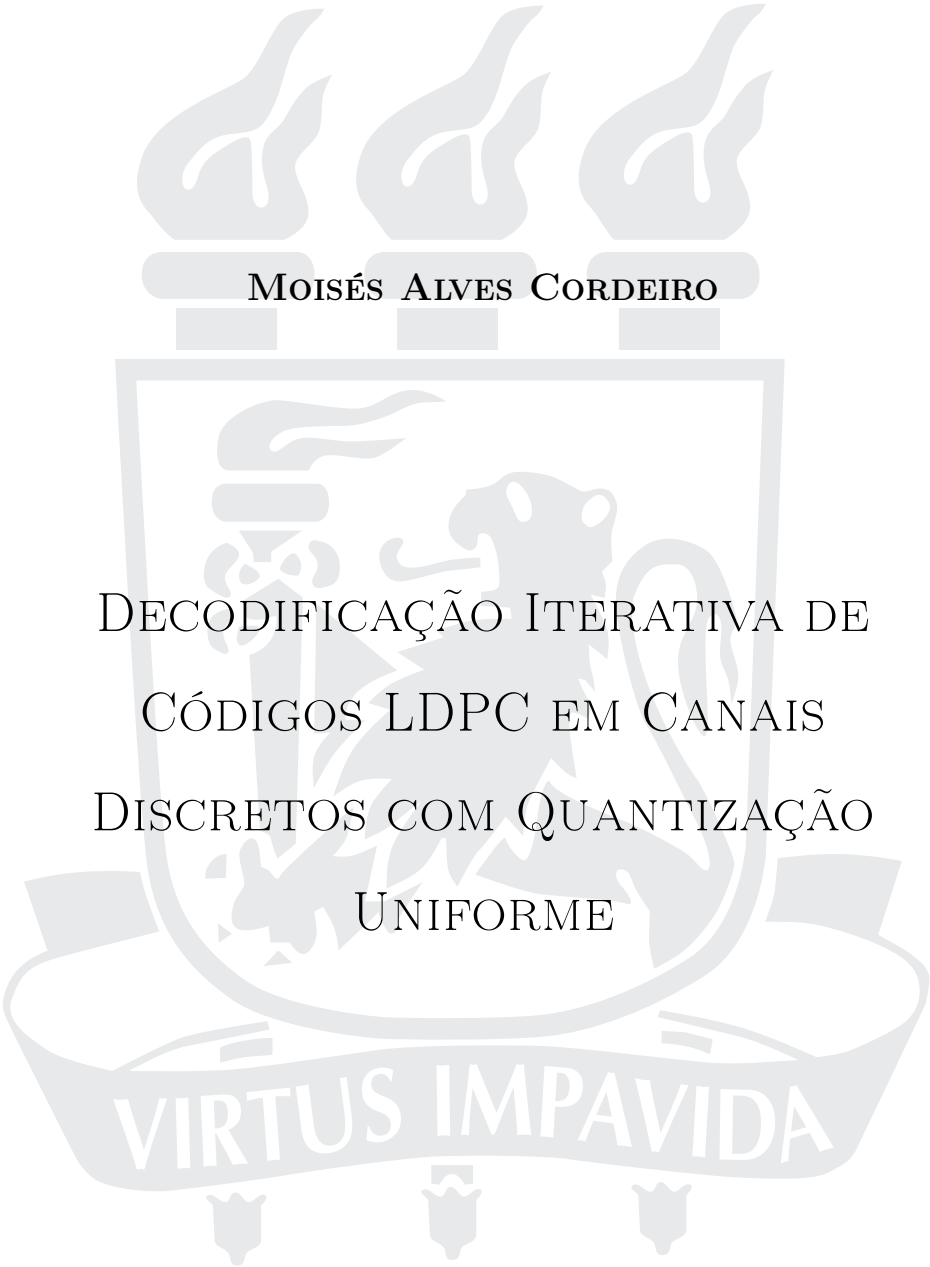
Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Programa de Pós-graduação em Engenharia Elétrica



Moisés Alves Cordeiro

Decodificação Iterativa de Códigos LDPC em Canais Discretos com Quantização Uniforme

Recife, Abril de 2010.

Moisés Alves Cordeiro

Decodificação Iterativa de Códigos LDPC em Canais Discretos com Quantização Uniforme

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco como parte dos requisitos para obtenção do grau de **Mestre em Engenharia Elétrica**

Orientador: Prof. Cecilio José Lins Pimentel, Ph.D.

Recife, Abril de 2010.

C794d **Cordeiro, Moises Alves.**
Decodificação iterativa de códigos LDPC em canais discretos com quantização uniforme / Moises Alves Cordeiro. - Recife: O Autor, 2010.
66 folhas, il : figs.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Elétrica, 2010.

Orientador: Prof. Cecílio José Lins Pimentel.

Inclui Bibliografia.

1. Engenharia Elétrica. 2.Códigos Corretores de Erro. 3.Decodificação Iterativa. 4.Grafo de Fatores. 5.Quantização Uniforme. I. Título.

UFPE

621.3 CDD (22. ed.) BCTG/2010-101

Universidade Federal de Pernambuco

Pós-Graduação em Engenharia Elétrica

PARECER DA COMISSÃO EXAMINADORA DE DEFESA DE DISSERTAÇÃO DO MESTRADO ACADÊMICO DE

MOISÉS ALVES CORDEIRO

TÍTULO

"DECODIFICAÇÃO ITERATIVA DE CÓDIGOS LDPC EM CANAIS DISCRETOS COM QUANTIZAÇÃO UNIFORME"

A comissão examinadora composta pelos professores: CECÍLIO JOSÉ LINS PIMENTEL, DES/UFPE, RICARDO MENEZES CAMPELLO DE SOUZA, DES/UFPE, DANIEL CARVALHO DA CUNHA, POLI/UPE e MARIA DE LOURDES MELO GUEDES ALCOFORADO, POLI/UPE sob a presidência do primeiro, consideram o candidato **MOISÉS ALVES CORDEIRO APROVADO.**

Recife, 23 de abril de 2010.

RAFAEL DUEIRE LINS
Coordenador do PPGEE

CECÍLIO JOSÉ LINS PIMENTEL
Orientador e Membro Titular Interno

DANIEL CARVALHO DA CUNHA
Co-Orientador e Membro Titular Externo

RICARDO MENEZES CAMPELLO DE SOUZA
Membro Titular Interno

MARIA DE LOURDES MELO GUEDES ALCOFORADO
Membro Titular Externo

Agradecimentos

A Deus, pela sua presença constante em minha vida, pelo dom da sabedoria para o desenvolvimento deste trabalho e por colocar tantas pessoas maravilhosas ao meu redor dentre as quais posso citar:

O meu pai Luiz Fernando e a minha mãe Maria do Carmo pelo amor, dedicação, apoio e incentivos para seguir em frente, contribuindo de forma positiva para minha formação e educação. Os meus irmãos, Luiz Filipe e Marília pela apoio e amizade me ajudando a tornar o caminho mais suave.

A minha família pela presença marcante em minha vida, começando por meus avós, motivo de toda fortaleza e união familiar, passando por meus tios que são novos pais em minha vida orientando, guiando e sempre me apoiando até chegar aos meus mais novos primos que são amigos e irmãos mesmo na distância, todos eles com a sua contribuição própria e peculiar. A minha namorada Danielly pela compreensão, amor, apoio, incentivos e por compartilhamento de cada momento.

Sou especialmente grato ao meu orientador, o professor e amigo Cecilio José Lins Pimentel, por ter acreditado no meu potencial e aceitado me orientar, partilhando comigo um pouco do seu conhecimento e experiência, além de ser um exemplo marcante de dedicação e profissionalismo. Agradeço ao meu co-orientador Daniel Carvalho da Cunha por toda ajuda e disponibilidade em ajudar quando o procurei e aos professores do PPGEE, em especial aos professores do grupo de comunicações e aos meus amigos da pós-graduação que muito colaboraram com estudos, incentivos e parcerias. De forma muito especial quero agradecer a Humberto Beltrão que contribuiu de forma muito significativa e determinante para as simulações.

Enfim, agradeço de forma mais sincera possível, às pessoas que contribuíram, direta e indiretamente, para a realização do presente trabalho.

Moisés Alves Cordeiro

Universidade Federal de Pernambuco

09 de Abril de 2010

Resumo da Dissertação apresentada à UFPE como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica

Decodificação Iterativa de Códigos LDPC em Canais Discretos com Quantização Uniforme

Moisés Alves Cordeiro

Abril/2010

Orientador: Prof. Cecilio José Lins Pimentel, Ph.D.
Área de Concentração: Comunicações
Palavras-chaves: Algoritmo soma-produto, códigos corretores de erro, decodificação iterativa, grafo de fatores, quantização
Número de páginas: 67

A decodificação de códigos baseados em matrizes de verificação de paridade esparsas (LDPC, do inglês *low-density parity-check*) é realizada através do algoritmo soma-produto (ASP). Este trabalho apresenta um estudo do funcionamento do ASP e do seu desempenho em um canal com ruído aditivo Gaussiano branco (AWGN, do inglês *additive white Gaussian noise*) através de simulações computacionais. Em seguida, esta análise é estendida quando um quantizador uniforme com 2^q níveis de quantização é incorporado ao sistema de comunicações. O passo de quantização ótimo é identificado para vários parâmetros do código e do canal. Este estudo indica que para $q = 1$ (quantização abrupta) ocorre uma perda de desempenho de aproximadamente 1,8 dB em relação ao canal AWGN enquanto que para $q = 4$ essa perda é reduzida para aproximadamente 0,12 dB.

Abstract of Dissertation presented to UFPE as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering

Iterative Decoding of LDPC Codes for Discrete Channels with Uniform Quantization

Moisés Alves Cordeiro

April/2010

Supervisor: Prof. Cecilio José Lins Pimentel, Ph.D.
Area of Concentration: Communications
Keywords: error correcting codes, factors graph, iterative decoding, quantization, sum-product algorithm
Number of pages: 67

The decoding of LDPC codes is carried out by using the Sum-Product algorithm (SPA). This work presents a study of the SPA and analyzes its performance over the AWGN channel by computer simulations. This analysis is extended to consider the incorporation of a uniform quantizer with 2^q levels into the communication system. The optimum quantization step is identified for several coding and channel parameters. This study indicates that for $q = 1$ (hard quantization) there is a performance loss of 1.8 dB relative to the AWGN channel, while for $q = 4$ this loss is reduced to 0.12 dB.

Sumário

Lista de Figuras

Lista de Tabelas

capítulo 1

Introdução

Os códigos baseados em matrizes de verificação de paridade esparsas (LDPC, do inglês *low-density parity check*) [1] são atrativos devido ao fato de apresentarem desempenho muito próximo ao limite estabelecido por Shannon [2] em canais com ruído aditivo Gaussiano branco (AWGN, do inglês *additive white Gaussian noise*) permitindo uma decodificação iterativa eficiente através do algoritmo soma-produto (ASP) [3], cuja complexidade cresce linearmente com o comprimento do bloco. Exemplos de sistemas de comunicações práticos que usam códigos LDPC para aumentar a confiabilidade da informação transmitida incluem wimax [4] e televisão digital via satélite [5].

Uma característica dos códigos LDPC está no fato de sua matriz de verificação de paridade $\mathbf{H}$ conter uma baixa quantidade de elementos não-nulos. A partir das distribuições destes elementos nas linhas e colunas de $\mathbf{H}$, os códigos LDPC podem ser classificados em regulares, quando esse número é constante em cada linha ou coluna, ou irregulares, caso contrário. As técnicas de construções de códigos LDPC são baseadas no preenchimento da matriz $\mathbf{H}$, como é o caso do código de Mackay [6], que busca evitar ciclos de comprimento curto, dos códigos LDPC quase-cíclicos [7], [8], em que se destaca a propriedade de que o deslocamento cíclico de w posições de uma palavra-código também é uma palavra-código [9] e do algoritmo de crescimento progressivo (PEG, do inglês *progressive edge-growth*) [10], [11] que tem o objetivo de obter uma representação gráfica do código de ciclo mínimo controlado. Esta dissertação apresenta um estudo do funcionamento do ASP e do seu desempenho em um canal AWGN através de simulações computacionais, analisando a influência de alguns parâmetros, tais como, o comprimento do código, o número de iterações, o polinômio de distribuição de grau,

abrangendo duas subclasses desse código: os regulares e os irregulares. Nessas simulações é possível constatar o ganho de codificação com o aumento do comprimento do código, verificar o número de iterações adequado e o ganho de codificação dos códigos irregulares sobre os regulares.

A implementação do ASP em circuitos digitais de alta velocidade usualmente requer quantização dos símbolos recebidos. Desta forma, após a quantização [12], os símbolos na entrada do ASP pertencem a um alfabeto finito, com cardinalidade denotada por 2^q. Neste trabalho, o desempenho do código LDPC será analisado via simulação quando a transmissão é realizada em um canal discreto que utiliza modulação por chaveamento de fase binária (BPSK, do inglês binary phase-shift keying), canal AWGN, demodulador, quantizador uniforme com 2^q níveis de quantização. Um dos objetivos desta dissertação é identificar o passo de quantização ótimo para cada valor de q e para vários parâmetros do código e do canal. Observa-se que o emprego de quantização abrupta ($q = 1$) implica em uma perda de desempenho de aproximandamente 1,8 dB em relação ao canal AWGN, enquanto que praticamente o desempenho do canal AWGN é obtido com $q = 4$.

1.1 Organização da Dissertação

Este trabalho está organizado em 6 capítulos, conforme detalhado a seguir.
No **Capítulo 2** será descrita a representação de códigos de blocos lineares por meio de grafos [13], [14]. No **Capítulo 3** será apresentado o ASP, descrevendo cada etapa desse algoritmo de passagem de mensagens. No **Capítulo 4** serão descritas técnicas de construção de códigos LDPC, incluindo os código de Mackay, os códigos quase-cíclicos e os códigos construídos pelo algoritmo PEG. Ainda neste capítulo será realizada uma análise de desempenho de códigos LDPC em canais AWGN. No **Capítulo 5** será apresentado um modelo de canal discreto com entrada binária e saída 2^q-ária obtido através do emprego de um quantizador uniforme. A determinação dos parâmetros do quantizador será realizada neste capítulo. No **Capítulo 6** serão apresentadas as conclusões deste trabalho e as sugestões para trabalhos futuros.

capítulo 2

Fundamentação Teórica

Este capítulo visa descrever alguns parâmetros de códigos de blocos lineares, incluindo a definição de função característica e sua representação gráfica, através do grafo de Tanner e do grafo de fatores. Serão considerados códigos de blocos lineares definidos por sua matriz geradora ou por sua matriz de verificação de paridade.

2.1 Códigos de Blocos Lineares Binários

Seja $u = \{u_1, u_2, ..., u_K\}$ um bloco de K bits gerados pela fonte de informação. Este bloco é utilizado para gerar vetores binários de dimensão N, $x = (x_1, ..., x_N)$, que são denominados palavras-código. Isto é feito através da adição de $N - K$ bits de redundância. O conjunto de palavras-código de um código de bloco linear binário de comprimento N, $C(N, K)$, forma um subespaço de dimensão K do espaço vetorial $\{0, 1\}^N$ (todas as N-uplas binárias). A taxa do código é $R = K/N$. Um código linear é especificado através de sua matriz geradora, denotada por $\mathbf{G}$, que tem dimensão $K \times N$ com linhas linearmente independentes [15]. A mensagem pode ser codificada pela operação $x{=}u\mathbf{G}$, sendo uma palavra-código a combinação linear dos vetores linhas $\vec{\mathbf{g}}_i$ da matriz $\mathbf{G}$, $x{=}u_1\vec{\mathbf{g}}_1{+}u_2\vec{\mathbf{g}}_2{+}...{+}u_K\vec{\mathbf{g}}_K$. Uma outra forma de especificar um código de bloco linear é através da matriz de verificação de paridade.

Definição 2.1 (Matriz de verificação de paridade) *A matriz de verificação de paridade, denotada por* $\mathbf{H}$*, tem dimensão* $(N - K) \times N$ *com* $N - K$ *linhas linearmente independentes, tal que qualquer vetor no espaço linha de* $\mathbf{G}$ *é ortogonal às linhas de* $\mathbf{H}$ *e qualquer vetor que é ortogonal às linhas de* $\mathbf{H}$ *está no espaço linha de* $\mathbf{G}$*.*

A partir da matriz $\mathbf{H}$ é possível determinar se uma palavra binária $x = (x_1, x_2, ..., x_N)$ pertence ou não ao código, utilizando-se a expressão $x\mathbf{H}^T$=$\mathbf{0}$ em que $\mathbf{H}^T$ representa a transposta da matriz $\mathbf{H}$ e $\mathbf{0}$ é um vetor de zeros[1]. Esta expressão estabelece um conjunto de equações conhecidas como equações de paridade. Para um mesmo código, existe mais de uma matriz de verificação de paridade possível. Observe que $\mathbf{G}\mathbf{H}^T = \mathbf{0}$.

Exemplo 2.1 *Uma matriz de verificação de paridade* $\mathbf{H}$ *do código de Hamming* $C(7,4)$ *é:*

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

Considere um vetor binário $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$*. As equações de paridade obtida por* $x\mathbf{H}^T = \mathbf{0}$ *para este código são:*

$$\begin{aligned} x_1 \oplus x_4 \oplus x_6 \oplus x_7 &= 0 \\ x_2 \oplus x_4 \oplus x_5 \oplus x_6 &= 0 \\ x_3 \oplus x_5 \oplus x_6 \oplus x_7 &= 0. \end{aligned} \tag{2.1}$$

em que o operador $\oplus$ *representa a soma módulo 2.*

Definição 2.2 (Função Característica) *A função característica de um código de bloco linear, denotada por* $\lambda_c(x_1, x_2, ..., x_N)$*, é uma função com domínio* $\{0,1\}^N$ *e imagem* $\{0,1\}$ *que mapeia um conjunto formado por todos os vetores binários de comprimento* N*,* $(x_1, x_2, ..., x_N)$*, em* 0*, se pelo menos uma das equações de paridade não for satisfeita, ou em* 1*, se todas as equações de paridade forem satisfeitas simultaneamente, isto é:*

$$\lambda_c(x) = \lambda_c(x_1, x_2, ..., x_N) = \begin{cases} 1, & se \;\; x\mathbf{H}^T = \mathbf{0} \\ 0, & caso\ contrário\,. \end{cases} \tag{2.2}$$

Podemos expressar a função característica do código $C(7,4)$ em termos das equações de paridade (2.1) aplicando-se a função delta de Kronecker[2] para cada uma dessas equações. Com isso, a partir de (2.1) obtemos $\delta(x_1 \oplus x_4 \oplus x_6 \oplus x_7 = 0)$, $\delta(x_2 \oplus x_4 \oplus x_5 \oplus x_6 = 0)$ e $\delta(x_3 \oplus x_5 \oplus x_6 \oplus x_7 = 0)$. Este código apresenta a função característica:

$$\begin{aligned} \lambda_c(x_1, x_2, ..., x_7) &= \delta(x_1 \oplus x_4 \oplus x_6 \oplus x_7 = 0)\delta(x_2 \oplus x_4 \oplus x_5 \oplus x_6 = 0)\delta(x_3 \oplus x_5 \oplus x_6 \oplus x_7 = 0) \\ &= f_1(x_1, x_4, x_6, x_7)\, f_2(x_2, x_4, x_5, x_6)\, f_3(x_3, x_5, x_6, x_7) \end{aligned} \tag{2.3}$$

[1] Neste trabalho a notação $\mathbf{0}$ indica um vetor ou matriz toda zero cuja dimensão fica clara no contexto.

[2] A função delta de Kronecker, denotada por $\delta(\cdot)$, assume o valor 1 se seu argumento é verdadeiro ou 0 caso contrário.

em que

$$f_1(x_1, x_4, x_6, x_7) = \delta(x_1 \oplus x_4 \oplus x_6 \oplus x_7 = 0)$$
$$f_2(x_2, x_4, x_5, x_6) = \delta(x_2 \oplus x_4 \oplus x_5 \oplus x_6 = 0)$$
$$f_3(x_3, x_5, x_6, x_7) = \delta(x_3 \oplus x_5 \oplus x_6 \oplus x_7 = 0).$$

Para o caso em que este conjunto de funções assume valores todos não nulos, isto significa que a palavra pertence ao código, diferente do caso em que pelo menos uma delas possui valor nulo, indicando que a palavra não pertence ao código.

2.2 Função Global

Um função $g(x_1, x_2, ..., x_N)$ é dita função global quando possui várias variáveis como argumento. Uma função global, , é denominada fatorável quando puder ser escrita como um produto de K_0 funções locais (os argumentos de cada função local são subconjuntos de $x_1, ..., x_N$). Uma função global fatorável pode ser representada da seguinte forma:

$$g(x_1, x_2, ..., x_N) = \prod_{j=1}^{K_0} f_j(X_j) \tag{2.4}$$

em que $f_j(X_j)$ é uma função local que tem os elementos de X_j como argumentos e X_j é um subconjunto de todas as variáveis, $x_1, x_2, ..., x_N$, que participam da função local correspondente.

Exemplo 2.2 *A função característica (2.3) é uma função global em que* $K_0 = 3$, $X_1 = \{x_1, x_4, x_6, x_7\}, X_2 = \{x_2, x_4, x_5, x_6\}, X_3 = \{x_3, x_5, x_6, x_7\}$ *e pode ser escrita da seguinte maneira:*

$$\lambda_c(x_1, x_2, ..., x_7) = \prod_{j=1}^{3} f_j(X_j) = \begin{cases} 1, & se\ (x_1, x_2, ..., x_n) \in C \\ 0, & se\ (x_1, x_2, ..., x_n) \notin C. \end{cases}$$

2.3 Representação Gráfica

Seja $G(\nu, \xi)$ um grafo com conjunto de nós ν e conjunto de ramos ξ. Um ramo $\varepsilon \in \xi$ será denotado por $e = (v_1, v_2)$ em que $v_1, v_2 \in \nu$ são os nós inicial e final de ε, respectivamente. Para simplificar a notação, um grafo será denotado simplesmente por G. Dois nós $v_i, v_j \in \nu$ são adjacentes se existe um ramo $\varepsilon \in \xi$, tal que $\varepsilon = (v_i, v_j)$. Um grafo pode ser classificado

como direcionado ou não direcionado [16], simples ou composto, cíclico ou acíclico etc. Um grafo é dito ser direcionado quando o par de nós que definem os ramos são pares ordenados, caso contrário, o grafo é dito não direcionado. Neste caso, os pares de nós são não ordenados implicando que $\varepsilon = (v_i, v_j)$ e $\varepsilon = (v_j, v_i)$ sejam considerados um único ramo. Em um grafo simples, todo par de nós adjacentes $v_i, v_j \in \nu$ corresponde a um único ramo. Um grafo composto permite a existência de ramos paralelos.

O número de nós de um grafo direcionado G [17] é definido como a ordem deste grafo e o número de ramos como o seu tamanho. O grau de saída referente a um nó v_i em um grafo direcionado é o número de ramos divergentes de v_i. O grau de entrada é o número de ramos convergentes em v_i. O grau de um nó v_i em um grafo direcionado é definido como a soma do grau de entrada e o grau de saída e para um grafo não direcionado é definido como a soma de todos os ramos dos quais esse nó faz parte. Um vértice de grau 1 é denominado nó folha.

Dado dois nós $v_1, v_2 \in \nu$, um caminho de comprimento ℓ de v_1 a v_2 é uma sequência $\varepsilon_1, ..., \varepsilon_j, ..., \varepsilon_\ell$ tal que v_1 é o nó inicial de ε_1, v_2 é o nó terminal de ε_ℓ e o nó inicial de ε_{j+1} é igual ao nó terminal de ε_j. Um ciclo é um caminho que inicia e termina no mesmo nó. O menor valor do comprimento dos ciclos do grafo é conhecido por ciclo mínimo.

Definição 2.3 (Grafo biparticionado) *Um grafo é denominado biparticionado quando o conjunto de nós é dividido em dois grupos distintos, em que nenhum par de nós que pertence a um mesmo conjunto é adjacente.*

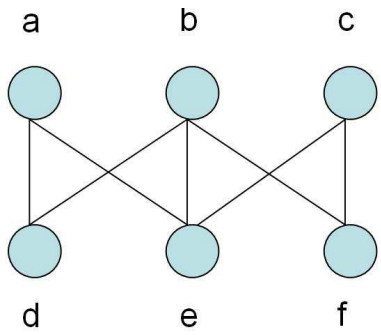


Figura 2.1: Grafo biparticionado.

A Figura 2.1 ilustra um grafo biparticionado com dois subconjuntos de nós $\{a, b, c\}$ e $\{d, e, f\}$. Observe que nesta figura não existe conexão entre os nós de um mesmo subconjunto. No contexto de códigos LDPC, definiremos um grafo biparticionado em que os dois grupos são denominados de nós de variável, rotulado por x_n, e nós de paridade representado por f_j.

Existe um nó de variável para cada um dos N bits da palavra-código e existe um nó de verificação de paridade para cada equação de paridade. O processo de fatoração definido em (2.4), pode ser representado através de um grafo de fatores.

Definição 2.4 (Grafo de Fatores) *O grafo de fatores é um grafo biparticionado em que o conjunto de nós é formado pelo grupo dos nós de variável e pelo grupo dos nós de função. Quando o nó de função f_j apresenta a variável x_n como argumento da função, existe uma conexão entre o nó de variável x_n e o nó de função f_j, formando assim o conjunto de ramos.*

O grafo de fatores [14] representa a estrutura de fatoração de uma função global e é uma forma gráfica de representar a dependência entre as variáveis. O grafo de fatores do código de Hamming $C(7,4)$ do Exemplo 2.1 é mostrado na Figura 2.2.

Pode-se representar a função característica de qualquer código de bloco linear através de um grafo de fatores que possui o número de nós de variável igual ao número de colunas da matriz $\mathbf{H}$, e o número de nós de função igual ao número equações de paridade, $N-K$. Esta representação gráfica foi proposta por Tanner [13].

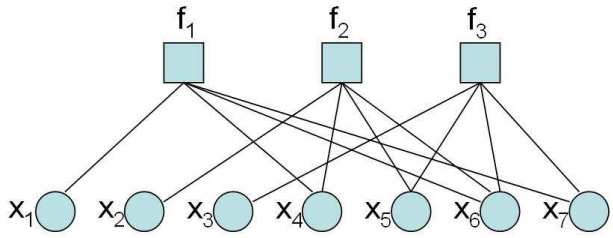


Figura 2.2: Grafo de Tanner do código $C(7,4)$ do Exemplo 2.1.

Definição 2.5 (Grafo de Tanner) *O grafo de Tanner de um código de bloco $C(N,K)$ é um grafo de fatores da função característica do código.*

O Grafo de Tanner que representa a função característica expressa em (2.3) é mostrado na Figura 2.2. Quando o elemento correspondente à n-ésima linha e à j-ésima coluna da matriz $\mathbf{H}$ for igual a 1 existirá um ramo entre os respectivos nós de diferentes grupos do grafo de Tanner e o número total de ramos do grafo é igual ao número de $1's$ da matriz $\mathbf{H}$. Assim, um grafo de Tanner para um código linear binário $C(N,K)$ é obtido a partir de qualquer sistema de equações de paridade do código. Cada nó de verificação de paridade está ligado a todos os

nós de variável relativos às variáveis da equação correspondente e de forma semelhante cada nó de variável está ligado a todos os nós de paridade relativo às equações das quais esta variável faz parte. Observe que modelar os códigos por meio de grafos [18] permite a aplicação de algoritmos de decodificação iterativos que buscam reduzir a complexidade da decodificação. Através deste grafo, é possível manipular funções complexas que envolvem várias variáveis como produto de funções locais mais simples através de alguns algoritmos de transferência de mensagens, dentre os quais se destaca o algoritmo soma-produto [3], que será descrito no Capítulo 3.

capítulo 3

Algoritmo Soma-Produto

O ASP efetua somas e produtos sobre funções locais ao longo de todas as suas variáveis. Este procedimento é realizado através da transferência de mensagens através dos ramos do grafo de fatores que representa a função global a ser marginalizada. Os nós podem ser considerados como processadores, quando atualizam as mensagens recebidas, e os ramos funcionam como canais de comunicações que servem como caminhos por onde as informações processadas são enviadas. Nesse capítulo serão apresentadas as etapas do ASP, descrevendo-se como ocorre a passagem das mensagens no grafo de fatores.

3.1 Marginalização de uma Função

Seja $g(x_1, ..., x_N)$ uma função global. A marginalização desta função em relação a uma variável x_n, denotada por $g_n(x_n)$, é definida por:

$$g_n(x_n) \triangleq \sum_{x_1} ... \sum_{x_{n-1}} \sum_{x_{n+1}} ... \sum_{x_N} g(x_1, ..., x_N).$$

Observe que no cálculo de $g_n(x_n)$ a função global é somada em todas as variáveis exceto em x_n. Para simplificar a notação esta operação é denotada por:

$$g_n(x_n) \triangleq \sum_{\sim x_n} g(x_1, ..., x_N).$$

Exemplo 3.1 *Seja uma função global de quatro variáveis $g(x_1, x_2, x_3, x_4)$, a função marginal $g_1(x_1)$ é expressa da seguinte forma:*

$$g_1(x_1) = \sum_{x_2} \sum_{x_3} \sum_{x_4} g(x_1, x_2, x_3, x_4) = \sum_{\sim x_1} g(x_1, x_2, x_3, x_4). \tag{3.1}$$

Na medida em que o número de variáveis cresce, aumenta o número total de operações, consequentemente, a complexidade. No caso da função global ser fatorável, existem vários termos em comum que podem ser reutilizados no cálculo da função marginal. Para diminuir o esforço computacional, utiliza-se o ASP que reutiliza alguns cálculos intermediários através do grafo de fatores que representa a função global.

3.2 Algoritmo Soma-Produto

3.2.1 Notação

Existem dois tipos de mensagens utilizadas no ASP. Um tipo destas mensagens é representado por $\mu_{x_i \to f_j}(x_i)$ quando a mensagem parte de um nó de variável x_i para um nó de função f_j. O outro tipo de mensagem faz o caminho inverso, partindo de um nó de função para um nó de variável e será representado por $\mu_{f_j \to x_i}(x_i)$. Observe que a mensagem enviada ou recebida por um nó de variável é sempre uma função que tem como argumento a respectiva variável. Aqui será assumido que $N(x_i)$ é o conjunto de nós de função dos quais x_i é argumento e $N(x_i)|f_j$ representa o conjunto $N(x_i)$ excluindo o elemento f_j. De forma semelhante, $N(f_j)$ é formado pelos nós de variável que estão conectados ao nó de função f_j e $N(f_j)|x_i$ representa o conjunto $N(f_j)$ excluindo o elemento x_i. Por exemplo, assumindo os nós de função $f_1(x_1, x_2, x_3), f_2(x_2, x_3, x_4, x_5), f_3(x_2, x_6), f_4(x_1, x_5, x_6)$, obtemos:

$$\begin{aligned} N(x_1) &= \{f_1, f_4\} \\ N(x_2) &= \{f_1, f_2, f_3\} \\ N(x_1)|f_1 &= \{f_4\} \\ N(f_1) &= \{x_1, x_2, x_3\} \\ N(f_1)|x_1 &= \{x_2, x_3\}. \end{aligned}$$

O ASP calcula funções marginais exatas para grafos que não possuem ciclos, enquanto calcula uma aproximação da função marginal quando o grafo possui ciclos, funcionando de forma iterativa.

3.2.2 Etapas do Algoritmo Soma-Produto para Grafos sem Ciclos

O ASP para um grafo sem ciclos, G, é definido em três etapas, sendo a primeira a inicialização, na qual são transmitidas mensagens pelos nós folha de G aos seus nós adjacentes.

Na segunda etapa, denominada de atualização, todos os nós de G recebem as mensagens provenientes de seus nós adjacentes, atualizam-nas e transmitem-nas aos demais nós adjacentes. Essa regra de atualização segue o *princípio da informação extrínseca*, onde uma mensagem que é enviada por um nó x_i ou f_j através de um ramo e_k não pode depender de nenhuma mensagem previamente recebida pelo ramo e_k. Por último, na fase de finalização são calculadas as funções marginais. Estas regras são descritas a seguir:
1) Inicialização - Os nós folha do grafo transmitem simultaneamente as seguintes mensagens:
Mensagem enviada de um nó de variável para um nó de função:

$$\mu_{x_i \to f_j}(x_i) = 1.$$

Mensagem enviada de um nó de função para um nó de variável:

$$\mu_{f_j \to x_i}(x_i) = f_j(x_i).$$

Com estas mensagens recebidas pelos nós adjacentes, os mesmos atualizam e transmitem as mensagens, caracterizando o processo conhecido por atualização.

2) Atualização - As regras de atualização se subdividem em duas, sendo uma regra para os nós de variável e a outra para os nós de verificação de paridade:
A mensagem atualizada pelo nó x_i repassada ao nó adjacente f_j é:

$$\mu_{x_i \to f_j}(x_i) = \prod_{v \in N(x_i) | f_j} \mu_{v \to x_i}(x_i). \tag{3.2}$$

De forma análoga, a mensagem repassada pelo nó f_j ao nó adjacente x_i é dada por:

$$\mu_{f_j \to x_i}(x_i) = \sum_{\sim x_i} f_j(x_j) \prod_{p \in N(f_j) | x_i} \mu_{p \to f_j}(p). \tag{3.3}$$

3) Finalização - Após todos os nós do grafo de fatores receberem mensagens provenientes de todos os seus nós adjacentes, a função marginal $g_n(x_n)$ é calculada por:

$$g_n(x_n) = \prod_{v \in N(x_n)} \mu_{v \to x_n}(x_n).$$

3.2.3 Decodificação de Máxima Verossimilhança

Nesta seção, consideraremos um canal discreto sem memória [19] (DMC, do inglês *discrete memoryless channel*) com alfabeto de entrada $\mathcal{X}$, alfabeto de saída $\mathcal{Y}$, descrito pela probabilidade condicional $P(y|x)$, em que $y \in \mathcal{Y}, x \in \mathcal{X}$.

Considere a transmissão de uma palavra-código $x = (x_1, ..., x_N)$ através de um canal DMC que irá produzir uma sequência recebida expressa pelo vetor $y = (y_1, ..., y_N)$. A partir da observação da sequência y a decodificação de máxima verossimilhança decide qual a sequência x mais provável de ter sido transmitida através da maximização da função densidade de probabilidade *a posteriori* $P(x|y)$. Mostraremos a seguir que $P(x|y)$ é proporcional a uma função global e então o seu grafo de fatores e o ASP podem ser empregados no processo de decodificação.

A distribuição de probabilidade *a posteriori* conjunta para as componentes do vetor transmitido x, dado um vetor recebido fixo y, é dada por:

$$\begin{aligned} P(x|y) &= \frac{P(x,y)}{P(y)} \\ &= \frac{P(y|x)P(x)}{P(y)} \end{aligned} \tag{3.4}$$

em que $P(x)$ é a distribuição de probabilidade *a priori* das palavras-código transmitidas, $P(y|x)$ é a densidade de probabilidade condicional do vetor y quando x é transmitido. Partindo da condição de que a distribuição $P(x)$ é uniforme sobre as palavras-código, pode-se escrever essa probabilidade como sendo:

$$P(x) = \frac{\lambda_c(x)}{|C|} \tag{3.5}$$

em que $\lambda_c(x)$ é a função característica do código e $|C|$ a sua cardinalidade. Utilizando-se (3.5) podemos reescrever (3.4) da seguinte forma:

$$P(x|y) = \frac{P(y|x)\lambda_c(x)}{|C|P(y)}. \tag{3.6}$$

Para y fixo, apenas o numerador de (3.6) é relevante no processo de maximização. Para o caso em que o canal é sem memória e sem realimentação, temos que a distribuição de probabilidade de transição $P(y|x)$ é:

$$P(y|x) = \prod_{i=1}^{N} p(y_i|x_i).$$

Lembrando que $\lambda_c(x_1, x_2, ..., x_N)$ e $P(y|x)$, são funções fatoráveis, podemos escrever o numerador da equação (3.6) como uma função global fatorável da seguinte forma:

$$g(x_1, x_2, ..., x_N) = \lambda_c(x_1, x_2, ..., x_N) \prod_{i=1}^{n} p(y_i|x_i). \tag{3.7}$$

Logo, o numerador de (3.6) que é uma função global fatorável pode ser representado por um grafo de fatores. A função densidade de probabilidade *a posteriori* marginal em relação à

variável x_n, $P(x_n|y)$, é proporcional à marginalização de $g(x_1, ..., x_N)$:

$$P(x_n|y) \sim g_n(x_n) = \sum_{\sim x_n} g(x_1, x_2, ..., x_N). \tag{3.8}$$

A decodificação símbolo a símbolo de cada componente de x, consiste na maximização de $g(x_n)$, $x_n \in \{0, 1\}$, onde a função marginalizada $g_n(x_n)$ é obtida pelo ASP. Em suma, o grafo de fatores que representa (3.7) e o ASP são usados na decodificação de máxima verossimilhança de um código de bloco. O próximo exemplo ilustra a obtenção de $g_n(x_n)$ em (3.8) para o código de repetição usando o ASP.

Exemplo 3.2 *Considere o código de repetição $C(3, 1)$ que apresenta matriz geradora* $\mathbf{G}$ *dada por:*

$$\mathbf{G} = [111]$$

e matriz de verificação de paridade $\mathbf{H}$*:*

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

Sendo $x = (x_1, x_2, x_3)$ uma palavra-código, então as seguintes equações de verificação de paridade devem ser satisfeitas:

$$x_1 \oplus x_2 = 0$$

$$x_2 \oplus x_3 = 0.$$

A função característica desse código é dada por:

$$\lambda_c(x_1, x_2, x_3) = \delta[x_1 \oplus x_2 = 0]\delta[x_2 \oplus x_3 = 0].$$

A função global (3.7) escreve-se:

$$g(x_1, x_2, x_3) = \delta[x_1 \oplus x_2 = 0]\delta[x_2 \oplus x_3 = 0] \prod_{i=1}^{3} P(y_i|x_i) \tag{3.9}$$

$$= \prod_{i=1}^{5} f_i(X_i) \tag{3.10}$$

em que

$$f_1(x_1) = P(y_1|x_1) \tag{3.11}$$

$$f_2(x_2) = P(y_2|x_2) \tag{3.12}$$

$$f_3(x_3) = P(y_3|x_3) \tag{3.13}$$

$$f_4(x_1, x_2) = \delta[x_1 \oplus x_2 = 0] \tag{3.14}$$

$$f_5(x_2, x_3) = \delta[x_2 \oplus x_3 = 0]. \tag{3.15}$$

A seguir descreveremos a troca de mensagens em cada passo do ASP e calcularemos a marginalização de $g(x_1, x_2, x_3)$, dada em (3.10), em relação a cada variável. O grafo de fatores desta função global é mostrado na Figura 3.1. O algoritmo é composto de 6 passos e a ordem cronológica em que estes são organizados é mostrada na Figura 3.2.

1) Inicialização - A inicialização começa pelos nós folha que nesse exemplo são os nós de verificação de paridade f_1, f_2, f_3.

Passo 1: Mensagens enviadas pelos nós de verificação de paridade f_1, f_2, f_3 aos respectivos nós de variável adjacentes x_1, x_2, x_3:

$$\begin{aligned} \mu_{f_1 \to x_1}(x_1) &= f_1(x_1) \\ \mu_{f_2 \to x_2}(x_2) &= f_2(x_2) \\ \mu_{f_3 \to x_3}(x_3) &= f_3(x_3). \end{aligned}$$

2) Atualização - Nesse processo a mensagem é atualizada com informação recebida na fase de inicialização e transmitida aos nós adjacentes:

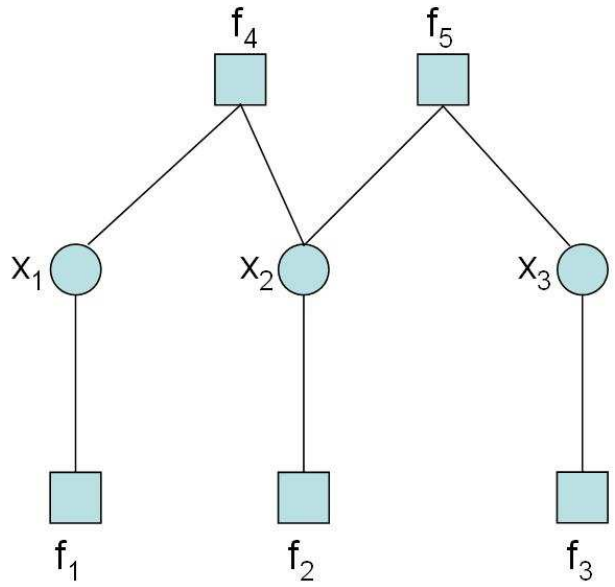


Figura 3.1: Grafo de fatores para o código de repetição C(3,1).

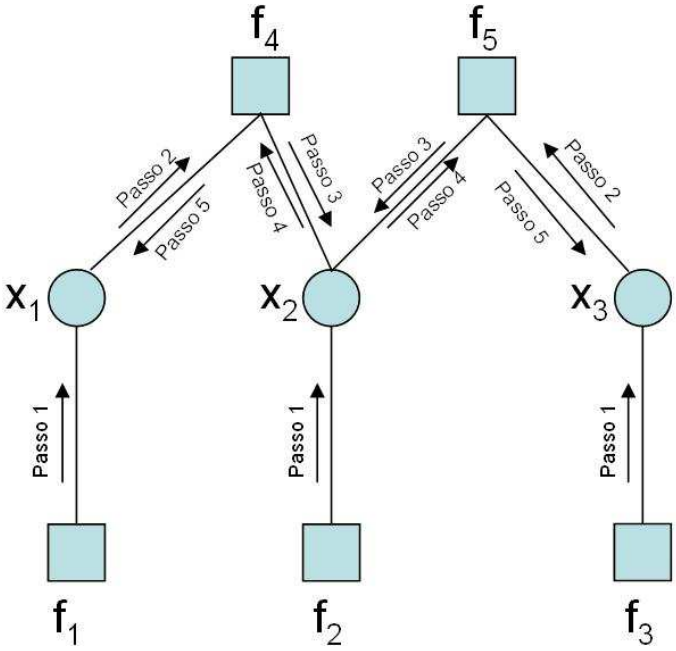


Figura 3.2: Troca de mensagens em todos os passos do ASP para o código de repetição C(3,1).

Passo 2: Mensagens enviadas dos nós de variável x_1 e x_3 aos respectivos nós adjacentes de verificação de paridade f_4 e f_5:

$$\begin{aligned} \mu_{x_1 \to f_4}(x_1) &= f_1(x_1) \\ \mu_{x_3 \to f_5}(x_3) &= f_3(x_3). \end{aligned}$$

Passo 3: Mensagens enviadas dos nós de verificação de paridade f_4 e f_5 para o nó de variável x_2:

$$\begin{aligned} \mu_{f_4 \to x_2}(x_2) &= \sum_{\sim x_2} f_4(x_1, x_2)\mu_{x_1 \to f_4}(x_1) \\ &= \sum_{x_1} f_4(x_1, x_2) f_1(x_1) \\ \mu_{f_5 \to x_2}(x_2) &= \sum_{\sim x_2} f_5(x_2, x_3)\mu_{x_3 \to f_5}(x_3) \\ &= \sum_{x_3} f_5(x_2, x_3) f_3(x_3). \end{aligned}$$

Passo 4: Mensagens enviadas do nó de variável x_2 para os nós de verificação paridade f_4, f_5:

$$\begin{aligned}
\mu_{x_2\to f_4}(x_2) &= \mu_{f_5\to x_2}(x_2)\mu_{f_2\to x_2}(x_2) \\
&= f_2(x_2)\sum_{x_3} f_5(x_2,x_3)f_3(x_3) \\
\mu_{x_2\to f_5}(x_2) &= \mu_{f_4\to x_2}(x_2)\mu_{f_2\to x_2}(x_2) \\
&= f_2(x_2)\sum_{x_1} f_4(x_1,x_2)f_1(x_1).
\end{aligned}$$

Passo 5:Mensagens enviadas dos nós de verificação de paridade f_4 e f_5 aos respectivos nós adjacentes de variável x_1 e x_3:

$$\begin{aligned}
\mu_{f_4\to x_1}(x_1) &= \sum_{\sim x_1} f_4(x_1,x_2)\mu_{x_2\to f_4}(x_3) \\
&= \sum_{x_2} f_4(x_1,x_2)f_2(x_2)\sum_{x_3} f_5(x_2,x_3)f_3(x_3) \\
\mu_{f_5\to x_3}(x_3) &= \sum_{\sim x_3} f_5(x_2,x_3)\mu_{x_2\to f_5}(x_2) \\
&= \sum_{x_2} f_5(x_2,x_3)f_2(x_2)\sum_{x_1} f_4(x_1,x_2)f_1(x_1).
\end{aligned}$$

3)<u>Finalização</u> - Com os resultados obtidos nas etapas anteriores é feito o cálculo das funções marginais:

Passo 6: Processo de marginalização com relação às variáveis x_1, x_2 e x_3:

Marginalização com relação à variável x_1:

$$\begin{aligned}
g_1(x_1) &= \mu_{f_1\to x_1}(x_1)\mu_{f_4\to x_1}(x_1) \\
&= f_1(x_1)\sum_{x_2} f_4(x_1,x_2)f_2(x_2)\sum_{x_3} f_5(x_2,x_3)f_3(x_3) \\
&= \sum_{x_2}\sum_{x_3} f_1(x_1)f_2(x_2)f_3(x_3)f_4(x_1,x_2)f_5(x_2,x_3) \qquad (3.16)\\
&= \sum_{\sim x_1} g(x_1,x_2,x_3).
\end{aligned}$$

Marginalização com relação à variável x_2:

$$\begin{aligned}
g_2(x_2) &= \mu_{f_4\to x_2(x_2)}\mu_{f_5\to x_2(x_2)}\mu_{f_2\to x_2(x_2)} \\
&= f_2(x_2)\sum_{x_1} f_5(x_2,x_3)f_3(x_3)\sum_{x_3} f_4(x_1,x_2)f_1(x_1) \\
&= \sum_{x_1}\sum_{x_3} f_1(x_1)f_2(x_2)f_3(x_3)f_4(x_1,x_2)f_5(x_2,x_3) \qquad (3.17)\\
&= \sum_{\sim x_2} g(x_1,x_2,x_3).
\end{aligned}$$

Marginalização com relação à variável x_3:

$$\begin{aligned} g_3(x_3) &= \mu_{f_3 \to x_3}(x_3)\mu_{f_5 \to x_3} \\ &= f_3(x_3)\sum_{x_2} f_5(x_2, x_3) f_2(x_2) \sum_{x_1} f_4(x_1, x_2) f_1(x_1) \\ &= \sum_{x_1}\sum_{x_2} f_1(x_1) f_2(x_2) f_3(x_3) f_4(x_1, x_2) f_5(x_2, x_3) \\ &= \sum_{\sim x_3} g(x_1, x_2, x_3). \end{aligned} \tag{3.18}$$

Observe que obtemos de forma exata a marginalização da função global em relação ás variáveis x_1, x_2 e x_3, sendo este o objetivo desse exemplo.

O canal binário simétrico (BSC, do inglês *binary symmetric channel*) é um caso especial de um canal DMC em que os símbolos de entrada e de saída pertencem ao alfabeto {0,1}. O canal é simétrico porque a probabilidade de receber o bit 1 dado que o bit 0 foi enviado é igual a probabilidade de receber o bit 0 dado que o bit 1 foi enviado. Essa probabilidade condicional é representada por p na Figura 3.3.

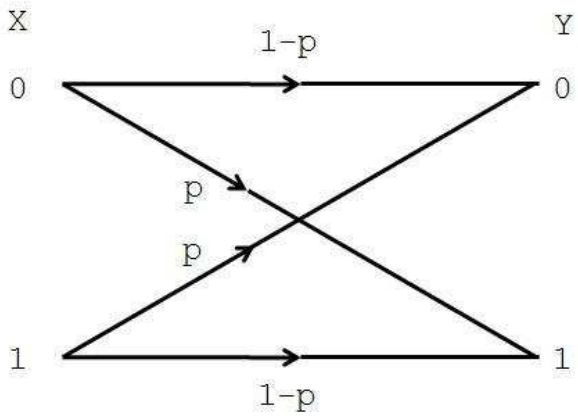


Figura 3.3: Representação de um canal BSC.

A seguir, iremos particularizar os resultados do Exemplo 3.2 para o canal BSC com probabilidade de transição $p < \frac{1}{2}$. Suponha que é transmitida a palavra-código $x = (111)$ e que a palavra recebida seja $y = (110)$. O processo de decodificação estima os valores binários $\hat{x}_1, \hat{x}_2$ e $\hat{x}_3$ que maximizam (3.16), (3.17), (3.18), respectivamente.

Desenvolvendo (3.16), obtemos:

$$\begin{aligned} g_1(x_1) &= \sum_{\sim x_1} f_1(x_1)f_2(x_2)f_3(x_3)f_4(x_1,x_2)f_5(x_2,x_3) \\ &= f_4(x_1,0)f_5(0,0)f_1(x_1)f_2(0)f_3(0) + f_4(x_1,0)f_5(0,1)f_1(x_1)f_2(0)f_3(1) + \\ &\quad f_4(x_1,1)f_5(1,0)f_1(x_1)f_2(1)f_3(0) + f_4(x_1,1)f_5(1,1)f_1(x_1)f_2(1)f_3(1). \quad (3.19) \end{aligned}$$

Para um vetor recebido fixo $y = (110)$, avaliaremos as funções (3.10) - (3.14) da seguinte forma:

$$\begin{aligned} f_5(0,1) &= f_5(1,0) = 0 \\ f_5(0,0) &= f_5(1,1) = 1 \\ f_1(0) &= f_2(0) = p \\ f_3(1) &= p \\ f_1(1) &= f_2(1) = 1-p \\ f_3(0) &= 1-p. \end{aligned}$$

Substituindo estes valores em (3.19) obtemos:

$$\begin{aligned} g_1(0) &= (1)(1)(p)(p)(1-p) + (1)(0)(p)(p)(p) + \\ &\quad (0)(0)(p)(1-p)(1-p) + (0)(1)(p)(1-p)(p) \\ &= p^2(1-p). \\ g_1(1) &= (0)(1)(1-p)(p)(1-p) + (0)(0)(1-p)(p)(p) + \\ &\quad (1)(0)(1-p)(1-p)(1-p) + (1)(1)(1-p)(1-p)(p) \\ &= (0)(1)(1-p)(p)(1-p) + (0)(0)(1-p)(p)(p) + \\ &\quad (1)(0)(1-p)(1-p)(1-p) + (1)(1)(1-p)(1-p)(p) \\ &= (1-p)^2p. \end{aligned}$$

Observe que $g_1(0) < g_1(1)$ se $p < \frac{1}{2}$, então $\hat{x}_1 = 1$. A estimação de $\hat{x}_2$ calculada com base em (3.17) é:

$$\begin{aligned} g_2(x_2) &= \sum_{\sim x_2} f_4(x_1,x_2)f_5(x_2,x_3)f_1(x_1)f_2(x_2)f_3(x_3) \quad (3.20) \\ &= f_4(0,x_2)f_5(x_2,0)f_1(0)f_2(x_2)f_3(0) + f_4(0,x_2)f_5(x_2,1)f_1(0)f_2(x_2)f_3(1) + \\ &\quad f_4(1,x_2)f_5(x_2,0)f_1(1)f_2(x_2)f_3(1) + f_4(1,x_2)f_5(x_2,1)f_1(1)f_2(x_2)f_3(1). \end{aligned}$$

o que avaliada nos valores binários fornece:

$$\begin{aligned} g_2(0) &= (1)(1)(p)(p)(1-p) + (1)(0)(p)(p)(p) + \\ &\quad (0)(1)(1-p)(p)(1-p) + (0)(0)(1-p)(p)(p) \\ &= p^2(1-p). \\ g_2(1) &= (0)(0)(p)(1-p)(1-p) + (0)(1)(p)(1-p)(p) + \\ &\quad (1)(0)(1-p)(1-p)(1-p) + (1)(1)(1-p)(1-p)(p) \\ &= p(1-p)^2. \end{aligned}$$

Se $p < \frac{1}{2}$, então $g_2(0) < g_2(1)$, logo $\hat{x}_2 = 1$. De forma análoga, partindo de (3.18) obtemos:

$$\begin{aligned} g_3(x_3) &= \sum_{\sim x_3} f_4(x_1, x_2) f_5(x_2, x_3) f_1(x_1) f_2(x_2) f_3(x_3) \qquad (3.21) \\ &= f_4(0,0) f_5(0, x_3) f_1(0) f_2(0) f_3(x_3) + f_4(0,1) f_5(1, x_3) f_1(0) f_2(1) f_3(x_3) + \\ &\quad f_4(1,0) f_5(0, x_3) f_1(1) f_2(0) f_3(x_3) + f_4(1,1) f_5(1, x_3) f_1(1) f_2(1) f_3(x_3). \end{aligned}$$

o que conduz a:

$$\begin{aligned} g_3(0) &= (1)(1)(p)(p)(1-p) + (1)(0)(p)(p)(1-p) + \\ &\quad (0)(1)(1-p)(p)(p) + (0)(0)(1-p)(p)(p) \\ &= p^2(1-p). \\ g_3(1) &= (0)(0)(p)(1-p)(1-p) + (0)(1)(p)(1-p)(p) + \\ &\quad (1)(0)(1-p)(1-p)(p) + (1)(1)(1-p)(1-p)(p) \\ &= (1-p)^2 p. \end{aligned}$$

Se $p < \frac{1}{2}$, então $g_3(0) < g_3(1)$, logo $\hat{x}_3 = 1$. Assim, conclui-se que o vetor $\hat{x} = (111)$ foi decodificado corretamente. Uma forma alternativa e mais eficiente de realizar a decodificação usando o ASP é através de passagem de mensagens pelo grafo de fatores, considerando cada mensagem como um vetor de duas componentes, correspondendo a avaliação da função associada a cada variável x_i igual a 0 (na primeira componente) e igual a 1 (na segunda componente). Considerando essa alternativa e reescrevendo os passos do ASP para o mesmo vetor recebido $y = (110)$ obtemos os seguintes passos:

1) Inicialização: Os nós folhas enviam as mensagens iniciais aos nós adjacentes:

Passo 1: Mensagens enviadas pelos nós f_1, f_2, f_3 aos respectivos nós de variável x_1, x_2, x_3:

$$\begin{aligned}
\mu_{f_1 \to x_1}(x_1) &= [P(1|x_1 = 0), P(1|x_1 = 1)] \\
&= [p, 1-p] \\
\mu_{f_2 \to x_2}(x_2) &= [P(1|x_2 = 0), P(1|x_2 = 1)] \\
&= [p, 1-p] \\
\mu_{f_3 \to x_3}(x_3) &= [P(0|x_3 = 0), P(0|x_3 = 1)] \\
&= [1-p, p].
\end{aligned}$$

2) Atualização:

Passo 2: Mensagens enviadas dos nós de variável x_1 e x_3 aos respectivos nós adjacentes de verificação de paridade f_4 e f_5:

$$\begin{aligned}
\mu_{x_1 \to f_4}(x_1) &= [P(1|x_1 = 0), P(1|x_1 = 1)] \\
&= [p, 1-p] \\
\mu_{x_3 \to f_5}(x_3) &= [P(0|x_3 = 0), P(0|x_3 = 1)] \\
&= [1-p, p].
\end{aligned}$$

Passo 3: Mensagens enviadas dos nós de verificação de paridade f_4 e f_5 para o nó de variável x_2:

$$\begin{aligned}
\mu_{f_4 \to x_2}(x_2) &= [f_4(0,0)f_1(0) + f_4(1,0)f_1(1), f_4(0,1)f_1(0) + f_4(1,1)f_1(1)] \\
&= [p, 1-p] \\
\mu_{f_5 \to x_2}(x_2) &= [f_5(0,0)f_3(0) + f_5(0,1)f_3(1), f_5(1,0)f_3(0) + f_5(1,1)f_3(1)] \\
&= [1-p, p].
\end{aligned}$$

Passo 4: Mensagens enviadas do nó de variável x_2 para os nós de verificação paridade f_4, f_5:

$$\begin{aligned}
\mu_{x_2\to f_4}(x_2) &= \left[f_2(0)\sum_{x_3} f_5(0,x_3)f_3(x_3), f_2(1)\sum_{x_3} f_5(1,x_3)f_3(x_3)\right] \\
&= [f_2(0)(f_5(0,0)f_3(0) + f_2(0)f_5(0,1)f_3(1), f_2(1)f_5(1,0)f_3(0) + \\
&\quad + f_2(1)f_5(1,1)f_3(1)] \\
&= [p(1-p), (1-p)p] \\
\mu_{x_2\to f_5}(x_2) &= \left[f_2(0)\sum_{x_1} f_4(x_1,0)f_1(x_1), f_2(1)\sum_{x_1} f_4(x_1,1)f_1(x_1)\right] \\
&= [f_2(0)f_4(0,0)f_1(0) + f_2(0)f_4(1,0)f_1(1), f_2(1)f_4(0,1)f_1(0) + \\
&\quad + f_2(1)f_4(1,1)f_1(1)] \\
&= [p^2, (1-p)(1-p)].
\end{aligned}$$

Passo 5: Mensagens enviadas dos nós de verificação de paridade f_4 e f_5 respectivamente para os nós de variável x_1 e x_3:

$$\begin{aligned}
\mu_{f_4\to x_1}(x_1) &= \left[\sum_{x_1} f_4(x_1,0)f_2(0)\sum_{x_3} f_5(0,x_3)f_3(x_3), \sum_{x_1} f_4(x_1,1)f_2(1)\sum_{x_3} f_5(1,x_3)f_3(x_3)\right] \\
&= [f_4(0,0)f_2(0)f_5(0,0)f_3(0) + f_4(0,0)f_2(0)f_5(0,1)f_3(1) + \\
&\quad f_4(1,0)f_2(0)f_5(0,0)f_3(0) + f_4(1,0)f_2(0)f_5(0,1)f_3(1), \\
&\quad f_4(0,1)f_2(1)f_5(1,0)f_3(0) + f_4(0,1)f_2(1)f_5(1,1)f_3(1) + \\
&\quad f_4(1,1)f_2(1)f_5(1,0)f_3(0) + f_4(1,1)f_2(1)f_5(1,1)f_3(1)] \\
&= [p(1-p), (1-p)p] \\
\mu_{f_5\to x_3}(x_3) &= \left[\sum_{x_2} f_5(x_2,0)f_2(x_2)\sum_{x_1} f_4(x_1,0)f_1(x_1), \sum_{x_2} f_5(x_2,1)f_2(x_2)\sum_{x_1} f_4(x_1,1)f_1(x_1)\right] \\
&= [f_5(0,0)f_2(0)f_4(0,0)f_1(0) + f_5(0,0)f_2(0)f_4(1,0)f_1(1) + \\
&\quad f_5(1,0)f_2(1)f_4(0,0)f_1(0) + f_5(1,0)f_2(1)f_4(1,0)f_1(1) + \\
&\quad f_5(0,1)f_2(0)f_4(0,1)f_1(0) + f_5(0,1)f_2(0)f_4(1,1)f_1(1) + \\
&\quad f_5(1,1)f_2(1)f_4(0,1)f_1(0) + f_5(1,1)f_2(1)f_4(1,1)f_1(1)] \\
&= [p^2, (1-p)(1-p)].
\end{aligned}$$

É possível utilizar-se do proposto em [20] para a passagem de vetores no ASP. Considere o caso mais simples em que chegam duas mensagens em um nó de variável, representadas pelos vetores $r = [r_0, r_1]$ e $s = [s_0, s_1]$, conforme ilustra a Figura 3.4.

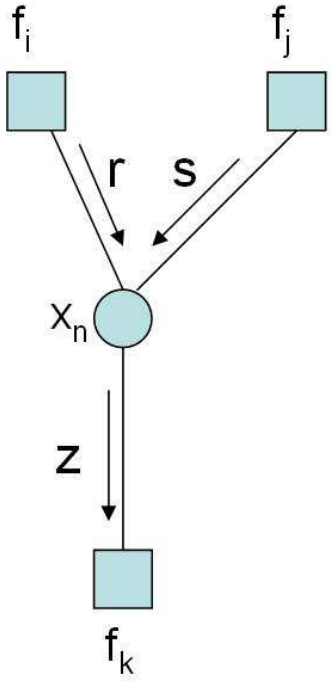


Figura 3.4: Passagem de vetores no ASP para nó de variável.

O vetor de saída do nó de variável é dado por [20]:

$$z = [z_0, z_1] = \left[\frac{r_0 s_0}{r_0 s_0 + r_1 s_1}, \frac{r_1 s_1}{r_0 s_0 + r_1 s_1} \right]. \tag{3.22}$$

Embora os denominadores das componentes do vetor em (3.22), $r_0 s_0 + r_1 s_1$, sejam utilizados como fator de normalização (a soma das componentes do vetor é unitária), nos cálculos a seguir este fator não será utilizado para não sobrecarregar a notação. Também é definida uma regra em [20] para a mensagem de saída de um nó de verificação de paridade, conforme está representado na Figura 3.5:

$$z = [z_0, z_1] = [r_0 s_0 + r_1 s_1, r_0 s_1 + r_1 s_0]. \tag{3.23}$$

Pode-se generalizar (3.22) e (3.23) para nós com grau maior que 2. Repetindo os passos 4 e 5, considerando agora (3.2) e (3.3) e as regras (3.22) e (3.23), respectivamente, obtemos:

Passo 4: Mensagens enviadas do nó de variável x_2 para os nós de verificação paridade f_4, f_5:

$$\begin{aligned} \mu_{x_2 \to f_4}(x_2) &= [p, 1-p][1-p, p] \\ &= [p(1-p), (1-p)p]. \\ \mu_{x_2 \to f_5}(x_2) &= [p, 1-p][p, 1-p] \\ &= [p^2, (1-p)^2]. \end{aligned}$$

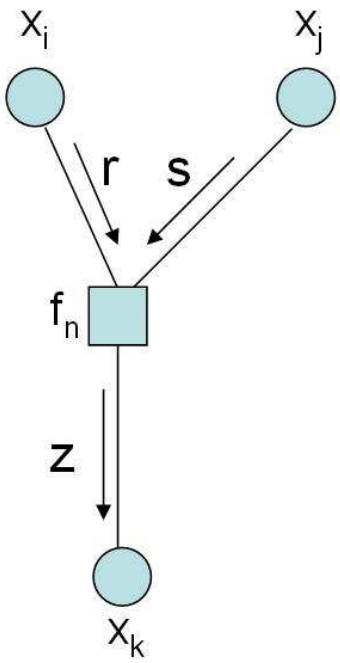


Figura 3.5: Passagem de vetores no ASP para nó de verificação de paridade.

Passo 5: Mensagens enviadas dos nós de verificação de paridade f_4 e f_5 respectivamente para os nós de variável x_1 e x_3:

$$\begin{aligned}
\mu_{f_4 \to x_1}(x_1) &= [p, (1-p)][(1-p), p] \\
&= [p(1-p), (1-p)p] \\
\mu_{f_5 \to x_3}(x_3) &= [p, (1-p)][p(1-p)] \\
&= [p^2, (1-p)^2].
\end{aligned}$$

A marginalização da função global $g(x_1, x_2, x_3)$ com relação às variáveis x_1, x_2 e x_3 utilizando a regra (3.22) de passagens de vetores é vista a seguir:

$$
\begin{aligned}
g_1(x_1) &= \mu_{f_4\to x_1}(x_1)\mu_{f_1\to x_1}(x_1) \\
&= [p(1-p),(1-p)p][p,1-p] \\
&= [p^2(1-p),p(1-p)^2]. \\
g_2(x_2) &= \mu_{f_4\to x_2}(x_2)\mu_{f_5\to x_2}(x_2)\mu_{f_2\to x_2}(x_2) \\
&= [p,1-p][1-p,p][p,1-p] \\
&= [p^2(1-p),p(1-p)^2]. \\
g_3(x_3) &= \mu_{f_5\to x_3}(x_3)\mu_{f_3\to x_3}(x_3) \\
&= [p^2,(1-p)(1-p)][1-p,p] \\
&= [p^2(1-p),p(1-p)^2].
\end{aligned}
$$

3.2.4 Regra da Tangente Hiperbólica

As mensagens podem ser transmitidas no ASP de diversas maneiras. Em vez de $[r_0, r_1]$, pode-se utilizar a diferença, $(r_0 - r_1)$, a razão (r_0/r_1) ou a representação mais comum, que é o logaritmo da razão entre as duas componentes, $\log(r_0/r_1)$, conhecido como logaritmo da razão de verossimilhança (LLR do inglês, *log likehood ratio*). Sendo assim, a mensagem na entrada do nó de variável é representada por $r = \log(r_0/r_1)$ e $s = \log(s_0/s_1)$ e, consequentemente, o formato da mensagem de saída de um nó de variável definida em (3.22) passa a ser:

$$
\begin{aligned}
\mu_{x_i\to f_j}(x_i) &= \log(z_0/z_1) = \log\left(\frac{r_0 s_0}{r_1 s_1}\right) \\
&= \log(r_0/r_1) + \log(s_0/s_1) \\
&= r + s.
\end{aligned}
$$

Para a mensagem de saída de um nó de verificação de paridade, podemos reescrever (3.23) da seguinte forma:

$$
\begin{aligned}
\mu_{f_j\to x_i}(x_i) &= \log(z_0/z_1) \\
&= \log\left(\frac{r_0 s_0 + r_1 s_1}{r_0 s_1 + r_1 s_0}\right).
\end{aligned}
$$

Dividindo o numerador e o denominador do argumento do logaritmo por $r_1 s_1$:

$$
\mu_{f_j\to x_i}(x_i) = = \log\left(\frac{\frac{r_0 s_0}{r_1 s_1} + 1}{\frac{r_0}{r_1} + \frac{s_0}{s_1}}\right).
$$

Fazendo $r_0/r_1 = e^r$ e $s_0/s_1 = e^s$ e utilizando-se da relação $\cosh(A) = \frac{e^A + e^{-A}}{2}$ obtemos:

$$\begin{aligned}
\mu_{f_j \to x_i}(x_i) &= \log\left(\frac{e^r e^s + 1}{e^r + e^s}\right) \\
&= \log\left(\frac{e^{\frac{r+s}{2}}\left(e^{\frac{r+s}{2}} + e^{\frac{-(r+s)}{2}}\right)}{e^{\frac{r+s}{2}}\left(e^{\frac{r-s}{2}} + e^{\frac{-(r-s)}{2}}\right)}\right) \\
&= \log\left(\frac{\cosh\left(\frac{r+s}{2}\right)}{\cosh\left(\frac{r-s}{2}\right)}\right). \qquad (3.24)
\end{aligned}$$

$$(3.25)$$

Utilizando-se da regra $\cosh(A \pm B) = \cosh A \cosh B \pm \sinh A \sinh B$, reescrevemos (3.25) da seguinte forma:

$$\mu_{f_j \to x_i}(x_i) = \log\left(\frac{\cosh\left(\frac{r}{2}\right)\cosh\left(\frac{s}{2}\right) + \sinh\left(\frac{r}{2}\right)\sinh\left(\frac{r}{2}\right)}{\cosh\left(\frac{r}{2}\right)\cosh\left(\frac{s}{2}\right) - \sinh\left(\frac{r}{2}\right)\sinh\left(\frac{r}{2}\right)}\right).$$

Dividindo o numerador e denominador do argumento do log por $\cosh(r/2)\cosh(s/2)$:

$$\mu_{f_j \to x_i}(x_i) = \log\left(\frac{1 + \tanh(\frac{r}{2})\tanh(\frac{s}{2})}{1 - \tanh(\frac{r}{2})\tanh(\frac{s}{2})}\right)$$

e finalmente, utilizando-se da expressão das $\tanh^{-1}(z) = \frac{1}{2}\log\left(\frac{1+z}{1-z}\right)$ chegamos a regra da tangente hiperbólica:

$$\mu_{f_j \to x_i}(x_i) = 2\tanh^{-1}\left[\tanh\left(\frac{r}{2}\right)\tanh\left(\frac{s}{2}\right)\right]. \qquad (3.26)$$

Esta regra é utilizada pelo ASP no domínio LLR para o cálculo das atualizações das mensagens nos nós de verificação de paridade. Em resumo, para entrada r e s no nó de variável a saída é $r + s$ e para estas entradas no nó de verificação de paridade as saídas serão dadas por (3.26).

3.2.5 Passagem de Mensagens em Grafos com Ciclos

Em grafos que possuem ciclos, as mensagens vão perdendo a precisão sobre as funções marginais, gerando um algoritmo iterativo e sub-ótimo, não sendo possível garantir a convergência do ASP. É necessário estabelecer um critério de parada tipicamente baseado em um número máximo de iterações ou quando resulte em decisões de $\hat{x}$ que formam uma palavra-código. O ASP utiliza algumas regras que visam organizar e definir o tempo e as atividades

a serem seguidas, como, por exemplo, definir os sentidos das mensagens trafegadas e quais os ramos serão ativos em cada iteração. Este conjunto de regras ficou conhecido por cronograma. O cronograma mais comum utilizado na decodificação de códigos LDPC é o cronograma invasivo, em que cada iteração do algoritmo possui apenas dois passos, sendo que no primeiro os nós de variável transmitem as mensagens e no segundo, as mensagens são enviadas pelos nós de verificação de paridade.

3.2.6 Utilização do ASP no Processo de Decodificação de um Código de Bloco

A decodificação de um código corretor de erros $C(N, K)$ baseada no ASP é feita símbolo a símbolo. O processo de decodificação consiste primeiramente em representar a distribuição de probabilidade conjunta *a posteriori* dos símbolos de informação $x = (x_1, ..., x_N)$ condicionadas à saída do canal $y = (y_1, ..., y_N)$. Uma vez obtido o grafo de fatores a partir da matriz $\mathbf{H}$ e das probabilidades de transição do canal $p(y_n|x_n)$, o ASP funciona de acordo com os seguintes passos:

1. Inicialização - O ASP inicia pelos nós de verificação de paridade que são nós folha. Cada um deles passa ao seu nó de variável adjacente uma mensagem LLR inicial $L(f_j \rightarrow x_n) = L(y_n|x_n) = \log(P(y_n|x_n = 0)/P(y_n|x_n = 1))$. Os respectivos nós de variável retransmitem aos seus nós adjacentes (excluindo os nós folha) a mensagem:

$$\begin{aligned} L(x_n \rightarrow f_j) &= L(y_n|x_n) \\ &= \log(p(y_n|x_n = 0)/p(y_n|x_n = 1)). \end{aligned}$$

em que $f_j \in N(x_n)$.

2. Atualização - No processo de atualização os nós de verificação de paridade recebem as mensagens $L(x_n \rightarrow f_j)$ e atualizam as mensagens $L(f_j \rightarrow x_n)$ conforme a seguinte equação:

$$L(f_j \rightarrow x_n) = 2 \tanh^{-1} \left[\prod_{t \in N(f_j)|x_n} \tanh \left(\frac{L(t \rightarrow f_j)}{2} \right) \right]. \tag{3.27}$$

Para o caso dos nós de variável a atualização das mensagens é :

$$L(x_n \rightarrow f_j) = L(y_n|x_n) + \sum_{v \in N(x_n)|f_j} L(v \rightarrow x_n).$$

3. Estimação da palavra recebida - Nessa etapa, o valor mais provável para cada símbolo transmitido é estimado. O decodificador utiliza-se da soma das mensagens transmitidas por todos os nós de verificação de paridade conectados à variável x_n para determinar a informação a posteriori sobre o símbolo x_n:

$$\Lambda_n = L(y_n|x_n) + \sum_{v \in N(x_n)} L(v \rightarrow x_n).$$

A decisão de $\hat{x}_n$ para o símbolo x_n é feita baseada na seguinte condição:

$$\hat{x}_n = \begin{cases} 0, & \text{se} \quad \Lambda_n \geq 0 \\ 1, & \text{se} \quad \Lambda_n < 0. \end{cases}$$

Se o número máximo de iterações for completado ou se o vetor $\hat{x} = (\hat{x}_1, ..., \hat{x}_N)$ satisfizer a condição $\hat{x}\mathbf{H}^T = \mathbf{0}$, o algoritmo é finalizado e apresenta o vetor $\hat{x}$ como palavra decodificada. Caso isso não ocorra, o algoritmo repete os passos 2 e 3 até que uma destas condições seja satisfeita. No caso da primeira condição ser atendida será decodificado o último vetor $\hat{x}$ estimado, mesmo que ele não seja uma palavra-código. Daí, a importância da escolha do número máximo de iterações.

3.3 Determinação das LLRs

As mensagens iniciais do ASP que são as LLRs denotados por $L(y_n|x_n), n = 1, ..., N$, são calculadas a partir do modelo do canal. Nesta subseção, determinaremos esta LLR para dois modelos de canal: BSC e AWGN.

3.3.1 Canal BSC

Para o caso de um canal BSC, calcula-se o logaritmo da razão de verossimilhança da seguinte forma:

$$L(y_n|x_n) \quad = \quad \log\left(\frac{p(y_n|x_n = 0)}{p(y_n|x_n = 1)}\right).$$

Consequentemente,

$$L(y_n|x_n) = \begin{cases} \log\frac{p}{1-p}, & \text{se} \quad y_n = 1 \\ \log\frac{1-p}{p}, & \text{se} \quad y_n = 0. \end{cases}$$

3.3.2 Canal AWGN

Considere um canal AWGN com entrada binária cujo o alfabeto é $\{-1, +1\}$, e o vetor recebido $y = (y_1, ..., y_N) \in R^N$. As LLRs são calculadas usando as funções densidades de

probabilidade Gaussiana de médias ± 1 e variância σ^2 da seguinte forma:

$$p_{y_n|x_n=0}(y_n) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{(y_n-1)^2}{2\sigma^2}}$$

$$p_{y_n|x_n=1}(y_n) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{(y_n+1)^2}{2\sigma^2}} .$$

A mensagem inicial processada pelo ASP, $L(y_n|x_n)$, é dada por:

$$\begin{aligned} L(y_n|x_n) &= \log\left(\frac{p_{y_n|x_n=0(y_n)}}{p_{y_n|x_n=1(y_n)}}\right) \\ &= \log\left(e^{\left(-\frac{(y_n-1)^2}{2\sigma^2}+\frac{(y_n+1)^2}{2\sigma^2}\right)}\right) \\ &= -\frac{(y_n-1)^2}{2\sigma^2} + \frac{(y_n+1)^2}{2\sigma^2} \\ &= \frac{-(y_n^2-2y_n+1)+(y_n^2+2y_n+1)}{2\sigma^2} \\ &= \frac{2}{\sigma^2}y_n. \end{aligned}$$

capítulo 4

Códigos lineares com matrizes de verificação de paridade esparsas

Neste capítulo, introduziremos alguns parâmetros de códigos LDPC. Em seguida, serão abordados alguns algoritmos de construção desses códigos, tal como, o algoritmo PEG. Por fim, faremos uma análise comparativa do desempenho de códigos LDPC construídos com o algoritmo PEG em um canal AWGN.

4.1 Códigos LDPC

O conceito de códigos LDPC foi proposto por Robert Gallager em 1962 [1] através de métodos de preenchimento da matriz **H**. Algum tempo depois esses códigos ganharam destaque através do trabalho de Mackay e Neal [21] onde foi verificado que é possível atingir uma probabilidade de erro muita próxima do limite de Shannon [22].

Um código LDPC é caracterizado por uma matriz de verificação de paridade **H** que apresenta uma baixa densidade de elementos não-nulos, ou seja, **H** é uma matriz esparsa. Isso implica em um número relativamente pequeno de ramificações no grafo de fatores com implicações na complexidade de decodificação. Baseado na distribuição de elementos não-nulos por linha ou por coluna da matriz **H**, divide-se os códigos LDPC em dois grupos: os regulares e os irregulares.

Os códigos regulares possuem a característica de que cada linha de **H** tem o mesmo número d_ℓ de $1's$ e cada coluna tem o mesmo número d_c de $1's$. Em [1] verifica-se que a distância mínima desse código regular aumenta linearmente com o comprimento do código. Os parâmetros d_ℓ, d_c, n e k especificam uma família de códigos LDPC e são escolhidos de tal modo que satisfaçam a condição:

$$(n-k)d_\ell = nd_c. \tag{4.1}$$

É importante salientar que o valor de cada membro da igualdade (4.1) fornece o número de elementos não-nulos da matriz **H**. O grafo de Tanner de um código LDPC regular é formado de n nós de variável de grau d_c e $(n-k)$ nós de verificação de paridade de grau d_ℓ. Sabendo-se que o número total de elementos da matriz é $(n-k) \times n$, pode-se escrever a densidade de **H**, denotada por d_H, em função dos parâmetros n, k e d_ℓ:

$$d_H = \frac{(n-k)d_\ell}{(n-k)n} = \frac{d_\ell}{n} \tag{4.2}$$

que também pode ser reescrita em função de n, k e d_c:

$$d_H = \frac{nd_c}{(n-k)n} = \frac{d_c}{(n-k)}. \tag{4.3}$$

Os códigos LDPC irregulares têm a característica que a sua matriz **H** não apresenta o número de $1's$ por linha ou por coluna constante, ou seja, os nós de verificação de paridade e de variável possuem graus variáveis. Uma classe de código irregular é definida por um par de polinômios $(\lambda(x), \rho(x))$ de distribuições de grau do grafo de Tanner expresso por:

$$\lambda(x) = \sum_i \lambda_i x^{i-1} \tag{4.4}$$

$$\rho(x) = \sum_i \rho_i x^{i-1} \tag{4.5}$$

em que $\lambda(x)$ é a distribuição de grau dos nós de variável e $\rho(x)$ é a distribuição de grau dos nós de verificação de paridade. Os coeficientes λ_i (ou ρ_i) expressam a fração de ramos no grafo de Tanner que estão conectados aos nós de variável (ou nós de verificação de paridade) de grau i. O grau máximo dos nós de variável do código é denotado por J e o grau médio dos nós de variável, denotado por J_M, é:

$$J_M = \sum_i i\lambda_i. \tag{4.6}$$

Os polinômios $\lambda(x)$ e $\rho(x)$ para um código regular em função de d_ℓ e d_c são dados por:

$$\lambda(x) = x^{d_c - 1} \tag{4.7}$$

$$\rho(x) = x^{d_\ell - 1}. \tag{4.8}$$

Técnicas de construção de códigos LDPC irregulares são descritas em [23] e [24]. A taxa de um código LDPC é dada por [20]:

$$R = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}. \tag{4.9}$$

Através de manipulações matemáticas na expressão (4.1) obtemos uma expressão para a taxa de um código LDPC regular:

$$R = 1 - \frac{d_c}{d_\ell}. \tag{4.10}$$

A partir de (4.10) observa-se que $d_c < d_\ell$.

4.2 Construção de Códigos LDPC

Nessa seção, são apresentadas algumas técnicas de construção de códigos LDPC baseadas no preenchimento da matriz **H** [25], [26].

4.2.1 Códigos de Mackay

A técnica proposta por Mackay [6], [21] consiste de construções semi-aleatórias da matriz **H** que visam obter códigos cujos grafos de fatores não apresentem ciclos curtos. A matriz **H** é construída com colunas de peso d_c e com linhas de peso constante d_ℓ de modo que quaisquer duas colunas não apresentem sobreposição maior que 1 buscando evitar ciclos de comprimento quatro [27]. Isso é verificado quando o produto interno entre duas colunas de **H** é sempre menor ou igual a 1. Mackay disponibilizou um arquivo de códigos LDPC em [28].

4.2.2 Códigos LDPC Quase-Cíclicos

A classe de códigos LDPC quase-cíclicos (LDPC-QC) possui a propriedade de que o deslocamento cíclico de w posições de uma palavra-código, múltiplo de d_ℓ, também é uma palavra-código [9]. Algumas técnicas de construções desta classe de códigos foram propostas em [7], [8], [9], [29] e [30]. Um família de códigos LDPC-QC irregular é apresentada em [31].

Exemplo 4.1 *Considere um código* $C(21, 17)$ *cuja matriz de paridade é apresentada em [7]:*

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

A matriz $\mathbf{H}$ *pode ser reescrita da seguinte maneira:*

$$\mathbf{H} = \begin{bmatrix}
\mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 \\
\mathbf{H}_3 & \mathbf{0} & \mathbf{0} & \mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{0} \\
\mathbf{H}_2 & \mathbf{H}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_0 & \mathbf{H}_1 \\
\mathbf{H}_1 & \mathbf{H}_2 & \mathbf{H}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_0
\end{bmatrix}$$

em que

$$\mathbf{H}_0 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{H}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{H}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{H}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Através da permutação das linhas e das colunas de $\mathbf{H}$, é possível obter um código LDPC-QC [7]. Esta classe de códigos apresenta cotas para a distância mínima em função dos parâmetros do código [7].

4.2.3 Códigos de Gallager e Códigos H-LDPC

Seja uma matriz $\mathbf{H}_t$ de dimensão $(n_t - k_t) \times n_t$. É possível construir uma matriz diagonal em blocos $\mathbf{H}_d$ de dimensão $d(n_t - k_t) \times dn_t$, utilizando-se de d blocos de $\mathbf{H}_t$ na diagonal principal de $\mathbf{H}_d$:

$$\mathbf{H}_d = \begin{bmatrix} \mathbf{H}_t & \mathbf{0} & \mathbf{0} & ... & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_t & \mathbf{0} & ... & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & ... & \mathbf{H}_t \\ & & & & \end{bmatrix} \tag{4.11}$$

Fazendo permutações aleatórias nas colunas de $\mathbf{H}_d$ obtemos novas matrizes denotadas genericamente por $\pi_i(\mathbf{H}_d)$. Esse conjunto de matrizes resulta na seguinte matriz esparsa:

$$\mathbf{H} = \begin{bmatrix} \pi_1(\mathbf{H}_d) \\ \pi_2(\mathbf{H}_d) \\ \vdots \\ \pi_i(\mathbf{H}_d) \end{bmatrix}. \tag{4.12}$$

Observe que a dimensão da matriz $\mathbf{H}$, $id(n_t - k_t) \times dn_t$, é função do número de blocos da matriz diagonal, das dimensões da matriz $\mathbf{H}_t$ e do número de permutações (i) feitas na matriz $\mathbf{H}_d$. Por exemplo, a construção definida em [32], conhecida como a matriz do código de Gallager [1], está na forma (4.12) em que $\mathbf{H}_t = (1, 1, ..., 1)$. Quando a matriz $\mathbf{H}_t$ é a matriz de verificação de paridade de um código de Hamming, $n_t = 2^{m_t} - 1$, $m_t = n_t - k_t$, forma-se o código H-LDPC [33]. Algumas construções com código de Hamming constituinte apresentam baixa complexidade na decodificação [34]. Um dos primeiros estudos com códigos LDPC baseados em códigos de Hamming (H-LDPC) foi feito em [35]. De forma análoga, a construção (4.12) pode ser feita com outros códigos constituintes, tal como, os códigos BCH [36], códigos Reed-Solomon [37].

4.2.4 Algoritmo PEG

Com objetivo de obter grafos de ciclo mínimo controlado pela conexão entre os nós de variável e os nós de verificação de paridade, surgiu o algoritmo PEG [10]. De acordo com a variação dos parâmetros desse algoritmo, é possível construir grafos colocando-se ramos progressivamente entre os nós de variável e os nós de verificação de paridade de modo a maximizar os ciclos mínimos locais de cada um dos nós de variável [10]. O objetivo do algoritmo é encontrar nós de verificação de paridade mais distantes de um determinado nó de variável x_i, construindo um subgrafo partindo desse nó e dos ramos nele incidentes, e assim conectá-lo através de ramos. O algoritmo seleciona o nó de menor grau quando encontra mais de um nó de verificação de paridade para conexão com o nó de variável x_i, fazendo com que a distribuição de graus dos nós de verificação de paridade seja o mais uniforme possível. No algoritmo PEG não há controle sobre o polinômio $\rho(x)$. A principal vantagem desse algoritmo é que ele é muito flexível e possibilita obter códigos com uma estrutura que permite um codificador com complexidade linear. Por isso, utilizaremos matrizes $\mathbf{H}$ construídas com este algoritmo na análise de desempenho de códigos LDPC que será realizada na próxima seção. Para a construção dessa matriz deve-se especificar alguns parâmetros, tais como, o comprimento e a taxa do código e o polinômio $\lambda(x)$ que especifica a distribuição de grau dos nós de variável. Em [11] é proposta uma melhoria no algoritmo PEG aumentando consideravelmente o comprimento do ciclo mínimo e quando isso não é possível, o algoritmo tenta diminuir o número de ciclos de menor comprimento.

Exemplo 4.2 *Utilizando o algoritmo PEG [10] para a construção de um código LDPC regular* $C(16, 8)$*, obtemos a matriz* $\mathbf{H}$ *:*

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

Observe que $d_\ell = 4$ *e* $d_c = 2$*. Utilizando-se (4.10) verificamos que a taxa do código é* $R = 0,5$*.*

Exemplo 4.3 *Considere o conjunto de códigos LDPC irregulares* $C(16, 8)$ *que apresenta a seguinte distribuição de grau dos nós de variável:*

$$\lambda(x) = 0,25x^1 + 0,5x^2 + 0,25x^3. \tag{4.13}$$

Uma matriz resultante com esta distribuição de grau utilizando o algoritmo PEG [10] é:

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

Observe que existem 4 colunas que possuem $d_c = 2$*, 8 colunas que possuem* $d_c = 3$ *e 4 colunas que possuem* $d_c = 4$*, o que equivale a constatar no grafo de fatores que existem 4 nós de variável de grau 2, 8 nós de variável de grau 3 e 4 nós de variável de grau 4. Observe também que o grau de distribuição nas linhas procurou manter uma regularidade obtendo assim um parâmetro constante* $d_\ell = 6$ *em cada linha, implicando que todos os nós de verificação de paridade tem grau* 6*.*

4.3 Resultado das Simulações

Nesta seção foram feitas simulações com a finalidade de analisar o desempenho dos códigos LDPC construídos com o algoritmo PEG de taxa igual a $0,5$ variando-se o comprimento do código, o número de iterações e o polinômio $\lambda(x)$. Foram transmitidas palavras-código todas nulas através de um canal AWGN utilizando a modulação BPSK e a decodificação feita pelo ASP. As curvas da taxa de erro de bit (BER, do inglês *bit error rate*) foram obtidas sobre os bits codificados. A simulação termina quando são encontrados 100 erros de bloco.

4.3.1 Desempenho de Códigos LDPC Regulares

A Figura 4.1 ilustra a BER versus a relação sinal ruído (SNR, do inglês *signal to noise ratio*) para um código LDPC regular $C(2000, 1000)$ com $d_c = 3$ utilizando o ASP com 25, 50

e 100 iterações. Quanto maior o número de iterações maior a confiabilidade dos bits decodificados, embora em termos práticos, o aumento desse número também aumenta o tempo de decodificação. Nas próximas análises iremos fixar o número de iterações em 100. As curvas de desempenho BER × SNR para um código regular $C(2000, 1000)$ com d_c igual a 2, 3, 4 e 5 são comparadas na Figura 4.2.

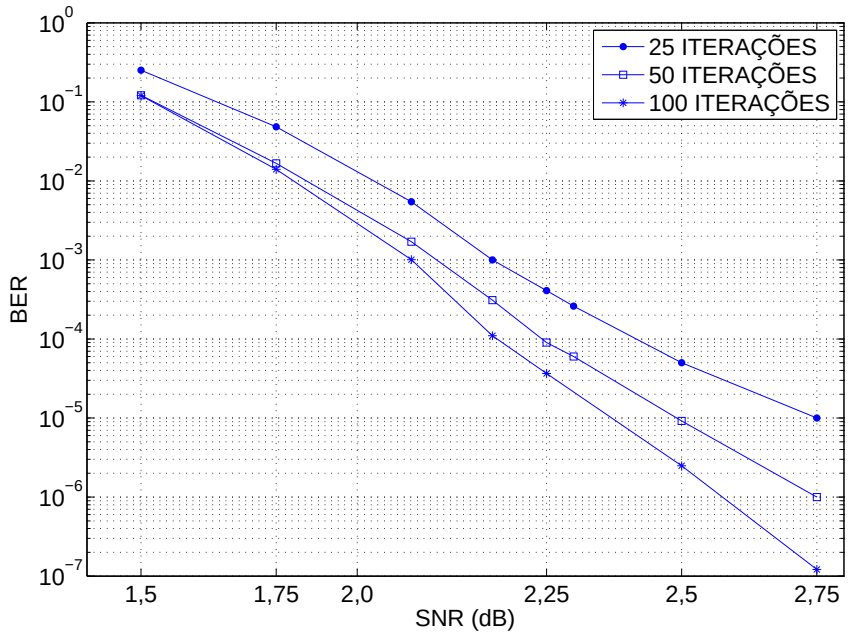


Figura 4.1: Desempenho do código regular $C(2000, 1000)$ *com* $d_c = 3$ *variando-se o número de iterações para 25, 50 e 100 iterações.*

Verifica-se que para $d_c = 2$, o desempenho é bem degradado quando comparados aos demais, dentre os quais se destaca o código construído com o parâmetro $d_c = 3$. Em virtude desta análise, consideraremos $d_c = 3$ nas próximas análises. O efeito do comprimento do código é ilustrado na Figura 4.3, em que considera-se códigos regulares de taxa 0,5, $d_c = 3$ e comprimentos 1000, 2000 e 4000. Observa-se um ganho de codificação em torno de 0,5 dB nas comparações entre os códigos $C(4000, 2000)$ e $C(2000, 1000)$ e entre $C(2000, 1000)$ e $C(1000, 500)$ na faixa de 2 dB a 3 dB para uma BER de 10^{-5}.

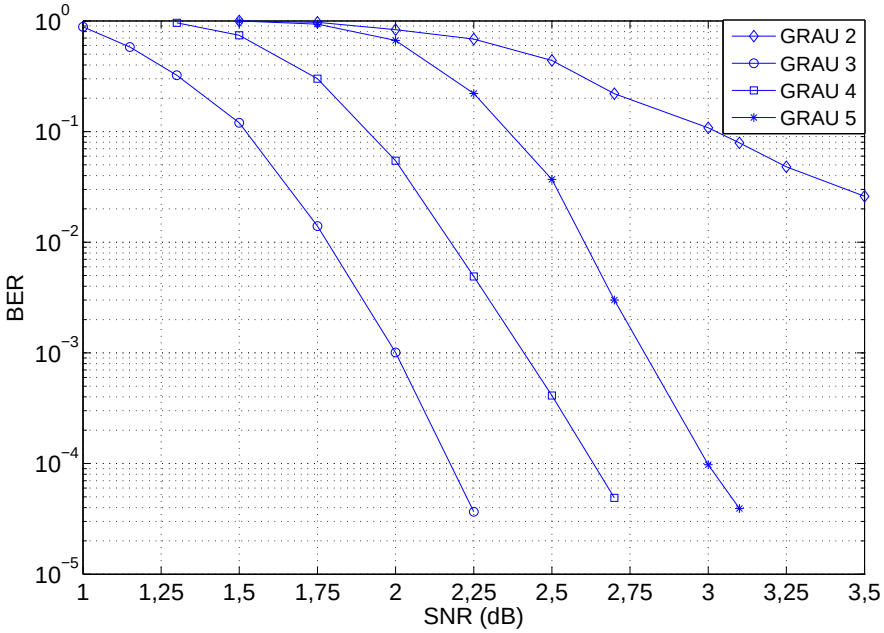


Figura 4.2: Desempenho do código regular $C(2000, 1000)$ variando-se o grau dos nós de variável, $d_c = 2, 3, 4, 5$ e mantendo fixo o número de iterações igual a 100.

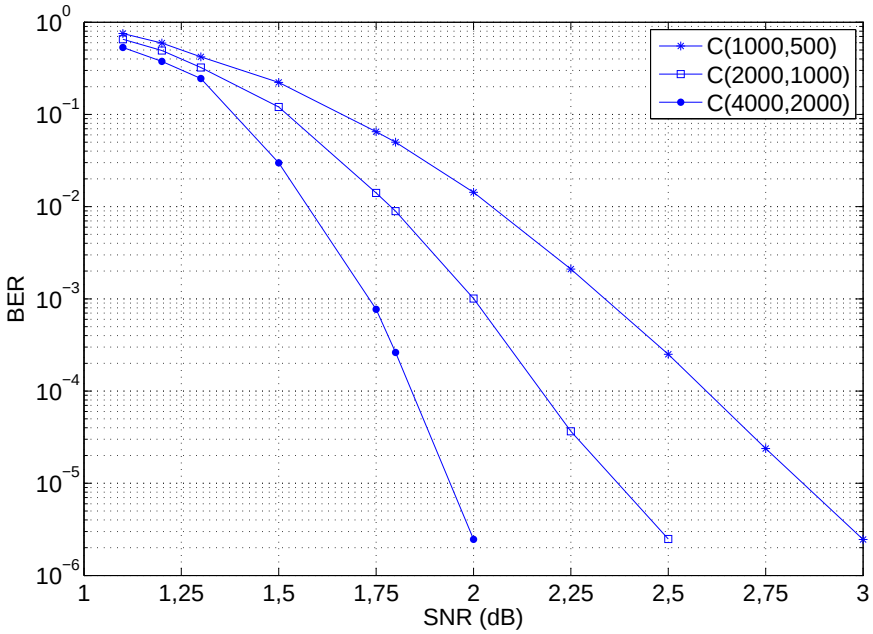


Figura 4.3: Desempenho de códigos LDPC regulares de taxa $0,5$, $d_c = 3$ para comprimentos 1000, 2000 e 4000 com o número de iterações igual a 100.

4.3.2 Desempenho de Códigos LDPC Irregulares

Compara-se dois códigos LDPC irregulares $C(2000, 1000)$ com o número máximo de ramos conectados aos nós de variável, J, igual a 4 e 15.
O polinômio $\lambda(x)$ adotado para $J = 4$ [23] é:

$$\lambda(x) = 0,38354x^1 + 0,04237x^2 + 0,57409x^3$$

e para $J = 15$ [23]:

$$\lambda(x) = 0,23802x^1 + 0,20997x^2 + 0,03492x^3 + 0,12015x^4 + 0,01587x^6 + 0,00480x^{13} + 0,37627x^{14}.$$

As curvas de desempenho BER $\times$ SNR destes dois códigos irregulares são apresentadas na Figura 4.4. Observamos que o código com $J = 15$ apresenta um desempenho melhor do que o código com $J = 4$. A comparação entre os códigos regulares e os irregulares de mesmo comprimento $C(2000, 1000)$ é mostrada na Figura 4.5. Considera-se um código irregular com grau máximo dos nós de variável igual a 4 ($J = 4$). Utilizando-se de (4.6) calculamos o grau médio dos nós de variável para o código irregular, $J_M = 0,38354 \times 2 + 0,04237 \times 3 + 0,57409 \times 4 = 3,019$. Com isso, torna-se interessante uma comparação desse código irregular com um código regular com $d_c = 3$ e $d_c = 4$.

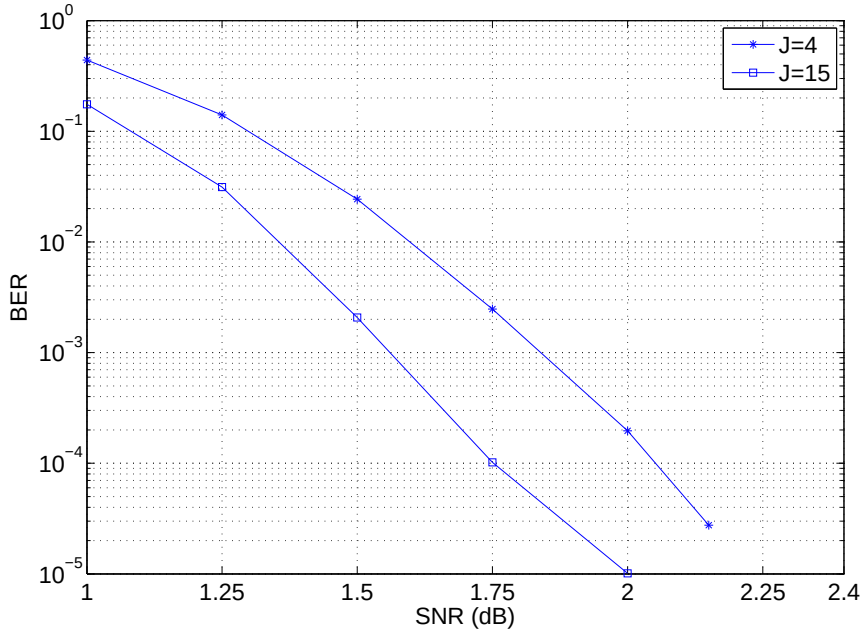


Figura 4.4: Desempenho de códigos LDPC irregulares $C(2000, 1000)$ para $J = 4$ e $J = 15$, 100 iterações.

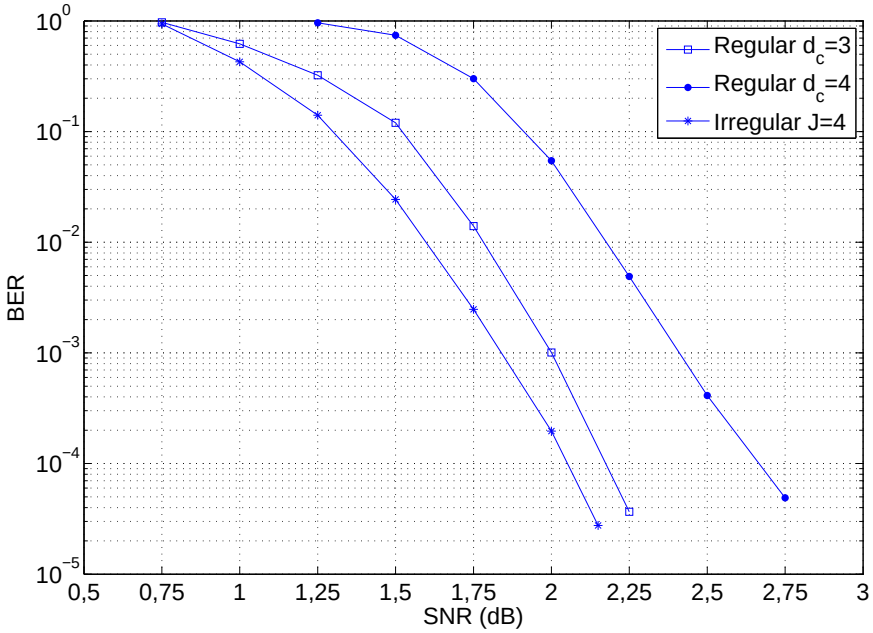


Figura 4.5: Desempenho de um código irregular $C(2000, 1000)$ *com* $J = 4$ *e códigos regulares* $C(2000, 1000)$ *com* $d_c = 3$ *e* $d_c = 4$*, 100 iterações.*

Quando se compara os desempenhos do código LDPC irregular $J = 4$ e do código LDPC regular $d_c = 4$, o código irregular apresenta um ganho de 0,6 dB para uma BER= 10^{-4}. De forma semelhante, quando a comparação do código irregular é feita com o código LDPC regular com $d_c = 3$, verifica-se que o ganho do código irregular sobre o regular é de 0,12 dB. Conclui-se que o desempenho obtido pelo código irregular é superior ao desempenho obtido pelos códigos regulares.

CAPÍTULO 5

ANÁLISE DO ASP EM UM CANAL AWGN COM QUANTIZAÇÃO SUAVE

Este capítulo visa descrever e analisar o desempenho de códigos LDPC utilizando a modulação BPSK e que são demodulados com um quantizador uniforme com 2^q níveis de quantização [12]. A decodificação com decisão suave tem por objetivo evitar a degradação de desempenho quando o sistema de comunicação utiliza quantização abrupta. Alguns algoritmos de decodificação que empregam decodificação suave podem ser vistos em [38]. Além disso, foram realizadas diversas simulações com a finalidade de definir o valor do limiar de quantização ótimo para cada valor de relação sinal ruído. Os códigos considerados neste capítulo foram obtidos com o algoritmo PEG.

5.1 Canal Discreto

Considere um canal discreto formado por um modulador BPSK, um canal AWGN, um demodulador e um quantizador que utiliza q bits de quantização, como é ilustrado na Figura 5.1. Sendo $X_k \in \{0, 1\}$ o símbolo de entrada do canal discreto no k-ésimo intervalo, o símbolo na entrada do quantizador é dado por:

$$R_k = \sqrt{E_s} S_k + N_k \tag{5.1}$$

em que E_s é a energia do sinal transmitido, $S_k = 2X_k - 1$ e N_k é uma variável aleatória Gaussiana de média zero e variância $\frac{N_0}{2}$. A variável aleatória R_k é a entrada de um quantizador

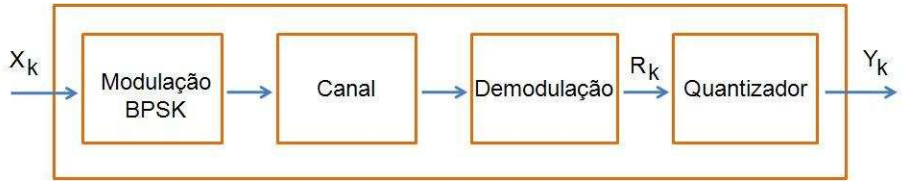


Figura 5.1: Diagrama de blocos de um canal discreto com entrada X_k e saída Y_k.

escalar uniforme com q bits de quantização, obtendo-se na saída do canal discreto a variável aleatória $Y_k \in \{0, 1, ..., 2^q - 1\}$. O quantizador uniforme é definido por um conjunto de limiares $\{T_j^1\}$ espaçados uniformemente.

A operação do quantizador fornece a saída do canal $Y_k = j$ se a condição $R_k \in (T_{j-1}^1, T_j^1)$ for satisfeita. Vale salientar que os limiares T_j^1 são espaçados uniformemente com o passo de quantização Δ da seguinte maneira:

$$T_j^1 = \begin{cases} -\infty, & \text{se } j = -1 \\ (j + 1 - 2^{q-1})\Delta, & \text{se } j = 0, 1, ..., 2^q - 2 \\ \infty, & \text{se } j = 2^q - 1. \end{cases}$$

Define-se o passo normalizado e os limiares normalizados, respectivamente, por $\delta = \Delta/\sqrt{E_s}$ e $T_j = \frac{T_j^1}{\sqrt{E_s}}$. Então, podemos definir $T_j = (j + 1 - 2^{q-1})\delta$, para $j = 0, 1, ...2^q - 2$.

O canal discreto pode ser modelado como um canal DMC com matriz de transição de probabilidade, $\mathbf{\Pi} = [\pi_{ij}]$, em que:

$$\pi_{ij} = P(Y_k = j | X_k = i)$$

para $i \in \{0, 1\}$, $j \in \{0, 1, ..., 2^q - 1\}$. Para o modelo considerado em (5.1), obtemos:

$$\begin{aligned} \pi_{ij} &= P(Y_k = j | X_k = i) \\ &= P(R_k \in (T_{j-1}^1, T_j^1) | X_k = i) \\ &= P(T_{j-1} - (2i - 1) < \frac{N_k}{\sqrt{E_s}} < T_j - (2i - 1)) \\ &= Q(\sqrt{2\gamma}(T_{j-1} - (2i - 1))) - Q(\sqrt{2\gamma}(T_j - (2i - 1))) \end{aligned}$$

em que $\gamma = E_s/N_0$ é a relação sinal ruído e $Q(x)$ é a função dada por:

$$Q(x) = 1/\sqrt{2\pi} \int_x^\infty e^{(-t^2/2)} dt.$$

Devido à simetria da constelação BPSK e das regiões de decisão, os elementos da matriz $\mathbf{\Pi}$ apresentam a seguinte propriedade:

$$\pi_{ij} = \pi_{0,\frac{j-(2^q-1)i}{(-1)^i}}. \tag{5.2}$$

A seguir, ilustraremos o cálculo da matriz $\mathbf{\Pi}$ para dois casos particulares, $q = 1$ e $q = 2$. O caso $q = 1$ corresponde ao canal BSC que é mostrado na Figura 3.3, cuja matriz $\mathbf{\Pi}$ é:

$$\mathbf{\Pi} = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$$

em que:

$$p = Q\left(\sqrt{\frac{2E_s}{N_0}}\right).$$

O caso $q = 2$ será descrito no exemplo a seguir.

Exemplo 5.1 *Para $q = 2$ temos que $X_k = \{0, 1\}$, $Y_k = \{0, 1, 2, 3\}$ e os elementos da matriz de probabilidade $\mathbf{\Pi}$ são dados por:*

$$\begin{aligned}
\pi_{00} &= P(R_k < -\Delta | X_k = 0) \\
&= P(-\sqrt{E_s} + N_k < -\delta\sqrt{E_s}) \\
&= P(N_k < -\delta\sqrt{E_s} + \sqrt{E_s}) \\
&= P(N_k < \sqrt{E_s}(1-\delta)) \\
&= 1 - Q\left(\frac{\sqrt{E_s}(-\delta+1)}{\sigma}\right) \\
&= Q\left(\sqrt{\frac{2E_s}{N_0}}(\delta-1)\right)
\end{aligned}$$

em que $\sigma^2 = \frac{N_0}{2}$.

$$\begin{aligned}
\pi_{01} &= P(-\delta\sqrt{E_s} < -\sqrt{E_s} + N_k < 0) \\
&= P((1-\delta)\sqrt{E_s} < N_k < \sqrt{E_s}) \\
&= Q\left(\sqrt{\frac{2E_s}{N_0}}(1-\delta)\right) - Q\left(\sqrt{\frac{2E_s}{N_0}}\right).
\end{aligned}$$

$$\begin{aligned}
\pi_{02} &= P(0 < -\sqrt{E_s} + N_k < \delta\sqrt{E_s}) \\
&= P(\sqrt{E_s} < N_k < \sqrt{E_s}(\delta+1)) \\
&= Q\left(\sqrt{\frac{2E_s}{N_0}}\right) - Q\left(\sqrt{\frac{2E_s}{N_0}}(\delta+1)\right).
\end{aligned}$$

$$\begin{aligned}\pi_{03} &= P(-\sqrt{E_s} + N_k > \delta\sqrt{E_s}) \\ &= Q\left(\sqrt{\frac{2E_s}{N_0}}(\delta+1)\right).\end{aligned}$$

Os elementos π_{1j} são obtidos via (5.2).

A sequência quantizada Y_k é a entrada do decodificador ASP que utilizará a seguinte LLR na fase de inicialização:

$$L\left(\frac{y_n}{x_n}\right) = \log\left(\frac{\pi_{0n}}{\pi_{1n}}\right), \text{ se } Y_n = j.$$

5.2 Resultado das Simulações

Será analisado o desempenho de diversos códigos LDPC com a finalidade de definir o passo de quantização ótimo para cada valor de SNR.

5.2.1 Códigos Regulares

Através da Figura 5.2 é possível analisar o desempenho do código LDPC $C(1000, 500)$ regular com $d_c = 3$, 4 níveis de quantização, $q = 2$, versus o passo de quantização δ para SNR iguais a 1,5 dB, 2 dB, 2,5 dB e 3 dB. Observa-se nesta figura que o valor ótimo de δ não varia significativamente para o intervalo de SNR analisado, ficando em torno de 0,6. O valor de δ que minimiza a BER para cada SNR é mostrada na Tabela 5.1. Estes serão considerados valores ótimos de δ. Um comportamento semelhante é observado para um código LDPC regular mais longo $C(2000, 1000)$ com $d_c = 3$, conforme é ilustrado na Figura 5.3 e na Tabela 5.2. Uma comparação do desempenho destes dois códigos de comprimentos diferentes pode ser vista na Figura 5.4. Os valores dos passos de quantização utilizados para cada SNR são indicados nas Tabelas 5.1 e 5.2 (primeira coluna).

Nas Figuras 5.5 e 5.6 são apresentados os desempenhos do código LDPC $C(2000, 1000)$ regular com $d_c = 3$ para SNR igual a 1,5 dB, 2 dB, 2,5 dB e 3 dB versus o passo de quantização, para $q = 3$ e $q = 4$, respectivamente. A Tabela 5.2 indica os valores ótimos de δ. Na Figura 5.7 é feita uma comparação do desempenho do código LDPC regular $C(2000, 1000)$ com $d_c = 3$, para $q = 1, 2, 3$ e 4. Observa-se que a medida que aumentamos o nível de quantização, o desempenho se aproxima do observado em um canal AWGN.

Tabela 5.1: Passo de quantização ótimo variando a SNR para o código LDPC regular $C(1000, 500)$, $d_c = 3$ *e* $q = 2$.

SNR	δ ótimo
1,50 dB	0,65
2,00 dB	0,60
2,50 dB	0,55
3,00 dB	0,50

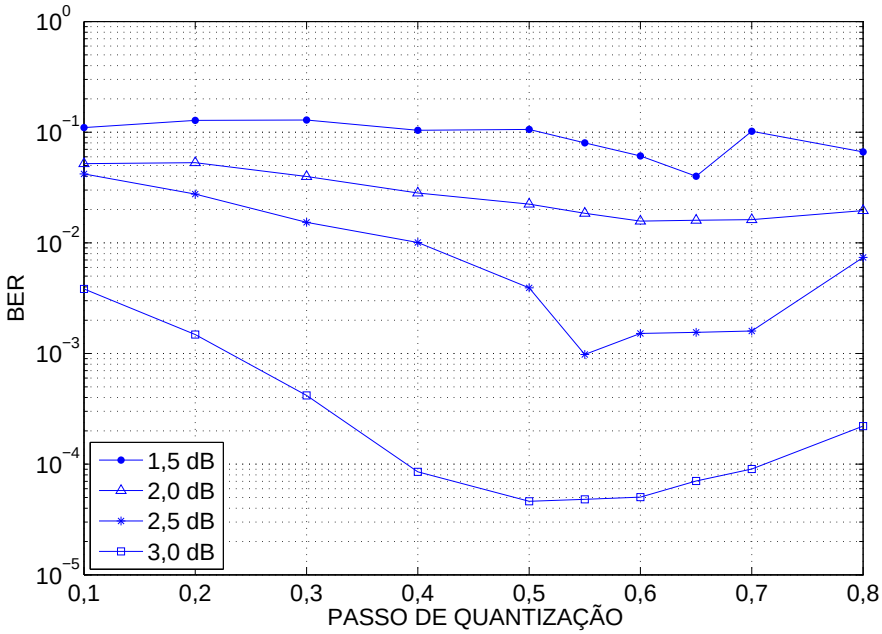


Figura 5.2: BER versus δ *para o código LDPC regular* $C(1000, 500)$ *com* $d_c = 3$, *para* $q = 2$.

Tabela 5.2: Passo de quantização ótimo variando a SNR para o código LDPC regular $C(2000, 1000)$, $d_c = 3$, *e* $q = 2, 3$ *e* 4.

SNR	δ ótimo para $q = 2$	δ ótimo para $q = 3$	δ ótimo para $q = 4$
1,50 dB	0,65	0,40	0,48
2,00 dB	0,63	0,33	0,46
2,50 dB	0,61	0,32	0,45
3,00 dB	0,60	0,20	0,43

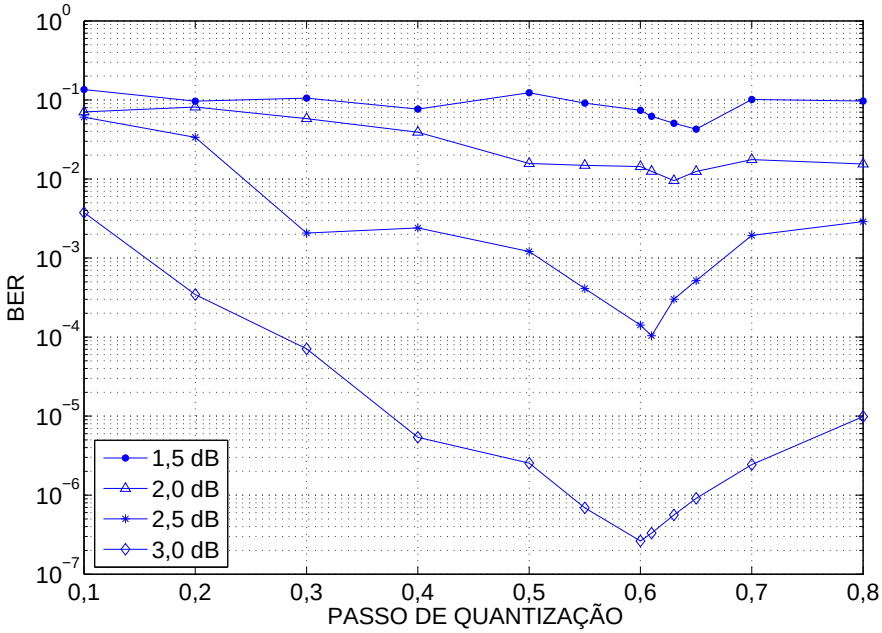


Figura 5.3: BER versus δ para o código LDPC regular com $d_c = 3$, $C(2000, 1000)$, para $q = 2$.

Observa-se também na Figura 5.7 que a perda de desempenho em relação ao AWGN para BER $= 10^{-5}$ é 1,75 dB para $q = 1$, 0,33 dB para $q = 2$, 0,17 dB para $q = 3$ e 0,10 dB para $q = 4$. O valor do passo de quantização ótimo para cada SNR é usado nesta figura e na Tabela 5.2. O emprego de decodificação abrupta ($q = 1$) provoca uma perda de desempenho de 1,75 dB em relação ao canal AWGN. Esta perda é consideravelmente reduzida com o emprego de apenas 4 níveis. O uso de $q = 3$ pode ser satisfatório, pois o desempenho é aproximado ao obtido com $q = 4$.

5.2.2 Códigos Irregulares

Uma análise semelhante à realizada na subseção anterior é feita nesta subseção para o código irregular $C(2000, 1000)$ com $J = 15$, conforme pode ser observado nas Figuras 5.8 - 5.10. A Tabela 5.3 mostra os valores ótimos dos níveis de quantização encontrados a partir destas figuras. Na Figura 5.11 é feita uma comparação do desempenho do código LDPC irregular $C(2000, 1000)$, $J = 15$, para $q = 1, 2, 3, 4$. Observa-se que a medida que aumentamos o nível de quantização, o desempenho do código se aproxima do desempenho do canal AWGN.

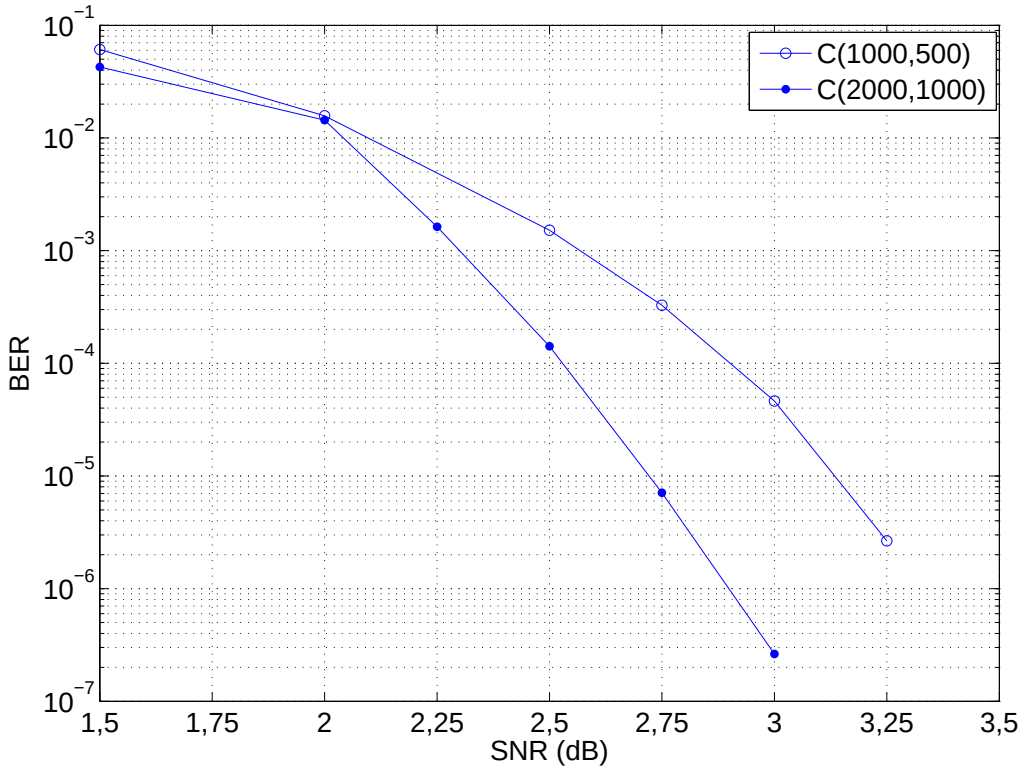


Figura 5.4: BER versus SNR para os códigos LDPC regulares $C(1000, 500)$ *e* $C(2000, 1000)$ *com* $d_c = 3$*, para* $q = 2$*.*

Tabela 5.3: Passo de quantização ótimo variando a SNR para o código LDPC irregular $C(2000, 1000)$ *com* $J = 15$*, e* $q = 2, 3$ *e* 4*.*

SNR	δ ótimo para $q = 2$	δ ótimo para $q = 3$	δ ótimo para $q = 4$
1,00 dB	0,37	0,26	0,30
1,50 dB	0,36	0,25	0,27
2,00 dB	0,35	0,24	0,25
2,50 dB	0,32	0,22	-

Verifica-se também na Figura 5.11 que a perda de desempenho em relação ao AWGN para BER $= 10^{-5}$ é $1,82$ dB para $q = 1$, $0,48$ dB para $q = 2$, $0,26$ dB para $q = 3$ e $0,14$ dB para $q = 4$. Estes valores são superiores ao caso do código regular analisado na seção anterior.

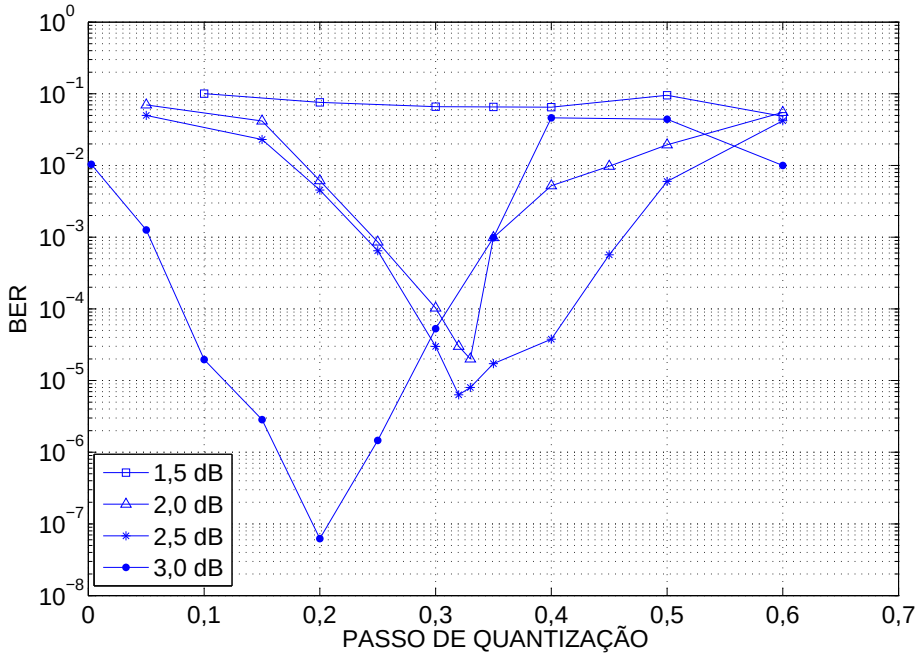


Figura 5.5: BER versus δ para o código LDPC regular $C(2000, 1000)$, $d_c = 3$, para $q = 3$.

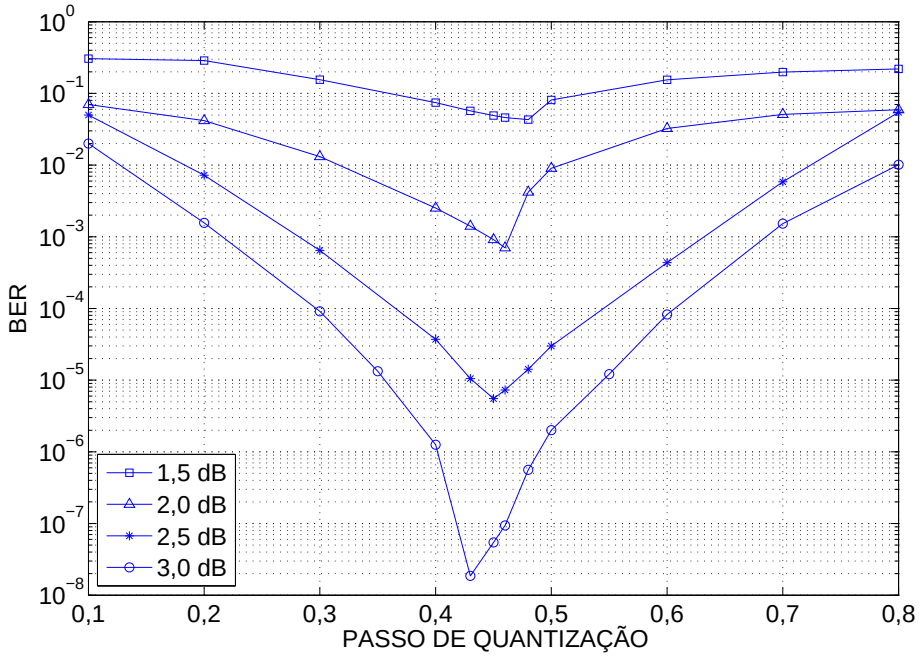


Figura 5.6: BER versus δ para o código LDPC regular $C(2000, 1000)$ com $d_c = 3$, para $q = 4$.

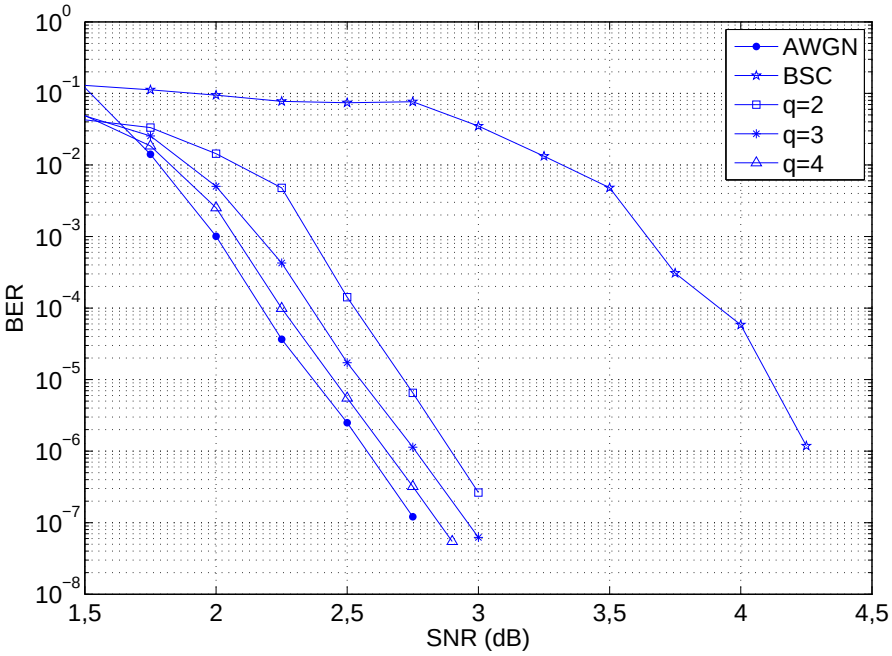


Figura 5.7: BER versus SNR para o código LDPC regular $C(2000, 1000)$ *com* $d_c = 3$*, para* $q = 1, 2, 3, 4$ *e o canal AWGN.*

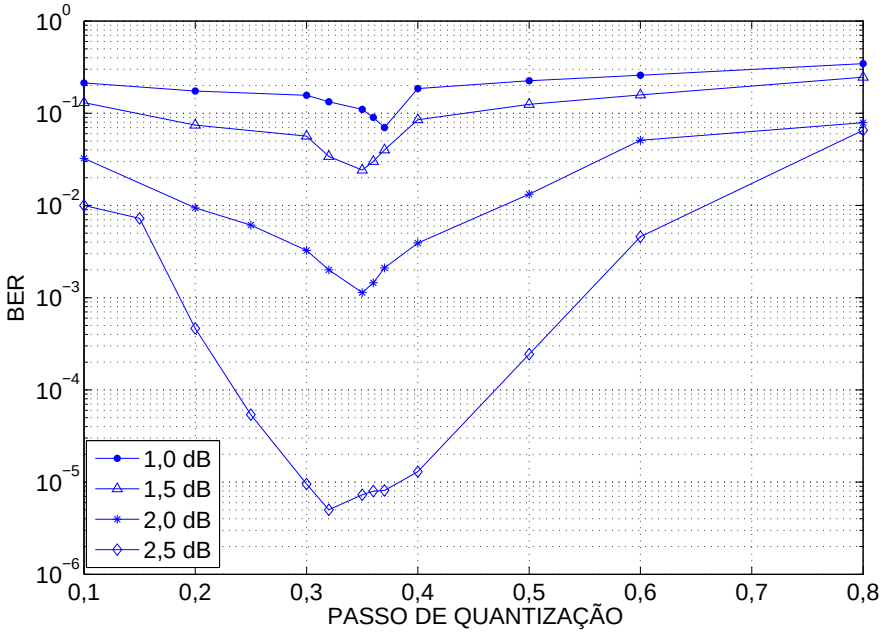


Figura 5.8: BER versus δ *para o código LDPC irregular* $C(2000, 1000)$ *com* $J = 15$*, para* $q = 2$.

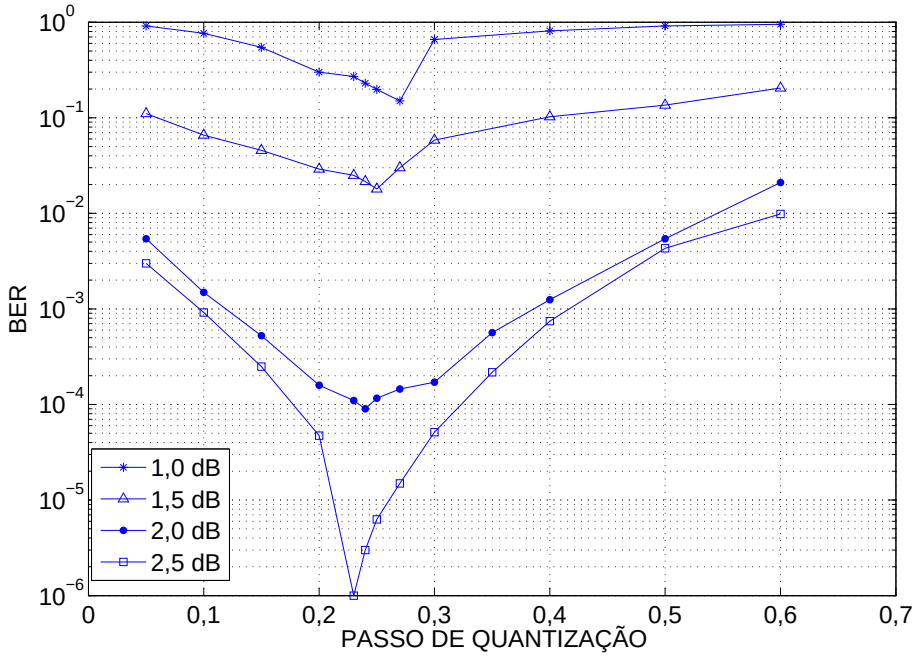


Figura 5.9: BER versus δ para o código LDPC irregular $C(2000, 1000)$ com $J = 15$, para $q = 3$.

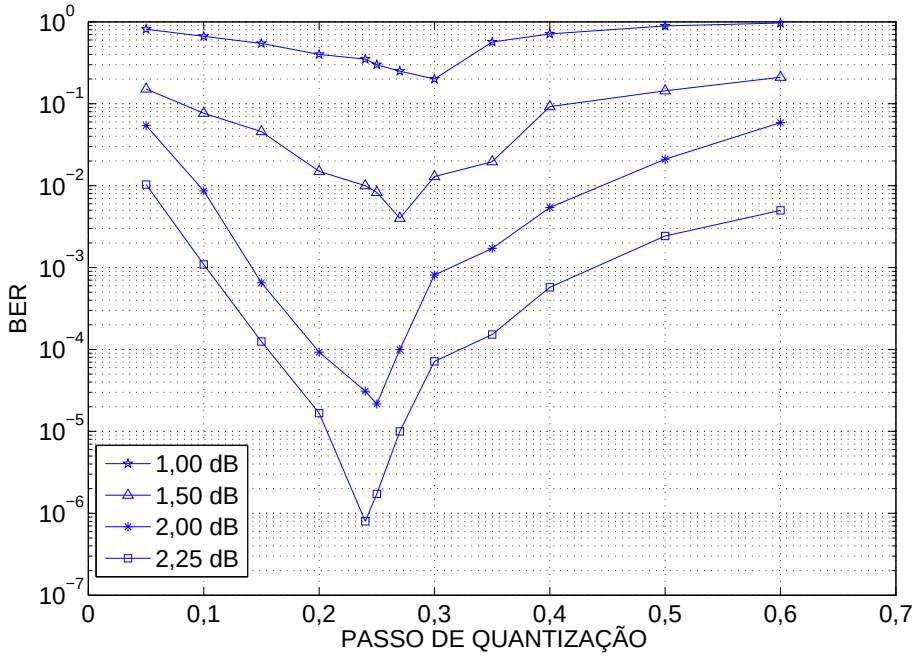


Figura 5.10: BER versus δ para o código LDPC irregular $C(2000, 1000)$ com $J = 15$, para $q = 4$.

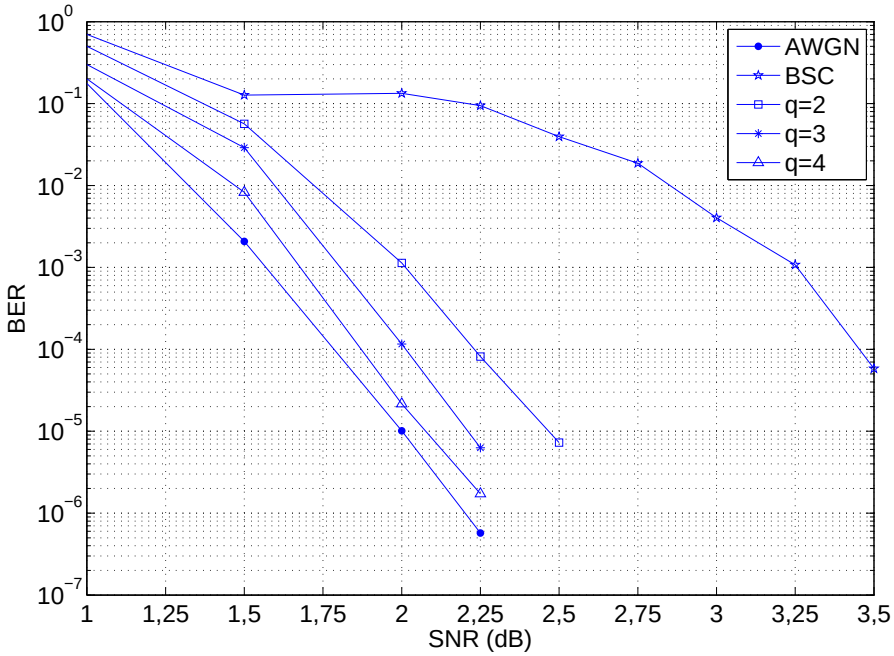


Figura 5.11: BER versus SNR para o código LDPC irregular $C(2000, 1000)$ *com* $J = 15$*, com* $q = 1, 2, 3, 4$ *e o canal AWGN.*

capítulo 6

Conclusões

Nesse trabalho foram descritas as etapas do algoritmo de transferência de mensagens soma-produto (ASP). Logo após, foram apresentadas técnicas de construções de códigos LDPC. Com os códigos obtidos pelo algoritmo PEG, foram realizadas simulações computacionais em um canal AWGN com modulação BPSK com a finalidade de analisar a influência de diversos parâmetros no desempenho do código. Com estas simulações pudemos verificar que o número de iterações do ASP deve ser igual a 100. Em seguida, consideramos um canal discreto (entrada binária, saída 2^q-ária) obtido com a incorporação de um quantizador uniforme ao sistema de comunicações. O passo de quantização ótimo para vários parâmetros do código e do canal foi identificado. Com esse estudo, verificou-se a importância da quantização através da comparação da perda de codificação entre o canal AWGN e o BSC ($q = 1$). Além disso, observou-se que o resultado obtido para $q = 3$, com uma perda de codificação em relação ao canal AWGN em torno de 0,2 dB, foi próximo do obtido com $q = 4$ em torno de 0,1 dB. Este ganho de codificação adicional obtido com $q = 4$ pode ser considerado pequeno, não compensando o aumento do nível de complexidade.

Dentre as principais contribuições deste trabalho, destacam-se a análise e a busca de níveis ótimos de quantização e do passo de quantização para os códigos LDPC regulares e irregulares. Enquanto esta análise é bastante difundida para códigos convolucionais [39], não encontramos na literatura estudo semelhante para códigos LDPC.

6.1 Sugestões de Trabalhos Futuros

Como sugestões para trabalhos futuros relativos aos temas apresentados nessa dissertação, podemos indicar:

- Projetar a distribuição de grau de um código irregular para o canal discreto, variando o valor de q.
- Estudo do desempenho de códigos LDPC usando um quantizador não uniforme.
- Analisar o efeito da quantização para códigos longos, por exemplo, $N = 10^4$ ou $N = 10^5$.
- Estudo do desempenho de códigos LDPC quantizados utilizando códigos de taxas diferentes de $1/2$.
- Aplicação das ferramentas apresentadas para os códigos turbos.

Bibliografia

[1] R.G. Gallager, "Low-density parity-check codes," *IEEE Transactions on Information Theory*, vol 8, no. 1, pp. 21-28, janeiro de 1962.

[2] S. Y. Chung, G. D. Forney, T. J. Richardson, R. Urbanke , "On the design of LDPC Codes within 0.0045 dB of the Shannon limit," *IEEE Comunications Letters,* vol. 5, no. 2, pp. 58-60, fevereiro de 2001.

[3] F. R. Kschischang, B. J. Frey, H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory,* vol. 47, no. 2, pp. 498-519, fevereiro de 2001.

[4] T. B. Iliev, G. V. Hristov, P. Z. Zahariev, M. P. Iliev, "Application and evaluation of the LDPC codes for the next generation communication systems," *Springer NetherlandBell,* pp. 532-536, agosto de 2008.

[5] T. F. Pegoraro, F. A. L. Gomes, F. Lumertz, R. R. Lopes, F. C. A. Oliveira, R. Gallo, M. C. Paiva, J. S. Panaro, "Codificação LDPC em sistemas de televisão digital," *Revista Telecomunicações-INATEL,* vol. 9, pp. 532-536, dezembro de 2006.

[6] D. J. C. Mackay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399-431, março de 1999.

[7] I. B. Bocharova, B. D. Kudryashov, R. V. Satyukov, "Short quasi-cyclic LDPC codes from convolutional codes," *in Proc. International Symposium on Information Theory,* Korea, 2009, pp. 551-554.

[8] W. M. Tam, F. C. M. Lau, C. K. Tse, "A class of QC-LDPC codes with low encoding complexity and good error performance," *IEEE Communications Letters*, vol. 14, no. 2, pp. 169-171, fevereiro de 2010.

[9] M. P. C. Fossorier, "Quasi-cyclic low-density partity-check codes from circulant permutation matrices," *IEEE Transactions on Information Theory,* vol. 50, no. 8, pp. 1788-1793, agosto de 2004.

[10] X. Y. Hu, E. Eleftheriou, D. M. Arnold, "Regular and irregular progressive edge growth Tanner graphs," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386-398, janeiro de 2005.

[11] A. Venkiah, D. Declercq, C. Poulliat, "Design of cages with a randomized progressive edge-growth algorithm," *IEEE Comunications Letters*, vol. 12, no. 4, pp. 301-303, abril de 2008.

[12] J. Singh, O. Dabeer, U. Madhow, "On the limits of communication with low-precision analog-to-digital conversion at the receiver," *IEEE Transactions on Communications*, vol. 57, no. 12, pp. 3629-3639, dezembro de 2009.

[13] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533-547, setembro de 1981.

[14] H. A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, pp. 28-41, janeiro de 2004.

[15] Shu Lin, D. J. Costello Jr, "**Error Control Coding**," *Prentice Hall*, 2004.

[16] D. A. M. Rocha, "Algoritmos avançados, Grafos," *Website, acessado em abril de 2010. [Online]. Disponível: http://www.slideshare.net/alexandrend/grafos-parte-1-introduo*.

[17] A. C. Mariani, "Conceito básicos da teoria de grafos," *Website, acessado em março de 2010. [Online]. Disponível: http://www.inf.ufsc.br/grafos/definicoes/definicao.html*.

[18] F. R. Kschischang, "Codes defined on graphs," *IEEE Communications Magazine*, pp. 118-125, agosto de 2003.

[19] T. K. Moon, "**Error Correction Coding**," *Wiley InterScience*, 2005.

[20] C. B. Schlegel, L. C. Pérez, "**Trellis and turbo coding**," *Wiley Interscience*, 2004.

[21] D. J. C. Mackay, R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457-458, março de 1997.

[22] C. E. Shannon, "A mathematical theory of comunication," *Bell System Technical Journal,* vol. 127, no. 1, pp. 379-422 e 623-656, outubro de 1948.

[23] T.J. Richardson, M. A. Shokrollahi, R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619-637, fevereiro de 2001.

[24] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 585-598, fevereiro de 2001.

[25] A. Prabhakar, K. R. Narayanan, "Pseudorandom construction of LDPC codes using linear congruential sequences," *IEEE Transactions on Communications*, vol. 50, no. 9, pp. 1389-1396, setembro de 2002.

[26] Y. Kou, S. Lin, M. P. C. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2711-2736, novembro de 2001.

[27] S. Lin, D. J. C. JR., "**Error Control Coding**," *Prentice Hall*, 2004.

[28] D. J. C. Mackay, "Online database of low-density parity-check codes," *Website, acessado em abril de 2009. [Online]. Disponível: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html*.

[29] R. M. Tanner, T. E. Fuja, D. J. Costello, "LDPC block and convolutional codes based on circulant matrices," *IEEE Transactions on Information Theory,* vol. 50, no. 12, pp. 2966-2984, dezembro de 2004.

[30] G. Liva, W. E. Ryan, M. Chiani, "Quasi-cyclic generalized LDPC codes with low error floors," *IEEE Transactions on Information Theory,* vol. 56, no. 1, pp. 49-57, janeiro de 2008.

[31] S. Johnson, S. R. Weller, "A family of irregular LDPC codes with low encoding complexity," *IEEE Communications Letters,* vol. 7, no. 2, pp. 79-81, fevereiro de 2003.

[32] P. Elias, " Error-free coding," *IEEE Transactions on Information Theory*, vol. 4, no. 4, pp. 29-37, setembro de 1954.

[33] M. Lentmaier, K. Zigangirov, "On generalized low-density parity-check codes based on Hamming component codes," *IEEE Commununications Letters,* vol. 3, no. 8, pp. 248-250, agosto de 1999.

[34] V. Zyablov, M. Loncar, R. Johannesson, P. Rybin, "On the asymptotic perfomance of low-complexity decoded LDPC codes with constituent Hamming Codes," *in Proc. 5th International Symposium on Turbo codes and related topics*, Suíça, outubro de 2008, pp. 174-179

[35] M. Lentmaier, K. Zigangirov, "Iterative decoding of generalized low-density parity-check codes," *in Proc. IEEE International Symposium on Information Theory*, Cambridge, MA, 1998, pp. 149.

[36] J. Boutros, O. Pothier, G. Zémmor, "Generalized low density (Tanner) codes," *in Proc. IEEE International Community Conference (ICC),* vol. 1, Vancouver, Canadá, junho de 1999, pp. 441-445.

[37] N. Miladinovic, M. Fossorier, "Generalized LDPC codes with Reed-Solomon and BCH codes as component codes for binary channels," *in Proc. IEEE Global Community Conference (GLOBECOM)*, St. Louis, MO, USA, novembro de 2005, pp. 1239-1244

[38] M. S. Alencar, "**Telefonia celular digital**," *Ed. Erica*, 2004.

[39] A.J. Viterbi, J.K. Omura, "**Principles of digital communications and coding**," *McGraw-Hill*, 1979.