, 1993). Em outras palavras, seja o vetor z_n, deve-se computar a distância dele para todos os que estão acima dele. Achamos então todos os que têm distância menor que d para z_n e guardamos os valores-g deles em uma lista. O $g(z_n)$ será o menor inteiro que não está nessa lista (AHMAD, 2005). Como exemplo, para $n = 5$ e $d = 3$, apresenta-se a Tabela 4.1, com o ordenamento B e respectivos valores-g:

Tabela 4.1: ordenamento e valores-g partindo da base ordenada lexicográfica padrão.

Ordem lexicográfica padrão de F_2^5	Lista	Valor-g do vetor	Palavra-código?
00000	**-**	**0**	**Sim**
00001	{0}	1	
00010	{0,1}	2	
00011	{0,1,2}	3	
00100	{0,1,2}	3	
00101	{0,1,3,3}	2	
00110	{0,2,3,3,2}	1	
00111	**{1,2,3,3,2,1}**	**0**	**Sim**
01000	{0,1,2,3}	4	
01001	{0,1,3,2,4}	5	
01010	{0,2,3,1,4,5}	6	
01011	{1,2,3,0,4,5,6}	7	
01100	{0,3,2,1,4,5,6}	7	
01101	{1,3,2,0,4,5,7,7}	6	
01110	{2,3,1,0,4,6,7,7,6}	5	
01111	{3,2,1,0,5,6,7,7,6,5}	4	
10000	{0,1,2,3,4}	5	
10001	{0,1,3,2,5,5}	4	
10010	{0,2,3,1,6,5,4}	7	
10011	{1,2,3,0,7,5,4,7}	6	
10100	{0,3,2,1,7,5,4,7}	6	
10101	{1,3,2,0,6,5,4,6,6}	7	
10110	{2,3,1,0,5,5,7,6,6,7}	4	
10111	{3,2,1,0,4,4,7,6,6,7,4}	5	
11000	{0,4,5,6,7,5,4,7,6}	1	
11001	**{1,4,5,7,6,5,4,6,7,1}**	**0**	**Sim**
11010	{2,4,6,7,5,5,7,6,4,1,0}	3	
11011	{3,5,6,7,4,4,7,6,5,1,0}	2	
11100	{3,4,7,6,5,5,6,7,4,1,0,3}	2	
11101	{2,5,7,6,4,4,6,7,5,1,0,2,2}	3	
11110	**{1,6,7,5,4,7,6,4,5,1,3,2,2,3}**	**0**	**Sim**
11111	{0,7,6,5,4,6,7,4,5,0,3,2,2,3,0}	1	

As palavras-código a partir desse algoritmo são os vetores que têm seus valores-g iguais a zero. Então, a partir da tabela acima podemos chegar à conclusão que o código lexicográfico gerado da base ordenada lexicográfica padrão de tamanho $n = 5$ e distância mínima $d = 3$ é $C = \{(0,0,0,0,0),(0,0,1,1,1),(1,1,0,0,1),(1,1,1,1,0)\}$.

Para diferentes valores de *n, k* e *d*, Ahmad (2005) assinalou algumas propriedades dos valores-g:

i) Se uma palavra *u* tem o valor-g igual a *i*, então G_i consiste de todas as palavras com valor-g igual a *i*. Cada palavra *u* com valor-g igual a 3 está em G_3.

ii) Se $u \oplus v$ está em C, ou seja, é uma palavra-código, então *u* e *v* têm o mesmo valor-g.

iii) Se $u \oplus v$ não está em C, então *u* e *v* têm diferentes valores-g.

iv) O número de palavras em cada G_i é o mesmo número de palavras-código.

v) Se C tem dimensão *k*, então existem exatamente 2^{n-k} diferentes valores-g. Por exemplo, no caso do código que tem $n = 7$, $k = 4$ e $d = 3$, o número de diferentes valores-g = $2^{7-4} = 8$.

vi) $g(u \oplus v) = g(u) + g(v)$ na expansão binária dos valores-g.

vii) $C \oplus u$ é o conjunto de palavras com valor-g igual ao valor-g de *u*.

Consideremos que *m* é o menor inteiro tal que $g(z) \leq 2^m - 1$ para todo *z* em F_2^n. Chama-se *m* de índice de uma dada ordem de F_2^m relativo a uma dada distância *d*. Cada inteiro *g(z)* pode ser considerado um vetor em F_2^m tomando seu valor na base 2 e então incluindo 0´s à esquerda, se necessário, até o tamanho *m*. Por isso, podemos também considerar *g* como um mapeamento $g: F_2^n \rightarrow F_2^m$, ou seja, transformamos os valores-g inteiros obtidos através de um ordenamento *B* em vetores binários. Nesse caso, já que a ordem natural dos inteiros é a mesma que a ordem lexicográfica dos seus vetores correspondentes em F_2^m, a definição de *g* ser o menor significa ser o menor lexicograficamente.

Pode ser conferido que o conjunto de todos os vetores com um valor-g igual é um *coset* do código *C*, como será provado no Teorema 4.2. Isso significa que o mapeamento *g:* $F_2^5 \rightarrow F_2^3$ é um homomorfismo (em álgebra, um homomorfismo é uma aplicação que preserva uma dada estrutura) com núcleo igual a *C*. Por isso, a matriz de paridade 3 x 5 será

$$H = [g(e_5)g(e_4)g(e_3)g(e_2)g(e_1)] \text{ ,} \tag{4.5}$$

e o valor-g de um vetor em F_2^5 é a sua síndrome relativa a essa matriz (BRUALDI, PLESS, 1993). Uma característica importante do ordenamento lexicográfico padrão é que, encontrada a matriz H para o *lexicode* padrão C de tamanho *n* e distância mínima *d* ímpar, então a matriz de paridade para o *lexicode* padrão C' de tamanho *n + 1* e distância mínima *d + 1* é

$$H' = \begin{bmatrix} & & 0 \\ & & . \\ & H & . \\ & & 0 \\ \delta_1 \delta_2 \ldots \delta_n & \ldots & 1 \end{bmatrix} \text{ ,} \tag{4.6}$$

onde os δ_i são escolhidos tal que cada coluna de H' contém um número par de 1's (BRUALDI, PLESS, 1993).

A dimensão do código C é encontrada através da fórmula

$$k = n - \log_2(\max(g) + 1) \text{ ,} \tag{4.7}$$

onde *max(g)* é o maior inteiro atribuído a *g*. Vale salientar que os valores-g na matriz H são tratados como vetores-coluna, nos quais as coordenadas menores correspondem às menores potências de 2. No nosso exemplo, sendo e_5 = (1,0,0,0,0), e_4 = (0,1,0,0,0), e_3 = (0,0,1,0,0), e_2 = (0,0,0,1,0) e e_1 = (0,0,0,0,1) e verificando a Tabela 4.1, chegamos aos valores de $g(e_5) = 5$, $g(e_4) = 4$, $g(e_3) = 3$, $g(e_2) = 2$, $g(e_1) = 1$. Transformando esses valores inteiros para os valores binários de tamanho *m*, observa-se que a matriz *H* é igual a

$$H = \begin{bmatrix} 11000 \\ 00110 \\ 10101 \end{bmatrix}. \tag{4.8}$$

A matriz de paridade *H* dos códigos *greedy* tem a característica de não poder ser reduzida à forma sistemática. Devido a isso, não é trivial gerar a matriz G. Por essa razão, neste trabalho se escolheu a geração da matriz G segundo Monroe (1994). Para ele, dadas as palavras-código de $C = \{c_0,...,c_j,...\}$, que são o conjunto de vetores com valores-g iguais a 0, gerados pelo algoritmo *greedy*, então podemos afirmar que uma matriz G pode ser formada tal que são escolhidos para suas linhas os vetores c_j, onde j é uma potência de 2. Esses vetores formam uma base para o código e qualquer vetor neste pode ser expresso como a soma desses elementos-base. No nosso exemplo, como temos C = {(0,0,0,0,0),(0,0,1,1,1),(1,1,0,0,1),(1,1,1,1,0)}., a matriz G será formada por *k* palavras-código, onde $k = 2$, que serão as que estão na posição 2^0 e 2^1, considerando a posição do vetor nulo sendo a posição 0, ou seja,

$$G = \begin{bmatrix} 00111 \\ 11001 \end{bmatrix}. \tag{4.9}$$

A prova de que esses vetores formam a base para o código *C* e que *C* é um código linear é dada pelo Lema 4.1 e o Teorema 4.1.

Lema 4.1*: Sejam u, v e w vetores binários. Se* $u < v \oplus w$ *e* $v < u \oplus w$*, então* $u \oplus v < w$*. (MONROE, 1994). Obs.: dizer que u < v significa que u vem antes de v em um dado ordenamento.*

Demonstração: Primeiro consideraremos o ordenamento lexicográfico padrão. Seja $u <_{Lex} v \oplus w$ e $v <_{Lex} u \oplus w$. Supomos que *u* e $v \oplus w$ diferem a primeira vez na posição *x* e chamemos os valores das primeiras $x - 1$ posições de *u* e $v \oplus w$ de *U*. Então, o valor da posição *x* de *u* será 0 e o valor da posição *x* de $v \oplus w$ será 1. Sejam os valores das $x - 1$ primeiras posições de $(v \oplus w) \oplus w = v$ chamados de *V*. Os valores das primeiras $x - 1$ posições de $u \oplus w$ também serão *V*, já que $v \oplus w$ e *u* não diferem nessas posições. O valor da posição *x* de *v* será 0 e o da posição *x* de $u \oplus w$ será 1, já que $v \oplus w$ e *u* diferem na posição *x* e $v <_{Lex} u \oplus w$. Então, $u \oplus v$ terá os valores das suas primeiras $x - 1$ posições iguais a $U \oplus V$ e 0 na posição *x*. Mas $w = u \oplus (u \oplus w)$, então isso força os valores das primeiras $x - 1$ posições de *w* serem $U \oplus V$ e a posição *x* ser 1. Isso, portanto, implica que $u \oplus v <_{Lex} w$.

Teorema 4.1*: Seja B o ordenamento B de vetores de tamanho n e seja d uma distância escolhida. Seja C = {c0,...,cj,...} o conjunto de vetores com valores-g iguais a zero, gerados pelo algoritmo greedy, na ordem que foram escolhidos. Então C é um código linear (MONROE, 1994).*

Demonstração: Para provar a linearidade devemos mostrar que os c_j, onde j é uma potência de 2, formam uma base para o código C e que cada vetor no código pode ser expresso como a soma desses elementos base. A prova é por indução de k, onde k é a dimensão do código gerado. O vetor 2^k é escolhido e nós mostraremos que os próximos 2^k-1 vetores são gerados em ordem adicionando-se o vetor 2^k a cada vetor previamente escolhido na ordem em que estão. Assume-se que isso é verdade para os primeiros 2^k vetores gerados. Então $\{c_0,...,c_{2^k-1}\}$ é um código linear tendo como base $\{c_{2^0},...,c_{2^{(k-1)}}\}$. Isso leva a:

$$c_{2^k+r} = c_{2^k} \oplus c_r \text{, para } 0 \leq r < 2^k . \tag{4.10}$$

Para r = 0, isso é trivial. Assumindo que $r > 0$, e que r é o primeiro inteiro tal que:

$$c_{2^k+r} \neq c_{2^k} \oplus c_r , \tag{4.11}$$

então

$$d(c_{2^k+r} \oplus c_{2^k}, c_j) = d(c_{2^k+r}, c_{2^k} \oplus c_j) = d(c_{2^k+r}, c_{2^k+j}) \geq d \text{, para } 0 \leq j \leq r, \tag{4.12}$$

assumindo o fato que c_{2^k+r} é escolhido para o código, ele deve ter distância d ou mais para os outros elementos do código. Isso significa que:

$$c_r < c_{2^k+r} \oplus c_{2^k} \quad , \tag{4.13}$$

já que pela Equação 4.11, $c_{2^k+r} \oplus c_{2^k}$ não é o r-ésimo vetor escolhido.

Similarmente,

$$d(c_{2^k+r} \oplus c_r, c_j) = d(c_{2^k+r}, c_r \oplus c_j) = d(c_{2^k+r}, c_r \oplus j) \geq d \text{, para } 0 \leq j \leq 2^k - 1, \tag{4.14}$$

o que implica

$$c_{2^k} < c_{2^k+r} \oplus c_r . \tag{4.15}$$

Finalmente,

$$d(c_{2^k} \oplus c_r, c_j) = d(c_{2^k} \oplus c_j, c_r) \geq d \text{, para } 2^k \leq j \leq 2^k + r, \tag{4.16}$$

$$d(c_{2^k} \oplus c_r, c_j) = d(c_{2^k}, c_r \oplus j) \geq d \text{, para } j < 2^k \text{ .} \quad (4.17)$$

Isso significa que:

$$c_{2^k+r} < c_{2^k} \oplus c_r \text{ .} \quad (4.18)$$

Tomadas juntas, as equações 4.13, 4.15 e 4.18 contradizem o Lema 4.1. Então confirmamos a Equação 4.10 e o teorema é provado. ■

O Teorema 4.2 implica que os valores-g dos vetores determinam os *cosets* do código e vice-versa.

Teorema 4.2: *Seja v um vetor que é o primeiro a ocorrer no ordenamento B com um dado valor-g igual a g e seja C = {c_0,...,c_j,...}. Então {$v \oplus c_0$,...,$v \oplus c_j$,...} é a lista completa dos vetores com valor g igual a g e $v + c_i < v \oplus c_j$ quando i < j. (MONROE, 1994)*

Demonstração: O teorema é verdadeiro para o conjunto de vetores com valor-g igual a zero, já que esses são vetores do código e *v* será o vetor nulo. Assuma que o teorema é válido para todos os valores-g que são menores que *g* e seja *v* o vetor com valor-g igual a *g* que ocorre primeiro no ordenamento *B*. É necessário que provemos duas declarações:

i) $v \oplus c_i$ é o *i*-ésimo vetor na ordem com valor-g igual a *g* (começando com *i* = 0);

ii) Todos os vetores que têm valor-g igual a *g* são da forma $v \oplus c_j$, para algum *j*.

Uma vez provadas, i) e ii) são suficientes para provar o teorema para o valor-g igual a *g*.

Prova da declaração i): por indução em *i,* $v \oplus c_0$ é certamente o 0-ésimo vetor no ordenamento *B* com valor-g igual a *g*, já que c_0 é o vetor nulo. Seja $v \oplus c_j$ o *j*-ésimo vetor na ordem com valor-g igual a *g* para *j* < *i*. Devemos mostrar que $v \oplus c_i$ é o *i*-ésimo vetor na ordem com valor-g igual a *g*. Se o valor-g de $v \oplus c_i$ é *h* < *g*, então a hipótese de indução dos valores-g implica que os vetores com valor-g igual a *h* formam um *coset*, então o valor-g de $v \oplus c_i \oplus c_i$ também é *h*. Mas $v \oplus c_i \oplus c_i = v$, que já tem o valor-g igual a *g*, o que nos leva a uma contradição. Então, o valor-g de $v \oplus c_i$ não é menor que *g*. Observemos que

$$d(v \oplus c_i, v \oplus c_j) = d(v, v \oplus c_i \oplus c_j) = wt(c_i \oplus c_j) \geq d \text{ ,} \quad (4.19)$$

então $v \oplus c_i$ se diferencia de todos os vetores previamente escolhidos com valor-g igual a *g* por pelo menos a distância *d*. Isso significa que a única forma de $v \oplus c_i$ não ser o *i*-ésimo vetor com valor-g igual a *g* é se existisse um vetor entre $v \oplus c_{i-1}$ e $v \oplus c_i$ tendo valor-g igual a *g*. Assuma que tal vetor existe e pode ser expresso como $v \oplus u$, para algum u e

$$v \oplus u < v \oplus c_i \ . \tag{4.20}$$

Agora,

$$d(c_j, u) = d(v \oplus c_j, v \oplus u) \geq d \text{, para todo j < i.} \tag{4.21}$$

Mas u não é o e-ésimo escolhido para estar no código, então $c_i < u$. Conseqüentemente,

$$c_i < (v \oplus u) \oplus v. \tag{4.22}$$

As Equações 4.20 e 4.22 implicam

$$(v \oplus u) \oplus c_i < v \ , \tag{4.23}$$

pelo Lema 4.1. Isso significa que $v \oplus u \oplus c$ tem valor-g igual a $h < g$, já que se *v* é o primeiro no ordenamento *B* com valor-g igual a *g* todos que ocorrem antes dele têm valor-g menor que *g*. Mas pela hipótese de indução dos valores-g (que determina que qualquer conjunto de elementos compreendendo todos os vetores com valores-g menores que *g* são o *coset* de um código), $v \oplus u \oplus c$ tem o mesmo valor-g que $(v \oplus u \oplus c) \oplus c_i = v \oplus u$ e esse valor-g assume-se que é igual a *g*. Chegamos então a uma contradição, então a declaração i) é provada e mostra-se que $v \oplus c_i$ é o *i*-ésimo vetor na ordem com valor-g igual a *g*.

Prova da declaração ii): Seja *w* um vetor com valor-g igual a *g*. Se *w* não é da forma $v \oplus c_i$, então *w* deve ocorrer depois de $v \oplus c_i$ para todos os c_i do código, então

$$d(w \oplus v, c_i) = d(w, v \oplus c_i) \geq d \ , \tag{4.24}$$

para todos os c_i do código, já que todos os $v \oplus c_i$ têm valor-g igual a *g*. Então, $w \oplus v$ tem valor-g igual a zero, por isso está no código. Isso significa que

$$w \oplus v = c_j \ , \tag{4.25}$$

para algum j e $w = v \oplus i$. Isso prova a declaração ii) e termina a prova do Teorema 4.2.

■

Para formarmos a tabela de *coset* para o algoritmo *greedy*, que será usada na decodificação da mensagem, devemos encontrar os líderes de *coset*, como definido na Seção 3.3. No nosso algoritmo desenvolvido no *Matlab*®, a função *syndtable* encontra os líderes de *coset* de cada linha. Podemos então formar nossa tabela de *coset* utilizando essa função e os vetores que geramos anteriormente para a base ordenada lexicográfica padrão. Como provado no Teorema 4.2, para organizarmos todos os vetores nas linhas de *coset* basta colocarmos na primeira linha as palavras-código e, chamando os líderes de *coset* de e_i, preenchemos a linha somando esses e_i a cada palavra-código.

Tabela 4.2: *cosets* do código lexicográfico padrão para $n = 5$ e $d = 3$.

		Palavras-código			
Padrão de erro	**00000**	**00000**	**00111**	**11001**	**11110**
	00001	00001	00110	11000	11111
	00010	00010	00101	11011	11100
	00100	00100	00011	11101	11010
	01000	01000	01111	10001	10110
	10000	10000	10111	01001	01110
	10100	10100	10011	01101	01010
	10010	10010	10101	01011	01100
		Líder do *coset*			

Para códigos com tamanho *n* pequeno, há uma grande semelhança entre as matrizes *H* e *G* e, conseqüentemente, entre as palavras-código. Entretanto, devido à inviabilidade de apresentar tabelas para tamanhos *n* grandes, iremos nos ater nas outras seções, como exemplos, aos códigos (5,2,3). Para darmos uma noção da diferença entre os códigos com tamanhos maiores, em cada seção subseqüente apresentaremos as matrizes *H* e *G* para o código (8,4,4), que será um dos códigos utilizados no Capítulo 5 para comparação entre eles. Para o ordenamento lexicográfico padrão, teremos as matrizes *H* e *G* conforme a Equação 4.26 e Equação 4.27 para o código (8,4,4).

$$H = \begin{bmatrix} 11110000 \\ 11001100 \\ 10101010 \\ 01101001 \end{bmatrix} . \tag{4.26}$$

$$G = \begin{bmatrix} 00001111 \\ 00110011 \\ 01010101 \\ 10010110 \end{bmatrix} . \qquad (4.27)$$

4.2 Ordenamento lexicográfico Gray

Nessa seção continuaremos nossa discussão com os códigos *greedy* de base ordenada lexicográfica *Gray*, ou seja, *lexicodes* com respeito a uma base *Gra*y de F_2^n. Ela é da seguinte forma: b_1 = (0,...,0,1), b_2 = (0,...,0,1,1),..., b_n = (1,1,0,...,0). Para n = 5 e d = 3, temos a base ordenada Gray como sendo b_1 = {0,0,0,0,1}, b_2 = {0,0,0,1,1}, b_3 = {0,0,1,1,0}, b_4 = {0,1,1,0,0} e b_5 = {1,1,0,0,0}. Aplicando o algoritmo gerado no *Matlab*® encontramos os seguintes vetores e seus valores-g inteiros e binários, chegando ao mesmo resultado encontrado em Brualdi e Pless (1993).

Tabela 4.3: ordenamento e valores-g partindo da base ordenada lexicográfica *Gray*.

Ordem lexicográfica Gray de F_2^5	Valor-g inteiro	Valor-g binário	Palavra-código?
00000	**0**	**000**	**Sim**
00001	1	001	
00011	2	010	
00010	3	011	
00110	1	001	
00111	**0**	**000**	**Sim**
00101	3	011	
00100	2	010	
01100	4	100	
01101	5	101	
01111	6	110	
01110	7	111	
01010	5	101	
01011	4	100	
01001	7	111	
01000	6	110	
11000	1	001	
11001	**0**	**000**	**Sim**
11011	3	011	

11010	2	010	
11110	**0**	**000**	**Sim**
11111	1	001	
11101	2	010	
11100	3	011	
10100	5	101	
10101	4	100	
10111	7	111	
10110	6	110	
10010	4	100	
10011	5	101	
10001	6	110	
10000	7	111	

O código C para este ordenamento é, portanto, C = {(0,0,0,0,0),(0,0,1,1,1),(1,1,0,0,1),(1,1,1,1,0)}, o mesmo que na Seção 4.1. Isso nem sempre acontece. Para códigos com *dmin* e *n* maiores há mudanças nas palavras-código. Assim como na Seção 4.1 o valor máximo do valor-g é 7 e $n = 5$, então $k = 2$. Analisando a Tabela 4.3 podemos visualizar que, na base 2, $g(e_5) = 111$, $g(e_4) = 110$, $g(e_3) = 010$, $g(e_2) = 011$, $g(e_1) = 001$. Então,

$$H = \begin{bmatrix} 11000 \\ 11110 \\ 10011 \end{bmatrix}. \tag{4.28}$$

A matriz G também pode ser gerada escolhendo para suas linhas as palavras-código de posição 2^0 e 2^1, como visto na Seção 3.1, então,

$$G = \begin{bmatrix} 00111 \\ 11001 \end{bmatrix}. \tag{4.29}$$

E finalmente, podemos gerar a tabela de *coset* para este ordenamento.

Tabela 4.4: *cosets* do código lexicográfico *Gray* para $n = 5$ e $d = 3$.

		Palavras-código			
Padrão de erro	**00000**	**00000**	**00111**	**11001**	**11110**
	00001	00001	00110	11000	11111
	00100	00100	00011	11101	11010
	00010	00010	00101	11011	11100
	10010	10010	10101	01011	01100
	10100	10100	10011	01101	01010
	01000	01000	01111	10001	10110
	10000	10000	10111	01001	01110
		Líder do *coset*			

Para o código (8,4,4) gerado pelo ordenamento lexicográfico *Gray*, teremos as matrizes G e H conforme a Equação 4.30 e Equação 4.31, que são diferentes das geradas para o código (8,4,4) do ordenamento lexicográfico padrão, o que também ocorrerá com os outros ordenamentos apresentados.

$$H = \begin{bmatrix} 11110000 \\ 00111100 \\ 01100110 \\ 11111111 \end{bmatrix}. \tag{4.30}$$

$$G = \begin{bmatrix} 00001111 \\ 00110011 \\ 01100110 \\ 11000011 \end{bmatrix}. \tag{4.31}$$

4.3 Ordenamento lexicográfico complementar

Nessa terceira seção abordaremos os códigos *greedy* de base complementar, ou seja, *lexicodes* com respeito a uma base complementar de F_2^n. Ela é da seguinte forma: $b_1 = (0,...,0,1)$, $b_2 = (0,...,0,1,1)$,..., $b_n = (1,1,1,...,1)$. Para $n = 5$ e $d = 3$, temos a base ordenada complementar como sendo $b_1 = \{0,0,0,0,1\}$, $b_2 = \{0,0,0,1,1\}$, $b_3 = \{0,0,1,1,1\}$, $b_4 = \{0,1,1,1,1\}$ e $b_5 = \{1,1,1,1,1\}$. Aplicando o algoritmo gerado no *Matlab*® encontramos os seguintes vetores e seus valores-g inteiros e binários:

Tabela 4.5: ordenamento e valores-g partindo da base ordenada lexicográfica complementar.

Ordem lexicográfica complementar de F_2^5	Valor-g inteiro	Valor-g binário	Palavra-código?
00000	**0**	**000**	**Sim**
00001	1	001	
00011	2	010	
00010	3	011	
00111	**0**	**000**	**Sim**
00110	1	001	
00100	2	010	
00101	3	011	
01111	4	100	
01110	5	101	
01100	6	110	
01101	7	111	
01000	4	100	
01001	5	101	
01011	6	110	
01010	7	111	
11111	1	001	
11110	**0**	**000**	**Sim**
11100	3	011	
11101	2	010	
11000	1	001	
11001	**0**	**000**	**Sim**
11011	3	011	
11010	2	010	
10000	5	101	
10001	4	100	
10011	7	111	
10010	6	110	
10111	5	101	
10110	4	100	
10100	7	111	
10101	6	110	

O código C para este ordenamento é, portanto, C = {(0,0,0,0,0),(0,0,1,1,1),(1,1,1,1,0),(1,1,0,0,1)}, o mesmo que na Seção 4.1 mas com a ordem trocada. Como dito na Seção 4.2 isso nem sempre acontece. Para códigos com *dmin* e *n* maiores há mudanças nas palavras-código. Assim como na Seção 4.1 o valor máximo do valor-g é 7 e $n = 5$, então $k = 2$. Analisando a Tabela 4.5 podemos visualizar que, na base 2, $g(e_5) = 101$, $g(e_4) = 100$, $g(e_3) = 010$, $g(e_2) = 011$, $g(e_1) = 001$. Então,

$$H = \begin{bmatrix} 11000 \\ 00110 \\ 10011 \end{bmatrix}. \tag{4.32}$$

A matriz G também pode ser encontrada como visto na Seção 3.1, então,

$$G = \begin{bmatrix} 00111 \\ 11110 \end{bmatrix}. \tag{4.33}$$

Tabela 4.6: *cosets* do código lexicográfico complementar para $n = 5$ e $d = 3$.

		Palavras-código			
Padrão de erro	**00000**	**00000**	**00111**	**11110**	**11001**
	00001	00001	00110	11111	11000
	00100	00100	00011	11010	11101
	00010	00010	00101	11100	11011
	01000	01000	01111	10110	10001
	10000	10000	10111	01110	01001
	10010	10010	10101	01100	01011
	10100	10100	10011	01010	01101
		Líder do *coset*			

Para o código (8,4,4) gerado pelo ordenamento lexicográfico complementar, teremos as matrizes *G* e *H* conforme a Equação 4.34 e Equação 4.35.

$$H = \begin{bmatrix} 11110000 \\ 11001100 \\ 01100110 \\ 11000011 \end{bmatrix}. \tag{4.34}$$

$$G = \begin{bmatrix} 00001111 \\ 00111100 \\ 01100110 \\ 11111111 \end{bmatrix}. \tag{4.35}$$

4.4 Ordenamento lexicográfico alternado

Nessa quarta seção abordaremos um novo código *greedy*, com uma base definida por nós baseados na definição dada por Brualdi e Pless (1993). Chamamos essa nova base de base ordenada lexicográfica alternada, ou seja, geramos *lexicodes* com respeito a uma nova base de F_2^n. Ela é gerada da seguinte forma: b_1 = (0,...,0,1), b_2 = (0,...,0,1,0),..., b_3 = {0,...,1,0,1} ,..., b_n = (1,0,1,0,...,0). Para n = 5 e d = 3, temos a base ordenada alternada como sendo b_1 = {0,0,0,0,1}, b_2 = {0,0,0,1,0}, b_3 = {0,0,1,0,1}, b_4 = {0,1,0,1,0} e b_5 = {1,0,1,0,0}. Aplicando o algoritmo gerado no *Matlab*® encontramos os seguintes vetores e seus valores-g inteiros e binários.

Tabela 4.7: ordenamento B e valores-g partindo da base ordenada lexicográfica alternada.

Ordem lexicográfica complementar de F_2^5	Valor-g inteiro	Valor-g binário	Palavra-código?
00000	**0**	**000**	**Sim**
00001	1	001	
00010	2	010	
00011	3	011	
00101	2	010	
00100	3	011	
00111	**0**	**000**	**Sim**
00110	1	001	
01010	4	100	
01011	5	101	
01000	6	110	
01001	7	111	
01111	6	110	
01110	7	111	
01101	4	100	
01100	5	101	
10100	4	100	
10101	5	101	
10110	6	110	
10111	7	111	
10001	6	110	
10000	7	111	
10011	4	100	
10010	5	101	
11110	**0**	**000**	**Sim**
11111	1	001	

11100	2	010	
11101	3	011	
11011	2	010	
11010	3	011	
11001	**0**	**000**	**Sim**
11000	1	001	

.

O código C para este ordenamento é, portanto, $C = \{(0,0,0,0,0),(0,0,1,1,1),(1,1,1,1,0),(1,1,0,0,1)\}$, mais uma vez o mesmo que na Seção 4.1 mas com a ordem trocada. Como dito nas seções anteriores isso nem sempre acontece. Para códigos com *dmin* e *n* maiores há mudanças nas palavras-código. Assim como na Seção 4.1 o valor máximo do valor-g é 7 e $n = 5$, então $k = 2$. Analisando a Tabela 4.7 podemos visualizar que, na base 2, $g(e_5) = 111$, $g(e_4) = 110$, $g(e_3) = 011$, $g(e_2) = 010$, $g(e_1) = 001$. Então,

$$H = \begin{bmatrix} 11000 \\ 11110 \\ 10101 \end{bmatrix}. \tag{4.36}$$

A matriz G também pode ser encontrada como visto na Seção 4.1, então,

$$G = \begin{bmatrix} 00111 \\ 11110 \end{bmatrix}. \tag{4.37}$$

Por último, podemos gerar a tabela de *coset* para este ordenamento.

Tabela 4.8: *cosets* do código lexicográfico alternado para n = 5 e d = 3.

		Palavras-código			
Padrão de erro	**00000**	**00000**	**00111**	**11110**	**11001**
	00001	00001	00110	11111	11000
	00010	00010	00101	11100	11011
	00100	00100	00011	11010	11101
	10100	10100	10011	01010	01101
	10010	10010	10101	01100	01011
	01000	01000	01111	10110	10001
	10000	10000	10111	01110	01001
		Líder do *coset*			

Para o código (8,4,4) gerado pelo ordenamento lexicográfico proposto por nós, teremos as matrizes *G* e *H* conforme a Equação 4.38 e Equação 4.39.

$$H = \begin{bmatrix} 11110000 \\ 00111100 \\ 10101010 \\ 01010101 \end{bmatrix}. \tag{4.38}$$

$$G = \begin{bmatrix} 00001111 \\ 00111100 \\ 01010101 \\ 10100101 \end{bmatrix}. \tag{4.39}$$

Ao final, considerando os ordenamentos lexicográficos padrão, *Gray*, complementar e alternado, podemos apresentar uma tabela comparativa entre as dimensões destes três ordenamentos, dados alguns valores específicos de *n* e *d*. Devido à falta de processadores mais potentes, só pudemos constatar a veracidade da tabela apresentada por Brualdi e Pless (1993) até os valores de $n = 14$. Por isso, não podemos afirmar que para o ordenamento lexicográfico alternado, proposto por nós, as dimensões sejam a partir desse *n* sejam iguais às da Tabela 4.9.

Tabela 4.9: comparação entre as dimensões dos ordenamentos lexicográficos padrão, *Gray*, complementar e alternado para n < 26 (BRUALDI, PLESS, 1993).

n : *d*	4	6	8	10	12
4	1	0	0	0	0
5	1	0	0	0	0
6	2	1	0	0	0
7	3	1	0	0	0
8	4	1	1	0	0
9	4	2	1	0	0
10	5	2	1	1	0
11	6	3	1	1	0
12	7	4	2	1	1
13	8	4	2	1	1
14	9	5	3	1	1
15	10	6	4	2	1
16	11	7	5	2	1
17	11	8[7]	5	2	1
18	12	9[8]	6	3	2
19	13	9	7	3	2
20	14	10	8	4	2
21	15	11	9	5	3
22	16	12	10	5	3
23	17	13(12)	11	6	4
24	18	13	12	7(6)[6]	5
25	19	14	12	7	5

Os números entre parênteses são as dimensões do ordenamento lexicográfico padrão quando esse difere das dimensões do ordenamento Gray e os números entre colchetes são as dimensões do ordenamento complementar quando esse difere das dimensões do ordenamento Gray. O que nós podemos afirmar analisando a tabela e baseando a qualidade dos nossos códigos pela sua dimensão é que os códigos gerados pelo ordenamento Gray são pelo menos tão bons quanto os gerados pelo ordenamento lexicográfico padrão e com uma exceção são pelo menos tão bons quanto o código complementar. Já o código gerado pelo ordenamento complementar é às vezes melhor e às vezes pior que o código gerado pelo ordenamento lexicográfico padrão. Para os códigos gerados pelo ordenamento alternado até $n = 14$ não se notou nenhuma diferença de dimensão entre ele e os demais, não podendo ser afirmado que isso se repetirá para n maiores.

4.5 Ordenamento lexicográfico classificado e classificado reverso

Nessa seção usaremos uma abordagem diferente, baseada no ordenamento lexicográfico padrão, chamada de ordenamento lexicográfico classificado. O número de palavras-código geradas por esse ordenamento é geralmente maior ou igual ao número de palavras-código criadas com os mesmos parâmetros usados no ordenamento lexicográfico padrão (ACOSTA, VIEIRA, MARTINEZ, 2001).

Para o ordenamento lexicográfico classificado, dados quaisquer x e y vetores binários pertencentes a F_2^n, dizemos que $x > y$ se $wt(x) > wt(y)$, ou se para $wt(x) = wt(y)$ o valor inteiro de x é maior que o valor inteiro de y, com respeito ao número binário que cada vetor representa. Exemplo: o ordenamento lexicográfico classificado de todas as palavras pertencentes a F_2^5 é mostrado na Figura 4.1 na matriz à esquerda. Um código *greedy* para $n = 5$ e $d = 3$ criado com o algoritmo desenvolvido por nós no *Matlab*® é representado na matriz à direita.

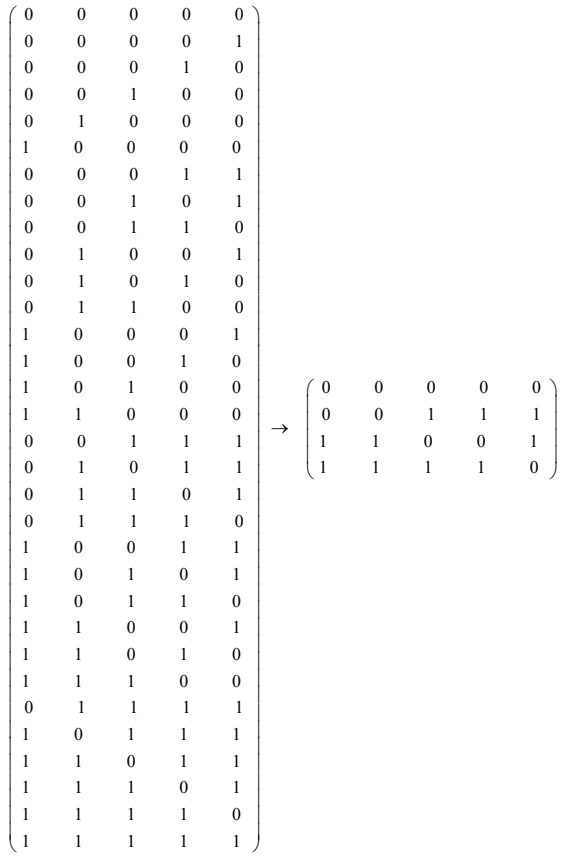


Figura 4.1: ordenamento lexicográfico classificado.

Pode-se notar que a matriz gerada com esse ordenamento é idêntica à matriz gerada pelo ordenamento lexicográfico padrão. Isso não é verdade para todos os casos, mas uma similaridade particular entre os parâmetros usados para gerar os códigos para os dois ordenamentos leva a uma conjectura interessante que será apresentada posteriormente de forma rápida.

Para o ordenamento lexicográfico classificado reverso, dado quaisquer x e y vetores binários pertencentes a F_2^n, dizemos que $x > y$ se $wt(x) > wt(y)$, ou se para $wt(x) = wt(y)$ então o resultado de $x - y$ (computados com a aritmética comum, não em módulo 2) têm o elemento não-nulo mais à direita negativo. Exemplo: o ordenamento lexicográfico classificado reverso de todas as palavras pertencentes a F_2^5 é mostrado na matriz à esquerda, da Figura 4.2. Um código *greedy* para $n = 5$ e $d = 3$ criado com o algoritmo desenvolvido por nós no *Matlab*® é representado na matriz à direita.

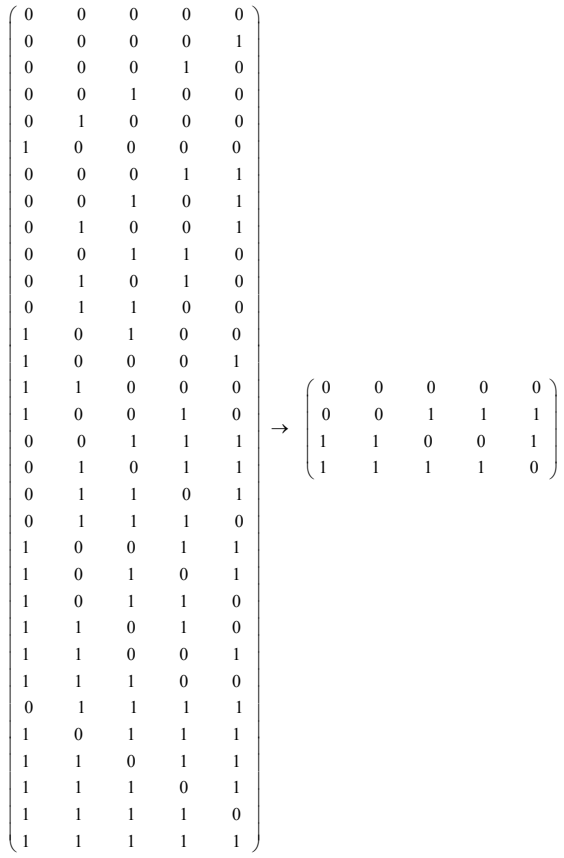


Figura 4.2: ordenamento lexicográfico classificado reverso.

Aqui, mais uma vez, a mesma matriz é apresentada, apesar de ter uma pequena mudança na forma como as palavras são escolhidas ao se criar o código. Mais uma vez esclarecemos que isso não acontece para todos os ordenamentos.

Ao examinarmos os códigos *greedy* criados com o ordenamento lexicográfico padrão e com esses dois novos ordenamentos, algumas similaridades aparecem nas palavras-código geradas separadamente em cada um. Muitos valores de n e d criam códigos que contêm exatamente o mesmo número de palavras. Mesmo quando esse é o caso, às vezes ocorre que as mesmas palavras-código são iguais entre os códigos, apesar de elas terem sido geradas num ordenamento diferente. Quando há diferenças no número de palavras que estão no código, é aparente que os ordenamentos que examinam os pesos das palavras primeiro (lexicográfico classificado e classificado reverso) tendem a produzir mais palavras-código para a maioria

dos valores de n e d, pelo menos para $n \leq 14$ (excluindo n = 12 com d = 6) e n = 16,17,18 com d = 9 (ACOSTA, VIEIRA, MARTINEZ, 2001).

A Tabela 4.10 mostra relações importantes entre os parâmetros em cada ordenamento. A coluna *CRev* representa o número de palavras-código geradas com o algoritmo *greedy* para o ordenamento lexicográfico classificado reverso. Similarmente, as colunas *Lex* e *CLex* representam o número de palavras-código para o ordenamento lexicográfico padrão e para o lexicográfico classificado, respectivamente. A coluna *SPB* representa o limite *sphere packing*, que indica o número máximo de palavras-código que um código com um dado tamanho n e distância mínima de *Hammin*g *d,* onde d = 2t + 1 ou d = 2t + 2, pode conter. Esse valor é dado pela Equação 4.40:

$$|C| \leq \frac{2^n}{\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{t}}. \tag{4.40}$$

Tabela 4.10: *CRev, Lex, GLex e SPB* para 2 < d < 5 (ACOSTA, VIEIRA, MARTINEZ, 2001).

(n,d)	*CRev*	*Lex*	*GLex*	*SPB*
(4,2)	8	8	8	16
(5,2)	16	16	16	32
(6,2)	32	32	32	64
(7,2)	64	64	64	128
(8,2)	128	128	128	256
(5,3)	4	4	4	5
(6,3)	8	8	8	9
(7,3)	16	16	16	16
(8,3)	16	16	16	28
(9,3)	32	32	32	51
(10,3)	64	64	64	93
(11,3)	128	128	128	171
(15,3)	2048	2048	2048	2048
(6,4)	4	4	4	9
(7,4)	8	8	8	16
(8,4)	16	16	16	28
(9,4)	16	16	16	52
(10,4)	32	32	32	93
(11,4)	64	64	64	171
(12,4)	128	128	128	315
(13,4)	256	256	256	585
(14,4)	512	512	512	1092
(15,4)	1024	1024	1024	2048
(16,4)	2048	2048	2048	3855

A Tabela 4.10 mostra que, para esses *n* e *d*, os códigos *greedy* gerados com esses ordenamentos têm todos os mesmos números de palavras-código e que, além disso, contêm exatamente as mesmas palavras-código (embora em ordem diferente). Isso, entretanto, não é verdade para todos os valores de *n* e *d*, como mostrado na Tabela 4.11.

Tabela 4.11: CRev, Lex, GLex e SPB para 4 < d < 10 (ACOSTA, VIEIRA, MARTINEZ, 2001).

(n,d)	CRev	Lex	GLex	SPB
(7,5)	2	2	2	4
(8,5)	4	4	4	7
(9,5)	6	4	6	11
(10,5)	12	8	12	18
(11,5)	12	16	24	31
(12,5)	25	16	24	52
(13,5)	33	32	40	89
(14,5)	54	64	67	155
(15,5)	97	128	103	271
(16,5)	164	256	164	478
(17,5)				851
(9,6)	4	4	4	11
(10,6)	6	4	4	18
(11,6)	12	8	8	31
(12,6)	12	16	12	52
(13,6)	25	16	20	89
(14,6)	34	32	33	155
(15,6)	54	64	56	271
(16,6)	97	128	95	478
(17,6)	164	256	156	851
(11,7)	4	4	4	9
(12,7)	4	4	4	14
(13,7)	8	8	8	22
(14,7)	16	16	16	35
(15,7)	32	32	32	57
(16,7)	32	32	32	94
(17,7)	64	64	64	157
(12,8)	4	4	4	14
(13,8)	4	4	4	22
(14,8)	8	8	8	35
(15,8)	16	16	16	57
(14,9)	4	4	4	11
(15,9)	4	4	4	17
(16,9)	6	4	6	26
(17,9)	8	8	10	41
(18,9)	20	8	20	65

Os resultados da Tabela 4.10 e da Tabela 4.11 revelam algumas características dos códigos apresentados. Por exemplo, o código gerado com n = 7 e d = 3 para os três ordenamentos examinados é o código de *Hamming*, que é um código perfeito, ou seja, o número de palavras-código é igual ao limite *sphere-packing*. Também pode ser confirmado que o código gerado para n = 15 e d = 3 com o ordenamento lexicográfico padrão tem essa mesma propriedade e o mesmo acontece para os outros dois ordenamentos.

Comparando o número de palavras geradas pelos algoritmos para alguns exemplos também se criam casos únicos quando n = 12 com d = 6 e quando n = 15, 16, 17 com d = 5, 6. Nesses casos o número de palavras geradas pelo ordenamento lexicográfico padrão é maior que o número de palavras nos outros dois ordenamentos. Entretanto, foi encontrado que para n = 16, 17, 18 e d = 9 os ordenamentos lexicográficos classificado e classificado reverso geram mais palavras que o ordenamento lexicográfico padrão. Também foi analisado que códigos gerados com distâncias mínimas que são potências de 2 geram exatamente as mesmas palavras-código para o ordenamento lexicográfico padrão e para o lexicográfico classificado. Além disso, as matrizes geradoras para esses dois ordenamentos, criados com distância d e que são potências de 2, consistem exclusivamente de linhas com peso d.

4.6 Códigos (2^M, M+1, 2^{M-1}), com M ≥ 3, usando blocos lexicográficos

Outra abordagem dos códigos lexicográficos foi definida por Alencar e Giraudo (2007), na qual foram gerados códigos (2^M, M+1, 2^{M-1}) usando blocos lexicográficos.

Definição 4.1: *Um bloco lexicográfico é uma matriz geradora de um código com distância mínima de Hamming $d = 2^{M-1}$, tal que, M ≥ 3, construída lexicograficamente a partir do vetor* $(0^d \mid 1^d)$ = (00...0011...11), *de d zeros e d uns, onde o peso de Hamming e a distância euclidiana entre os seus vetores linha, tomados dois a dois, são iguais a d*. (ALENCAR, GIRAUDO, 2007)

Escolheu-se como parâmetro no trabalho em questão M = 3, gerando então o código (8, 4, 4), que tem a sua matriz geradora G disposta conforme a Equação 4.41.

Essa matriz é construída a partir do vetor $(0^4 \mid 1^4)$ = (00001111), mediante a escolha de vetores linearmente independentes, considerando uma lexicografia na qual esses vetores possuem a mesma distância de Hamming, dois a dois.

Como propriedade fundamental tem-se que a distância máxima da matriz *G* é igual à sua distância mínima *d*. Além disso, essa nova construção lexicográfica permite que todos os vetores geradores da matriz *G* se encontrem, dois a dois, à mesma distância euclidiana. (ALENCAR, GIRAUDO, 2007).

$$G = \begin{bmatrix} 00001111 \\ 00110011 \\ 01010101 \\ 10011001 \end{bmatrix}. \tag{4.41}$$

Cada bloco lexicográfico que dá origem a uma matriz geradora, dá origem a outro bloco lexicográfico que é matriz geradora para outro código em seqüência. A matriz *B* da Equação 4.42 é o bloco lexicográfico que dá origem a *G*, na Equação 4.41. Nesse caso a matriz *G* na Equação 4.41 é um bloco lexicográfico, e dessa forma pode-se obter uma nova matriz geradora *N*, como definida na Equação 4.43, do código *C* (16, 5, 8). (ALENCAR, GIRAUDO, 2007).

$$B = \begin{bmatrix} 0011 \\ 0101 \\ 1001 \end{bmatrix}. \tag{4.42}$$

$$N = \begin{bmatrix} 0000000011111111 \\ 0000111100001111 \\ 0011001100110011 \\ 0101010101010101 \\ 1001100110011001 \end{bmatrix}. \tag{4.43}$$

Podemos perceber que Alencar e Giraudo (2007) chegaram a um resultado interessante. As matrizes *G* e *N* quando multiplicadas pelas suas transpostas dão como valor de retorno a matriz nula, ou seja, $GG^T = 0$ e $NN^T = 0$. Como, por definição, para códigos de bloco, $GH^T = 0$, podemos chegar à conclusão que a matriz de paridade *H* é igual à matriz geradora *G* para o código (8,4,4) gerado por essa construção. O mesmo já não pode ser afirmado para a matriz geradora N do código (16,5,8), pois apesar de $NN^T = 0$, a dimensão de N é (16,5), diferente da dimensão da matriz de paridade para esse código que deve ser (11,16).

4.7 Códigos lexicográficos desenvolvidos por Trachtenberg (2002)

Trachtenberg (2002) desenvolveu uma forma diferente de gerar os códigos lexicográficos das que eram conhecidas na literatura, a qual ele chamou de construção lexicográfica. Em suma ela é um caso especial dos algoritmos *greedy* definidos por Brualdi e Pless (1993), com a diferença que alguns limites são dados para facilitar a construção, como o raio de cobertura.

Toma-se primeiramente o conjunto $S = \{0\}$ e adiciona, até a exaustão, o vetor que é o primeiro lexicograficamente a ter a distância de *Hamming* para S pelo menos igual a d. Assim, pode-se descrever completamente um *lexicode* de dimensão k encontrando-se k vetores-base (que são chamados de geradores) usando a construção lexicográfica. K iterações dessas formam o código de dimensão k C_k^d (TRACHTENBERG, 2002). A Tabela 4.12 demonstra essa construção para o código binário com $d = 3$, $n = 7$ e $k = 4$.

Tabela 4.12: Matriz geradora para o código binário C_4^3, de dimensão 4 e distância mínima 3 (TRACHTENBERG, 2002).

0000111
0011001
0101010
1001011

Para se entender mais analiticamente a construção lexicográfica é necessário fazer uso do raio de cobertura de cada código C_i^d intermediário. O raio de cobertura de um código de tamanho n é o inteiro ρ, onde ρ é a distância máxima de qualquer vetor em F_q^n para o código. Cada iteração da construção lexicográfica de um código intermediário C que tem o raio de cobertura ρ e distância mínima d pode ser entendida como a adição de um vetor gerador:

$$\left\langle 1^{d-\rho} \mid f_{lexi}(C) \right\rangle, \tag{4.43}$$

onde $1^{d-\rho}$, conhecido como bloco gerador, corresponde a $(d - \rho)$ sucessivos 1´s, e a função $f_{\mathbf{lexi}}(C)$ retorna o primeiro vetor lexicograficamente que tem a distância ρ para C. O símbolo

$\langle . | . \rangle$ significa concatenação. Os códigos lineares gerados por essa construção são precisamente os *lexicodes* (TRACHTENBERG, 2002).

O código de tamanho $n = 8$, dimensão $k = 4$ e distância mínima de Hamming $d_{min} = 4$, mencionado por Trachtenberg (2002), é exemplo de código ótimo e a sua construção lexicográfica é dada a partir de sua matriz geradora G, na Equação 4.44.

$$G = \begin{bmatrix} 00001111 \\ 00111100 \\ 01011010 \\ 11110000 \end{bmatrix}. \tag{4.44}$$

Pode-se perceber que, assim como a matriz geradora do código de Alencar e Giraudo (2007), a matriz geradora do código de Trachtenberg (2002) possui a característica de que $GG^T = 0$, então também podemos concluir que a matriz de paridade H para esse código é igual à matriz geradora G dele.

5. SIMULAÇÃO E RESULTADOS

Neste capítulo, apresentamos resultados experimentais obtidos para os algoritmos descritos no Capítulo 4, realizando uma comparação crítica segundo o ponto de vista do desempenho.

Para exprimir o desempenho de um sistema de transmissão digital usaremos a taxa de erros de *bit* (BER) na saída do decodificador de canal. A dúvida surge quando procuramos saber se a taxa de erros em questão diz respeito aos *bits* da palavra-código decodificada ou, simplesmente, aos bits de mensagem que correspondem a essa palavra-código. Se muitos autores, na apresentação de resultados relativos ao estudo de desempenho de um dado código, não clarificam devidamente a que BER se referem, é comum considerar a taxa de erros entre os bits de informação (ou mensagem). (GOMES, 2003)

Por exemplo, a análise do desempenho de um código binário (*n,k)* , não sistemático com um *n* grande, é um estudo extremamente demorado. Em alternativa é usual considerar-se que se a palavra recebida contém *x* erros, então, o número médio de *bits* de mensagem errados é *x*R*, e nesse caso,

$$BER_{\text{bits informação}} = BER_{\text{bits codigo}} \qquad (5.1)$$

o que simplifica o cálculo da BER.

Nem todos os erros ocorridos durante a transmissão são detectados. Referimos no Capítulo 3 que no caso de um código linear, usando decodificação por síndrome, só havia detecção de erros se a síndrome fosse diferente do vetor nulo. Assim, num dado esquema de decodificação poderão ocorrer quatro tipos de situações para $s=rH^T$:

- A palavra recebida, r, é a palavra-código que foi transmitida ou que o decodificador de canal corrige corretamente e, logo, não há erros;
- A palavra recebida, r, é uma palavra-código diferente da que foi transmitida, portanto esse erro não é detectado;
- A palavra recebida, r**,** não é uma palavra-código, e o decodificador de canal corrige-a corretamente.

− A palavra recebida, **r**, não é uma palavra-código e o decodificador de canal não corrige-a corretamente.

Definida a medida de desempenho adotada, o problema que se coloca é como comparar de forma justa o desempenho de códigos com diferentes dimensões e taxas de informação.

O uso de um código de canal com taxa R = k/n, tem algumas conseqüências. Uma delas é que a energia de transmissão aumenta para nE_s em vez de kE_s, sendo E_s a energia transmitida por símbolo. De forma a comparar o desempenho de diferentes esquemas de codificação, a energia do sinal passa a ser expressa em termos da energia enviada por bit de informação, E_b, com

$$E_b = E_s/R \tag{5.2}$$

Assim, considerando o caso de um canal Gaussiano, os gráficos de desempenho passam a ser expressos em termos da figura de mérito, E_b/N_0, diretamente relacionada com o SNR do canal em que:

$$\frac{E_b}{N_0} = \frac{E_s}{2R\sigma^2} \tag{5.3}$$

sendo σ^2 a variância do ruído AWGN que é adicionado a cada símbolo da palavra de código transmitida, $N_0/2$ a densidade espectral de potência do ruído, e E_b/N_0 é expresso normalmente em dB.

Outro termo comum na análise do desempenho de um esquema de codificação é o *ganho de codificação,* que é uma medida da potência adicional que seria necessário transmitir no caso de não usarmos qualquer método de codificação, para obter o mesmo desempenho.

Fixando a energia transmitida por bit E_b, a utilização de um código de taxa R provoca um aumento da taxa de transmissão, R_s, e uma diminuição da energia enviada por símbolo, E_s, ou seja, uma diminuição da SNR na saída do canal. Como conseqüência, verifica-se o aumento da taxa de erros da seqüência recebida, em comparação com a que seria obtida se não fosse usado qualquer esquema de codificação. No entanto, a redundância introduzida com o código de canal, permite que após a decodificação muitos dos erros sejam corrigidos, garantindo um desempenho superior ao sistema sem codificação, como pode ser visualizado na Figura 5.1 (GOMES, 2003).

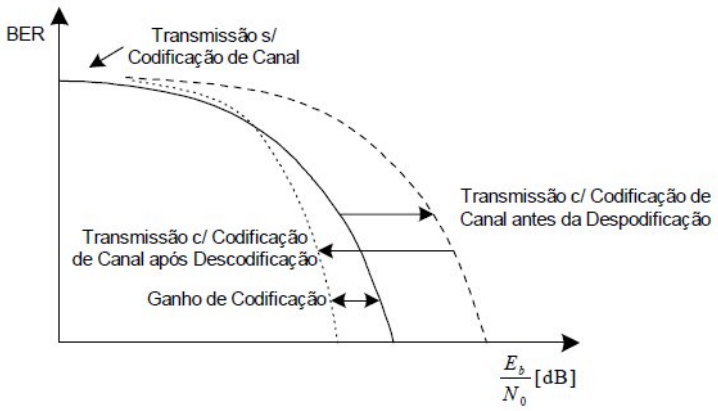


Figura 5.1: Conceito de ganho de codificação (GOMES, 2003)

Para nossa simulação utilizamos o programa *Matlab®*, usando o método de Monte Carlo, simulando um sistema de transmissão digital, como foi definido no Capítulo 2. Para cada código lexicográfico estudado, foram computadas as matrizes *G* e *H*, com a finalidade de utilizá-las na codificação e decodificação de canal, respectivamente. O algoritmo utilizado para a simulação de um sistema de transmissão digital segue abaixo:

i) Gera mensagens *u* aleatórias de dimensão *k*;

ii) Codifica essas mensagens e obtém a palavra-código *v*, onde $v = uG$;

iii) Passa essa palavra por um modulador *BPSK*;

iv) Adiciona ruído gaussiano branco;

v) Passa essa saída por um demodulador *BPSK*, gerando o vetor *r*;

vi) Computa a síndrome desse vetor, $s = rH^T$;

vii) Procura a síndrome na tabela de síndrome;

viii) Achando a síndrome na tabela, encontra qual o líder de *coset* para aquela síndrome, sendo esse líder um padrão de erro *e*;

ix) Decodifica a palavra recebida, $v\_estimado = r + e$;

x) Compara *v_estimado* com *v* e computa em quantas posições eles diferenciaram;

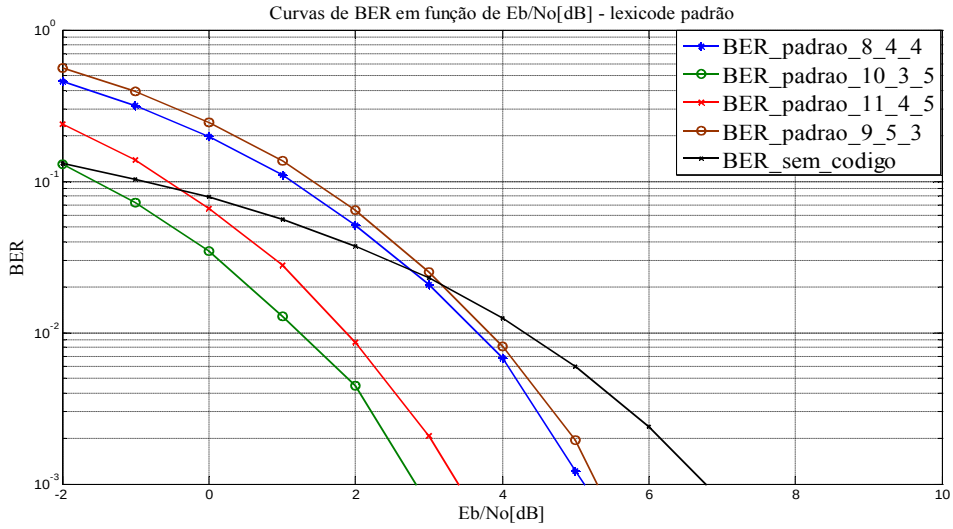
xi) Computa a quantidade de *bits* de *v*;

xii) Repete-se os passos de i) a xi) até a quantidade de iterações definidas;

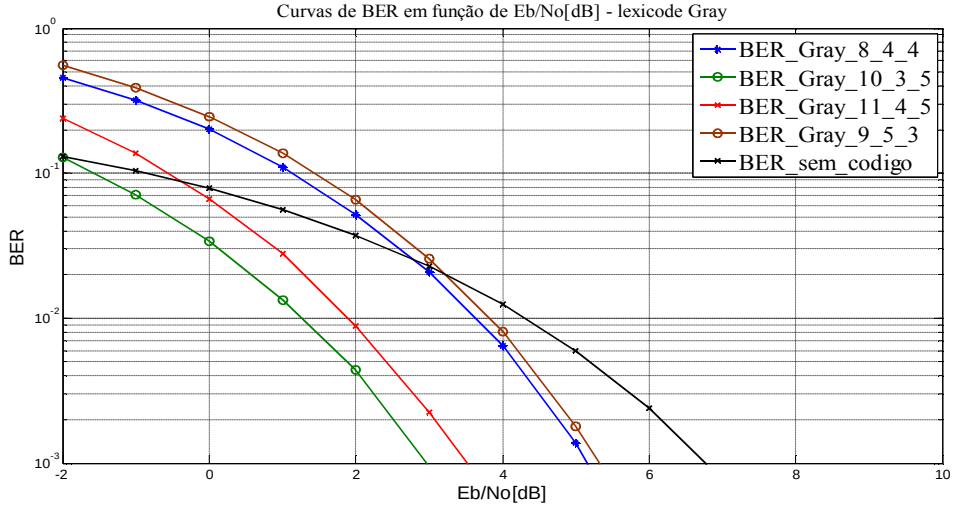
xiii) Para cada $E_b/N_0[dB]$ faz-se $BER = \frac{n^o\_total\_de\_erros}{n^o\_total\_de\_bits}$;

xiv) Plota o gráfico *BER x* $E_b/N_0[dB]$.

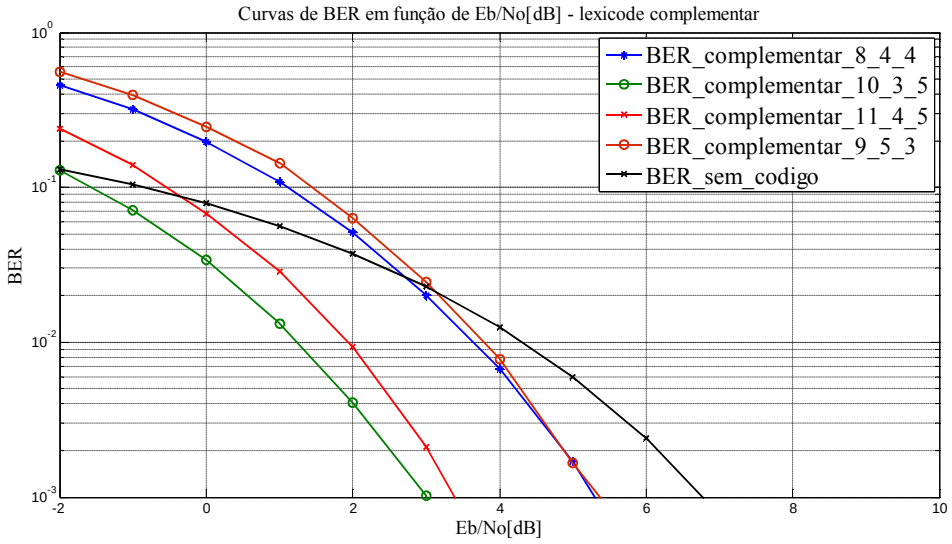
Usando esse algoritmo para os ordenamentos lexicográficos padrão, *Gray*, complementar e alternado obtivemos os resultados da Figura 5.2. Os *(n,k,d)* de comparação utilizados foram (8,4,4), (9,5,3), (10,3,5) e (11,4,5), respectivamente, além da simulação para o sistema não-codificado.



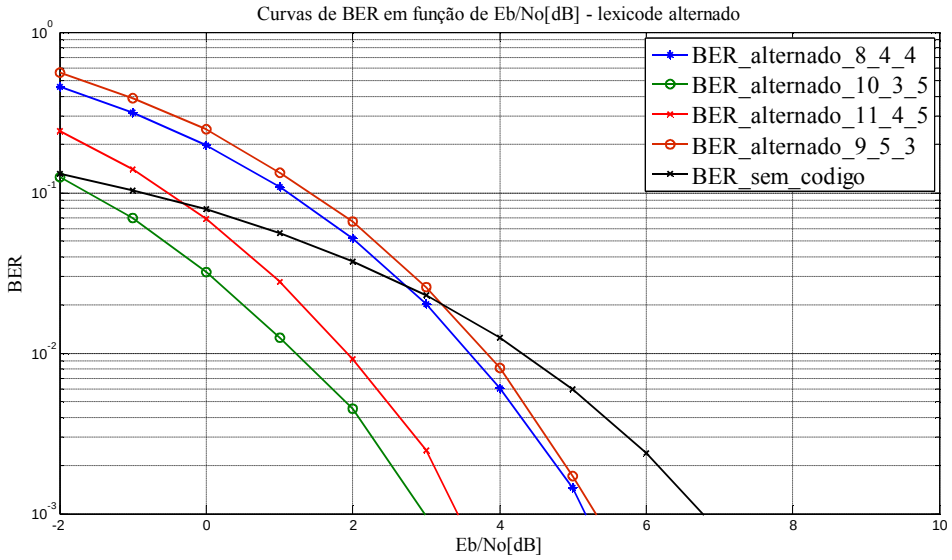
(a)



(b)



(c)



(d)

Figura 5.2: BER x E_b/N_0[dB] para os ordenamentos lexicográficos. (a) padrão, (b) *Gray*, (c) complementar, (d) alternado.

Tabela 5.1: Ganho de codificação para o código padrão (9,5,3), (8,4,4), (11,4,5) e (10,3,5).

BER	**E_b/N_0[dB] (9,5,3)**	**Eb/N0[dB] (sem_código)**	**Ganho de codificação**
0.1	1.4	-0.9	-2.3
0.02	3.2	3.2	0
0.01	3.8	4.3	0.5
0.001	5.3	6.8	1.5
BER	**E_b/N_0[dB] (8,4,4)**	**Eb/N0[dB] (sem_código)**	**Ganho de codificação**
0.1	1.1	-0.9	-2.0
0.03	3.0	3.0	0
0.01	3.6	4.3	0.7
0.001	5.2	6.8	1.6
BER	**E_b/N_0[dB] (11,4,5)**	**Eb/N0[dB] (sem_código)**	**Ganho de codificação**
0.1	-0.5	-0.9	-0.4
0.08	-0.2	-0.2	0
0.01	1.9	4.3	2.4
0.001	3.4	6.8	3.4
BER	**E_b/N_0[dB] (10,3,5)**	**Eb/N0[dB] (sem_código)**	**Ganho de codificação**
0.13	-1.9	-1.9	0
0.1	-1.6	-0.9	0.7
0.01	1.2	4.3	3.1
0.001	2.9	6.8	3.9

Podemos notar que ao compararmos cada ordenamento lexicográfico, os códigos de mesmo *(n,k,d)* têm a mesma BER e E_b/N_0[dB], o que já era esperado, já que essa relação é função dos três parâmetros. Entretanto, ao analisarmos cada código em separado, entre *(n,k,d)* diferentes, o ganho de codificação também é diferente, como visualizado na Figura 5.2 e na Tabela 5.1. Mais especificamente, observa-se que há a diminuição da BER com o aumento da relação E_b/N_0[dB] do sistema codificado em relação ao não-codificado.

Percebe-se também que para uma dada taxa de erro, pelo menos para valores menores que 10^{-3}, o sistema codificado, com as construções lexicográficas simuladas, possui taxas de erros menores que o sistema não-codificado, precisando de menos potência de transmissão. Para os códigos (8,4,4), essa afirmação é verdadeira a partir de E_b/N_0[dB] = 3dB, quando o ganho de codificação é nulo, passando a ser positivo a partir desse ponto. Para os códigos (9,5,3), (10,3,5) e (11,4,5) o ganho de codificação passa a ser positivo a partir de E_b/N_0[dB] = 3.2dB, -1.9dB e -0.2dB, respectivamente.

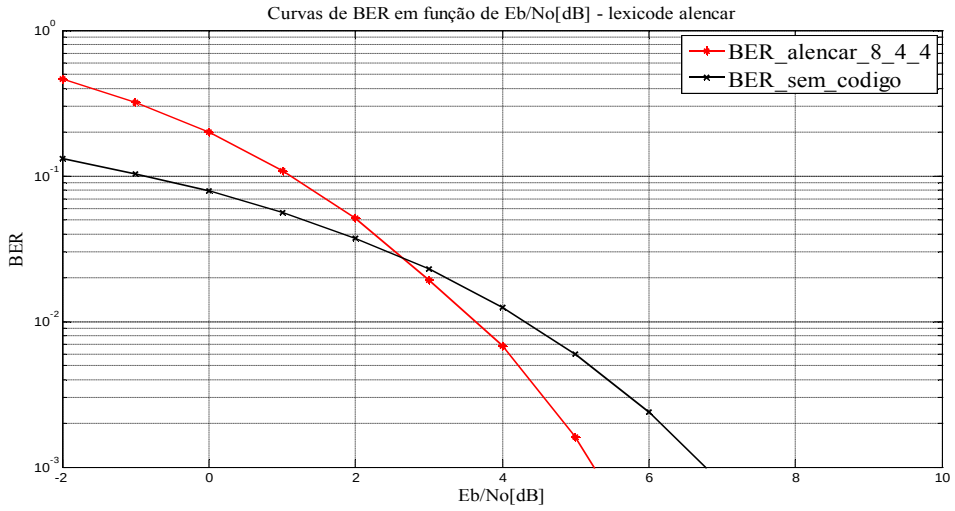
Na Tabela 5.2 é mostrado que para os códigos lexicográficos com *dmin* maiores, para um dado E_b/N_0[dB], há uma BER menor. Isso já era esperado, pois para cada código a

capacidade de correção é dada pela Equação 3.12, na qual é visto que essa capacidade é diretamente proporcional a *dmin*.

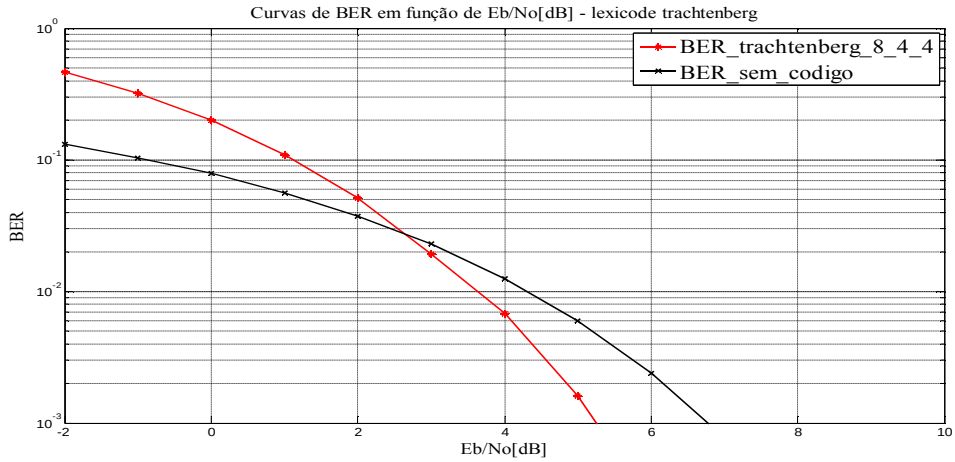
Tabela 5.2: Valores de BER para o código padrão (9,5,3), (8,4,4), (11,4,5) e (10,3,5), para valores constantes de E_b/N_0[dB].

E_b/N_0[dB]	**BER (9,5,3)**	**BER (8,4,4)**	**BER (11,4,5)**	**BER (10,3,5)**	**BER (sem_código)**
-2	0.5598	0.4562	0.2383	0.1289	0.1306
-1	0.3941	0.3195	0.1387	0.0708	0.1038
0	0.2466	0.1975	0.0675	0.0341	0.0786
1	0.1438	0.1083	0.0287	0.0131	0.0563
2	0.0628	0.0513	0.0093	0.0041	0.0375
3	0.0245	0.0202	0.0021	0.0010	0.0229
4	0.0078	0.0067	0.0003	0.0002	0.0125
5	0.0017	0.0017	0.0001	0.0000	0.0060

Para os algoritmos desenvolvidos por Trachtenberg (2002) e Alencar e Giraudo (2007) obtivemos os resultados da Figura 5.3. O (*n,k,d*) de comparação utilizado foi o (8,4,4).



(a)



(b)

Figura 5.3: BER x E_b/N_0[dB] para os códigos de: (a) Alencar e Giraudo (2007) (b) Trachtenberg (2002).

Aqui nota-se também que a partir de E_b/N_0[dB] = 3dB, para ambos os códigos, o ganho passa a ser positivo, ou seja, para valores maiores que 3dB é preciso menos potência de transmissão para o sistema codificado que para o sistema não-codificado.

Na Figura 5.4 fazemos a comparação entre todos os códigos lexicográficos para os quais geramos os algoritmos. Para (*n,k,d*) = (8,4,4), podemos perceber que para todos eles teremos o mesmo ganho de codificação. Podemos então concluir que todos os códigos, para (*n,k,d*) iguais entre si, apesar de terem as matrizes geradoras e de paridade diferentes, precisam da mesma potência de transmissão.

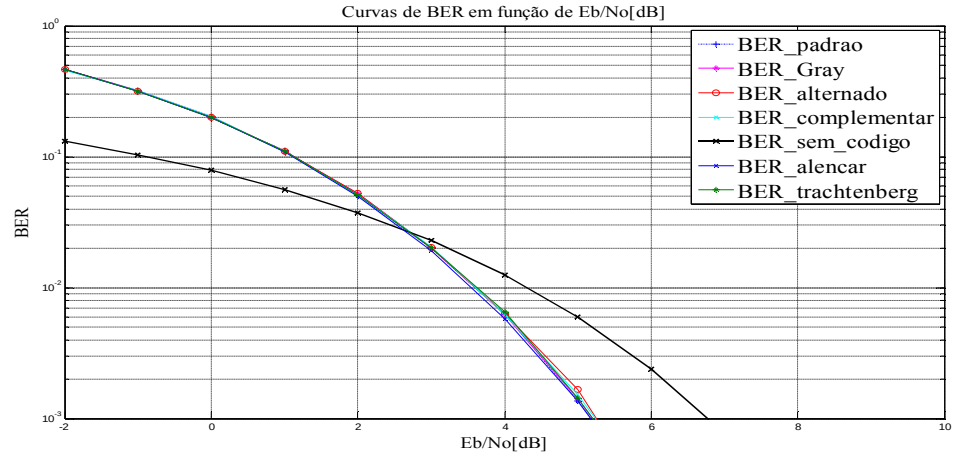


Figura 5.4: BER x E_b/N_0[dB] para os códigos lexicográficos simulados – (8,4,4).

6. CONCLUSÃO

Após analisarmos os conceitos, desenvolvidos por diversos autores, que se relacionam com a teoria dos códigos corretores de erros lexicográficos e fundamentado o estudo, pudemos implementar os principais códigos lexicográficos existentes na literatura, sobre o campo de Galois de característica 2, ou seja, binário.

A geração dos *lexicodes* se deu através do algoritmo *greedy*, que foi apresentado por Brualdi e Pless (1993). Entretanto, esse algoritmo só fornecia como saída a matriz de paridade, as palavras-código e a tabela de *coset*, sendo que por meio dele era dificultoso resolver a equação $GH^T = 0$ para obter a matriz geradora, que é de extrema importância na codificação. Como a matriz *H* desse código não pode ser reduzida à forma sistemática trivialmente, uma contribuição desse trabalho foi gerar a matriz *G* por outro método, no qual esta é encontrada tal que são escolhidas para suas linhas as palavras-código c_j, onde j é uma potência de 2.

Ademais, para o algoritmo *greedy* foi proposta uma base lexicográfica diferente das existentes na literatura, a qual chamamos base ordenada lexicográfica alternada.

Outra contribuição foi mostrar o desempenho dos *lexicodes* através da simulação da BER como função de E_b/N_0[dB]. Obtemos como resultado que, para E_b/N_0[dB] cada vez maiores, tem-se taxas de erro de bit cada vez menores, ao compararmos o sistema codificado e o não-codificado. Além disso, observou-se que para uma dada probabilidade de erro, o sistema codificado com a construção lexicográfica precisa de menos potência de transmissão que o sistema não-codificado.

TRABALHOS FUTUROS

Como trabalhos futuros pretendemos:

- Aumentar a lista de códigos lexicográficos gerados através do algoritmo *greedy* para $n > 14$ e analisar a capacidade de correção para esses códigos.

- Continuar o trabalho com os códigos lexicográficos desenvolvidos por Trachtenberg (2002) e desenvolver nosso algoritmo baseado na estrutura apresentada por ele para códigos com *dmin* maiores.

- Propor novas bases lexicográficas para o algoritmo *greedy*, a fim de analisar suas características para tamanhos n e dimensões k maiores.

- Implementar os códigos lexicográficos sobre campos de Galois com características ternárias e quaternárias.

REFERÊNCIAS BIBLIOGRÁFICAS

ACOSTA, K.; VIEIRA, M. H.; MARTÍNEZ, J. E. **Lexicographic and Non-Lexicographic Greedy Codes**. 2001.

AHMAD, K. **G-value Decoding of Greedy Codes**. Proc.\1st International Conference on Information and Com. Tech. IEEE Catalog No.05EX1176 p. 99-100, 2005.

AHMAD, K.; DIN, S. U. **Graph Representation of Binary Greedy Codes, Coloring and Decoding**. European Journal of Scientific Research, ISSN 1450-216X Vol.30 No.4, p.579-583, 2009.

ALENCAR, F. A. M.; GIRAUDO, E. C. **Construção dos Códigos (2^M, M+1, 2^{M-1}), com M ≥ 3, Usando Blocos Lexicográficos**. Rev. Bras. Biom., São Paulo, v. 25, n. 4, p.157-163, 2007.

BRUALDI, R. A.; PLESS, V. **Greedy codes**. J. Comb. Theory Ser. *A*, New York, v.64, p.10–30, 1993.

CONWAY, J. H. **Integral lexicographic codes**. Discrete Math., Amsterdam, v.83, p.219–235, 1990.

CONWAY, J.H.; SLOANE, N.J.A. **Lexicographic Codes: Error-Correcting Codes from Game Theory**. Transaction on Information Theory, Vol.IT-32, No.3, 1986.

COSTA, F. M. da; AGUSTINI, E. **Códigos Corretores de Erros Lineares Equivalentes**. 14º Simpósio Internacional de Iniciação Científica da USP, 2006, São Paulo - SP. Anais do 14º SIICUSP, 2006. Vol. 01. p. 01-01, 2006.

GIRAUDO, E. C. **Curso de Pós-Graduação em Engenharia de Telecomunicações**. Comunicações Digitais III, FIEC-UFC, 2001, Fortaleza.

GOMES, M. A. C. **Códigos Binários Definidos por Matrizes de Teste de Paridade Esparsas Algoritmos de Descodificação.** Universidade de Coimbra, Engenharia Eletrotécnica e de Computadores, 2003.

HAYKIN, S. **Sistemas de Comunicação Analógicos e Digitais**, 4ª ed., Bookman, Porto Alegre, 2004.

HEFEZ, A.; VILELA, M. L. T. **Códigos corretores de erros**. Rio de Janeiro: IMPA, 206p, 2002.

HERSCOVICI, D. S. **Minimal Distance Lexicographic Codes Over an Infinite Alphabet**. IEEE Transactions on Information Theory, Vol.37, No.5, 1991.

HUFFMAN, W. C.; PLESS, V. **Fundamentals of Error-Correcting Codes**. Cambridge University Press, 2003.

JENKINS, B. **Table of lexicodes**. Disponível em: <http://www.burtleburtle.net/bob/math/lexicode.html >. Acesso em 10 Out 2009.

LEVENSHTEIN, V. I. **A class of systematic codes**. Soviet Math. Dokl., Providence, v.1, n.1, p.368-371, 1960.

LIN, S.; COSTELLO Jr. D. **Error control coding**. Prentice Hall, 1260p, 1983.

MONROE. L. **Binary Greedy Codes**. Congressus Numerantium, Vol. 104, p.49-63, 1994.

MORELOS, H. R.; ZARAGOZA. **The Art of Error Correcting Coding**. John Wiley & Sons, 2002.

SPASOV, D. **Implementing the Lexicographic Construction**. Boston University, 2006.

SUPARTA, N. **Counting Sequences, Gray Codes, and Lexicodes.** Ph.D. thesis, Delft University of Technology, 2006.

TRACHTENBERG, A. **Designing lexicographic codes with a given trellis complexity**. IEEE Trans. Inform. Theory, New York, v.48, n.1, p.89-100, 2002.

TRACHTENBERG, A. **Error-Correcting Codes on Graphs: Lexicodes, Trellises, and Factor Graphs**, Ph.D. thesis, University of Illinois at Urbana-Champaign, 2000,

Disponível em: <http://ipsit.bu.edu/documents/phdthesis.ps>. Acesso em 20 Nov 2009.

TRACHTENBERG, A.; VARDY, A. **Lexicographic codes: Constructions, bounds, and trellis complexity.** Proc. 31st Annual Conference on Information Sciences and Systems, Baltimore, MD, P. 521-527, 1997.

ZANTEN, A. J. **Lexicographic Order and Linearity**. Designs, Codes and Cryptography, Vol.10, p.85-97, 1997.