Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica e de Computadores

CÓDIGOS BINÁRIOS DEFINIDOS POR MATRIZES DE TESTE DE PARIDADE ESPARSAS ALGORITMOS DE DESCODIFICAÇÃO

Marco Alexandre Cravo Gomes

Coimbra
Novembro 2003

CÓDIGOS BINÁRIOS DEFINIDOS POR MATRIZES DE TESTE DE PARIDADE ESPARSAS ALGORITMOS DE DESCODIFICAÇÃO

por

Marco Alexandre Cravo Gomes

Licenciado em Engenharia Electrotécnica e Computadores
pelo Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade Ciências e Tecnologia da Universidade Coimbra

Dissertação submetida para a satisfação parcial dos requisitos do grau de Mestre em Engenharia Electrotécnica e de Computadores Especialização em Sistemas de Telecomunicações

Departamento de Engenharia Electrotécnica e de Computadores
Faculdade Ciências e Tecnologia da Universidade Coimbra

Coimbra
Novembro 2003

Dissertação realizada sob a orientação de

Vítor Manuel Mendes da Silva

Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores

da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

AGRADECIMENTOS

Ao meu orientador, Doutor Vítor Silva, pelo incentivo, pela inesgotável dedicação e sobretudo pela sua amizade.

Ao Instituto de Telecomunicações, pelos meios disponibilizados.

Aos meus amigos e às pessoas do DEEC (professores, alunos e funcionários), por todo o seu apoio.

Aos meus pais, pelo seu amor que fez de mim o que hoje sou.

Ao meu irmão, por ser um “Irmão”.

À Sara pelo amor, o carinho e aquele abraço revigorante de todos os dias.

A todos,

Muito Obrigado

À Sara e aos meus Pais . . .

RESUMO

Esta dissertação aborda o estudo de algoritmos iterativos para a descodificação de Códigos Binários Definidos por Matrizes de Teste de Paridade Esparsas, também conhecidos por *Low Density Parity-Check Codes* (LDPC). São considerados como uma das classes de códigos que melhor desempenho apresentam e que num futuro próximo poderão integrar várias normas de comunicação digital. O desenvolvimento científico deste tema, verificado ao longo da última década, é impressionante.

Apresentamos um estudo sobre técnicas de descodificação iterativas baseadas no algoritmo Soma de Produtos (SPA). É feita uma análise comparativa e crítica dos vários algoritmos do ponto de vista do desempenho e da complexidade computacional. São abordadas várias técnicas de simplificação do algoritmo SPA, com particular destaque para o algoritmo Soma Mínima.

Finalmente, propomos para o algoritmo SPA uma nova técnica de normalização, cujos resultados obtidos evidenciam uma melhoria significativa do desempenho para códigos LDPC longos.

ABSTRACT

The Iterative Decoding of Low Density Parity-Check Codes (LDPC) is the main subject of this thesis. LDPC are seen as one of most powerful classes of error correcting codes which in a near future will probably be included in new digital data transmission standards. In the last decade, the scientific development of this theme was impressive.

The Sum Product Algorithm (SPA) and their simplifications are deeply study, with special attention to Min-Sum algorithm. A critical and comparison analysis of different SPA variants is made from the performance and decoding complexity point of view.

Finally, we propose a new normalization technique which significantly improves the performance of SPA for long LDPC codes.

PALAVRAS-CHAVE:

códigos definidos por matrizes de teste de paridade esparsas (LDPC), descodificação iterativa, algoritmo soma de produtos (SPA), algoritmo soma de produtos normalizado, algoritmo soma mínima (MS-LSPA), algoritmo soma mínima normalizado.

KEYWORDS:

low density parity-check codes (LDPC), iterative decoding, sum product algorithm (SPA), belief propagation (BP), normalized sum product algorithm, min-sum algorithm (MS-LSPA), normalized min-sum algorithm.

CONTEÚDO

ÍNDICE DE ALGORITMOS

GLOSSÁRIO

ADC	Conversor Analógico Digital (*Analog Digital Converter*)
APP	Probabilidade À Posteriori (*A Posteriori Probability*)
APP-LSPA	*A-Posteriori Probability Logarithmic Sum Product Algorithm*
AWGN	Ruído Branco Aditivo Gaussiano (*Additive White Gaussian Noise*)
BCH	Código de Bose, Chaudry e Hocquehen (*Bose-Chaudry-Hocquehen Code*)
BER	Taxa de Bits Erros (*Bit Error Rate*)
BF	Algoritmo de Troca de Bits (*Bit Flipping Algorithm*)
BN	Nodo de Bit (*Bit Node*)
BP	*Belief Propagation*
BPSK	Modulação de Desvio de Fase Binário (*Binary Phase Shift Keying*)
BSC	Canal Binário Simétrico (*Binary Symmetric Channel*)
BWBF	Algoritmo de Troca de Bits Pesado com Arranque Programado (*Bootstrapped Weight Bit Flipping Algorithm*)
CCSDS	Comité Consultivo para os Sistemas de Dados Espaciais (*Consultative Committee for Space Data Systems*)
CN	Nodo de Teste (*Check Node*)
DSC	*Difference-Set Cyclic Codes*
DSP	Processador Digital de Sinal (*Digital Signal Processor*)
DVB	Transmissão de Vídeo Digital (*Digital Video Broadcast*)

GF(*n*)	Campo de Galois de ordem *n* (*Galois Field*)
HD	Descodificação por Decisão Firme (*Hard Decoding*)
LLR	Máxima Verosimilhança Logarítmica (*Log Likelihood Ratio*)
LSPA	Algoritmo Soma de Produtos no Domínio Logarítmico (*Logarithmic Sum Product Algorithm*)
MER	Taxa de Mensagens Erradas (*Message Error Rate*)
MS-LSPA	Algoritmo Soma Mínima (*Min-Sum Sum Product Algorithm*)
SD	Descodificação por Decisão Programada (*Soft Decoding*)
SNR	Relação Sinal Ruído (*Signal-to-Noise Ratio*)
SPA	Algoritmo Soma de Produtos (*Sum Product Algorithm*)
SPA-GNP	Algoritmo Soma de Produtos com Gestão Não Probabilística
SPA-GP	Algoritmo Soma de Produtos com Gestão Probabilística
TC	Turbo Códigos (*Turbo Codes*)
TG	Gráfico de Tanner (*Tanner Graph*)
WBF	Algoritmo de Troca de Bits Pesado (*Weight Bit Flipping Algorithm*)

CAPÍTULO 1

INTRODUÇÃO

Em 1948, Shannon [Shan] lançou as bases da Teoria de Informação. Cedo se compreendeu a importância de dispor de técnicas de codificação de canal que permitissem uma transmissão e um armazenamento robusto de dados.

Shannon não só determinou os limites teóricos para a taxa máxima de transmissão de dados através de um canal de comunicação, como também, colocou um enorme desafio de investigação: determinar uma técnica de codificação de canal capaz de atingir o limite por ele deduzido. A pesquisa não cessou desde então. Se durante quase 50 anos o progresso foi lento, em 1993 Berrou, Glavieux e Thitimajshimi [BGT] apresentaram uma nova técnica de codificação designada por *Turbo Coding* (TC), que revolucionou toda a investigação realizada até então. Baseado em técnicas de descodificação iterativas, foi possível pela primeira vez aproximar o limite de Shannon a menos de uma fracção de dB. Os melhoramentos conseguidos foram tão surpreendentes e a técnica proposta tão revolucionária que foram precisos vários anos para compreender e absorver os novos conceitos.

À medida que a comunidade científica concentrava a sua atenção no novo método, tornou-se claro que os seus fundamentos tinham sido lançados muitos anos antes, em 1960 por Robert Gallager [Gal1], [Gal2], que na sua tese de doutoramento tinha proposto uma nova classe de códigos baseados em *Matrizes de Teste de Paridade Esparsas*, conhecidos por *Low Density Parity-Check Codes* (LDPC), e um algoritmo de descodificação iterativo designado por *Algoritmo Soma de Produtos* (SPA). Foi Makay e Neal [Mac2], [MN1], [MN2], que redescobriram os códigos LDPC e que confirmaram

as suas excelentes propriedades de correcção de erros, tendo provado que à semelhança dos *Turbo Codes*, os LDPC's conseguiam atingir uma probabilidade de erro muito próxima do limite de Shannon [Shan], existindo mesmo alguns exemplos com um desempenho superior relativamente aos melhores *Turbo Codes* conhecidos.

1.1 MOTIVAÇÃO

A recente adopção pela nova norma de transmissão digital de vídeo via satélite (DVB-S2) de uma solução de codificação de canal baseado num código LDPC de comprimento 64800 capaz de suportar taxas de informação de $1/2$, $2/3$, $3/4$, $4/5$, $5/6$, $7/8$, $8/9$ e $9/10$, associado a um código BCH, demonstra bem a importância dos códigos LDPC no panorama actual, como uma das classes de códigos de correcção de erros que melhor desempenho apresentam e que num futuro próximo poderá a vir a fazer parte de outras normas[1]. Os desafios que daqui advêm na construção de sistemas codificadores e descodificadores capazes de cumprir os requisitos temporais impostos pelas elevadas taxas de transmissão dos sistemas de hoje, aliado às não menos importantes restrições de custo, têm motivado um estudo aprofundado dos códigos LDPC ao nível das melhores técnicas de codificação e descodificação a serem implementadas quer em *software* quer em *hardware*.

1.2 OBJECTIVOS

Neste trabalho focaremos apenas a problemática da descodificação dos códigos LDPC. Os objectivos fundamentais deste estudo foram:

---

[1] Recentemente o *Jet Propulsion Laboratory* apresentou uma proposta baseada nos códigos LDPC ao Comité Consultivo para os Sistemas de Dados Espaciais (CCSDS) e a indústria de armazenamento de dados mostrou já um sério interesse em integrar os códigos LDPC nos sistemas de armazenamento do futuro.

- Simplificação do algoritmo de descodificação SPA proposto por Gallager [Gal1], [Gal2], com vista a uma implementação mais eficiente tanto em *software* como em *hardware*.

- Análise de técnicas de melhoria do desempenho do algoritmo SPA e das suas simplificações.

- Realizar uma análise comparativa e crítica das várias soluções propostas do ponto de vista do desempenho e da complexidade computacional.

1.3 CONTRIBUIÇÕES DA DISSERTAÇÃO

Nesta dissertação propomos um novo método de normalização para o algoritmo LSPA que melhora significativamente o desempenho do algoritmo para códigos LDPC longos, sem que tal implique um aumento significativo da complexidade computacional.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Este documento é composto por 6 capítulos que abordam o trabalho realizado no âmbito desta dissertação de mestrado.

Sendo os códigos LDPC, lineares e descritos por matrizes de paridade esparsas, faremos no capítulo 2 uma introdução aos códigos lineares. São abordadas as suas características, formas de representação, capacidades de detecção e a descodificação pelo *método do síndroma*.

No capítulo 3, será feita uma descrição das características principais dos códigos LDPC. Procurar-se-á evidenciar o seu excelente desempenho e a importância da representação de *Tanner* e do conceito de *giro* no projecto de bons códigos LDPC. Serão apresentadas algumas estratégias para a construção de códigos. Por fim, será abordada de forma muito sucinta como poderá ser realizada a sua codificação.

A descodificação de códigos LDPC é abordada com detalhe no capítulo 4. Começaremos por abordar os *algoritmos de troca de bits* (BF) essenciais para uma melhor compreensão dos princípios que regem a descodificação iterativa SPA. A

generalidade do capítulo será dedicada ao estudo do algoritmo SPA e suas simplificações.

No capítulo 5 serão apresentados resultados experimentais relativos aos diversos algoritmos apresentados no capítulo 4, procurando realizar uma comparação crítica segundo o ponto de vista do desempenho e da complexidade computacional. Será também proposto um novo método de normalização para o algoritmo SPA.

No capítulo 6 apresentaremos as principais conclusões extraídas do trabalho realizado, terminando com algumas sugestões de trabalho futuro.

CAPÍTULO 2

CÓDIGOS DE BLOCO LINEARES

Uma classe muito importante de códigos de detecção e correcção de erros são os códigos lineares. Sendo os códigos LDPC lineares e descritos por matrizes de paridade esparsas, apresentaremos neste capítulo uma pequena introdução à teoria subjacente a este tipo de códigos.

2.1 CARACTERÍSTICAS GERAIS E SUA REPRESENTAÇÃO

Um código corrector é de bloco se a cada mensagem de símbolos do alfabeto X, de comprimento fixo k, o codificador de canal fizer corresponder uma palavra de código de comprimento fixo n, com $n > k$. A diferença $n - k$ representa o número de símbolos de teste utilizados na detecção e correcção de erros. No caso de o alfabeto ser binário, $X = \{0,1\}$, o código diz-se binário[2].

Uma classe importante de códigos correctores são os códigos lineares, na medida em que estes podem ser descritos matematicamente com base na teoria dos espaços vectoriais. Existem várias formas de definir um código linear. Para o caso dos códigos binários lineares utiliza-se aritmética módulo 2 e, como tal, dizemos que um código

---

[2] Nota: Neste trabalho consideramos apenas códigos binários, embora existam exemplos de bons códigos não-binários lineares como seja o caso dos Reed-Solomon.

binário é linear se a soma (módulo 2) de quaisquer palavras de código é ainda uma palavra de código. Tendo em conta as propriedades da adição módulo 2, facilmente concluímos que a soma de uma palavra de código consigo própria é o vector nulo, pelo que uma das condições necessárias (mas não suficiente) para que um código seja linear é que uma das palavras de código seja o vector nulo.

A teoria de Galois [Mac1], permite generalizar a definição dada para um código binário linear descrito em GF(2) para um código linear não binário definido em GF(q).

Existem outras definições [Wel1], [Nay] que nos conduzem às várias formas de representação de um código binário linear, que passaremos a referir.

2.1.1 MATRIZ GERADORA

Dado um código binário linear (n,k), ou seja, um código em que as palavras são vectores de dimensão n do tipo,

$$\mathbf{c} = \left[c_0\ c_1 \cdots c_{n-1}\right] \tag{2.1}$$

e as mensagens a codificar são vectores de dimensão k (com $n > k$) do tipo

$$\mathbf{m} = \left[m_0\ m_1 \cdots m_{k-1}\right], \tag{2.2}$$

a forma mais simples de representar esse código é com base numa matriz $\mathbf{G}$ de dimensões $k \times n$. As palavras de código podem ser obtidas a partir da seguinte equação

$$\mathbf{c} = \mathbf{m}\,\mathbf{G}\,. \tag{2.3}$$

Devido ao facto de cada palavra de código ser gerada a partir do produto da mensagem $\mathbf{m}$ pela matriz $\mathbf{G}$, esta é designada por *Matriz Geradora*.

Facilmente se conclui, a partir da equação (2.3), que cada bit da palavra de código é uma combinação linear dos bits da mensagem. No entanto, esta observação não é suficiente para afirmar que o conjunto dos vectores $\left\{\mathbf{c}_0, \mathbf{c}_1, \cdots, \mathbf{c}_{2^k-1}\right\}$, obtidos a partir da equação (2.3) para o conjunto de mensagens $\left\{\mathbf{m}_0, \mathbf{m}_1, \cdots, \mathbf{m}_{2^k-1}\right\}$, constitui um código útil. De facto, estamos perante um código se mensagens diferentes derem origem a palavras de código diferentes. Só desta forma é possível a sua descodificação. Ora, tendo por base (2.3), facilmente se conclui que cada elemento de $\left\{\mathbf{c}_0, \mathbf{c}_1, \cdots, \mathbf{c}_{2^k-1}\right\}$ é uma combinação linear das linhas da matriz $\mathbf{G}$, pelo que é necessário que estas sejam linearmente independentes e, nesse caso, todos os vectores $\mathbf{c}_i$, com $i = 0, \cdots, 2^k - 1$, são distintos.

Resumindo, podemos dizer que qualquer matriz $\mathbf{G}$, de dimensões $k \times n$ (com $n > k$), com característica k, define um código binário linear (n, k).

No caso de o código ser sistemático as palavras de código são (por ex.) do tipo

$$\mathbf{c} = [c_0\ c_1 \cdots c_{n-1}] = [b_0\ b_1 \cdots b_{n-k-1}\ m_0\ m_1 \cdots m_{k-1}], \tag{2.4}$$

onde os bits b_i, com $i = 0, \cdots, n-k-1$, são designados por bits de teste de paridade. Neste caso os bits de mensagem são transmitidos de forma separada dos bits de teste de paridade, sendo estes obtidos como combinação linear dos primeiros,

$$\mathbf{b} = [b_0\ b_1 \cdots b_{n-k-1}] = \mathbf{m}\,\mathbf{P}, \tag{2.5}$$

com $\mathbf{P}$ a *Matriz de Paridade* de dimensão $k \times (n-k)$. Neste caso, a matriz geradora pode ser representada na forma sistemática

$$\mathbf{G} = [\mathbf{P} | \mathbf{I}_k], \tag{2.6}$$

em que $\mathbf{I}_k$ é a matriz identidade de dimensão $k \times k$.

2.1.2 MATRIZ TESTE DE PARIDADE

Designando por c_i o bit i de uma palavra de código, uma outra definição possível para um código binário linear (n, k) é a seguinte: um código binário diz-se linear se todas as 2^k palavras de código satisfazem um conjunto de $n-k$ equações lineares homogéneas em c_i com $i = 0, \cdots, n-1$, isto é, um sistema de $n-k$ equações homogéneas a n incógnitas. Qualquer sistema linear de equações homogéneas define pois um código linear. Estas equações são designadas por *Equações de Teste de Paridade*.

O sistema de equações pode ser descrito na forma matricial

$$\mathbf{H}\,\mathbf{c}^T = \mathbf{0}_{(n-k)\times 1}, \tag{2.7}$$

em que $\mathbf{H}$ é a *Matriz de Teste Paridade*, de dimensões $(n-k) \times n$, formada pelos coeficientes do sistema de $n-k$ equações homogéneas a n incógnitas.

À semelhança do que foi dito para a matriz $\mathbf{G}$, é necessário que as linhas da matriz $\mathbf{H}$ sejam linearmente independentes para que o código seja (n, k). No caso de as linhas de $\mathbf{H}$ não serem linearmente independentes a matriz define um código de bloco (n, k') com $k' > k$ (por eliminação das linhas linearmente dependentes).

Pelas equações (2.3) e (2.7) resulta que:

$$\mathbf{H}\,\mathbf{c}^T = \mathbf{0}_{(n-k)\times 1} \Leftrightarrow \mathbf{c}\,\mathbf{H}^T = \mathbf{0}_{1\times(n-k)} \Leftrightarrow \mathbf{m}\,\mathbf{G}\,\mathbf{H}^T = \mathbf{0}_{1\times(n-k)} \quad \Rightarrow \quad \mathbf{G}\,\mathbf{H}^T = \mathbf{0}_{k\times(n-k)}. \tag{2.8}$$

Um código linear definido por uma matriz $\mathbf{H}$ de máxima característica é sempre sistematizável e, nesse caso, por aplicação de operações linear às linhas de $\mathbf{H}$, esta poderá ser pode ser escrita na forma sistemática,

$$\mathbf{H} = \left[\mathbf{I}_{n-k} \middle| \mathbf{P}^T \right], \tag{2.9}$$

com $\mathbf{I}_{n-k}$ a matriz identidade de dimensões $(n-k)\times(n-k)$, de acordo com (2.6) e (2.8).

2.1.3 GRÁFICOS DE TANNER

Os *Gráficos de Tanner*[3] (TG) [KFL], [Ryan], são uma das formas mais simples de representar um código binário linear.

Sabemos que um código binário linear (n,k) pode ser definido por um sistema de $(n-k)$ equações lineares homogéneas a n incógnitas (os bits das palavras de código). Podemos construir a partir deste sistema de equações, um gráfico bipartido formado por dois tipos de nodos:

- $(n-k)$ nodos designados por *nodos de teste* (CN's - *check nodes*), um por cada uma das equações lineares homogéneas do sistema.
- n nodos designados por *nodos das variáveis* (BN's - *bit nodes*[4]), um por cada uma das variáveis do sistema de equações.

Cada CN é ligado a todos os BN's que intervêm na equação à qual o CN está associado. Por sua vez, cada BN associado a um dado bit c_i da palavra de código, é ligado a todos os CN's correspondentes às equações de paridade na qual o bit c_i intervém. Devido a este facto, apenas existem ligações entre BN's e CN's e nunca entre nodos do mesmo tipo (gráfico bipartido). A um gráfico deste tipo dá-se o nome de *Gráfico de Tanner*.

Dada a matriz de teste de paridade $\mathbf{H}$ de um código binário linear (n,k), facilmente se pode obter o seu TG e vice-versa. Assim, basta atender a que existem

---

[3] Na literatura aparece muitas vezes a designação genérica de *Factor Graph* que engloba não só o caso particular dos TG's, utilizados para descrever os códigos binário lineares, mas também outros tipos de gráficos aos quais é possível aplicar técnicas de factorização [KFL].

[4] Os BN's são também designados na literatura por *variable nodes* ou *symbol nodes*.

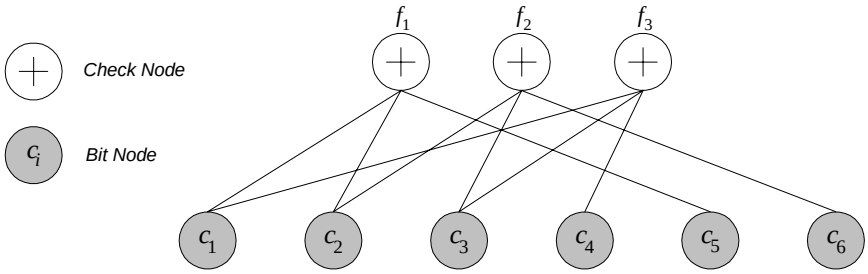
$(n-k)$ CN's associados a cada linha da matriz $\mathbf{H}$, n BN's associados a cada coluna de $\mathbf{H}$ e que cada CN j é ligado ao BN i sempre que na matriz $\mathbf{H}$ o elemento h_{ji} =1.

Exemplo 2.1:

Considere-se o código binário linear $(6,3)$, definido pela matriz de teste de paridade:

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

O TG associado a este código é:



A importância dos TG como forma de representação de códigos binários lineares reside no facto de muitos dos algoritmos de descodificação destes códigos terem por base esta representação gráfica. É sobre alguns desses algoritmos, em particular, o algoritmo SPA e suas simplificações que incidiu o trabalho de investigação realizado e que se encontra documentado de forma mais pormenorizada ao longo dos capítulos 4 e 5 desta dissertação.

2.2 CAPACIDADES DE DETECÇÃO E CORRECÇÃO DE ERROS

2.2.1 DISTÂNCIA DE HAMMING, PESO DE HAMMING E PESO MÍNIMO

Dado um código binário de bloco, define-se *distância de Hamming*, d_H, entre duas palavras de código como sendo o número de posições para as quais as duas palavras diferem.

Da mesma forma, define-se *Peso de Hamming*, ω_H, de uma palavra de código não nula como sendo o número de 1's dessa palavra.

Exemplo 2.2:

Dadas as palavras $\mathbf{a}=\begin{bmatrix}1 & 0 & 1 & 1 & 0 & 0\end{bmatrix}$ e $\mathbf{b}=\begin{bmatrix}0 & 1 & 1 & 0 & 1 & 1\end{bmatrix}$ tem-se:

- distancia de Hamming: $d_H(\mathbf{a},\mathbf{b})=5$
- peso de Hamming: $\omega_H(\mathbf{a})=3,\qquad \omega_H(\mathbf{b})=4$

Define-se também *Distância Mínima* de um código C, $d_{\min}(C)$, como sendo a menor distância de Hamming verificada para todos os pares distintos de palavras de código.

Como usamos aritmética módulo 2, concluímos que para duas palavras quaisquer de um código binário de bloco, $\mathbf{a}$ e $\mathbf{b}$, então:

$$d_H(\mathbf{a},\mathbf{b})=\omega_H(\mathbf{a}+\mathbf{b}). \tag{2.10}$$

Atendendo a que para um código linear a soma de quaisquer palavras de código é uma palavra de código, então podemos concluir que a distância mínima de um código binário linear é igual ao peso de Hamming mínimo de todas as palavras de código diferentes do vector nulo. Como todas as palavras de código verificam a equação (2.7) então, a distância mínima de um código binário linear é igual ao menor número de colunas de $\mathbf{H}$ que somadas geram o vector nulo.

2.2.2 DESCODIFICAÇÃO – REGRAS DE DECISÃO

Normalmente durante a transmissão de dados através de um canal ocorrem erros. Suponhamos que transmitimos uma palavra de código $\mathbf{a}$ através de um canal ruidoso e que a palavra recebida é $\mathbf{b}$. O objectivo de qualquer boa regra de decisão é minimizar a probabilidade de erro, ou seja, considerando um código binário linear (n,k) devemos escolher a palavra de código $\mathbf{a}'\in\{\mathbf{c}_0,\cdots,\mathbf{c}_{2^k-1}\}$ para a qual a probabilidade de erro é menor, isto é:

$$p(\mathbf{a}'|\mathbf{b})\geq p(\mathbf{c}_i|\mathbf{b})\quad \forall i=0,\cdots,2^k-1. \tag{2.11}$$

Fazendo uso do teorema de Bayes e admitindo que todas as palavras de código ocorrem com igual probabilidade podemos optar por uma descodificação de máxima verosimilhança, ou seja, por escolher $\mathbf{a}'\in\{\mathbf{c}_0,\cdots,\mathbf{c}_{2^k-1}\}$ para a qual:

$$p(\mathbf{b}|\mathbf{a}')\geq p(\mathbf{b}|\mathbf{c}_i)\quad \forall i=0,\cdots,2^k-1. \tag{2.12}$$

2.2.3 CAPACIDADE DE DETECÇÃO E CORRECÇÃO DE UM CÓDIGO BINÁRIO LINEAR

Uma das regras de decisão de máxima verosimilhança para a descodificação de um código binário linear (n,k) baseia-se precisamente na distância de Hamming. Assim, considerando que as palavras de código ocorrem com igual probabilidade, que são transmitidas através de um canal binário simétrico (BSC), com probabilidade $q < 0.5$ de um bit transmitido estar errado, e admitindo que ao ser transmitida a palavra de código $\mathbf{a}$, a palavra recebida é $\mathbf{b}$, então, sendo $D = \omega_H(\mathbf{a},\mathbf{b})$, temos que

$$p(\mathbf{b}|\mathbf{a}) = q^D (1-q)^{n-D}. \tag{2.13}$$

Uma regra de decisão de máxima verosimilhança consiste em escolher a palavra de código $\mathbf{a}' \in \{\mathbf{c}_0, \cdots, \mathbf{c}_{2^k-1}\}$ para a qual,

$$d_H(\mathbf{a}',\mathbf{b}) \le d_H(\mathbf{c}_i,\mathbf{b}) \quad \forall i = 0,\cdots,2^k - 1. \tag{2.14}$$

Para o caso em que $\mathbf{b}$ não é uma palavra de código mas existe uma única palavra de código $\mathbf{a}'$ que verifica (2.14), então, dir-se-á que os erros de transmissão foram corrigidos. Caso exista mais do que uma palavra de código que verifique (2.14), então, a palavra recebida não poderá ser corrigida.

As capacidades de correcção e detecção de erros de um código são expressas pelas seguintes propriedades [Wel1]:

Detecção de Erros

Um código C detecta todos os t erros, se e só se, a sua distância mínima for maior que t, isto é,

$$d_{\min}(C) > t. \tag{2.15}$$

Correcção de Erros

Um código C é capaz de corrigir todos os t erros, se e só se, a sua distância mínima for maior que $2t$, isto é,

$$d_{\min}(C) > 2t. \tag{2.16}$$

2.3 DESCODIFICAÇÃO – MÉTODO DO SÍNDROMA

Um dos métodos mais eficientes na descodificação de códigos de bloco lineares (n,k), para baixos valores de n, consiste na utilização do conceito de *Síndroma*.

Considere-se então que ao transmitirmos uma dada palavra de código $\mathbf{c}$, a palavra recebida é

$$\mathbf{r} = \mathbf{c} + \mathbf{e}, \tag{2.17}$$

com $\mathbf{e}$ o padrão de erro introduzido pelo canal de transmissão. A primeira etapa da descodificação consiste em averiguar se a palavra recebida é uma palavra de código. Para tal, é verificada a equação (2.7), pelo que é efectuado o seguinte cálculo

$$\mathbf{s} = \mathbf{r} \times \mathbf{H}^T. \tag{2.18}$$

O vector $\mathbf{s}$ assim obtido, de dimensões $1 \times (n-k)$ é designado por *Síndroma*.

No caso de o síndroma ser o vector nulo a palavra recebida é uma palavra de código e portanto, a descodificação é imediata. A palavra mensagem pode neste caso ser obtida rapidamente a partir da palavra de código recebida. No caso de um código sistemático, é conhecida automaticamente a mensagem por eliminação dos $(n-k)$ bits de teste de paridade. Para códigos não sistemáticos a forma mais simples é dispor de uma tabela em memória que faz corresponder a cada palavra de código a respectiva palavra mensagem. Facilmente se conclui que esta forma de afectação de palavras de código às respectivas mensagens levanta problemas de realização para códigos com valores de k muito elevados.

O caso mais comum é a obtenção de um síndroma $\mathbf{s} \neq \mathbf{0}$, o que significa que a palavra recebida não é uma palavra de código (contém erros). Torna-se pois necessário corrigir os erros ocorridos em função do síndroma, $\mathbf{s}$. Assim, por (2.17) e (2.7) tem-se substituindo em (2.18):

$$\begin{aligned} \mathbf{s} = \mathbf{r} \times \mathbf{H}^T \;&\Leftrightarrow \mathbf{s} = (\mathbf{c} + \mathbf{e}) \times \mathbf{H}^T \Leftrightarrow \mathbf{s} = \mathbf{c} \times \mathbf{H}^T + \mathbf{e} \times \mathbf{H}^T \Leftrightarrow \\ &\Leftrightarrow \mathbf{s} = \mathbf{0} + \mathbf{e} \times \mathbf{H}^T \Leftrightarrow \mathbf{s} = \mathbf{e} \times \mathbf{H}^T. \end{aligned} \tag{2.19}$$

Da equação (2.19) podemos concluir que o síndroma, $\mathbf{s}$, depende unicamente do padrão de erro, $\mathbf{e}$. Este é dado pela soma das colunas de $\mathbf{H}$ correspondentes aos bits errados, isto é, dado pelas posições dos bits a 1 do padrão de erro. Conclui-se ainda a partir de (2.19) que padrões de erro que diferem entre si de uma palavra de código produzem o mesmo síndroma. Existem portanto 2^{n-k} síndromas distintos tendo cada um

deles associado a si um conjunto de 2^k vectores de erro distintos designado por *Classe Complementar* associada ao síndroma[5].

Visto que associados a cada síndroma existem 2^k vectores de erro distintos, o problema que se coloca é saber qual deve ser seleccionado.

Considerando descodificação de máxima verosimilhança para um canal *BSC* com probabilidade de erro q, chegamos à conclusão que o padrão de erro a seleccionar deverá ser o que contém o menor número de bits a 1 (*Elemento Principal* da classe associado a esse síndroma). Se o elemento principal associado a um dado síndroma for único, então, diz-se que o erro pode ser corrigido. Caso contrário, apenas poderemos detectar a ocorrência de erros de transmissão, mas não os poderemos corrigir (não existe certeza).

O algoritmo para a descodificação de uma palavra recebida é:

A1. Algoritmo de descodificação por cálculo do síndroma:

(0) Cálculo do síndroma: $\mathbf{s} = \mathbf{r} \times \mathbf{H}^T$ *;*

(1) Determinação do elemento principal, $\mathbf{e}$*, da classe complementar associado a esse síndroma;*

Nota: A forma mais simples consiste em dispor de uma tabela em memória que faz corresponder a cada síndroma o respectivo elemento principal. No entanto, tal solução levanta problemas de implementação para códigos em que $(n-k)$ é um valor elevado.

(2) Determinação da palavra de código transmitida por $\mathbf{c} = \mathbf{r} + \mathbf{e}$ *;*

(3) Determinação da mensagem por tabela de consulta (código não sistemático) ou por remoção dos bits de teste (código sistemático).

[5] Podemos constatar que:

(Número de Classes Complementares) x (*Número de Elementos de Cada Complementar)* = 2^n

CAPÍTULO 3

LDPC – CÓDIGOS LINEARES DEFINIDOS POR MATRIZES DE TESTE DE PARIDADE ESPARSAS

Neste capítulo iniciaremos o estudo dos códigos definidos por matrizes de paridade esparsas (LDPC). O conhecimento das suas características e a interiorização de alguns conceitos são fundamentais para o estudo dos algoritmos iterativos utilizados na sua descodificação, objectivo primeiro do nosso trabalho. São essas características que levam a que sejam hoje considerados uma das mais poderosas famílias de códigos correctores de erros.

3.1 CARACTERÍSTICAS GERAIS E SUA REPRESENTAÇÃO

Um código LDPC é caracterizado por uma matriz de teste de paridade $\mathbf{H}$ com uma baixa densidade de 1's. Dentro dos códigos LDPC é comum fazer-se a distinção entre códigos *regulares* e *irregulares*.

3.1.1 CÓDIGOS LDPC REGULARES E IRREGULARES

Os códigos LDPC regulares foram propostos pela primeira vez por Robert Gallager em 1962, na sua dissertação de Doutoramento [Gal1]. Gallager definiu um código LDPC regular (n, k, w_c), com $w_c \ll m = n - k$, como sendo um código de bloco linear

cuja matriz de teste de paridade **H** tem dimensões $(n-k)\times n$, contendo exactamente w_c 1's por coluna e $w_r = w_c \times \frac{n}{m}$ 1's por linha[6]. Obviamente, existe um conjunto enorme de códigos que cumprem este requisito.

Gallager provou ainda que fazendo $w_c \geq 3$, o conjunto de códigos LDPC (n,k,w_c) que podem ser obtidos possuem, na sua grande maioria, uma distância mínima elevada, bastando para tal seguir algumas regras simples de construção como, por exemplo, garantir que quaisquer duas colunas da matriz **H** possuam quanto muito um só 1 em comum. Devido à baixa densidade de 1's da matriz **H** garantimos que o número mínimo de colunas de **H** que são necessárias somar de forma a obter o vector nulo é elevado e, logo, que o código possua uma distância mínima elevada de acordo com o que foi referido na secção 2.2.

Apesar das suas excelentes características, os códigos LDPC, salvo raras excepções, não foram alvo de qualquer atenção por parte da comunidade científica até meados dos anos 90, altura em que foram redescobertos por Makay e Neal [Mac2], [MN1] e [MN2]. Makay e Neal provaram que estes conseguem atingir uma probabilidade de erro muito próxima do limite de Shannon [Shan] (existindo mesmo alguns com desempenho superior relativamente aos melhores *Turbo Codes* conhecidos).

A razão do seu esquecimento durante tantos anos talvez se tenha devido à elevada complexidade computacional necessária a todos os níveis: na geração da matriz **H** que garanta um boa distância mínima do código, na codificação (código normalmente não sistemático) e ainda na sua descodificação, tendo por base o algoritmo *Soma de Produtos* (SPA) inicialmente proposto por Gallager. No entanto, os avanços tecnológicos, o surgimento de simplificações do algoritmo SPA, bem como, de formas alternativas de projecto fazendo uso de métodos algébricos e geométricos [PN], [KLF], [Weld], [JW], tornaram-nos num dos principais alvos de investigação científico nos dias de hoje.

Mais recentemente começaram a ser estudados os códigos LDPC irregulares cuja matriz de teste de paridade **H** possui uma baixa densidade de 1's mas em que o número de 1's por coluna e por linha não é constante. Em [RU] é feita a prova de que os códigos

---

[6] A nomenclatura utilizada w_c significa *column weight* e w_r significa *row weight*.

LDPC irregulares são superiores aos regulares para blocos longos. No entanto, a realização em *hardware* dos códigos LDPC regulares é mais simples visto que a sua estrutura pode ser explorada com vista a simplificar o descodificador. Além disso, para comprimentos de bloco mais curtos, os códigos irregulares possuem, na maioria dos casos, uma distância mínima inferior aos regulares, daí a preferência por estes últimos nessa situação.

Uma outra explicação para o melhor desempenho dos códigos LDPC irregulares tem por base a sua representação em TG, assim como, a forma como estes são descodificados.

3.1.2 REPRESENTAÇÃO GRÁFICA DE TANNER

Michael Tanner [Tan1] foi um dos poucos investigadores que estudou os códigos LDPC antes do seu ressurgimento por via de Mackay e Neal [Mac2], [MN1], [MN2]. O seu contributo veio-se a revelar de extrema importância visto que muita da investigação actual na pesquisa de bons códigos LDPC regulares e irregulares, bem como, o desenvolvimento de muitos dos algoritmos de descodificação assenta nos TG's por ele introduzidos.

Como foi referido na secção 2.1.3, Tanner considerou que qualquer código linear de bloco e, em particular, qualquer código LDPC, podia ser representado por intermédio de um TG. Este gráfico bipartido é formado por dois tipos de nodos, BN's e CN's, em que cada BN só pode estar ligado a CN's e cada CN a BN's (não podem existir ligações entre nodos do mesmo tipo, daí se dizer que o gráfico é bipartido).

Feita que foi uma breve introdução aos TG's na secção 2.1.3, convém agora introduzir novos conceitos. As ligações entre os nodos designam-se por *caminhos* (*edges*). O número de caminhos que chegam a cada nodo é designado por o *grau* desse nodo. Dado um código LDPC (n,k), onde w_{ci} é o grau do BN i e w_{rj} é o grau do CN j, concluímos que

$$\sum_{i=1}^{n} w_{ci} = \sum_{j=1}^{n-k} w_{rj}\,, \tag{3.1}$$

ou seja, a soma dos graus de todos os BN's é igual à soma dos graus de todos os CN's.

No caso de se tratar de um código LDPC regular (n,k,w_c), então, todos os BN's têm o mesmo grau, w_c, e todos os CN's têm o mesmo grau, w_r, e atendendo às dimensões da matriz **H** temos

$$n \times w_c = (n-k) \times w_r \Leftrightarrow w_r = w_c \times \frac{n}{n-k}, \tag{3.2}$$

tal como havia sido referido aquando da definição dos códigos LDPC regulares. O TG possui neste caso uma estrutura altamente regular (ver figura 3-1) que pode ser explorada na descodificação do código.

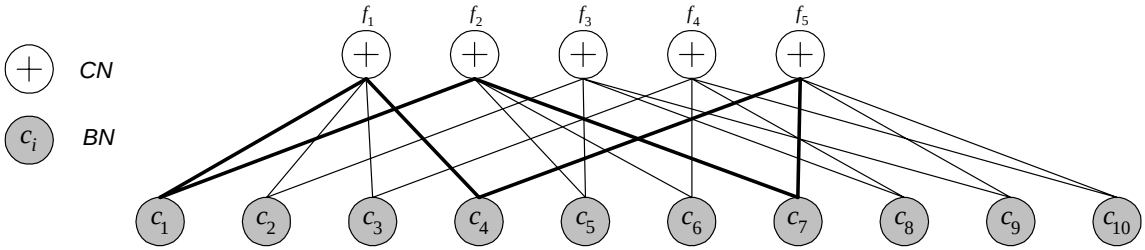


Figura 3-1 – Código LDPC regular $(10,5,2)$.

3.1.3 CONCEITO DE GIRO

Um dos mais importantes conceitos relativos aos TG's é a definição de *giro* (*girph*) de comprimento *l* como sendo um percurso fechado formado por *l* caminhos [LS], [MB1]. Por exemplo, tendo por base a figura 3-1 podemos observar um ciclo de comprimento 6 que se encontra assinalado a negrito. O menor comprimento de todos os ciclos existentes num TG é designado por giro.

Na prática, ao projectar um código LDPC procura-se evitar a existência de ciclos de pequeno comprimento no seu TG de forma a melhorar o desempenho do algoritmo SPA. De facto, prova-se que o algoritmo SPA tem desempenho óptimo quando aplicado a gráficos sem ciclos [KF], [Forn], [KFL], [Ksch]. Na presença de gráficos com ciclos, a sua eficiência diminui, sendo inferior para códigos com um baixo giro. Por outro lado, prova-se também que TG sem ciclos não suportam bons códigos [ETV], pelo que é necessário obedecer a algumas regras na construção dos códigos LDPC por forma a que estes possuam boas propriedades (distância mínima e um giro elevados).

Identificação de ciclos e determinação do giro de um código LDPC

Qualquer ciclo de um TG tem um comprimento par e o seu valor mínimo é 4 correspondendo a uma matriz de teste de paridade em que existem duas colunas com dois 1's em comum (ver figura 3-2). Uma das regras de construção da matriz **H**, para evitar a existência de ciclos de dimensão 4, consiste em garantir que quaisquer duas colunas da matriz **H**, possuam quanto muito um só 1 em comum.

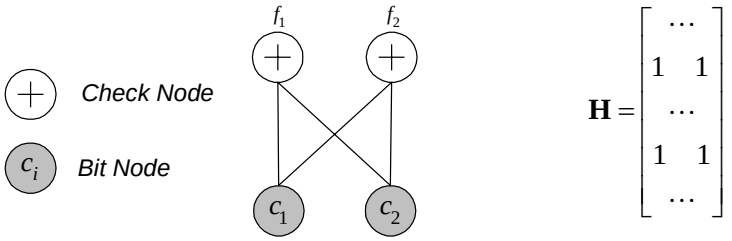


Figura 3-2 – Exemplo de um ciclo de comprimento 4 num TG e a respectiva matriz de teste de paridade.

Já os ciclos de comprimento 6 são mais difíceis de identificar na matriz **H**. Na figura 3-3 apresentamos um exemplo.

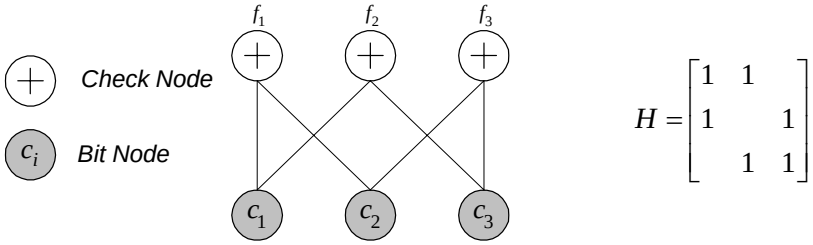


Figura 3-3 – Exemplo de um ciclo de comprimento 6 num TG e a respectiva matriz de teste de paridade.

Mao e Banihashemi apresentam [MB2] uma forma simples de determinar o giro associado a cada BN, isto é, o ciclo mais curto que passa por cada BN. O método consiste em construir para cada BN uma árvore a partir do TG ou da matriz **H** do código. Assim, considera-se como raiz da árvore o BN_i cujo giro pretendemos determinar. A árvore é então construída passo a passo. No nível k de descendência da árvore são incluídos todos os nodos a uma distância k do BN_i. O procedimento é repetido até ao nível de descendência n em que é incluído um nodo que se encontra ligado no TG do código a pelo menos dois nodos já incluídos no nível de descendência $n-1$. Isto identifica a formação do primeiro ciclo, sendo $2n$ o giro do BN_i.

Em resumo, o método consiste em adicionar como descendentes, a cada nó j da árvore, todos os nós que a ele se encontram ligados no TG, com excepção do nó pai,

terminando o algoritmo quando é encontrado um nó que é descendente de mais do que um nó diferente.

Na figura 3-4 podemos observar a aplicação do método anteriormente descrito na determinação do giro do BN c_1 do código descrito pelo TG da figura 3-1. Como pode ser observado, só no 3º nível de descendência da árvore encontramos um nó descendente simultaneamente de mais do que um nó, pelo que se conclui que o giro do BN c_1 é 6. A negrito encontra-se identificado um dos ciclos de comprimento 6 que contém o BN c_1.

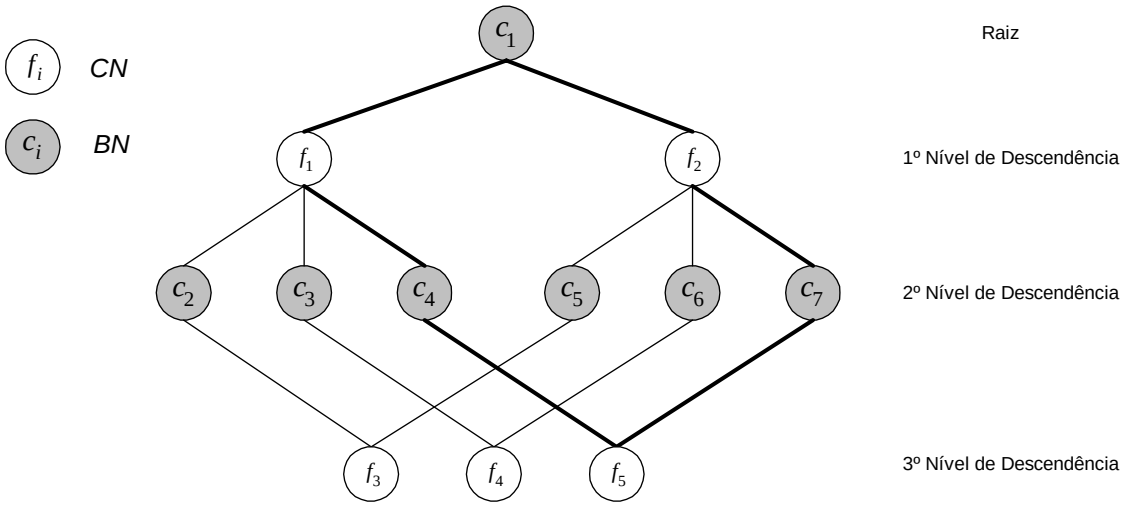


Figura 3-4 – Determinação do giro do BN c_1 do código LDPC descrito pelo TG da figura 3-1.

McGowan e Williamson apresentam em [MW] um método algébrico de determinação do giro de cada BN, baseado no conceito de *matriz adjacente*. Dado um código LDPC descrito por uma matriz de teste de paridade $\mathbf{H}$, a sua matriz adjacente é definida por,

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{H} \\ \mathbf{H}^T & \mathbf{0} \end{bmatrix}, \tag{3.3}$$

pelo que cada nó do TG do código (BN ou CN) é representado por uma linha e por uma coluna da matriz $\mathbf{A}$.

McGowan e Williamson provaram que o elemento (i, j) da matriz $\mathbf{A}^n$ é o número de percursos com comprimento n entre os nós i e j do código. Neste sentido, cada elemento diagonal da matriz $\mathbf{A}^n$, a_{ii}^n, representa o número de ciclos de dimensão n que contém o nó i. O método descrito permite determinar rapidamente o giro de cada BN, bem como, o giro do código.

3.2 CONSTRUÇÃO DE CÓDIGOS LDPC

A forma mais simples de projectar um código LDPC (n,k), consiste na construção da matriz de teste de paridade **H** que cumpra um conjunto requisitos pretendidos, como sejam, definir um código regular (caracterizado por um dado peso para cada coluna, w_c) ou irregular (podendo ser especificado o grau de cada BN e CN), ou ainda, garantir um dado giro mínimo.

Obviamente, para qualquer conjunto de restrições existe um conjunto imenso de códigos que cumprem as referidas especificações. A construção da matriz **H** é feita de forma quase aleatória, mas seguindo algumas regras que maximizam a probabilidade do código obtido possuir um bom desempenho. No entanto, essas regras não nos dão qualquer garantia de que tal aconteça. Poderão inclusive suceder situações em que a matriz **H** obtida não é de característica máxima e, portanto, define um código LDPC (n,k') com $k'>k$.

Makay e Neal apresentam em [Mac2] um conjunto de estratégias para gerar códigos LDPC. Estas estratégias são apresentadas de forma numerada, sendo convicção dos autores que as de ordem superior maximizam a probabilidade do código obtido possuir um melhor desempenho. No entanto, eles próprios reconhecem não possuir qualquer prova deste facto.

Estratégias apresentadas por Mackay e Neal para construção de códigos LDPC:

(i) A matriz **H** é gerada partindo de uma matriz de zeros de dimensões $(n-k)\times n$ e aleatoriamente negando w_c bits em cada coluna (o código assim gerado poderá ser irregular);

(ii) A matriz **H** é gerada criando aleatoriamente colunas de peso de Hamming w_c;

(iii) A matriz **H** é gerada criando aleatoriamente colunas de peso de Hamming w_c e procurando uniformizar ao máximo o peso de Hamming w_r de cada linha;

(iv) A matriz **H** é gerada com colunas de peso de Hamming w_c, linhas de peso de Hamming w_r, e não possuindo quaisquer duas colunas com mais de um 1 em comum;

(v) A matriz **H** é gerada de acordo com o procedimento do ponto anterior mas tendo como objectivo a maximização do giro do código;

(vi) A matriz **H** é gerada de acordo com o procedimento referido em (iv) procurando obter uma matriz **H** de característica máxima, de preferência na forma $\mathbf{H} = \left[\mathbf{H}_1 \middle| \mathbf{H}_2\right]$ com $\mathbf{H}_1$ ou $\mathbf{H}_2$ invertível.

Mackay e Neal apresentam ainda em [MN1] um conjunto de técnicas que fazem uso de algumas das estratégias antes referidas. Essas técnicas são vulgarmente conhecidas por 1A, 2A, 1B e 2B, sendo usadas por muitos investigadores na classificação dos códigos LDPC estudados.

A estratégia 1A diz respeito ao ponto (iii) com $w_c = 3$, e em que é introduzida a restrição de o código possuir um giro superior a 4.

A estratégia 2A baseia-se na construção da matriz **H** com $M/2$ colunas (com M sendo o número de linhas da matriz **H**) de peso 2, sem que exista qualquer 1 em comum entre elas, e em que as restantes colunas têm peso 3, Simultaneamente, procura-se que o peso w_r seja o mais uniforme possível, impondo como restrição o facto de quaisquer duas colunas de **H** não terem mais do que um 1 em comum. Consiste pois numa implementação da estratégia (iv) anteriormente mencionada.

As estratégias 1B e 2B resultam da eliminação de um pequeno número de colunas da matriz **H** obtidas segundo as estratégias 1A e 2A, respectivamente, de forma a eliminar todos os ciclos do código de tamanho inferior a um dado valor *l* estipulado, procurando, desta forma, maximizar o giro de acordo com a estratégia (v).

Um dos problemas que se colocado é precisamente a forma de eliminar os ciclos de comprimento inferior a um dado valor. McGowan e Williamson baseados no conceito de matriz adjacente, definida anteriormente, apresentam um método em [MW] para remoção desses ciclos por alteração da matriz **A**, garantindo, simultaneamente, que não são gerados novos ciclos do mesmo tamanho ou inferior.

Como já foi referido, seguindo qualquer uma destas estratégias, o conjunto de códigos admissíveis é muito grande, tornando-se necessário dispor de alguns critérios

de selecção por forma a escolher os susceptíveis de apresentarem um melhor desempenho.

Mao e Banihashemi sugerem em [MB2], como critério de selecção a escolha do código que apresente o giro mais elevado. Este método implica a determinação do giro de cada um dos códigos o que coloca, desde logo, limitações à sua aplicação em comprimentos de bloco elevados. Estes autores mostraram que para códigos obtidos segundo o método de construção 2A, o critério de selecção por eles proposto, conduzia à escolha de um código com uma menor probabilidade de erro de descodificação.

Um outro critério de selecção consiste na escolha do código que possui uma maior distância mínima (sem dúvida, um bom parâmetro de avaliação do desempenho de um código linear). A determinação da distância mínima de um código linear é, no entanto, uma tarefa complexa para códigos com comprimentos de bloco elevados. Berrou, Vaton, Jézéquel e Douillard apresentam em [BVJD] um método iterativo para a determinação da distância mínima de um código linear, que designam por *Método do Erro Impulsivo*. O método baseia-se na resposta de um descodificador iterativo de máxima verosimilhança do tipo *Soft-In / Soft-Out*[7] a um erro do tipo impulsivo[8] num canal Gaussiano. Os autores provaram que a amplitude mínima de ruído que provoca um erro de descodificação é igual à distância mínima do código em causa.

Abordagens alternativas para a construção de códigos LDPC

Existem outras formas de construção de códigos LDPC. Por exemplo, Prabhakar e Narayanan apresentam em [PN] um método algébrico de construção de códigos LDPC regulares. Este apresenta como grande vantagem o facto de a estrutura do TG poder ser gerada algebricamente usando um procedimento recursivo.

O método algébrico apresentado por Prabhakar e Narayanan pretende construir um código LDPC regular de dimensões (n,k), sendo w_c o peso de cada BN e w_r o peso de

---

[7] O conceito de descodificação iterativa do tipo Soft-In / Soft-Out será abordado detalhadamente no capítulo 4.

[8] A palavra de código $\mathbf{c}=[c_0\ c_1 \cdots c_{i-1}\ c_i\ c_{i+1} \cdots c_n]$ transmitida é apenas corrompida num bit com ruído impulsivo de amplitude A_i, dando origem a uma palavra recebida do tipo $\mathbf{r}=[c_0\ c_1 \cdots c_{i-1}\ (c_i + \mathrm{A}_i)\ c_{i+1} \cdots c_n]$.

cada CN. Existem $n \times w_c$ ligações a BN's (*conexão BN*) e a CN's (*conexão CN*) em que cada caminho do TG liga uma conexão BN a uma conexão CN. Designando por A_i a conexão CN à qual liga a conexão BN_i, com A_i pertencente ao CN $\lfloor A_i / w_r \rfloor$ e a conexão BN_i pertencente ao BN $\lfloor i / w_c \rfloor$, então, Prabhakar e Narayanan propuseram que a sequência $A_0, A_1, A_2, \cdots, A_{n \times w_c - 1}$ fosse obtida da seguinte forma recursiva:

$$A_{i+1} = (a.A_i + b) \bmod M \quad \text{c/} \quad M = n \times w_c \quad \text{e} \quad 0 \le i \le M , \tag{3.4}$$

com a e b números inteiros satisfazendo um conjunto de condições por eles deduzidas e A_0 um número inteiro entre 0 e M.

Ao todo são nove as restrições colocadas por Prabhakar e Narayanan que garantem que o código gerado por este método é LDPC e não possui ciclos de comprimento inferior a seis. Estas condições não são muito restritivas, permitindo que seja gerado um vasto conjunto de códigos LDPC com diferentes comprimentos e taxas de informação. O desempenho do código gerado depende também dos valores escolhidos para a, b e A_0.

Uma abordagem alternativa para a construção de bons códigos LDPC irregulares é apresentada por Richardson, Shokrollahi e Urbanke em [RSU] e Chung, Forney, Richardson e Urbanke em [CFRU] que se baseia num estudo sobre a distribuição dos pesos de cada BN que optimiza o desempenho do código.

3.3 CODIFICAÇÃO

Antes de referir outros métodos de construção de códigos LDPC, torna-se necessário ter uma ideia de como é realizada a codificação conhecida a matriz **H** ou o TG de um código.

Sendo os códigos LDPC lineares e tendo em conta o que foi referido na secção 2.1.1, a forma obvia de realizar a codificação seria partindo do conhecimento da matriz geradora **G** determinar as palavras de código de acordo com a equação (2.3), ou seja, fazendo $\mathbf{c} = \mathbf{m}\mathbf{G}$. No entanto, os métodos utilizados na construção de códigos LDPC, assentam na obtenção da sua matriz de teste de paridade **H** ou no equivalente TG. Na

grande maioria das vezes a matriz **H** obtida não é sistemática nem de característica máxima. A matriz **H** pode, no entanto, ser expressa na forma sistemática (2.9)[9] usando o método de Gauss. A obtenção da matriz **G** é depois imediata. Podem, no entanto, surgir situações em que para obter **H** na forma (2.9) se torne necessário efectuar troca de colunas, obtendo-se desta forma um código LDPC diferente mas com o mesmo desempenho do original.

Embora de fácil implementação, o método anterior é extremamente dispendioso em termo do número de operações a realizar na codificação de cada palavra de código. Tal deve-se ao facto de a sistematização da matriz **H** não conduzir, necessariamente, à obtenção de uma matriz **G** com baixa densidade de 1's, pelo que a codificação por (2.3) exige a realização de um número extremamente elevado de operações.

Uma aproximação alternativa para a solução deste problema consiste na obtenção de códigos LDPC por métodos algébricos e geométricos em que a codificação possa ser realizado por circuitos simples baseados em registos de deslocamento. É o caso dos códigos apresentados por Johnson e Weller em [JW], que sugerem uma família de códigos LDPC quase cíclicos com giro não inferior a 6. É também o caso de Kou, Lin e Fossorier que apresentam em [KLF] uma abordagem para a construção de códigos LDPC baseada em linhas e pontos de uma geometria finita, como a geometria Euclidiana e a geometria projectiva em campos finitos. Os códigos construídos por este método são também cíclicos ou quase cíclicos.

---

[9] No caso da matriz **H** não ser de característica máxima ao aplicarmos o algoritmo de Gauss vamos obter uma ou mais linhas de zeros que poderão ser eliminadas obtendo uma matriz de paridade $\mathbf{H}'$ equivalente a **H**, expressa na forma da equação (2.9).

CAPÍTULO 4

DESCODIFICAÇÃO ITERATIVA

Devido ao seu desempenho os códigos LDPC são hoje reconhecidos como a classe de códigos que melhor se aproxima do limite de eficiência de Shannon [Shan]. De facto, o sucesso dos códigos LDPC deve-se em grande medida à sua simplicidade e ao algoritmo de descodificação iterativo introduzido por Gallager [Gal1] e cuja redescoberta por MacKay e Neal [MN1], [MN2], veio tornar clara a sua importância.

O algoritmo apresentado por Gallager [Gal1] tem por base a representação gráfica dos códigos LDPC e é vulgarmente conhecido como algoritmo *Soma de Produtos* (SPA). No entanto, o seu campo de aplicação vai muito para além da descodificação de códigos LDPC, abrangendo áreas do processamento de sinal, das comunicações digitais e da inteligência artificial, onde é conhecido por *Belief Propagation*[10] (BP) devido à sua aplicação em redes Bayesianas.

---

[10] A designação do algoritmo SPA por *Belief Propagation* é comum no contexto da descodificação de códigos LDPC. Ambos os termos serão empregues ao longo do texto de forma indiferenciada.

4.1 INTRODUÇÃO

Na descodificação de um código, existem duas abordagens possíveis: *Hard Decoding* (HD) e *Soft Decoding* (SD)[11]. A primeira, considera que o conjunto de símbolos diferentes à entrada do descodificador é finito, ou seja, no caso de um código binário à entrada do descodificador teremos apenas bits, ao passo que na descodificação SD é tida em conta a distribuição probabilística dos símbolos recebidos.

Se por um lado, os algoritmos do tipo HD são na sua grande maioria menos complexos e menos exigentes computacionalmente, também é verdade que tomando em conta a distribuição probabilística dos dados recebidos pelo descodificador, os algoritmos SD apresentem um melhor desempenho em termos de uma menor taxa de erros quer BER quer MER.

No contexto dos códigos LDPC ambas as abordagens de descodificação têm vindo a ser objecto de investigação procurando um compromisso entre complexidade, latência e desempenho. Os algoritmos iterativos HD propostos são normalmente designados por algoritmos de troca de bits (*bit flipping algorithms*) [Rich], e os SD por algoritmos de troca de mensagens (*message passing algorithms*).

Começamos por descrever o algoritmo de troca de bits (BF - *Bit Flipping Algorithm*) proposto por Gallager [Gal1], [Gal2], bem como duas das suas variantes: o algoritmo de troca de bits pesado (WBF – *Weight Bit Flipping Algorithm*) proposto por Kou, Lin e Fossorier [KLF] e o algoritmo de troca de bits pesado com arranque programado (BWBF – *Bootstraped Weight Bit Flipping*) proposto por Nouh e Banihashemi [NB]. O algoritmo BF além de ser o mais simples de entre os diversos algoritmos propostos para a descodificação dos códigos LDPC, permite lançar as bases para uma melhor compreensão do algoritmo SPA e suas variantes sobre o qual incidiu grande parte do nosso estudo.

---

[11] Existem no entanto alguns algoritmos de descodificação cuja abordagem é um misto de HD com SD. É o caso dos algoritmos WBF e BWBF a que será feita referência.

4.2 ALGORITMOS BF E SUAS VARIANTES

4.2.1 ALGORITMOS BF

Os algoritmos BF [Gal1], [Gal2], [KLF], [Rich] são algoritmos HD de baixa complexidade. A ideia base [Ryan], [LMSS1], [KLF] consiste em negar o número mínimo de bits da palavra de código recebida por forma a que todas as equações de paridade expressas pela matriz $\mathbf{H}$ sejam satisfeitas. Este processo torna-se mais simples quando a matriz $\mathbf{H}$ possui uma baixa densidade de 1's e quando o grau dos BN's e dos CN's do TG que representa o código é baixo.

A forma mais simples de descrever a descodificação HD é através de um exemplo:

Exemplo 4.1:

Consideremos o código de Hamming $(7,4)$, definido pela matriz de teste de paridade

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

A palavra de código transmitida foi

$$\mathbf{c}=\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 1 & 1\end{bmatrix},$$

e a palavra recebida

$$\mathbf{y}=\mathbf{c}+\begin{bmatrix}0 & 0 & 1 & 0 & 0 & 0 & 0\end{bmatrix}=\begin{bmatrix}1 & 0 & 1 & 0 & 0 & 1 & 1\end{bmatrix}.$$

A palavra recebida $\mathbf{y}$ deverá ser descodificada de forma a corrigir o erro ocorrido.

Resolução:

Uma forma simples e instrutiva de compreender o processo de descodificação HD consiste na representação esquemática do conjunto de restrições de teste de paridade. Cada restrição é representada por um círculo contendo o conjunto de variáveis que intervêm nessa restrição em que a soma de todas as variáveis contidas dentro do círculo deverá ser nula. Para o caso do código dado, obtemos a representação esquemática da figura 4-1 em que y_i, com $i=1,\cdots,7$, representam os bits da palavra recebida $\mathbf{y}$. Substituindo os y_i pelos respectivos valores, verificamos que as restrições f_2 e f_3 passam a ser violadas, conforme se pode verificar pela figura 4-2.

Da intersecção dos círculos f_2 e f_3 verificamos que as variáveis comuns a ambas as restrições são y_3 e y_7, pelo que negando um destes bits ambas as restrições passariam a ser satisfeitas. No

entanto, y_7 faz também parte da restrição f_1, restrição que passaria a ser violada caso o bit y_7 fosse negado. Logo, o bit a negar deverá ser y_3 e a palavra descodificada será,

$$\overline{y}' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

que de facto corresponde à palavra transmitida.

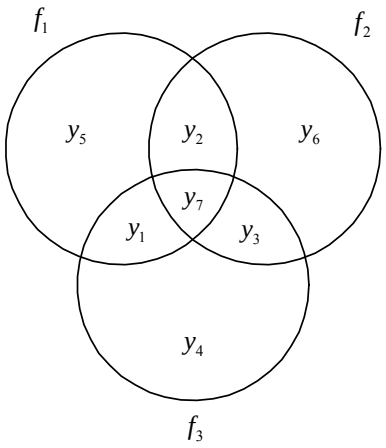


Figura 4-1 – Representação esquemática do conjunto de restrições de teste de paridade expressas pela matriz **H**.

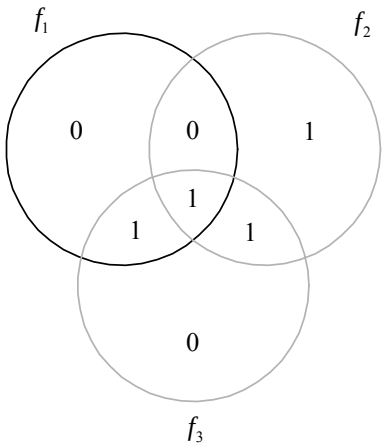


Figura 4-2 – Ao substituir os bits da palavra recebida **y** constata-se que as restrições f_2 e f_3 são violadas.

O exercício poderia também ser resolvido recorrendo ao TG do código (figura 4-3).

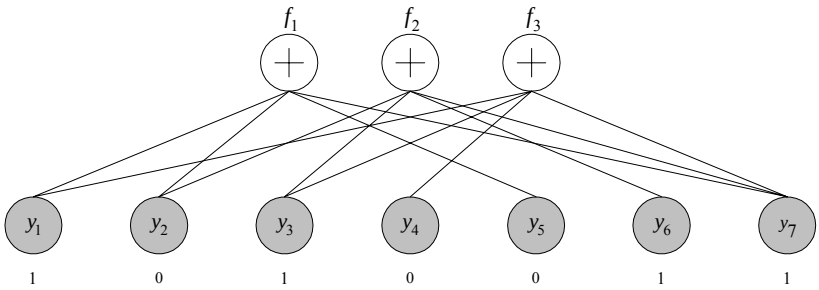


Figura 4-3 – TG associado ao código de Hamming $(7,4)$. Para a palavra recebida $\overline{y}$ as restrições f_2 e f_3 são violadas.

A única restrição que não é violada é f_1, pelo que o CN f_1 envia a informação aos BN's y_1, y_2, y_5 e y_7 de que estes bits deverão permanecer inalterados. Restam como possíveis candidatos a serem alterados y_3, y_4 e y_6. No entanto, tendo em conta a regra de descodificação HD (negar o menor número de bits da palavra recebida de forma a satisfazer todas as condições), facilmente concluímos que somente a alteração de y_3 faz com que as três equações de paridade sejam simultaneamente verificadas, correspondendo, de facto, ao bit que estava errado na palavra recebida.

Como pudemos verificar no exemplo anterior, a restrição f_1 não era violada. Os bits nela envolvidos fornecem um tipo de informação extrínseca às restantes restrições f_2 e f_3, ajudando a identificar os possíveis bits errados. É este conceito de informação extrínseca e de passagem de informação/mensagens entre nodos que está na base do algoritmo SPA e das suas variantes.

Na descodificação BF desenvolvida por Gallager [Gal1], [Gal2], [KLF], a palavra recebida é previamente binarizada. Assim, consideremos um código LDPC (N,K) [12] cujas palavras de código, $\mathbf{c}$, são transmitidas através de um canal Gaussiano aditivo. Para tal, os bits da palavra de código $c_n \in \{0,1\}$, com $n = 1,\cdots,N$, são modulados em BPSK, fazendo-lhes corresponder os símbolos $x_n = (-1)^{c_n}$. Os símbolos da palavra recebida $\mathbf{y}$, podem então ser expressos por, $y_n = x_n + n_n$, com n_n ruído Gaussiano de média nula e desvio padrão σ. A palavra binária recebida $\mathbf{r}$ é então obtida fazendo, $r_n = 0$ se $y_n > 0$ e $r_n = 1$ se $y_n < 0$.

De acordo com (2.18) o descodificador calcula o síndroma, $\mathbf{s}$. Todos os bits contidos em mais do que um dado número δ de equações de teste de paridade não satisfeitas (equações correspondes aos bits a 1 de $\mathbf{s}$) são então negados e o sindroma recalculado. O processo repete-se iterativamente até que todas as equações sejam satisfeitas, ou até que um número máximo de iterações seja atingido, situação em que é declarado uma falha na descodificação.

O parâmetro de projecto δ, designado por limiar, deverá ser escolhido de forma a minimizar o BER e o MER da descodificação e, simultaneamente, o número de iterações do algoritmo. O valor de δ depende dos parâmetros do código e da relação sinal ruído (SNR) do canal.

O número de iterações do algoritmo é uma variável aleatória função da SNR do canal e das características do código. Um facto de realce no contexto da descodificação iterativa de um código LDPC, é que o conceito de distância mínima perde importância [KLF], na medida em que é possível corrigir muitos padrões de erro com peso superior às capacidades de correcção expressas pela equação (2.16). Devido à estrutura dos códigos LDPC e ao facto de a descodificação ser iterativa, a rápida convergência de

---

[12] Por conveniência de notação passamos a designar as dimensões de um código por (N,K).

alguns BN's para o seu valor correcto, ajuda à convergência dos restantes BN's incorrectos.

O parâmetro δ tem pois influência no desempenho do algoritmo BF. Um dos algoritmos BF mais simples é o seguinte:

A2. Algoritmo BF:

(0) Calcular o sindroma $\mathbf{s} = \mathbf{r} \times \mathbf{H}^T$ *. Se* $\mathbf{s} = \mathbf{0}$ *parar a descodificação* $\Rightarrow$ *palavra correctamente descodificada.*

(1) Calcular para cada BN_n*, o número de equações de teste de paridade não satisfeitas* f_n*, que o contêm, com* $n = 1, \cdots, N$*;*

(2) Identificar o conjunto Ω *de BN's para os quais* f_n *é maior.*

(3) Negar os bits do conjunto Ω*, obtendo uma nova palavra* $\mathbf{r}'$*.*

(4) Repetir com a nova palavra $\mathbf{r}$ *os passos (0) a (3) até que todas as equações de teste de paridade sejam satisfeitas, ou, um número máximo de iterações seja atingido, situação em que é declarada falha na descodificação.*

4.2.2 ALGORITMOS WBF E BWBF

Os algoritmos WBF [KLF], [NB] e BWBF [NB] são já de alguma forma do tipo SD. Embora as palavras descodificadas continuem a ser obtidas por troca de bits da palavra recebida, $\mathbf{r}$, essa troca é pesada por informação recebida à saída do canal (*soft information*). A sua descrição nesta secção permite lançar os fundamentos para uma melhor compreensão da descodificação SD, da qual faz parte o algoritmo SPA e as suas variantes.

Algoritmo WBF

Consideremos um código LDPC (N, K) cujas palavras de código, $\mathbf{c}$, são transmitidas através de um canal Gaussiano, com modulação BPSK e seja $\mathbf{y} = (y_1, y_2, \cdots, y_N)$, a palavra recebida à saída do canal. Uma medida da fiabilidade da decisão binária tomada para cada símbolo, y_n, recebido é o seu módulo, $|y_n|$. De facto, ao considerar, $r_n = 0$ se $y_n > 0$ e $r_n = 1$ se $y_n < 0$, então, quanto maior for $|y_n|$ maior é probabilidade de a decisão tomada se encontrar correcta. O algoritmo WBF toma precisamente em consideração este facto.

Designemos por:

- $N(m)$ o conjunto de BN's ligados ao CN_m ou seja:

$$N(m) = \{n: \quad h_{mn} = 1\}. \tag{4.1}$$

- $N(m) \setminus n$ o conjunto de BN's ligados ao CN_m excluindo o BN_n.
- $M(n)$ o conjunto de CN's ligados ao BN_n, ou seja:

$$M(n) = \{m: \quad h_{mn} = 1\}. \tag{4.2}$$

- $M(n) \setminus m$ o conjunto de CN's ligados ao BN_n excluindo o CN_m.

Por (2.18) calculamos o síndroma $\mathbf{s}$ associado à palavra recebida $\mathbf{r}$, determinando para cada CN_m, a necessidade de negar, ou não, algum dos BN's a ele ligados[13]. Para cada CN_m, com $m = 1, \cdots, M$, podemos determinar o menor valor absoluto dos símbolos recebidos correspondentes aos bits que intervêm nesse CN, ou seja,

$$y_{\min}^{(m)} = \min\{|y_n| : n \in N(m)\}. \tag{4.3}$$

Este valor determina o peso do CN_m, na decisão de negar, ou não, os BN's que a ele se encontram ligados. Desta forma, para cada BN_n, com $n = 1, \cdots, N$, é calculado um peso com base na informação fornecida pelos CN's a ele ligados, sendo este peso dado por,

$$E_n = \sum_{m \in M(n)} (2s_m - 1) y_{\min}^{(m)}. \tag{4.4}$$

O BN com maior peso é então negado, e o processo de cálculo do sindroma e dos pesos é repetido até que $\mathbf{s} = 0$, ou um número máximo de iterações seja atingido.

O algoritmo pode pois ser expresso na seguinte forma:

---

[13] Se o elemento s_m de $\mathbf{s}$ for igual a 0 tal significa que a equação de teste de paridade associada ao CN_m é verificada e, como tal, não existe necessidade de negar nenhum dos bits de $\mathbf{r}$ que intervêm no cálculo dessa equação de paridade. Caso $s_m \neq 0$, pelo menos um dos bits que intervêm na equação de paridade m terá de ser negado.

A3. Algoritmo WBF

(0) Calcular o sindroma $\mathbf{s} = \mathbf{r} \times \mathbf{H}^T$ *. Se* $\mathbf{s} = \mathbf{0}$ *parar a descodificação* $\Rightarrow$ *palavra correctamente descodificada.*

(1) Calcular para cada CN_m*, com* $m = 1, \cdots, M$ *,*

$$y_{\min}^{(m)} = \min\left\{\left|y_n\right| : n \in N(m)\right\}.$$

(2) Calcular para cada BN_n*, com* $n = 1, \cdots, N$ *,*

$$E_n = \sum_{m \in M(n)} \left(2s_m - 1\right) y_{\min}^{(m)} .$$

(3) Negar o bit r_n *para o qual* E_n *é máximo.*

(4) Repetir com a nova palavra $\mathbf{r}$ *os passos (0), (2) e (3), até obter* $\mathbf{s} = \mathbf{0}$ *ou até que o número máximo de iterações estipulado seja atingido. Nesta situação é declarada falha na descodificação.*

Algoritmo BWBF

O algoritmo BWBF difere apenas do algoritmo WBF no que toca aos símbolos da palavra recebida $\mathbf{y}$ sobre os quais é efectuada a decisão binária. Sendo $\left|y_n\right|$ uma medida de fiabilidade da decisão binária tomada para o símbolo y_n, é lógico pensar que para valores de $\left|y_n\right|$ inferiores a um dado limiar, α, a informação recebida não é fidedigna e, como tal, a decisão binária sobre o bit r_n não deverá ser tomada tendo em conta y_n, mas sim, a informação fidedigna proveniente dos CN's nos quais esse bit intervém. Neste sentido, um CN_m é considerado fidedigno em relação a um BN_n, se todos os BN's a ele ligados, com excepção do BN_n, forem considerados BN fidedignos, isto é,

$$\left|y_{n'}\right| > \alpha, \quad \forall\, n' \in N(m) \setminus n . \tag{4.5}$$

A descodificação começa por identificar os BN's não fidedignos e os CN's fidedignos. Para um BN_n, não fidedigno ligado a pelo menos um CN fidedigno, a informação recebida, y_n, é corrigida tendo por base a informação proveniente dos CN's fidedignos, de acordo com a equação

$$y_n' = y_n + \sum_{m \in M_f(n)} \left[\left(\prod_{n' \in N(m) \setminus n} \operatorname{sgn}\left(y_{n'}\right) \right) \times \min_{n' \in N(m) \setminus n} \left|y_{n'}\right| \right], \tag{4.6}$$

com:

- $M_f(n)$ o conjunto de CN's fidedignos ligados ao BN_n.

O factor de correcção pode ser compreendido tendo em conta alguns factos já referidos anteriormente. Sabemos que o sinal de cada símbolo recebido, y_n, está na base da decisão binária r_n tomada para cada BN. Assim, para um dado CN_m fidedigno, ligado ao BN_n, sabemos que $\prod_{n' \in N(m) \setminus n} \operatorname{sgn}(y_{n'})$ é o produto dos sinais de todas as mensagens recebidas correspondentes aos BN's que intervêm no CN_m, com excepção do BN_n e, como tal, não é mais do que o sinal que deveria possuir y_n por forma a que a restrição correspondente ao CN_m fosse verificada.

No entanto, de entre os CN's fidedignos ligados ao BN_n, poderá haver alguns que requeiram $y_n > 0$, ao passo que outros requeiram $y_n < 0$. Torna-se pois necessário pesar a fiabilidade da decisão imposta por cada um desses CN's. Tal é tomada em conta pelo termo $\min_{n' \in N(m) \setminus n} |y_{n'}|$ que determina para cada CN_m fidedigno a menor amplitude dos símbolos fidedignos a ele ligados. De facto, tendo já sido referido que $|y_n|$ é uma medida de fiabilidade da decisão binária tomada para o símbolo y_n, se $\min_{n' \in N(m) \setminus n} |y_{n'}|$ é um valor elevado, então, com grande probabilidade se pode concluir que as mensagens $y_{n'}$ (com $n' \in N(m) \setminus n$) são fidedignas e, como tal, estará correcta a decisão de sinal $\prod_{n' \in N(m) \setminus n} \operatorname{sgn}(y_{n'})$ imposta pelo CN_m fidedigno, para o BN_n não fidedigno.

O somatório presente em (4.6) não é mais do que uma pesagem das contribuições de cada CN fidedigno na decisão a ser tomada para o BN_n não fidedigno.

Corrigidos os valores y_n recebidos, o método de descodificação segue os mesmos passos do algoritmo WBF. De referir ainda que o parâmetro α é função do código e do seu TG [NB], bem como, do SNR do canal.

O algoritmo pode ser pois expresso na seguinte forma:

A4. Algoritmo BWBF

(0) Identificação do conjunto I dos BN's não fidedignos de acordo com a equação

$$|y_n| < \alpha \, .$$

(1) Identificação do conjunto $M_f(n)$ dos CN's fidedignos ligados a cada $BN_n \in I$.

(2) Calcular:

$$y'_n = \begin{cases} y_n + \sum_{m \in M_f(n)} \left[\left(\prod_{n' \in N(m) \setminus n} \operatorname{sgn}(y_{n'}) \right) . \min_{n' \in N(m) \setminus n} |y_{n'}| \right] & , se \quad n \in I \\ y_n & , se \quad n \notin I \end{cases} ,$$

e determinar $\mathbf{r}$ em que:

$$r_n = \begin{cases} 0 & , se \; y'_n > 0 \\ 1 & , se \; y'_n < 0 \end{cases} .$$

(3) Calcular o sindroma $\mathbf{s} = \mathbf{r} \times \mathbf{H}^T$. Se $\mathbf{s} = \mathbf{0}$ parar a descodificação $\Rightarrow$ palavra correctamente descodificada.

(4) Calcular para cada CN_m, com $m = 1, \cdots, M$,

$$y_{\min}^{(m)} = \min \left\{ |y_n| : n \in N(m) \right\} .$$

(5) Calcular para cada BN_n, com $n = 1, \cdots, N$,

$$E_n = \sum_{m \in M(n)} (2s_m - 1) \, y_{\min}^{(m)} \, .$$

(6) Negar o bit r_n para o qual E_n é máximo.

(7) Repetir com a nova palavra $\mathbf{r}$ os passos (3), (5) e (6), até obter $\mathbf{s} = \mathbf{0}$ ou até que o número máximo de iterações estipulado seja atingido. Nesta situação é declarada falha na descodificação.

4.3 O ALGORITMO SOMA DE PRODUTOS

As melhorias de desempenho [KLF], [NB], dos algoritmos WBF e BWBF relativamente ao algoritmo BF, tornam bem patentes a importância do valor absoluto da informação recebida, $\mathbf{y}$, presente à entrada do descodificador. Tomando consciência deste facto, Gallager [Gal1], [Gal2] propôs também um algoritmo de descodificação iterativo do tipo SD que toma em conta a distribuição probabilística dos símbolos recebidos pelo descodificador. Este algoritmo é vulgarmente conhecido por Algoritmo Soma de Produtos (SPA), sendo também designado por *Belief Propagation* (BP).

O algoritmo SPA opera sobre o TG de um código binário linear e funciona, basicamente, como um algoritmo de passagem de mensagens entre nodos. De uma forma simplista, cada nodo pode ser visto como um processador de mensagens ("opiniões") recebidas dos seus vizinhos (nodos a ele se ligados), aos quais devolve mensagens actualizadas. As mensagens recebidas por um BN_n dos CN's que a ele se encontram ligados ($M(n)$), ou as enviadas desse BN para esses CN's, não são mais do que "opiniões", ou "convicções"[14], sobre o valor lógico desse BN. Essas convicções são expressas em termos de probabilidades que têm em conta, como já se referiu, a distribuição probabilística dos símbolos recebidos pelo descodificador.

O algoritmo SPA apresenta na descodificação de códigos LDPC o melhor desempenho, tendo Chung, Forney, Richardson e Urbanke [CFRU] demonstrado ser possível aproximar (a menos de $0.0045\,\text{dB}$) a capacidade de um canal Gaussiano aditivo considerando no limite um código de comprimento infinito. Nesta medida, em toda a literatura científica em que é abordada a descodificação SD de códigos LDPC, o algoritmo SPA é descrito na forma original, e no domínio logarítmico [MN1], [MN2], [Mac1], [Mac2], [KFL], [Ksch], [Ryan], [CF1], [CF2], [PL], [LLWP], [HEAD], [EMD], [FMI] e [HM], entre muitos outros. No entanto, apesar do seu desempenho, a sua implementação é limitada pela grande complexidade, pelo que na maioria dos artigos anteriormente referidos são propostas simplificações.

De seguida, apresentaremos o algoritmo SPA na forma original, precedido de uma breve referência ao canal de transmissão.

4.3.1 O CANAL DE TRANSMISSÃO

Como foi já referido, o algoritmo SPA toma em conta a distribuição probabilística dos símbolos recebidos. Para tal, é necessário conhecer previamente as características do canal. Geralmente, os canais considerados no estudo do desempenho de um dado algoritmo de descodificação são os chamados canais "bem comportados", como sejam, o canal Binário Simétrico (BSC), o canal Binário Erróneo (BEC) e o canal aditivo Gaussiano (AWGN) [Bos] [Carl]. Na realidade, as características dos canais de comunicação usados variam ao longo do tempo, pelo que a sua perfeita caracterização

[14] O conceito de "convicção" ou "opinião" está na base da designação de *Belief Propagation* (BP).

estatística torna-se complicada, senão impossível, o que origina um pior desempenho dos algoritmos de descodificação.

No que diz respeito à descodificação de códigos LDPC, tendo por base o algoritmo SPA, a grande maioria dos investigadores considera que o canal é Gaussiano (com características parecidas com as comunicações em espaço livre e de mais fácil tratamento teórico). Neste sentido, na descrição que a seguir se apresenta do algoritmo SPA, consideramos também que o canal é aditivo Gaussiano, sem que isso, no entanto, impeça a sua generalização para outro tipo de canais.

4.3.2 ALGORITMO SPA: A FORMA ORIGINAL

Consideremos um código binário LDPC (N,K) cujas palavras de código são transmitidas através de um canal aditivo Gaussiano. Para tal, os bits da palavra de código $c_n \in \{0,1\}$, com $n = 1,\cdots,N$, são modelados em BPSK, fazendo-lhes corresponder os símbolos, $x_n = (-1)^{c_n}$. Os símbolos recebidos são designados por y_n, com, $y_n = x_n + n_n$, e n_n ruído Gaussiano de média nula e desvio padrão σ.

O TG pode pois ser modificado de forma a incluir a informação recebida, como o objectivo de facilitar a descrição do algoritmo (ver exemplo 4.2).

Exemplo 4.2:

Considere-se o código de Hamming $(7,4)$ do exemplo 4.1. O TG deste código pode ser alterado de forma a tomar em conta os símbolos recebidos, obtendo-se o seguinte gráfico:

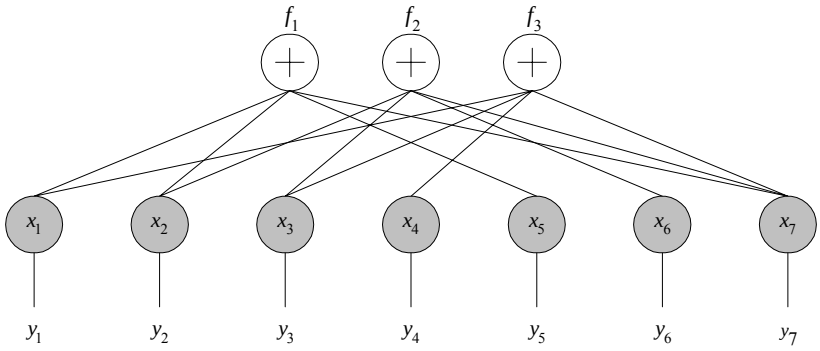


Figura 4-4 – TG modificado por forma a incluir a informação recebida.

Como já foi dito, o algoritmo SPA baseia-se na passagem de informação entre nodos. Assim, cada BN_n envia para cada um dos CN's a ele ligados toda a informação

que ele recebeu com excepção da informação que o CN em causa já possui. Ou seja, sendo m um CN ligado ao BN_n, a informação enviada por este BN para o CN_m inclui a informação que recebeu do canal, y_n, e a informação recebida de todos os outros CN's ligados ao BN_n, designada por *Informação Extrínseca*.

Da mesma forma, cada CN_m envia para cada BN que a ele se encontra ligado toda a informação extrínseca que ele recebeu dos restantes BN's a ele ligados. No entanto, neste caso a informação enviada tem que verificar a restrição associada a esse CN.

Exemplo 4.3:

Considere-se os seguintes gráficos parciais obtidos do TG da figura 4-4:

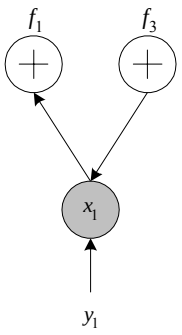


Figura 4-5

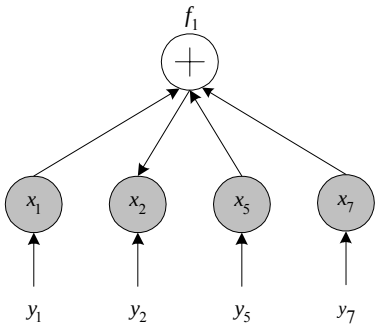


Figura 4-6

Na figura 4-5, a mensagem que o BN x_1 envia para o CN f_1 é formada pela informação y_1 recebido do canal e pela informação extrínseca recebida do CN f_3.

Da mesma forma, na figura 4-6, a informação que o CN f_1 envia para o BN x_2 é formada pela informação extrínseca recebida dos nodos x_1, x_5 e x_7.

Tratando-se de um algoritmo do tipo SD que considera a distribuição probabilística dos símbolos recebidos, facilmente se conclui que as mensagens a serem transmitidas entre os nodos, não são mais do que probabilidades. O objectivo do algoritmo SPA é maximizar a probabilidade à *posteriori* de um bit c_n, da palavra descodificada $\mathbf{c}$, ser 1 ou 0 tendo em conta a palavra recebida $\mathbf{y}$ e o conjunto de restrições S_n que o bit c_n tem de satisfazer. O que pretendemos é pois calcular:

$$\Pr\left(c_n = 0 \middle| \mathbf{y}, S_n\right), \tag{4.7}$$

e

$$\Pr\left(c_n = 1 \middle| \mathbf{y}, S_n\right). \tag{4.8}$$

Designemos por:

- $p_n = \Pr\left(c_n = 1 \middle| y_n\right)$ com y_n o i-ésimo símbolo da palavra recebida, $\mathbf{y}$;
- $c_{kj} \rightarrow$ k-ésimo bit da j-ésima equação de teste de paridade envolvendo c_n;
- $y_{kj} = (-1)^{c_{kj}} + n$, com n AWGN $\rightarrow$ Símbolo recebido correspondente a c_{kj};
- $p_{kj} = \Pr\left(c_{kj} = 1 \middle| y_{kj}\right)$.

Assumindo que os símbolos da palavra recebida $\mathbf{y}$ são estatisticamente independentes, Gallager [Gal1], [Gal2] apresentou um teorema no qual provou que a razão entre as probabilidades à *posteriori* (4.7) e (4.8) é dada por:

$$\frac{\Pr\left(c_n = 0 \middle| \mathbf{y}, S_n\right)}{\Pr\left(c_n = 1 \middle| \mathbf{y}, S_n\right)} = \frac{1 - p_n}{p_n} \frac{\prod_{m \in M(n)} \left(1 + \prod_{n' \in N(m) \setminus n} \left(1 - 2p_{n'm}\right)\right)}{\prod_{m \in M(n)} \left(1 - \prod_{n' \in N(m) \setminus n} \left(1 - 2p_{n'm}\right)\right)}. \tag{4.9}$$

Para provar (4.9) Gallager fez uso do lema L1 (ver anexo A) que diz que dada uma sequência de m dígitos binários estatisticamente independentes, $\mathbf{a} = (a_1, a_2, \cdots, a_m)$, em que $p_k = \Pr(a_k = 1)$, então, a probabilidade de $\mathbf{a}$ conter um número par de 1's é

$$\frac{1}{2} + \frac{1}{2} \prod_{k=1}^{m} (1 - 2p_k) \tag{4.10}$$

e a probabilidade de $\mathbf{a}$ conter um número impar de 1's é

$$\frac{1}{2} - \frac{1}{2} \prod_{k=1}^{m} (1 - 2p_k). \tag{4.11}$$

Por aplicação do teorema de Bayes chegamos à conclusão que:

$$\begin{aligned} \frac{\Pr\left(c_n = 0 \middle| \mathbf{y}, S_n\right)}{\Pr\left(c_n = 1 \middle| \mathbf{y}, S_n\right)} &= \frac{\Pr\left(c_n = 0, \mathbf{y}, S_n\right) \times \Pr\left(\mathbf{y}, S_n\right)}{\Pr\left(c_n = 1, \mathbf{y}, S_n\right) \times \Pr\left(\mathbf{y}, S_n\right)} \\ &= \frac{\Pr\left(S_n \middle| c_n = 0, \mathbf{y}\right) \times \Pr\left(c_n = 0 \middle| \mathbf{y}\right) \times \Pr\left(\mathbf{y}\right)}{\Pr\left(S_n \middle| c_n = 1, \mathbf{y}\right) \times \Pr\left(c_n = 1 \middle| \mathbf{y}\right) \times \Pr\left(\mathbf{y}\right)}. \end{aligned} \tag{4.12}$$

Partindo do pressuposto que o canal de comunicação não tem memória, os símbolos da palavra recebida $\mathbf{y}$ são estatisticamente independentes e, como tal,

$$\Pr\left(c_n = 1 \middle| \mathbf{y}\right) = \Pr\left(c_n = 1 \middle| y_n\right) = p_n, \tag{4.13}$$

pelo que obtemos

$$\frac{\Pr\left(c_n = 0 \middle| \mathbf{y}, S_n\right)}{\Pr\left(c_n = 1 \middle| \mathbf{y}, S_n\right)} = \frac{(1 - p_n)}{p_n} \frac{\Pr\left(S_n \middle| c_n = 0, \mathbf{y}\right)}{\Pr\left(S_n \middle| c_n = 1, \mathbf{y}\right)}. \tag{4.14}$$

Considerando $c_n = 0$, os restantes bits a 1 duma equação de teste de paridade envolvendo c_n, deverão ser em número par por forma a que a equação seja verificada. Assim, seja m uma das equações de teste de paridade que depende de c_n. Se $c_n = 0$, então, pelo Lema L1, a probabilidade de existir um número par de 1's entre os restantes bits que fazem parte da equação de teste de paridade m é:

$$\frac{1}{2} + \frac{1}{2} \prod_{n' \in N(m) \setminus n} \left(1 - 2p_{n'm}\right). \tag{4.15}$$

Logo, a probabilidade de todas as equações de teste de paridade dependentes de c_n, $M(n)$, serem satisfeitas para $c_n = 0$ é

$$\Pr\left(S_n \middle| c_n = 0, \mathbf{y}\right) = \prod_{m \in M(n)} \left(\frac{1}{2} + \frac{1}{2} \prod_{n' \in N(m) \setminus n} \left(1 - 2p_{n'm}\right) \right). \tag{4.16}$$

De forma idêntica, poderíamos concluir que para $c_n = 1$, temos

$$\Pr\left(S_n \middle| c_n = 1, \mathbf{y}\right) = \prod_{m \in M(n)} \left(\frac{1}{2} - \frac{1}{2} \prod_{n' \in N(m) \setminus n} \left(1 - 2p_{n'm}\right) \right), \tag{4.17}$$

pelo que substituindo (4.16) e (4.17) em (4.14) obtemos a razão (4.9).

À semelhança de Gallager, facilmente podemos concluir que o cálculo da razão (4.9) entre as probabilidades à *posteriori* é demasiado complexo. Gallager propôs um algoritmo iterativo para determinar essas probabilidades à *posteriori*, isto é, o algoritmo SPA.

Com vista a descrever o algoritmo é necessário referir alguma notação adicional. Assim, designemos:

- $q_{nm}(b)$ com $b \in \{0,1\}$ → mensagem que é enviada do BN_n para o CN_m indicando a probabilidade de $c_n = b$, baseado na informação recebida do canal, y_n, e em todos os CN's dependentes de c_n com excepção do CN_m;
- $r_{mn}(b)$ com $b \in \{0,1\}$ → mensagem que é enviada do CN_m para o BN_n indicando a probabilidade de $c_n = b$, baseado em todos os BN's ligados ao CN_m com excepção do BN_n;
- $Q_n(b)$ com $b \in \{0,1\}$ → pseudo-probabilidade à *posteriori* de $c_n = b$.

Tendo em conta a definição de $r_{mn}(b)$, o Lema L1 expresso pelas equações (4.10) e (4.11) e o facto de que a informação que o CN_m recebe de todos os BN's, $n' \in N(m) \setminus n$, é dada pelos valores $q_{n'm}$, então, podemos concluir que

$$r_{mn}(0) = \frac{1}{2} + \frac{1}{2} \prod_{n' \in N(m) \setminus n} (1 - 2p_{n'm}) = \frac{1}{2} + \frac{1}{2} \prod_{n' \in N(m) \setminus n} (1 - 2q_{n'm}(1)), \tag{4.18}$$

$$r_{mn}(1) = \frac{1}{2} - \frac{1}{2} \prod_{n' \in N(m) \setminus n} (1 - 2p_{n'm}) = \frac{1}{2} - \frac{1}{2} \prod_{n' \in N(m) \setminus n} (1 - 2q_{n'm}(1)), \tag{4.19}$$

e logo, a razão (4.9) pode ser expressa na forma,

$$\frac{\Pr(c_n = 0 | \mathbf{y}, S_n)}{\Pr(c_n = 1 | \mathbf{y}, S_n)} = \frac{1 - p_n}{p_n} \frac{\prod_{m \in M(n)} r_{mn}(0)}{\prod_{m \in M(n)} r_{mn}(1)}. \tag{4.20}$$

De igual modo, tendo em conta a definição de $q_{nm}(b)$ e de acordo com as ideias veiculadas no exemplo 4.3, podemos concluir que

$$q_{nm}(0) = (1 - p_n) \prod_{m' \in M(n) \setminus m} r_{m'n}(0), \tag{4.21}$$

$$q_{nm}(1) = p_n \prod_{m' \in M(n) \setminus m} r_{m'n}(1). \tag{4.22}$$

A decisão sobre o valor binário do bit *n* da mensagem recebida é então tomada tendo por base as pseudo-probabilidades à *posteriori*,

$$Q_n(0) = (1 - p_n) \prod_{m \in M(n)} r_{mn}(0), \tag{4.23}$$

$$Q_n(1) = p_n \prod_{m \in M(n)} r_{mn}(1), \tag{4.24}$$

onde o bit n da palavra descodificada $\hat{\mathbf{c}}$ é dado por

$$\hat{c}_n = \begin{cases} 1 & \Leftarrow \quad Q_n(1) > 0.5 \\ 0 & \Leftarrow \quad Q_n(1) < 0.5 \end{cases} \tag{4.25}$$

As equações (4.21) e (4.22), bem como, (4.23) e (4.24) deverão ser normalizadas por forma a que $q_{nm}(0) + q_{nm}(1) = 1$ e $Q_n(0) + Q_n(1) = 1$.

O algoritmo pode ser descrito do seguinte modo:

A5. Algoritmo SPA

Para todos os pares (BN_n, CN_m)*, ou seja, todos os pares* (n, m) *para os quais na matriz de teste de paridade* $\mathbf{H}$ *se tem* $h_{mn} = 1$.

(0) Inicialização:

$$q_{nm}(0) = 1 - p_n,$$

$$q_{nm}(1) = p_n.$$

(1) Calcular a mensagem que o CN_m *envia para o* BN_n*:*

$$r_{mn}(0) = \frac{1}{2} + \frac{1}{2} \prod_{n' \in N(m) \setminus n} (1 - 2q_{n'm}(1)),$$

$$r_{mn}(1) = 1 - r_{mn}(0).$$

(2) Calcular a mensagem que o BN_n *envia para o* CN_m*:*

$$q_{nm}(0) = k_{nm}(1 - p_n) \prod_{m' \in M(n) \setminus m} r_{m'n}(0),$$

$$q_{nm}(1) = k_{nm} p_n \prod_{m' \in M(n) \setminus m} r_{m'n}(1),$$

onde as constantes k_{nm} *são escolhidas por forma a que* $q_{nm}(0) + q_{nm}(1) = 1$.

(3) Calcular as pseudo-probabilidades à posteriori:

$$Q_n(0) = k_n(1 - p_n) \prod_{m \in M(n)} r_{mn}(0),$$

$$Q_n(1) = k_n p_n \prod_{m \in M(n)} r_{mn}(1),$$

onde as constantes k_n *são escolhidas por forma a que* $Q_n(0) + Q_n(1) = 1$.

(4) $\forall n$, *fazer:*

$$\hat{c}_n = \begin{cases} 1 & \Leftarrow \quad Q_n(1) > 0.5 \\ 0 & \Leftarrow \quad Q_n(1) < 0.5 \end{cases}.$$

Se a palavra descodificada $\hat{\mathbf{c}}$ *verificar as equações de teste de paridade (*$\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*) ou o número máximo de iterações tiver sido atingido, então, parar. Caso contrário, voltar a (1).*

Em caso de paragem devolver: - Palavra descodificada $\hat{\mathbf{c}}$ *se* $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*;*

- Erro se (# iterações = max_iterações).

Como podemos verificar a dedução do algoritmo foi feita apenas no pressuposto do canal não ter memória e, como tal, os símbolos recebidos, y_n, são estatisticamente independentes entre si. Neste sentido, o algoritmo é perfeitamente genérico, qualquer que seja o canal de transmissão sem memória considerado. Apenas na inicialização do

mesmo, temos que considerar a distribuição estatística dos símbolos y_n recebidos que é determinada pelas características do canal.

No caso de um canal AWGN, com média nula e desvio padrão σ, admitindo que os símbolos c_n, transmitidos com modulação BPSK, são de igual probabilidade, temos,

$$p_n = \Pr\left(c_n = 1 \middle| y_n\right) = \Pr\left(x_n = -1 \middle| y_n\right) = \frac{1}{1 + e^{\frac{2y_n}{\sigma^2}}}, \tag{4.26}$$

$$1 - p_n = \Pr\left(c_n = 0 \middle| y_n\right) = \Pr\left(x_n = +1 \middle| y_n\right) = \frac{1}{1 + e^{-\frac{2y_n}{\sigma^2}}}, \tag{4.27}$$

conforme é demonstrado no lema L2 do anexo A.

No caso de um canal BSC com probabilidade de erro q e probabilidade de acerto $p = 1 - q$, os símbolos y_n recebidos são binários e, como tal, obtemos de acordo com o lema L3 do anexo A,

$$p_n = q^{1-y_n} p^{y_n}, \tag{4.28}$$

$$1 - p_n = q^{y_n} p^{1-y_n}. \tag{4.29}$$

A primeira iteração do algoritmo SPA pode, neste caso, ser simplificada na medida em que por (4.28) verificamos que p_n toma apenas um de dois valores, p ou q, conforme $y_n = 1$ ou $y_n = 0$, respectivamente. Assim sendo, nas expressões (4.18) e (4.19) o termo $1 - 2q_{n'm}(1)$ poderá ser substituído por

$$\begin{aligned} 1 - 2q_{n'm}(1) &= 1 - 2p_{n'} \\ &= \begin{cases} 1 - 2q & \Leftarrow \quad y_{n'} = 0 \\ 1 - 2p & \Leftarrow \quad y_{n'} = 1 \end{cases} \\ &= (p - q)(-1)^{y_{n'}} \end{aligned}, \tag{4.30}$$

e logo, os valores $r_{mn}(b)$ da primeira iteração poderão ser calculados a partir das expressões:

$$\begin{aligned} r_{mn}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{n' \in N(m) \setminus n} (p - q)(-1)^{y_{n'}} \\ &= \frac{1}{2} + \frac{1}{2}(p - q)^{[\#N(m)]-1} \prod_{n' \in N(m) \setminus n} (-1)^{y_{n'}}, \end{aligned} \tag{4.31}$$

$$r_{mn}(1) = \frac{1}{2} - \frac{1}{2}(p - q)^{[\#N(m)]-1} \prod_{n' \in N(m) \setminus n} (-1)^{y_{n'}}. \tag{4.32}$$

Como se pode constatar, as mensagens $r_{mn}(b)$ enviadas pelo CN_m tomam apenas um de dois valores e dependem unicamente do facto de a restrição associada a esse CN ser ou não satisfeita. Assim, se a restrição for satisfeita:

$$r_{mn}(0) = r_{mn}(1) = \frac{1}{2} + \frac{1}{2}(p-q)^{[\#N(m)]-1} \tag{4.33}$$

e caso contrário

$$r_{mn}(0) = r_{mn}(1) = \frac{1}{2} - \frac{1}{2}(p-q)^{[\#N(m)]-1}. \tag{4.34}$$

Quando o código LDPC é regular, então, o termo $(p-q)^{[\#N(m)]-1}$ é igual qualquer que seja o CN_m considerado, o que reduz ainda mais a complexidade computacional.

As pseudo probabilidades à *posteriori* podem neste caso ser calculadas por,

$$\begin{aligned} Q_n(0) &= k_n(1-p_n)\prod_{m\in M(n)} r_{mn}(0) \\ &= k_n(1-p_n)\left[\frac{1}{2}+\frac{1}{2}(p-q)^{[\#N(m)]-1}\right]^S\left[\frac{1}{2}-\frac{1}{2}(p-q)^{[\#N(m)]-1}\right]^{[\#M(n)]-S}, \end{aligned} \tag{4.35}$$

$$Q_n(1) = k_n p_n\left[\frac{1}{2}+\frac{1}{2}(p-q)^{[\#N(m)]-1}\right]^S\left[\frac{1}{2}-\frac{1}{2}(p-q)^{[\#N(m)]-1}\right]^{[\#M(n)]-S}, \tag{4.36}$$

onde S é o número de CN's ligados ao BN_n, cujas equações de teste de paridade são satisfeitas.

4.3.3 CONSIDERAÇÕES SOBRE O ALGORITMO SPA

SPA um algoritmo "óptimo"

Prova-se que o algoritmo Soma de Produtos (SPA) [Gal] é de máxima verosimilhança quando aplicado a códigos descritos por gráficos de Tanner sem ciclos, ou seja, calcula com exactidão as probabilidades à *posteriori* (APP) [Pearl], [KFL], [Tan1], [ETV].

De facto, é possível obter ao fim de um número finito de iterações as probabilidades à *posteriori* $\Pr(c_n = 0|\mathbf{y}, S_n)$ e $\Pr(c_n = 1|\mathbf{y}, S_n)$, na medida em que as pseudo-probabilidades à *posteriori*, $Q_n(0)$ e $Q_n(1)$, convergem para esses valores à medida que o número de iterações aumenta. No entanto, gráficos de Tanner sem ciclos não suportam bons códigos [ETV] em virtude de estes possuírem uma baixa distância mínima.

Quando aplicado a códigos cujos TG possuem ciclos, o algoritmo deixa de ser óptimo na medida em que os ciclos introduzem uma realimentação positiva nas mensagens enviadas entre nodos. De facto, dado um código LDPC com giro g , apenas as mensagens enviadas ao longo das primeiras $g/2$ iterações são independentes, passando haver correlação entre elas nas iterações seguintes. Este é o motivo pelo qual na construção dos códigos LDPC se procura maximizar o giro [LS], [Lech]. No entanto, a sua aplicação a certos códigos LDPC provou ser possível aproximar a capacidade de um canal Gaussiano a menos de 0.0045dB [CFRU] [Mac]. Apesar de não ser óptimo, os resultados experimentais mostram que o algoritmo SPA apresenta um bom desempenho na descodificação de códigos LDPC, sobretudo nos de maior giro e nos códigos com comprimentos de bloco mais longos.

Erros de Descodificação

Outra consideração importante a tecer acerca do algoritmo SPA é o facto de ele permitir não só corrigir erros ocorridos, como também, detectar a existência de erros na medida em que a matriz de teste de paridade **H** é conhecida. Esta é uma grande vantagem relativamente ao algoritmo Viterbi utilizado na descodificação dos códigos convolucionais e do algoritmo BCJR utilizado na descodificação dos Turbo Codes, uma vez que estes algoritmos não nos informam da eventualidade da sequência descodificada ainda possuir erros.

No que diz respeito aos erros de descodificação é de referir o estudo realizado por Lechner e Sayir [LS], [Lech], sobre os diversos tipos de erros que podem ocorrer na descodificação. Lechner e Sayir identificaram três tipos de erros:

1. Convergência para uma palavra de código errada;
2. Convergência das pseudo-probabilidades à *posteriori* $Q_n(x)$ para um máximo local;
3. A não convergência das pseudo-probabilidades à *posteriori* $Q_n(x)$ para um ponto de equilíbrio.

Os dois últimos tipos de erros podem ser detectados, com o algoritmo a declarar falha na descodificação ao atingir o número máximo de iterações estipulado. Estes devem-se sobretudo à existência de ciclos e à realimentação positiva que estes originam.

Já o primeiro tipo de erros é mais grave na medida em que estes não podem ser detectados. Este tipo de erros está directamente relacionado com a distância mínima do

código e, como tal, não poderá ser evitado por uma simples melhoria do algoritmo de descodificação. Lechner e Sayir chegaram, no entanto, à conclusão que para códigos LDPC com comprimentos de bloco superiores a 1000 esse tipo de erros era praticamente desprezável.

Códigos LDPC Regulares vs Irregulares

Aquando da definição dos códigos LDPC regulares e irregulares no capítulo 3, ficou por explicar a razão do melhor desempenho dos códigos irregulares face aos códigos regulares.

Luby, Mitzenmacher, Shokrollahi e Spielman apresentam em [LMSS1] [LMSS2] uma das razões para esse melhor desempenho. Tendo em conta o algoritmo SPA, concluímos que quanto maior for o grau de cada BN, maior será a quantidade de informação que ele recebe e, logo, mais facilmente ele poderá inferir sobre qual o valor correcto que deve possuir. Por outro lado, do ponto de vista de cada CN quanto menor for o seu grau, menor é o número de BN's dos quais depende e logo, menor é a probabilidade de transmitir mensagens incorrectas para cada BN a ele ligado. Concluímos, portanto, que o grau de cada BN deverá ser o maior possível e o de cada CN o menor possível. Estes dois requisitos são contraditórios. No entanto, a opção por códigos LDPC irregulares, em que os nodos têm graus diferentes, permite contrabalançar estes dois requisitos levando a uma melhoria do desempenho do algoritmo SPA [LS]. De facto, os BN's de maior grau tendem mais rapidamente para o valor correcto, fornecendo ao CN's a ele ligados informação fiável que é transmitida aos BN's de menor grau, fazendo com que estes tendam para o valor correcto.

4.4 SIMPLIFICAÇÕES DO ALGORITMO SPA

Face ao bom desempenho do algoritmo SPA, são poucos os investigadores que se têm preocupado em melhorar o seu desempenho na descodificação dos códigos LDPC [MB1] [Foss]. Pelo contrário, a implementação eficiente do algoritmo SPA quer em hardware quer em software tem sido um tópico de crescente interesse.

São vários os desafios colocados ao nível do desempenho, expressa em termos de BER e MER, da latência e da complexidade computacional do algoritmo. Relativamente a estes dois últimos pontos há que salientar o elevado número de operações aritméticas,

nomeadamente, multiplicações, que são necessárias executar. Há também a referir os elevados requisitos de memória para armazenar os valores das mensagens enviadas entre nodos. A implementação do algoritmo SPA usando processadores de vírgula fixa, onde é necessário realizar a quantificação dos valores das mensagens enviadas entre nodos, mostrou também ser sensível aos erros de quantificação [PL].

Das várias modificações propostas ao algoritmo há a destacar a implementação no domínio logarítmico [CF1], [CF2], [HM], [HEAD]. De facto, sistemas de descodificação usando máximas verosimilhanças logarítmicas (LLR's) apresentam grandes vantagens uma vez que as multiplicações são substituídas por adições e o passo de normalização é eliminado. Neste sentido, quase todas as alterações propostas são no domínio logarítmico (LSPA).

4.4.1 AUMENTO DA EFICIÊNCIA COMPUTACIONAL DO ALGORITMO SPA E MINIMIZAÇÃO DA SUA SENSIBILIDADE AOS ERROS DE QUANTIFICAÇÃO

Antes de abordarmos a implementação do algoritmo SPA no domínio logarítmico, convém referir algumas técnicas que nos permitem melhorar a eficiência computacional do algoritmo e minimizar a sua sensibilidade aos erros de quantificação.

Uma forma simples de melhorar a eficiência computacional do algoritmo SPA em termos do número de operações realizadas, consiste na aplicação do método *frente e verso* (FB) no cálculo das mensagens enviadas entre CN's e BN's [EMD] [PL]. No cálculo da mensagem que um dado CN envia para um dado o BN a ele ligado são efectuados um conjunto de cálculos comuns à mensagem que esse mesmo CN envia para qualquer outro BN a ele ligado. O método FB minimiza o número de operações a realizar, procurando tirar o máximo de partido dos cálculos comuns às mensagens que um dado CN envia para os BN's a ele ligados e, vice-versa.

Dado um CN_m, com grau $k = \#N(m)$, ou seja, ligado a k BN's, facilmente se conclui que o número de multiplicações envolvidas no cálculo de todas as mensagens r_{mn} que esse CN envia para todos BN a ele ligados é de $k \times (k-1)$, isto é, de complexidade $O(k^2)$. No entanto, por aplicação do algoritmo FB, o número de multiplicações pode ser reduzido para cerca de $3 \times \left[\#N(m)\right] - 5$, ou seja, $O(k)$. Para códigos LDPC longos, em que o grau de cada CN é elevado, existe a uma redução significativa do número de multiplicações a realizar.

Como exemplo, veja-se uma das formas de aplicar o algoritmo FB no cálculo das mensagens que um dado CN_m envia para todos BN a ele ligados. Seja $k = \#N(m)$ e $\{n_1, n_2, \cdots, n_k\}$ os k elementos de $N(m)$. Do algoritmo SPA temos:

$$\begin{aligned} r_{mn}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{n_i \in N(m)\setminus n} \left(1 - 2q_{n_i m}(1)\right) \\ &= \frac{1}{2} + \frac{1}{2} \prod_{n_i \in N(m)\setminus n} \left(q_{n_i m}(0) - q_{n_i m}(1)\right). \end{aligned} \tag{4.37}$$

Definindo:

(*passagem para a frente*) $$\varepsilon_i = \varepsilon_{i-1} \times \left(q_{n_i m}(0) - q_{n_i m}(1)\right) \text{ c/ } i = 1, \cdots, k-1 \text{ e } \varepsilon_0 = \frac{1}{2}, \tag{4.38}$$

(*passagem para trás*) $$\nu_i = \nu_{i+1} \times \left(q_{n_i m}(0) - q_{n_i m}(1)\right) \text{ c/ } i = k, \cdots, 2 \text{ e } \nu_{k+1} = 1, \tag{4.39}$$

podemos, então, calcular com uma única passagem:

$$r_{mn_i}(0) = \frac{1}{2} + \varepsilon_{i-1} \times \nu_{i+1} \text{ c/ } i = 1, \cdots, k, \tag{4.40}$$

$$r_{mn_i}(1) = 1 - r_{mn_i}(0). \tag{4.41}$$

O algoritmo FB pode ser aplicado também no cálculo das mensagens que cada BN envia para os diversos CN's a ele ligados, $q_{nm}(x)$, com $x = \{0,1\}$. No entanto, por observação do algoritmo SPA podemos concluir que neste caso o número de operações, envolvidas (quase exclusivamente multiplicações) é ainda mais elevado. É necessário calcular $q_{nm}(0)$, $q_{nm}(1)$ e efectuar a normalização destes valores por forma a que $q_{nm}(0) + q_{nm}(1) = 1$. O mesmo ocorre no cálculo de $Q_n(x)$ com $x \in \{0,1\}$.

O número de operações a realizar é drasticamente reduzido se em vez do cálculo das probabilidades, p_n, $r_{mn}(x)$, $q_{nm}(x)$ e $Q_n(x)$, com $x \in \{0,1\}$, optarmos por calcular as máximas verosimilhanças,

$$u_n = \frac{1-p_n}{p_n},\ w_{mn} = \frac{r_{mn}(0)}{r_{mn}(1)},\ \nu_{nm} = \frac{q_{nm}(0)}{q_{nm}(1)} \text{ e } V_n = \frac{Q_n(0)}{Q_n(1)}, \tag{4.42}$$

usando a notação de Ping e Leung [PL].

Estes autores mostraram que a implementação directa do algoritmo SPA em *hardware* ou em processadores de vírgula fixa, levanta enormes problemas, motivados pela degradação do desempenho do algoritmo em virtude de o cálculo das mensagens $r_{mn}(x)$ ser baseado nas diferenças de probabilidades, $q_{nm}(0) - q_{nm}(1)$. Estes cálculos são muito sensíveis aos erros de quantificação, provocando uma acumulação de erros ao longo das várias iterações do algoritmo. Como alternativa, propuseram um método

baseado na *função de paridade de máxima verosimilhança*, $\mathrm{plr}(x)$, perfeitamente equivalente em termos aritméticos ao algoritmo SPA, mas muito menos sensível aos erros de quantificação.

4.4.2 ALGORITMO PLRA-SPA

De forma a apresentar o algoritmo SPA baseado na função de paridade de máxima verosimilhança proposto por Ping e Leung [LP1] e designado por PLRA-SPA, temos que fazer uso do lema L4 (ver anexo A). Dado um conjunto de variáveis aleatórias binárias, $x_1, x_2, \cdots, x_n$, com $n \geq 2$, e sendo y_n a variável aleatória resultante da soma módulo 2 das variáveis aleatórias x_i, com $i = 1, \cdots, n$, ou seja, $y_n = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, então, a máxima verosimilhança de y_n é função das máximas verosimilhanças das variáveis x_i, dada por

$$\mathrm{mv}(y_n) = \frac{1 + \prod_{i=1}^{n} \left(\frac{\mathrm{mv}(x_i) - 1}{\mathrm{mv}(x_i) + 1} \right)}{1 - \prod_{i=1}^{n} \left(\frac{\mathrm{mv}(x_i) - 1}{\mathrm{mv}(x_i) + 1} \right)}, \tag{4.43}$$

ou na forma recursiva por

$$\mathrm{mv}(y_n) = \frac{\mathrm{mv}(y_{n-1})\mathrm{mv}(x_n) + 1}{\mathrm{mv}(y_{n-1}) + \mathrm{mv}(x_n)}, \tag{4.44}$$

em que a máxima verosimilhança de uma variável aleatória x é definida por

$$\mathrm{mv}(x) = \frac{\Pr(x = 0)}{\Pr(x = 1)}. \tag{4.45}$$

O resultado apresentado pode ser transposto para o cálculo das mensagens enviadas de um CN para cada BN a ele ligado [PL] [HM] [EMD]. De facto, designemos por, $\{c_1, c_2, \cdots, c_N\}$, as variáveis aleatórias binárias associadas a cada um dos BN do código. Relembrando a definição de $r_{mn}(x)$, com $x \in \{0,1\}$, sabemos que $r_{mn}(x)$ é a mensagem que é enviada do CN_m para o BN_n indicando a probabilidade de $c_n = x$, baseado em todos os BN's ligados ao CN_m, com excepção do BN_n, ou seja,

$$r_{mn}(x) = \Pr\left(\sum_{n' \in N(m) \setminus n} \oplus c_{n'} = x \right), \tag{4.46}$$

onde $\sum\oplus$ é um somatório módulo 2 e em que as probabilidades dos $c_{n'}$ envolvidos no cálculo são os $q_{n'm}$.

Logo, a máxima verosimilhança w_{mn} da mensagem $r_{mn}(x)$ pode ser calculada a partir de (4.43) e (4.44) do lema L4, em que a máxima verosimilhança de cada uma das variáveis aleatórias $c_{n'}$, com $n' \in N(m) \setminus n$, envolvidas no cálculo é $v_{n'm}$, tal como foi definida em (4.42). Se dúvidas houvessem quanto à veracidade do resultado (4.46), poderíamos simplesmente recorrer à definição de w_{mn}, isto é,

$$w_{mn} = \frac{r_{mn}(0)}{r_{mn}(1)} = \frac{\frac{1}{2} + \frac{1}{2}\prod_{n' \in N(m)\setminus n}\left(1-2q_{n'm}(1)\right)}{\frac{1}{2} - \frac{1}{2}\prod_{n' \in N(m)\setminus n}\left(1-2q_{n'm}(1)\right)} = \frac{1+\prod_{n' \in N(m)\setminus n}\left(1-2q_{n'm}(1)\right)}{1-\prod_{n' \in N(m)\setminus n}\left(1-2q_{n'm}(1)\right)}, \tag{4.47}$$

o que atendendo a, $q_{n'm}(0) + q_{n'm}(1) = 1$ vem,

$$\begin{aligned} w_{mn} &= \frac{1+\prod_{n' \in N(m)\setminus n}\left(\frac{q_{n'm}(0)-q_{n'm}(1)}{q_{n'm}(0)+q_{n'm}(1)}\right)}{1-\prod_{n' \in N(m)\setminus n}\left(\frac{q_{n'm}(0)-q_{n'm}(1)}{q_{n'm}(0)+q_{n'm}(1)}\right)} \\ &= \frac{1+\prod_{n' \in N(m)\setminus n}\left(\frac{\frac{q_{n'm}(0)}{q_{n'm}(1)}-1}{\frac{q_{n'm}(0)}{q_{n'm}(1)}+1}\right)}{1-\prod_{n' \in N(m)\setminus n}\left(\frac{\frac{q_{n'm}(0)}{q_{n'm}(1)}-1}{\frac{q_{n'm}(0)}{q_{n'm}(1)}+1}\right)} \\ &= \frac{1+\prod_{n' \in N(m)\setminus n}\left(\frac{v_{n'm}-1}{v_{n'm}+1}\right)}{1-\prod_{n' \in N(m)\setminus n}\left(\frac{v_{n'm}-1}{v_{n'm}+1}\right)} \\ &= \mathrm{mv}\left(\sum_{n' \in N(m)\setminus n}\oplus c_{n'}\right) \end{aligned} \quad . \tag{4.48}$$

Tendo em conta este facto, Ping e Leung [PL] propuseram uma nova forma de implementação do algoritmo SPA, baseado na *função razão de paridade de máxima verosimilhança*, definida na seguinte forma:

Função Razão de Paridade de Máxima Verosimilhança

Seja $Z=\{z_1,z_2,\cdots,z_K\}$ *um conjunto de valores reais positivos. Representando o símbolo* $\cup$ *a união de dois subconjuntos de* Z *sem elementos em comum, a função* $\mathrm{plr}(x)$ *é definida por,*

$$\mathrm{plr}(Z)\equiv\begin{cases} z_1 & \Leftarrow \quad \#Z=1 \quad i.e. \quad Z=\{z_1\} \\ f\left(\mathrm{plr}(X),\mathrm{plr}(Y)\right) & \Leftarrow \quad \#Z>1 \quad e \quad Z=X\cup Y \\ & \qquad c/ \quad \#X>0 \quad e \quad \#Y>0 \end{cases}, \tag{4.49}$$

em que a função $f(x)$ *é dada por*

$$f(a,b)=\frac{ab+1}{a+b}. \tag{4.50}$$

Facilmente se conclui por análise da equação (4.50) e por comparação com (4.44) que esta nos dá a máxima verosimilhança da soma módulo 2 de duas variáveis aleatórias binárias cujas máximas verosimilhanças são a e b.

Designando por

$$V_m=\{v_{nm}:n\in N(m)\} \tag{4.51}$$

chegamos à conclusão que o algoritmo SPA pode agora ser escrito na forma:

A6. Algoritmo PLRA-SPA

Para todos os pares (BN_n,CN_m)*, ou seja, todos os pares* (n,m) *para os quais na matriz de teste de paridade* $\mathbf{H}$ *se tem* $h_{mn}=1$.

(0) Inicialização:

$$v_{nm}=\frac{1-p_n}{p_n}.$$

(1) Calcular a mensagem que o CN_m *envia para o* BN_n*:*

$$w_{mn}=\mathrm{plr}\left(V_m\setminus\{v_{nm}\}\right).$$

(2) Calcular a mensagem que o BN_n *envia para o* CN_m*:*

$$v_{nm}=\frac{(1-p_n)}{p_n}\prod_{m'\in M(n)\setminus m}w_{m'n}.$$

(3) Calcular as pseudo-probabilidades à posteriori:

$$V_n=\frac{(1-p_n)}{p_n}\prod_{m\in M(n)}w_{mn}.$$

(4) $\forall n$, *fazer:*

$$\hat{c}_n = \begin{cases} 1 & \Leftarrow \quad V_n < 1 \\ 0 & \Leftarrow \quad V_n > 1 \end{cases}.$$

Se a palavra descodificada $\hat{\mathbf{c}}$ *verificar as equações de teste de paridade (*$\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*) ou o número máximo de iterações tiver sido atingido, então, parar. Caso contrário, voltar a (1).*

Em caso de paragem devolver: *- Palavra descodificada* $\hat{\mathbf{c}}$ *se* $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*;*

- Erro se (# iterações = max_iterações).

Considerações sobre o algoritmo PLRA-SPA

Podemos constatar que no algoritmo PLRA-SPA o passo de normalização necessário no cálculo dos valores $q_{nm}(x)$ e $Q_n(x)$ foi eliminado. Além disso, podemos observar que o algoritmo acima descrito pode usar também o algoritmo FB no cálculo dos w_{mn}, v_{nm} e V_n. Estes dois últimos valores são calculados simultaneamente.

- **Cálculo dos w_{mn} recorrendo ao método FB** [PL]

Sejam $\{n_1, n_2, \cdots, n_k\}$ os índices dos $k = \#N(m)$ BN's ligados ao CN_m. Por (4.48) sabemos que no cálculo das mensagens w_{mn}, basta ter em conta que

$$\begin{aligned} w_{mn_i} &= \mathrm{mv}\left(c_{n_1} \oplus \cdots \oplus c_{n_{i-1}} \oplus c_{n_{i+1}} \oplus \cdots \oplus c_{n_k}\right) \\ &= \mathrm{mv}\left(z_1 \oplus z_2\right) \\ &= f\left(\mathrm{plr}(Z_1), \mathrm{plr}(Z_2)\right) \end{aligned} \quad , \tag{4.52}$$

com $z_1 = c_{n_1} \oplus \cdots \oplus c_{n_{i-1}}$, $z_2 = c_{n_{i+1}} \oplus \cdots \oplus c_{n_k}$, $Z_1 = \{c_{n_1}, c_{n_2}, \cdots, c_{n_{i-1}}\}$ e $Z_2 = \{c_{n_{i+1}}, c_{n_{i+2}}, \cdots, c_{n_k}\}$ para $\forall i \in \{1, \cdots, k\}$. Assim, os valores dos w_{mn} podem ser calculados da seguinte forma: Seja $A_{n_i m} = \{v_{n_1 m}, \cdots, v_{n_i m}\}$, com $i = 1, \cdots, k-1$, e $B_{n_i m} = \{v_{n_i m}, \cdots, v_{n_k m}\}$, com $i = 2, \cdots, k$, então, podemos escrever,

$$V_m \setminus v_{mn_i} = A_{n_{i-1} m} \cup B_{n_{i+1} m}, \tag{4.53}$$

em que se considera $A_{n_0 m}$ e $B_{n_{k+1} m}$ como sendo conjuntos vazios. Então, a passagem para a frente do algoritmo FB pode ser definida por,

$$\mathrm{plr}\left(A_{n_i m}\right) = f\left(\mathrm{plr}\left(A_{n_{i-1} m}\right), v_{n_i m}\right), \tag{4.54}$$

com $i = 2, \cdots, k-1$ e

$$\mathrm{plr}\left(A_{n_1 m}\right) = v_{n_1 m}\,. \tag{4.55}$$

De forma análoga a passagem para trás pode ser definida por

$$\mathrm{plr}\left(B_{n_i m}\right) = f\left(\mathrm{plr}\left(B_{n_{i+1} m}\right), v_{n_i m}\right), \tag{4.56}$$

com $i = k-1,\cdots,2$ e

$$\mathrm{plr}\left(B_{n_k m}\right) = v_{n_k m}\,. \tag{4.57}$$

Então de acordo com (4.52) podemos escrever,

$$w_{mn_i} = \mathrm{plr}\left(V_m \setminus v_{mn_i}\right) = f\left(\mathrm{plr}\left(A_{n_{i-1} m}\right), \mathrm{plr}\left(B_{n_{i+1} m}\right)\right), \tag{4.58}$$

onde são realizadas desta forma $3\times\left[\#N(m)\right]-6$ operações do tipo $f(a,b)$ (compostas por duas adições e duas multiplicações) nos cálculos de w_{mn} para cada CN_m.

- **Cálculo dos v_{nm} recorrendo ao método FB** [PL]

No cálculo das mensagens v_{nm} consideremos $j = \#M(n)$ e $\left\{m_1, m_2, \cdots, m_j\right\}$ os índices dos j CN's ligados ao BN_n. A passagem para a frente do algoritmo FB é definida por

$$\alpha_{m_i n} = \alpha_{m_{i-1} n}.w_{m_i n}\,, \tag{4.59}$$

com $i = 1,\cdots,j$ e

$$\alpha_{m_0 n} = u_n\,. \tag{4.60}$$

A passagem para trás consiste no cálculo de

$$\beta_{m_i n} = \beta_{m_{i+1} n}.w_{m_i n}\,, \tag{4.61}$$

com $i = j,\cdots,2$ e em que

$$\beta_{m_{j+1} n} = 1\,. \tag{4.62}$$

Podemos, então, calcular os valores v_{nm} através de

$$v_{nm_i} = \alpha_{m_{i-1} n}.\beta_{m_{i+1} n}\,, \tag{4.63}$$

com $i = 1,\cdots,j$ e em que a razão de máxima verosimilhança das pseudo-probabilidades à *posteriori* é dada por

$$V_n = \alpha_{m_j n}\,, \tag{4.64}$$

onde são realizadas $3\times\left[\#N(m)\right]$ multiplicações por cada BN.

Se do ponto de vista da eficiência expressa em termos do número de operações realizadas, o algoritmo PLRA-SPA é praticamente equivalente ao algoritmo SPA, o

mesmo já não se pode dizer no que diz respeito à minimização dos erros de quantificação (implementação em vírgula fixa).

Em primeiro lugar, é necessário definir a forma como são quantificados os valores envolvidos nos cálculos. Ping e Leung [PL] adoptaram no caso do algoritmo SPA, um esquema de quantificação uniforme. Esta opção faz todo o sentido na medida em que o cálculo das mensagens $r_{mn}(x)$ é feito com base nas diferenças, $q_{nm}(0)-q_{nm}(1)$, que tomam valores entre 0 e 1. Demonstraram que são necessários 12 bits de quantificação para obter um desempenho idêntico à do algoritmo SPA sem quantificação (para valores de BER $\leq 10^{-5}$). Já para o algoritmo PLRA-SPA tendo em conta que os valores envolvidos nos cálculos são máximas verosimilhanças (valores entre 0 e ∞), o esquema de quantificação adoptado por Ping e Leung define um conjunto de níveis de quantificação dados por s^i, com $s>1$ (inteiro), $i=0,\pm 1,\pm 2,\cdots,\pm\left(2^{L-1}-1\right)$ e L o número de bits de quantificação, o que corresponde a um quantificador uniforme no domínio logarítmico. Com este tipo de quantificação provaram ser necessários apenas de 6 bits para obterem um desempenho (para valores de BER $\leq 10^{-5}$) idêntico à do algoritmo SPA sem quantificação. No entanto, em termos do algoritmo PLRA-SPA, para obtermos o desempenho reportado é necessário truncar os valores de entrada da função (4.50) de forma a evitar erros grosseiros de cálculo [PL]. Assim, do ponto de vista da eficiência computacional expressa em termos de memória, verificamos que o algoritmo PLRA-SPA é mais vantajoso do que o SPA.

4.4.3 ALGORITMO SPA NO DOMÍNIO LOGARÍTMICO (LSPA)

Uma análise do algoritmo SPA na forma original [Gal1], [Gal2] [Mac2] [MN1] ou na forma proposta por Ping e Leung (algoritmo PLRA-SPA) [PL], mesmo aplicando o algoritmo FB no cálculo das mensagens enviadas entre nodos, permite concluir que estes requerem um número elevado de multiplicações. Uma vez que as multiplicações são mais dispendiosas computacionalmente do que as adições, vamos transferir os cálculos para o domínio logarítmico onde as multiplicações são convertidas em adições. Este é o princípio que está na origem da implementação do algoritmo SPA no domínio logaritmo [CF1], [CF2], [HM], [HEAD], [Ryan].

Comecemos por definir as máximas verosimilhanças logarítmicas das mensagens, r_{mn} e q_{nm}, transmitidas entre os nodos, bem como, as probabilidades à *posteriori* p_n e as pseudo-probabilidades à *posteriori* Q_n. Seja:

$$L(c_n) = \ln \frac{\Pr(c_n = 0 | y_n)}{\Pr(c_n = 1 | y_n)} = \ln \frac{1 - p_n}{p_n}, \tag{4.65}$$

$$L(r_{mn}) = \ln \frac{r_{mn}(0)}{r_{mn}(1)}, \tag{4.66}$$

$$L(q_{nm}) = \ln \frac{q_{nm}(0)}{q_{nm}(1)}, \tag{4.67}$$

$$L(Q_n) = \ln \frac{Q_n(0)}{Q_n(1)}. \tag{4.68}$$

A condição de inicialização do algoritmo SPA passa a ser

$$L(q_{nm}) = L(c_n) = \ln \frac{1 - p_n}{p_n}. \tag{4.69}$$

No caso do canal aditivo Gaussiano, e de acordo com as expressões (4.26) e (4.27) para p_n e $(1 - p_n)$, a condição de inicialização é uma expressão muito menos exigente do ponto de vista computacional (apenas 2 multiplicações). De facto, por substituição em (4.69) vem

$$\begin{aligned} L(q_{nm}) = L(c_n) = \ln \frac{1 - p_n}{p_n} &= \ln \frac{\left(1 + e^{-\frac{2y}{\sigma^2}}\right)^{-1}}{\left(1 + e^{\frac{2y}{\sigma^2}}\right)^{-1}} \\ = \ln \frac{1 + e^{\frac{2y}{\sigma^2}}}{1 + e^{-\frac{2y}{\sigma^2}}} &= \ln \frac{e^{\frac{y}{\sigma^2}}\left(e^{\frac{y}{\sigma^2}} + e^{-\frac{y}{\sigma^2}}\right)}{e^{-\frac{y}{\sigma^2}}\left(e^{\frac{y}{\sigma^2}} + e^{-\frac{y}{\sigma^2}}\right)} = \frac{2y}{\sigma^2}. \end{aligned} \tag{4.70}$$

No que toca à mensagem que cada BN_n envia para cada CN_m, por substituição de (4.21) e (4.22) em (4.67) resulta,

$$\begin{aligned} L(q_{nm}) &= \ln \frac{q_{nm}(0)}{q_{nm}(1)} \\ &= \ln \frac{K_{nm}(1-p_n) \prod_{m' \in M(n) \setminus m} r_{m'n}(0)}{K_{nm} p_n \prod_{m' \in M(n) \setminus m} r_{m'n}(1)} \\ &= \ln \frac{(1-p_n)}{p_n} + \sum_{m' \in M(n) \setminus m} \frac{r_{m'n}(0)}{r_{m'n}(1)} \\ &= L(c_n) + \sum_{m' \in M(n) \setminus m} L(r_{m'n}) \end{aligned} \quad . \tag{4.71}$$

Da mesma forma, para as pseudo-probabilidades à *posteriori*, resulta por substituição de (4.23) e (4.24) em (4.68) a seguinte expressão,

$$L(Q_n) = \ln \frac{Q_n(0)}{Q_n(1)} = L(c_n) + \sum_{m \in M(n)} L(r_{mn}) \, . \tag{4.72}$$

As equações (4.71) e (4.72) envolvem apenas somas pelo que o seu cálculo é muito menos exigente do que o cálculo dos q_{nm} e Q_n do algoritmo SPA, e dos v_{nm} e V_n do algoritmo PLRA-SPA. Além disso, podemos verificar que o cálculo de (4.71) e de (4.72) pode ser realizado numa única passagem. De facto, podemos verificar facilmente que:

$$L(q_{nm}) = L(Q_n) - L(r_{mn}) \tag{4.73}$$

ou seja, necessitamos de realizar apenas, e só, $2 \times [\#M(n)]$ adições para calcular simultaneamente todas as $\#M(n)$ mensagens que um dado BN_n envia para todos os CN's a ele ligados e a máxima verosimilhança logarítmica da pseudo probabilidade à *posteriori* associada a esse BN. A eficiência computacional é melhorada uma vez que o número de operações a realizar é menor e são menos complexas (adições).

Quanto ao cálculo da mensagem que cada CN_m envia para cada BN_n não basta substituir (4.18) e (4.19) em (4.66), pois o resultado obtido seria demasiado complexo de calcular. De facto, a eficiência computacional do algoritmo SPA no domínio logarítmico depende da forma como são calculadas as mensagens que cada CN envia para todos os BN's a ele ligados. São dois os parâmetros determinantes da complexidade computacional deste cálculo. O primeiro é a topologia do código directamente relacionada com o grau de cada CN e, logo, com a taxa de informação[15]. O

[15] Códigos com taxas de informação elevadas possuem gráficos de Tanner em que cada CN está ligado a muitos BN's, ou seja, em que o grau de cada CN é elevado.

segundo parâmetro é a forma como é implementada a operação base (fundamental) do cálculo das as mensagens que cada CN envia para todos os BN's a ele ligados.

Uma solução possível para o problema foi proposta por Gallager [Gal1], [Gal2], [Ryan] e [ZWP]. Comecemos por considerar o seguinte resultado:

$$\tanh\left(\frac{1}{2}\log\left(\frac{p_0}{p_1}\right)\right) = p_0 - p_1 = 1 - 2p_1 . \tag{4.74}$$

onde p_0 e p_1 são probabilidades com, $p_0 + p_1 = 1$.

Subtraindo (4.18) e (4.19) obtemos,

$$r_{mn}(0) - r_{mn}(1) = \prod_{n' \in N(m)} \left(1 - 2q_{n'm}(1)\right), \tag{4.75}$$

donde por (4.74) resulta que os termos $\left[r_{mn}(0) - r_{mn}(1)\right]$ e $\left[1 - 2q_{n'm}(1)\right]$ podem ser expressos por:

$$r_{mn}(0) - r_{mn}(1) = \tanh\left(\frac{1}{2}\log\left(\frac{r_{mn}(0)}{r_{mn}(1)}\right)\right) = \tanh\left(\frac{1}{2}L\left(r_{mn}\right)\right), \tag{4.76}$$

$$1 - 2q_{n'm}(1) = \tanh\left(\frac{1}{2}\log\left(\frac{q_{n'm}(0)}{q_{n'm}(1)}\right)\right) = \tanh\left(\frac{1}{2}L\left(q_{n'm}\right)\right). \tag{4.77}$$

Logo, a máxima verosimilhança logarítmica da mensagem que o CN_m envia para o BN_n é

$$L\left(r_{mn}\right) = 2\tanh^{-1}\left(\prod_{n' \in N(m)\setminus n} \tanh\left(\frac{1}{2}L\left(q_{n'm}\right)\right)\right). \tag{4.78}$$

A expressão obtida para $L\left(r_{mn}\right)$ ainda envolve multiplicações, ao contrário das expressões obtidas para $L\left(q_{nm}\right)$ e $L\left(Q_n\right)$. Além disso, podemos constatar que a expressão (4.78) depende da função tangente hiperbólica, cuja implementação é difícil [HMO], [LLHT], [HM].

Com vista a simplificar (4.78), vamos rescrever $L\left(q_{nm}\right)$ na forma [Gal1], [Gal2] [Ryan],

$$L\left(q_{nm}\right) = \alpha_{nm}\beta_{nm}, \tag{4.79}$$

com

$$\alpha_{nm} = \operatorname{sgn}\left(L\left(q_{nm}\right)\right), \tag{4.80}$$

$$\beta_{nm} = \left|L\left(q_{nm}\right)\right|. \tag{4.81}$$

Atendendo a que a função $\tanh(x)$ é impar resulta de (4.78), (4.79), (4.80) e (4.81) que

$$\tanh\left(\frac{1}{2}L(r_{mn})\right) = \prod_{n'\in N(m)\setminus n} \tanh\left(\frac{1}{2}L(q_{n'm})\right)$$
$$= \prod_{n'\in N(m)\setminus n} \alpha_{n'm} \prod_{n'\in N(m)\setminus n} \tanh\left(\frac{1}{2}\beta_{n'm}\right) \quad (4.82)$$

e logo,

$$L(r_{mn}) = \left(\prod_{n'\in N(m)\setminus n} \alpha_{n'm}\right) \times 2\tanh^{-1}\left(\prod_{n'\in N(m)\setminus n} \tanh\left(\frac{1}{2}\beta_{n'm}\right)\right)$$
$$= \left(\prod_{n'\in N(m)\setminus n} \alpha_{n'm}\right) \times 2\tanh^{-1}\log^{-1}\left[\log\left(\prod_{n'\in N(m)\setminus n} \tanh\left(\frac{1}{2}\beta_{n'm}\right)\right)\right]$$
$$= \left(\prod_{n'\in N(m)\setminus n} \alpha_{n'm}\right) \times 2\tanh^{-1}\log^{-1}\left(\sum_{n'\in N(m)\setminus n} \log\tanh\left(\frac{1}{2}\beta_{n'm}\right)\right)$$
$$= \left(\prod_{n'\in N(m)\setminus n} \alpha_{n'm}\right) \times \Phi\left(\sum_{n'\in N(m)\setminus n} \Phi(\beta_{n'm})\right) \quad (4.83)$$

onde

$$\Phi(x) = -\log\tanh\left(\frac{1}{2}x\right) = \log\left(\frac{e^x+1}{e^x-1}\right) \quad (4.84)$$

e

$$\Phi^{-1}(x) = \Phi(x). \quad (4.85)$$

Comparando agora as expressões (4.83) e (4.78), podemos constatar que todas as multiplicações foram convertidas em adições o que representa uma melhoria significativa da eficiência computacional do algoritmo. É claro que estamos a excluir o cálculo do produto dos sinais dos vários termos que intervêm no cálculo da mensagem $L(r_{mn})$ (simples operação EXOR - soma módulo 2).

A função base na qual assenta o cálculo das mensagens $L(r_{mn})$ é $\Phi(x)$, cuja implementação deve ser o mais eficiente possível. Como pode ser observado na figura 4-7 a função $\Phi(x)$ é contínua para valores de $x > 0$, apresentando, no entanto, uma de descontinuidade para $x = 0$.

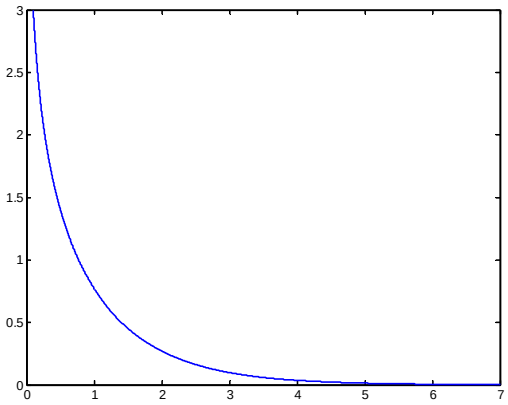


Figura 4-7 – Função $\Phi(x)$ para $x>0$.

Como forma de ultrapassar este problema e, simultaneamente, diminuir a complexidade do algoritmo, a função $\Phi(x)$ é muitas vezes implementada com recurso a uma tabela de consulta ou a uma aproximação linear por partes em que o declive de cada segmento de recta é uma potência de 2. De referir que esta última solução é bastante simples de realizar em *hardware* fazendo uso de registos de deslocamento. Obviamente, ao aproximar a função $\Phi(x)$, estamos a cometer pequenos erros de cálculo com consequências a nível do desempenho do algoritmo [HEAD].

O algoritmo SPA no domínio logarítmico pode ser descrito do seguinte modo:

A7. Algoritmo LSPA

Para todos os pares (BN_n, CN_m)*, ou seja, todos os pares* (m,n) *para os quais na matriz de teste de paridade* $\mathbf{H}$ *se tem* $h_{mn}=1$.

(0) Inicializacão:

$$L(q_{nm}) = L(c_n) = \frac{2y_n}{\sigma^2}$$

(1) Calcular a máxima verosimilhança logarítmica da mensagem que o CN_m *envia para o* BN_n*:*

$$L(r_{mn}) = \left(\prod_{n' \in N(m) \setminus n} \alpha_{n'm} \right) \Phi\left(\sum_{n' \in N(m) \setminus n} \Phi(\beta_{n'm}) \right)$$

com

$$\alpha_{nm} = \operatorname{sgn}\left(L(q_{nm})\right),$$

$$\beta_{nm} = \left|L(q_{nm})\right|,$$

$$\Phi(x) = -\log\tanh\left(\frac{1}{2}x\right) = \log\left(\frac{e^x+1}{e^x-1}\right).$$

(2) Calcular a máxima verosimilhança logarítmica da mensagem que o BN_n envia para o CN_m:

$$L(q_{nm}) = L(c_n) + \sum_{m'\in M(n)\setminus m} L(r_{m'n}).$$

(3) Calcular a máxima verosimilhança logarítmica das pseudo-probabilidades à posteriori:

$$L(Q_n) = L(c_n) + \sum_{m'\in M(n)} L(r_{m'n}).$$

(4) $\forall n$, *fazer:*

$$\hat{c}_n = \begin{cases} 1 & \Leftarrow \quad L(Q_n) < 0 \\ 0 & \Leftarrow \quad L(Q_n) > 0 \end{cases}$$

Se a palavra descodificada $\hat{\mathbf{c}}$ *verificar as equações de teste de paridade (*$\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*) ou o número máximo de iterações tiver sido atingido, então, parar. Caso contrário, voltar a (1).*

Em caso de paragem devolver: - Palavra descodificada $\hat{\mathbf{c}}$ *se* $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*;*

- Erro se (# iterações = max_iterações).

4.4.4 SIMPLIFICAÇÕES DO ALGORITMO LSPA

Uma análise atenta do algoritmo LSPA permite concluir que a sua complexidade reside no cálculo das mensagens enviadas de cada CN_m para cada BN_n, à semelhança do que acontece com as outras variantes já apresentadas do algoritmo SPA. Em [LLWP], [HEAD] e [HM] podemos encontrar um tratamento análogo ao seguido por Gallager [Gal1], [Gal2] para o cálculo das mensagens $L(r_{mn})$, que não só resolve o problema da descontinuidade da função $\Phi(x)$, como também, serve de ponto de partida para algumas simplificações do algoritmo LSPA, nomeadamente, o *min-sum* (MS-LSPA).

O desenvolvimento presente em [LLWP], [HEAD] e [HM] faz uso da função de máxima verosimilhança logarítmica e do lema L5 (ver anexo A). Seja y_n a variável aleatória resultante da soma módulo 2 de variáveis aleatórias x_i, com $i = 1, \cdots, n$, ou seja, $y_n = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, então, a máxima verosimilhança logarítmica de y_n é função das máximas verosimilhanças logarítmicas das variáveis x_i dadas por,

$$\mathrm{mvl}(y_n) = \frac{1+\prod_{i=1}^{n}\left(\frac{\mathrm{e}^{\mathrm{mvl}(x_i)}-1}{\mathrm{e}^{\mathrm{mvl}(x_i)}+1}\right)}{1-\prod_{i=1}^{n}\left(\frac{\mathrm{e}^{\mathrm{mvl}(x_i)}-1}{\mathrm{e}^{\mathrm{mvl}(x_i)}+1}\right)}, \tag{4.86}$$

ou na forma recursiva por,

$$\mathrm{mvl}(y_n) = \ln\left(\frac{\mathrm{e}^{\mathrm{mvl}(y_{n-1})+\mathrm{mvl}(x_n)}+1}{\mathrm{e}^{\mathrm{mvl}(y_{n-1})}+\mathrm{e}^{\mathrm{mvl}(x_n)}}\right), \tag{4.87}$$

em que a máxima verosimilhança de uma variável aleatória x é definida por

$$\mathrm{mvl}(x) = \ln\left(\frac{\Pr(x=0)}{\Pr(x=1)}\right). \tag{4.88}$$

Por uma questão de simplificação, vamos adoptar a notação seguida por Hu e Mittelholzer [HM] que definem a operação $\boxplus$ como sendo:

$$\mathrm{mvl}(x_1) \boxplus \mathrm{mvl}(x_2) \equiv \mathrm{mvl}(x_1 \oplus x_2) = \ln\left(\frac{\mathrm{e}^{\mathrm{mvl}(x_1)+\mathrm{mvl}(x_2)}+1}{\mathrm{e}^{\mathrm{mvl}(x_1)}+\mathrm{e}^{\mathrm{mvl}(x_2)}}\right) \tag{4.89}$$

e de forma idêntica podemos escrever:

$$\mathrm{mvl}(y_n) = \mathrm{mvl}(x_1 \oplus x_2 \oplus \cdots \oplus x_K) = \boxplus_{i=1}^{K}\left[\mathrm{mvl}(x_i)\right] \tag{4.90}$$

Podemos constatar que (4.86) e (4.87) resultam simplesmente da aplicação da função logaritmo natural às expressões (4.43) e (4.44) do lema L4, usadas na formulação do algoritmo PLRA-SPA. O desenvolvimento presente em [LLWP], [HEAD] e [HM] é pois em tudo idêntico ao que foi seguido no algoritmo PLRA-SPA.

Tomemos o resultado obtido em (4.46) no qual provámos que sendo, $\{c_1, c_2, \cdots, c_N\}$, as variáveis aleatórias binárias associadas a cada um dos BN do código, a mensagem $r_{mn}(x)$ que é enviada do CN_m para o BN_n indicando a probabilidade de $c_n = x$ com $x \in \{0,1\}$, é dada por

$$r_{mn}(x) = \Pr\left(\sum_{n' \in N(m)\backslash n} \oplus c_{n'} = x\right). \tag{4.91}$$

Recorrendo à definição de máxima verosimilhança logarítmica concluímos que a máxima verosimilhança logarítmica da mensagem que é enviada do CN_m para o BN_n é dado por:

$$L(r_{mn}) = \mathrm{mvl}\left(\sum_{n' \in N(m)\backslash n} \oplus c_{n'}\right) = \boxplus_{n' \in N(m)\backslash n}\left[L(q_{n'm})\right]. \tag{4.92}$$

A expressão assim obtida é perfeitamente equivalente à expressão (4.48) obtida para o algoritmo PLRA-SPA, em que a simples aplicação da função logaritmo natural a (4.48) conduz à obtenção de (4.92).

Pode-se também provar que (4.92) é equivalente à expressão (4.78) obtida por Gallager [Gal1], [Gal2], de acordo com o lema L6 (ver anexo A) ou seja,

$$2\tanh^{-1}\left(\tanh\left(\lambda/2\right)\tanh\left(\mu/2\right)\right)=\log\left(\frac{1+e^{\lambda+\mu}}{e^{\lambda}+e^{\mu}}\right). \tag{4.93}$$

Implementação série do algoritmo LSPA

A operação $\boxplus$ torna possível a aplicação do algoritmo FB no cálculo das mensagens $L\left(r_{mn}\right)$ à semelhança do realizado para o algoritmo PLRA-SPA [HEAD], [HM].

De facto, sejam $\left\{n_1,n_2,\cdots,n_k\right\}$ os índices dos $k=\#N\left(m\right)$ BN's ligados ao CN_m. A passagem para a frente do algoritmo pode então ser definida por

$$A_i=A_{i-1}\boxplus L\left(q_{n_i m}\right), \tag{4.94}$$

com $i=2,\cdots,k-1$ e

$$A_1=L\left(q_{n_1 m}\right). \tag{4.95}$$

De forma análoga, a passagem para trás pode ser definida por

$$B_i=B_{i+1}\boxplus L\left(q_{n_i m}\right), \tag{4.96}$$

com $i=k-1,\cdots,2$ e

$$B_k=L\left(q_{n_k m}\right). \tag{4.97}$$

Então, de acordo com (4.52), podemos escrever:

$$L\left(r_{mn_i}\right)=\begin{cases}B_2 & \Leftarrow i=1\\ A_{i-1}\boxplus B_{i+1} & \Leftarrow i=1,\cdots,k-1,\\ A_{k-1} & \Leftarrow i=k\end{cases} \tag{4.98}$$

sendo realizadas desta forma apenas $3\times\left[\#N(m)\right]-6$ operações do tipo $\boxplus$.

A aplicação do algoritmo FB no cálculo das mensagens $L\left(r_{mn}\right)$ que cada CN_m envia para os $k=\#N(m)$ BN's a ele ligados corresponde a uma implementação série do algoritmo LSPA que opera sobre uma árvore do tipo da figura 4-8 e, como tal, a latência do algoritmo assim implementado, na actualização das mensagens enviadas por cada

BN_n, é da ordem de $O(\#N(m))$. A operação $\boxplus$ permite também uma implementação paralela com uma latência da ordem de $O\left[\log_2(\#N(m))\right]$.

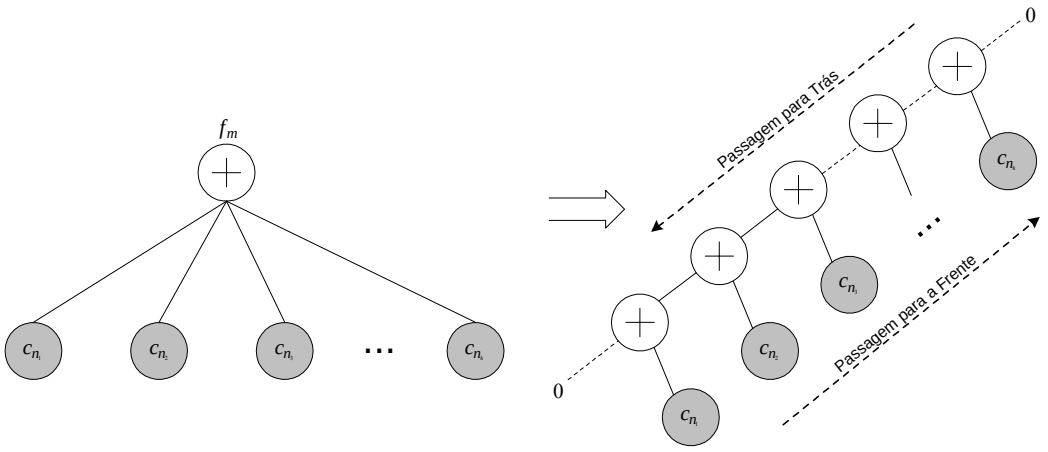


Figura 4-8 – Configuração série para o cálculo das mensagens que cada CN_m envia para os BN's a ele ligados.

Implementação eficiente da operação $\boxplus$

A partir de (4.89) onde a operação $\boxplus$ é definida, verificamos que

$$\mathrm{mvl}(x_1)\boxplus\mathrm{mvl}(x_2)=\begin{cases}0 & \Leftarrow \quad \mathrm{mvl}(x_1)=0\vee\mathrm{mvl}(x_2)=0\\ \mathrm{mvl}(x_2) & \Leftarrow \quad \mathrm{mvl}(x_1)=\infty\\ \mathrm{mvl}(x_1) & \Leftarrow \quad \mathrm{mvl}(x_2)=\infty\end{cases}, \tag{4.99}$$

pelo que 0 é o elemento absorvente da operação e que ∞ é o elemento neutro. Se o problema da descontinuidade da função $\Phi(x)$ aparentemente parece resolvido com a nova operação $\boxplus$, é necessário, no entanto, uma implementação eficiente desta pois é fundamental no cálculo dos $L(r_{mn})$, nos quais reside a complexidade do algoritmo LSPA.

Comecemos por aplicar o lema L7 (ver anexo A) à definição da operação $\boxplus$. O referido lema diz que dados dois números reais quaisquer x e y, então,

$$\log_a\left(a^x+a^y\right)=\max(x,y)+\log_a\left(1+a^{-|x-y|}\right). \tag{4.100}$$

pelo que a expressão (4.89) pode ser escrita na forma

$$\begin{aligned}\mathrm{mvl}(x_1)\boxplus\mathrm{mvl}(x_2) &= \ln\left(\frac{e^{\mathrm{mvl}(x_1)+\mathrm{mvl}(x_2)}+1}{e^{\mathrm{mvl}(x_1)}+e^{\mathrm{mvl}(x_2)}}\right)\\ &= \ln\left(e^{\mathrm{mvl}(x_1)+\mathrm{mvl}(x_2)}+1\right)-\ln\left(e^{\mathrm{mvl}(x_1)}+e^{\mathrm{mvl}(x_2)}\right)\\ &= \max\left(0,\mathrm{mvl}(x_1)+\mathrm{mvl}(x_2)\right)+\ln\left(1+e^{-|\mathrm{mvl}(x_1)+\mathrm{mvl}(x_2)|}\right)\\ &\quad-\max\left(\mathrm{mvl}(x_1),\mathrm{mvl}(x_2)\right)-\ln\left(1+e^{-|\mathrm{mvl}(x_1)-\mathrm{mvl}(x_2)|}\right). \qquad (4.101)\end{aligned}$$

Atendendo ao lema L8 (ver anexo A) que diz que dados dois números quaisquer, x e y, então,

$$\max(0,x+y)-\max(x,y)=\mathrm{sgn}(x)\mathrm{sgn}(y)\min\left(|x|,|y|\right), \qquad (4.102)$$

vem que

$$\begin{aligned}\mathrm{mvl}(x_1)\boxplus\mathrm{mvl}(x_2) &= \max\left(0,\mathrm{mvl}(x_1)+\mathrm{mvl}(x_2)\right)+\ln\left(1+e^{-|\mathrm{mvl}(x_1)+\mathrm{mvl}(x_2)|}\right)\\ &\quad-\max\left(\mathrm{mvl}(x_1),\mathrm{mvl}(x_2)\right)-\ln\left(1+e^{-|\mathrm{mvl}(x_1)-\mathrm{mvl}(x_2)|}\right)\\ &= \mathrm{sgn}\left[\mathrm{mvl}(x_1)\right]\mathrm{sgn}\left[\mathrm{mvl}(x_2)\right]\min\left(|\mathrm{mvl}(x_1)|,|\mathrm{mvl}(x_2)|\right)\\ &\quad+\ln\left(1+e^{-|\mathrm{mvl}(x_1)+\mathrm{mvl}(x_2)|}\right)-\ln\left(1+e^{-|\mathrm{mvl}(x_1)-\mathrm{mvl}(x_2)|}\right) \qquad . \qquad (4.103)\end{aligned}$$

Definindo a função

$$g(x)=\ln\left(1+e^{-|x|}\right), \qquad (4.104)$$

podemos constatar que a operação $\boxplus$ pode ser implementada à custa de 4 adições, uma comparação e duas correcções, correspondentes aos termos $g\left[\mathrm{mvl}(x_1)+\mathrm{mvl}(x_2)\right]$ e $g\left[\mathrm{mvl}(x_1)-\mathrm{mvl}(x_2)\right]$.

Como pode ser observado na figura 4-9, a função $g(x)$ é contínua, não apresentando qualquer descontinuidade para $x=0$, ao contrário do que acontecia com a função $\Phi(x)$, o que torna a sua implementação mais simples. A função $g(x)$ pode, à semelhança da função $\Phi(x)$ ser implementada com recurso a uma tabela de consulta ou utilizando uma aproximação linear por partes. Em [HEAD] é mostrado ser possível implementar a função com uma tabela de quantificação de 3 bits, cometendo um erro de quantificação máximo inferior a 0.05, com uma reduzida diminuição do desempenho do algoritmo LSPA. Como já foi referido anteriormente a implementação da função recorrendo a uma aproximação linear por partes em que o declive dos segmentos de recta é potência de 2 é bastante simples de realizar em *hardware* fazendo uso de registos de deslocamento [HEAD].

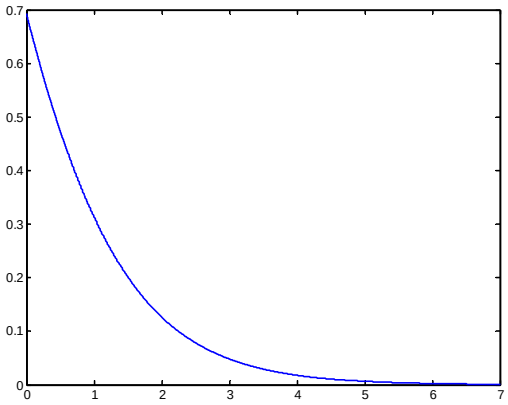


Figura 4-9 – Função $g(x)$ para $x > 0$.

A questão que pode ser colocada é a razão de implementar o algoritmo LSPA recorrendo à operação $\boxplus$ com a função $g(x)$ e não antes optar pela função $\Phi(x)$. Aparentemente, a opção pela operação $\boxplus$ requer que sejam realizadas mais 4 adições e 1 comparação por cálculo.

Tanto para a operação $\boxplus$, como para a função $\Phi(x)$, o resultado (máxima verosimilhança logarítmica) é um valor pertencente ao intervalo $]-\infty;+\infty[$, o que origina erros de quantificação elevados para valores de x próximos de 0, na implementação da função $\Phi(x)$ com recurso a uma tabela de consulta. Já no caso da implementação da operação $\boxplus$ por (4.103) o resultado é determinado pelo termo $\operatorname{sgn}\left[\operatorname{mvl}(x_1)\right]\operatorname{sgn}\left[\operatorname{mvl}(x_2)\right]\min\left(\left|\operatorname{mvl}(x_1)\right|,\left|\operatorname{mvl}(x_2)\right|\right)$ com os termos $g\left[\operatorname{mvl}(x_1)+\operatorname{mvl}(x_2)\right]$ e $g\left[\operatorname{mvl}(x_1)-\operatorname{mvl}(x_2)\right]$ a corresponderem apenas a pequenas correcções ao resultado (ver contradomínio da função $g(x)$, $]0;\ln(2)]$). Logo, a diminuição de desempenho do algoritmo LSPA, devido aos erros de quantificação da função $g(x)$ é muito menor do que no caso da função $\Phi(x)$.

Algoritmo Soma Mínima (*Min-Sum*)

Uma das principais versões simplificadas do algoritmo LSPA, resulta da aproximação,

$$\operatorname{mvl}(x_1)\boxplus\operatorname{mvl}(x_2)\cong\operatorname{sgn}\left[\operatorname{mvl}(x_1)\right]\operatorname{sgn}\left[\operatorname{mvl}(x_2)\right]\min\left(\left|\operatorname{mvl}(x_1)\right|,\left|\operatorname{mvl}(x_2)\right|\right) \quad (4.105)$$

em que os restantes termos da expressão (4.103) correspondem, como já foi dito, a uma correcção do valor (4.105). Desta forma, a operação $\boxplus$ passa a poder a ser

implementada à custa de apenas 1 comparação. A multiplicação dos sinais de $\mathrm{mvl}(x_1)$ e $\mathrm{mvl}(x_2)$ pode ser realizada de forma extremamente eficiente recorrendo a uma simples operação EXOR [CF1], [CF2].

Desta forma, no algoritmo LSPA o cálculo das mensagens $L(r_{mn})$ pode ser simplificado por

$$\begin{aligned} L(r_{mn}) &= \boxplus_{n'\in N(m)\setminus n} \left[L(q_{n'm}) \right] \\ &= \prod_{n'\in N(m)\setminus n} \left[\mathrm{sgn}\left(L(q_{n'm}) \right) \right] \times \min_{n'\in N(m)\setminus n} \left| L(q_{n'm}) \right| . \end{aligned} \tag{4.106}$$

Na medida em que o somatório da equação (4.83) é aproximado pelo menor dos seus termos, o algoritmo designa-se por *Soma Mínima* (MS-LSPA[16]) [CF1], [CF2], [HEAD], [HM], [Ryan].

Uma análise atenta da equação (4.106) permite concluir que na configuração série do cálculo das mensagens que o CN_m envia para os $k = [\#N(m)]$ CN's a ele ligados, estas podem ser calculadas numa única passagem da árvore da figura 4-8, sendo realizadas apenas $k-1$ operações $\boxplus$ compostas por 1 multiplicação de sinais e uma comparação. De facto, considere-se:

$$\alpha_{n_i m} = \mathrm{sgn}\left[L\left(q_{n_i m} \right) \right], \tag{4.107}$$

$$\beta_{n_i m} = \left| L\left(q_{n_i m} \right) \right| \tag{4.108}$$

com $n_i \in N(m)$ e $i = 1, \cdots, k$.

Nessa passagem única na árvore é efectuado o produto dos sinais de todas as mensagens $L\left(q_{n_i m}\right)$ recebidas pelo CN_m, ou seja, é calculado

$$\mathrm{A} = \prod_{n'\in N(m)} \alpha_{n'm} . \tag{4.109}$$

Simultaneamente, são determinados os dois menores valores $\beta_{n_i m}$ recebidos pelo CN_m. Designemos por $n_{\min}$ e $n_{\sec\min}$, os índices do menor e do segundo menor valor $\beta_{n_i m}$, respectivamente. O envio das mensagens do CN_m para os k BN's a ele ligados é realizado com uma segunda passagem na árvore mas, envolvendo apenas 1 multiplicação de sinal por mensagem. Assim, as mensagens enviadas são,

---

[16] O algoritmo referido pela sigla MS-LSPA é designado em [CF1] e [CF2] por BP-Based Algorithm.

$$L\left(r_{mn_i}\right)=\begin{cases} A\alpha_{n_i m}\beta_{n_{\min} m} & \Leftarrow \quad n_i \neq n_{\min} \\ A\alpha_{n_i m}\beta_{n_{\sec\min} m} & \Leftarrow \quad n_i = n_{\min} \end{cases}. \tag{4.110}$$

Desta forma, relativamente ao LSPA, não só o número de operações aritméticas que são necessárias realizar para cada CN sofre uma redução acentuada, como também a latência diminui.

Outro ponto importante na análise dos diversos algoritmos já apresentados diz respeito à memória necessária para guardar em cada iteração as mensagens enviadas dos CN's para os BN's e, vice-versa. Todos os algoritmos da família SPA estudados até agora, com excepção do MS-LSPA, requerem para cada CN_m, $k=\#N(m)$ unidades de memória, e requerem para cada BN_n, $j=\#M(n)$ unidades de memória. Para o algoritmo MS-LSPA são necessárias apenas duas unidades de memória por CN, o que corresponde a uma redução acentuada, sobretudo para códigos LDPC com taxa de informação elevada (o grau médio de cada CN é significativo[17]).

Devido à simplificação efectuada, o algoritmo MS-LSPA apresenta, para o mesmo número de iterações, um menor desempenho relativamente ao LSPA como é mostrado em [CF1], [CF2], [HEAD], [HM]. No entanto, a sua baixa complexidade computacional (apenas adições, comparações e multiplicações de sinais), bem como, os baixos requisitos de memória tornam-no num dos mais adequados à implementação em processadores digitais de sinal (DSP's) ou mesmo em *hardware* [ZWP], [YNA] [BNK].

O algoritmo MS-LSPA pode pois ser descrito do seguinte modo:

---

[17] Seja $\overline{\#N(m)}$ e $\overline{\#M(n)}$ o peso médio de cada CN e cada BN, respectivamente. Para um código LDPC regular (N,K), temos que $N\times\overline{\#M(n)}=(N-K)\times\overline{\#N(m)}$. Como $\overline{\#M(n)}$ é normalmente 3 ou 4, verificamos que para códigos com taxa de informação elevada $\overline{\#N(m)}$ é considerável.

A8. Algoritmo MS-LSPA

Para todos os pares $\left(BN_n, CN_m\right)$*, ou seja, todos os pares* $\left(m, n\right)$ *para os quais na matriz de teste de paridade* $\mathbf{H}$ *se tem* $h_{mn}=1$.

(0) Inicializacão:

$$L\left(q_{nm}\right)=L\left(c_n\right)=\frac{2y_n}{\sigma^2}$$

(1) Calcular a máxima verosimilhança logarítmica da mensagem que o CN_m *envia para o* BN_n*:*

$$L\left(r_{mn}\right)=\prod_{n'\in N(m)\setminus n}\left[\operatorname{sgn}\left(L\left(q_{n'm}\right)\right)\right]\times\min_{n'\in N(m)\setminus n}\left|L\left(q_{n'm}\right)\right|$$

(2) Calcular a máxima verosimilhança logarítmica da mensagem que o BN_n *envia para o* CN_m*:*

$$L\left(q_{nm}\right)=L\left(c_n\right)+\sum_{m'\in M(n)\setminus m}L\left(r_{m'n}\right).$$

(3) Calcular máxima verosimilhança logarítmica das pseudo-probabilidades à posteriori:

$$L\left(Q_n\right)=L\left(c_n\right)+\sum_{m'\in M(n)}L\left(r_{m'n}\right).$$

(4) $\forall n$, *fazer:*

$$\hat{c}_n=\begin{cases}1 & \Leftarrow \quad L\left(Q_n\right)<0\\ 0 & \Leftarrow \quad L\left(Q_n\right)>0\end{cases}$$

Se a palavra descodificada $\hat{\mathbf{c}}$ *verificar as equações de teste de paridade (*$\hat{\mathbf{c}}\mathbf{H}^T=\mathbf{0}$*) ou o número máximo de iterações tiver sido atingido, então, parar. Caso contrário, voltar a (1).*

Em caso de paragem devolver: - Palavra descodificada $\hat{\mathbf{c}}$ *se* $\hat{\mathbf{c}}\mathbf{H}^T=\mathbf{0}$*;*

- Erro se (# iterações = max_iterações).

Outra grande vantagem do algoritmo MS-LSPA reside na fase de inicialização. Na medida em que os passos (1) e (2) envolvem apenas adições, comparações e multiplicações de sinais, logo a multiplicação dos $L\left(c_n\right)$ por uma constante positiva não altera o desempenho do algoritmo. Neste sentido, o passo de inicialização poderá ser substituído por

$$L\left(c_n\right)=y_n, \tag{4.111}$$

de onde se conclui que não é necessário conhecer à partida a SNR do canal (sem qualquer redução do desempenho do algoritmo [CF1][18]).

Os algoritmos APP-LSPA e UMP-LSPA

A simplificação do algoritmo LSPA pode também ser realizada ao nível do processamento realizado nos BN's [CF2].

No algoritmo LSPA, para cada BN_n são realizados dois tipos de cálculos. O primeiro realizado no passo (2), calcula para cada CN_m ligado a esse BN a mensagem que este lhe envia, $L(q_{nm})$. Esta depende da informação recebida de todos os CN's ligados ao BN_n com excepção do CN_m pelo que, de alguma forma, $L(q_{nm})$ não se correlaciona com a mensagem $L(r_{mn})$ recebida previamente pelo BN_n do CN_m.

O segundo cálculo é efectuado no passo (3) e consiste em calcular a máxima verosimilhança logarítmica das pseudo-probabilidades à *posteriori*, $L(Q_n)$, com base na qual é efectuada uma decisão sobre o valor binário do bit n recebido.

No algoritmo *Iterative* APP-LSPA[19] proposto por Chen e Fossorier [CF2], o passo (2) do algoritmo LSPA é eliminado e cada BN_n envia a todos os CN's a ele ligados a mesma mensagem, $L(Q_n)$. A sigla APP diz respeito ao facto de o processamento realizado nos BN's, se basear apenas no cálculo das probabilidades à *posteriori*. Neste sentido, o algoritmo pode ser descrito por:

---

[18] Este algoritmo é designado em [CF1] e [CF2] por Uniformly Most Powerful BP-Based Algorithm (UMP BP-Based Algorithm).

[19] Passamos a designar o algoritmo *Iterative* APP-LSPA apenas por APP-LSPA.

A9. Algoritmo APP-LSPA

Para todos os pares (BN_n, CN_m)*, ou seja, todos os pares* (m, n) *para os quais na matriz de teste de paridade* $\mathbf{H}$ *se tem* $h_{mn} = 1$.

(0) Inicialização:

$$L(q_{nm}) = L(c_n) = \frac{2y_n}{\sigma^2}$$

(1) Calcular a máxima verosimilhança logarítmica da mensagem que o CN_m *envia para o* BN_n*:*

$$L(r_{mn}) = \left(\prod_{n' \in N(m) \setminus n} \alpha_{n'} \right) \Phi\left(\sum_{n' \in N(m) \setminus n} \Phi(\beta_{n'}) \right)$$

com

$$\alpha_n = \operatorname{sgn}(L(Q_n)),$$

$$\beta_n = |L(Q_n)|,$$

$$\Phi(x) = -\log \tanh\left(\frac{1}{2}x\right) = \log\left(\frac{e^x + 1}{e^x - 1}\right).$$

(2) Calcular máxima verosimilhança logarítmica das pseudo-probabilidades à posteriori:

$$L(Q_n) = L(c_n) + \sum_{m' \in M(n)} L(r_{m'n}).$$

(3) $\forall n$, *fazer:*

$$\hat{c}_n = \begin{cases} 1 & \Leftarrow \quad L(Q_n) < 0 \\ 0 & \Leftarrow \quad L(Q_n) > 0 \end{cases}$$

Se a palavra descodificada $\hat{\mathbf{c}}$ *verificar as equações de teste de paridade (*$\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*) ou o número máximo de iterações tiver sido atingido, então, parar. Caso contrário, voltar a (1).*

Em caso de paragem devolver: - Palavra descodificada $\hat{\mathbf{c}}$ *se* $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*;*

- Erro se (# iterações = max_iterações).

A grande vantagem do algoritmo APP-LSPA advém do facto de ser apenas necessário calcular e armazenar um valor por cada BN. Assim, comparando com a implementação mais eficiente do algoritmo LSPA, em que é feito uso do algoritmo FB no cálculo das mensagens enviadas por cada BN_n para os $j = \#M(n)$ CN's a ele ligados, verificamos que são realizadas com o algoritmo APP-LSPA menos j adições e é preciso apenas uma unidade de memória em vez de j.

No entanto, esta redução de complexidade computacional e das necessidades de armazenamento não é suficiente para compensar a diminuição de desempenho verificada para muitos códigos LDPC. Esta diminuição deve-se ao facto de no algoritmo APP-LSPA existir correlação entre a mensagem $L(Q_n)$, que cada BN_n envia para todos os CN's a ele ligados e as mensagens $L(r_{mn})$ que ele recebe desses mesmos CN's. Esta correlação é tanto maior quanto menor é o peso desse BN, o que provoca uma diminuição acentuada do desempenho do algoritmo APP-LSPA face ao LSPA, conforme pode ser verificado em [CF2]. Esta redução de desempenho pode, no entanto, ser compensada por normalização [CF2].

Já para códigos LDPC cujos BN's possuem um maior peso, como seja o caso dos códigos DSC[20], a diminuição de desempenho do algoritmo APP-LSPA é muito menos significativa, tornando-o numa boa alternativa face à complexidade do LSPA [CF2].

Simplificando o algoritmo LSPA simultaneamente ao nível do processamento realizado nos BN's [CF2] e nos CN's [CF1], [CF2], [HEAD], [HM], [Ryan], podemos reduzir ao mínimo as necessidades computacionais e de armazenamento. O algoritmo assim obtido, não é mais do que uma combinação dos algoritmos MS-LSPA e APP-LSPA. Designado por Chen e Fossorier [CF2] por *Iterative* APP-*Based Algorithm*, pode ser descrito por:

[20] Os DSC (*Difference-Set Cyclic Codes*) são uma das classes de códigos LDPC, referidos no capítulo 3, obtidos por métodos algébricos cuja geometria quasi-cíclica permite a sua implementação com base em registos de deslocamento. No entanto, a sua matriz $\mathbf{H}$ possui um peso por coluna bastante superior a 3 ou 4 como é característico dos códigos LDPC comuns.

A10. Algoritmo Iterative APP-Based

Para todos os pares (BN_n, CN_m)*, ou seja, todos os pares* (m, n) *para os quais na matriz de teste de paridade* $\mathbf{H}$ *se tem* $h_{mn} = 1$.

(0) Inicializacão:

$$L(q_{nm}) = L(c_n) = \frac{2y_n}{\sigma^2}$$

(1) Calcular a máxima verosimilhança logarítmica da mensagem que o CN_m *envia para o* BN_n*:*

$$L(r_{mn}) = \prod_{n' \in N(m) \setminus n} \left[\operatorname{sgn}\left(L(Q_{n'}) \right) \right] \times \min_{n' \in N(m) \setminus n} \left| L(Q_{n'}) \right|$$

(2) Calcular máxima verosimilhança logarítmica das pseudo-probabilidades à posteriori:

$$L(Q_n) = L(c_n) + \sum_{m' \in M(n)} L(r_{m'n}) .$$

(3) $\forall n$, *fazer:*

$$\hat{c}_n = \begin{cases} 1 & \Leftarrow \quad L(Q_n) < 0 \\ 0 & \Leftarrow \quad L(Q_n) > 0 \end{cases}$$

Se a palavra descodificada $\hat{\mathbf{c}}$ *verificar as equações de teste de paridade (*$\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*) ou o número máximo de iterações tiver sido atingido, então, parar. Caso contrário, voltar a (1).*

Em caso de paragem devolver: - Palavra descodificada $\hat{\mathbf{c}}$ *se* $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*;*

- Erro se (# iterações = max_iterações).

A diminuição de desempenho que se verifica relativamente ao algoritmo LSPA para códigos LDPC convencionais[21], é praticamente a mesma que ocorria para o algoritmo APP-LSPA, o que mostra que a combinação de ambas as simplificações não acarreta uma degradação adicional do desempenho, como atestam os resultados experimentais apresentados em [CF2]. O mesmo já não se passa com os códigos LDPC geométricos para os quais se verifica uma degradação acentuada de desempenho.

[21] Designamos por códigos LDPC convencionais, aqueles que são construídos segundo as estratégias propostas por Mackay e Neal (ver capítulo 3).

Métodos de melhoria de desempenho dos algoritmos MS-LPSA, APP-LSPA e Iterative APP-Based

Os algoritmos resultantes da simplificação do algoritmo LSPA colocam menores restrições ao nível das necessidades computacionais e de armazenamento à custa duma diminuição de desempenho expresso em termos de BER e MER (para o mesmo número de iterações). No entanto, é necessário referir que alguns destes algoritmos conseguem descodificar a mesma mensagem correctamente à custa de um maior número de iterações. Tal não pode ser considerado uma vantagem na medida em que o ganho de eficiência (diminuição da complexidade computacional) é perdido com o maior número de iterações necessárias para conseguir o mesmo desempenho.

Portanto, em cada aplicação prática há que pesar devidamente os diversos factores, como sejam: a capacidade de processamento e armazenamento disponível, o desempenho pretendido expresso em termo da taxa máxima de erros de bit (BER) ou de bloco (MER) para um dado SNR do canal, os requisitos de transmissão (por ex: transmissão em tempo real), entre outros.

Seria desejável melhorar o desempenho dos algoritmos MS-LSPA, APP-LSPA e Iterative APP-Based sem que isso implique um aumento significativo da sua complexidade computacional (grande vantagem).

A diminuição do desempenho resulta, no caso da simplificação do processamento nos CN's, da estimação por excesso das mensagens $L(r_{mn})$ que é realizada ao aproximar (4.92) por (4.106) devido à simplificação (4.105). Hu, Eleftheriou, Arnold e Dholakia sugerem em [HEAD] à semelhança do que é proposto em [EMD], a aproximação da operação $\boxplus$ por

$$\begin{aligned}\mathrm{mvl}(x_1)\boxplus\mathrm{mvl}(x_2) &= \mathrm{sgn}\left[\mathrm{mvl}(x_1)\right]\mathrm{sgn}\left[\mathrm{mvl}(x_2)\right]\min\left(\left|\mathrm{mvl}(x_1)\right|,\left|\mathrm{mvl}(x_2)\right|\right)\\ &\quad +g\left(\mathrm{mvl}(x_1)+\mathrm{mvl}(x_2)\right)-g\left(\mathrm{mvl}(x_1)-\mathrm{mvl}(x_2)\right)\\ &\cong \mathrm{sgn}\left[\mathrm{mvl}(x_1)\right]\mathrm{sgn}\left[\mathrm{mvl}(x_2)\right]\min\left(\left|\mathrm{mvl}(x_1)\right|,\left|\mathrm{mvl}(x_2)\right|\right)\\ &\quad +\begin{cases} c & \Leftarrow \left|\mathrm{mvl}(x_1)\right|<2\wedge\left|\mathrm{mvl}(x_2)\right|>2\left|\mathrm{mvl}(x_1)\right| \\ -c & \Leftarrow \left|\mathrm{mvl}(x_2)\right|<2\wedge\left|\mathrm{mvl}(x_1)\right|>2\left|\mathrm{mvl}(x_2)\right| \\ 0 & \Leftarrow \text{Outros Casos}\end{cases}, \end{aligned} \tag{4.112}$$

com c uma constante pré-determinada de acordo com o SNR do canal e as características do código. Desta forma, a operação base $\boxplus$ é implementada à custa de duas comparações, uma multiplicação de sinal e uma adição, pelo que o aumento da

complexidade computacional é muito baixo, quando comparado com o aumento do desempenho do algoritmo, conforme mostram os resultados presentes em [HEAD] e [EMD].

Chen e Fossorier propõem em [CF1], [CF2] uma abordagem diferente que procura resolver, simultaneamente, o problema da simplificação do processamento ao nível dos BN's em que passa a existir correlação entre a mensagem $L(Q_n)$, que cada BN$_n$ envia para todos os CN's a ele ligados e as mensagens $L(r_{mn})$ que ele recebe desses mesmos CN's. Esta correlação contribui de forma significativa para uma má estimação dos valores das mensagens $L(r_{mn})$ na iteração seguinte que, por sua vez, vai contribuir para uma má estimação das pseudo-probabilidades à *posteriori* $L(Q_n)$, ou seja, um ciclo com acumulação de erros que provoca uma diminuição efectiva do desempenho expresso em termos de BER e MER. Neste sentido, Chen e Fossorier [CF1], [CF2], propõem uma normalização das mensagens $L(r_{mn})$ obtidas por (4.106). Assim sendo, seja L_1 o valor obtido por (4.78), para a máxima verosimilhança logarítmica da mensagem enviada de um CN$_m$ para um BN$_n$ segundo o algoritmo LSPA, e por L_2 o valor obtido para a mesma mensagem por (4.106), ou seja,

$$L_1 = 2\tanh^{-1}\left(\prod_{n'\in N(m)\setminus n}\tanh\left(\frac{1}{2}L(q_{n'm})\right)\right), \tag{4.113}$$

$$L_2 = \prod_{n'\in N(m)\setminus n}\left[\operatorname{sgn}\left(L(q_{n'm})\right)\right]\times \min_{n'\in N(m)\setminus n}\left|L(q_{n'm})\right|. \tag{4.114}$$

Facilmente se prova que [CF1]

$$\text{sinal}(L_1) = \text{sinal}(L_2), \tag{4.115}$$

$$|L_2| > |L_1|, \tag{4.116}$$

pelo que Chen e Fossorier sugerem uma normalização do valor L_2, ou seja, a divisão de L_2 por um factor $\alpha > 1$ por forma a que o valor calculado para a máxima verosimilhança logarítmica da mensagem $L(r_{mn})$ enviada de um CN$_m$ para um BN$_n$, seja o mais próximo possível do valor L_1.

Com vista a determinar α, Chen e Fossorier sugerem que a média do módulo dos valores normalizados $|L_2|/\alpha$ seja igual à média do módulo dos valores $|L_1|$, ou seja,

$$\alpha = \frac{E(|L_2|)}{E(|L_1|)}. \tag{4.117}$$

Com este pressuposto, o valor α pode ser calculado com base em simulações de Monte Carlo (sugestão de Chen e Fossorier). Em [CF1] é apresentado o desenvolvimento teórico do valor de α a utilizar na primeira iteração no caso do algoritmo MS-LSPA. Chen e Fossorier além de afirmarem que a dedução teórica dos valores de α para as iterações seguintes não é simples, observaram também, experimentalmente, que apenas uma pequena melhoria é produzida com a utilização de valores de α diferentes, de iteração para iteração. Observaram ainda que apesar do valor óptimo de α ser função da SNR do canal, este poderá ser considerado fixo, desde que o valor escolhido para α seja o correspondente à SNR para o qual o BER $\in\left[10^{-4},10^{-3}\right]$. Para cada código LDPC passa a estar associado um valor específico de α pré-conhecido que depende do algoritmo de descodificação. Como é obvio, este é menor para o algoritmo MS-LSPA do que para o Iterative APP-Based.

A complexidade computacional é neste caso um pouco maior do que a da proposta apresentada por Hu, Eleftheriou, Arnold e Dholakia em [HEAD], na medida em que por cada mensagem $L(r_{mn})$ é efectuada mais uma divisão. No entanto, refira-se que se o valor de α for uma potência de 2, esta pode ser implementada eficazmente com base em registos de deslocamento. A proposta de Chen e Fossorier permite também compensar a diminuição de desempenho provocada pela simplificação ao nível dos BN's. O mesmo já não se verifica com a proposta apresentada por Hu, Eleftheriou, Arnold e Dholakia.

Implementação paralela do algoritmo LSPA e suas simplificações

A operação $\boxplus$ permite uma implementação paralela do algoritmo LSPA com uma latência da ordem de $O\left[\log_2\left(\#N(m)\right)\right]$. De facto, em aplicações com elevadas taxas de transmissão, o algoritmo FB (apesar de minimizar o número de operações $\boxplus$ a realizar no cálculo das mensagens $L(r_{mn})$), obriga a que a árvore da figura 4-8 seja percorrida 2 vezes e, como tal, a latência seja da ordem de $O\left[\#N(m)\right]$.

Hu, Eleftheriou, Arnold e Dholakia propõem em [HEAD] uma implementação paralela do algoritmo LSPA. Considere-se,

$$L(S_m)=\mathrm{mvl}\left(\sum_{n'\in N(m)}\oplus c_{n'}\right)=\boxplus_{n'\in N(m)}\left[L\left(q_{n'm}\right)\right], \tag{4.118}$$

como sendo a "soma" $\boxplus$ de todas as mensagens recebidas pelo CN_m dos $k = \#N(m)$ a ele ligados. O valor anterior pode ser calculado rapidamente recorrendo à estrutura paralela da figura 4-10, isto é, com uma latência da ordem de $O\left[\#N(m)\right]$.

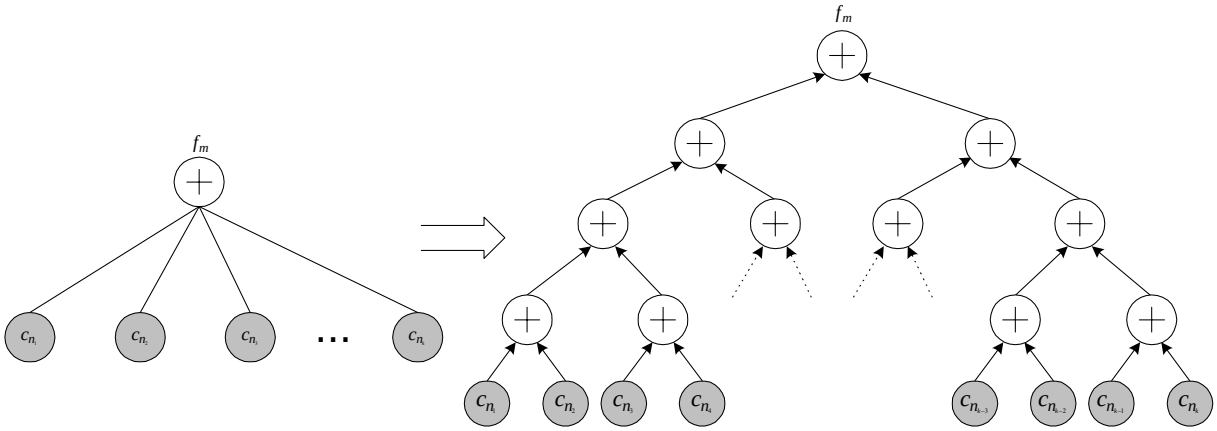


Figura 4-10 – Árvore equilibrada correspondente à configuração paralela para cálculo das mensagens que cada CN_m envia para os BN's a ele ligados.

Conhecido $L(S_m)$ podemos calcular simultaneamente todas as mensagens que o CN_m envia para cada um dos k BN's a ele ligados. De facto, de acordo com (4.118) vem,

$$L(S_m) = \mathrm{mvl}\left(c_n \oplus \sum_{n' \in N(m) \setminus n} \oplus c_{n'} \right) = L(q_{nm}) \boxplus L(r_{mn}), \tag{4.119}$$

pelo que efectuando alguma manipulação algébrica, com base na definição da operação $\boxplus$ temos que,

$$\begin{aligned} L(r_{mn}) &= \ln\left(\frac{e^{L(q_{nm})+L(S_m)} - 1}{e^{L(q_{nm})-L(S_m)} - 1} \right) - L(S_m) \\ &= \ln\left| e^{L(q_{nm})+L(S_m)} - 1 \right| - \ln\left| e^{L(q_{nm})+L(S_m)} - 1 \right| - L(S_m). \end{aligned} \tag{4.120}$$

Definindo a operação $\boxminus$ como sendo,

$$L(r_{mn}) = L(S_m) \boxminus L(q_{nm}) = \ln\left| e^{L(q_{nm})+L(S_m)} - 1 \right| - \ln\left| e^{L(q_{nm})+L(S_m)} - 1 \right| - L(S_m), \tag{4.121}$$

podemos obter uma implementação paralela do cálculo das mensagens enviadas por cada CN, representada na figura seguinte,

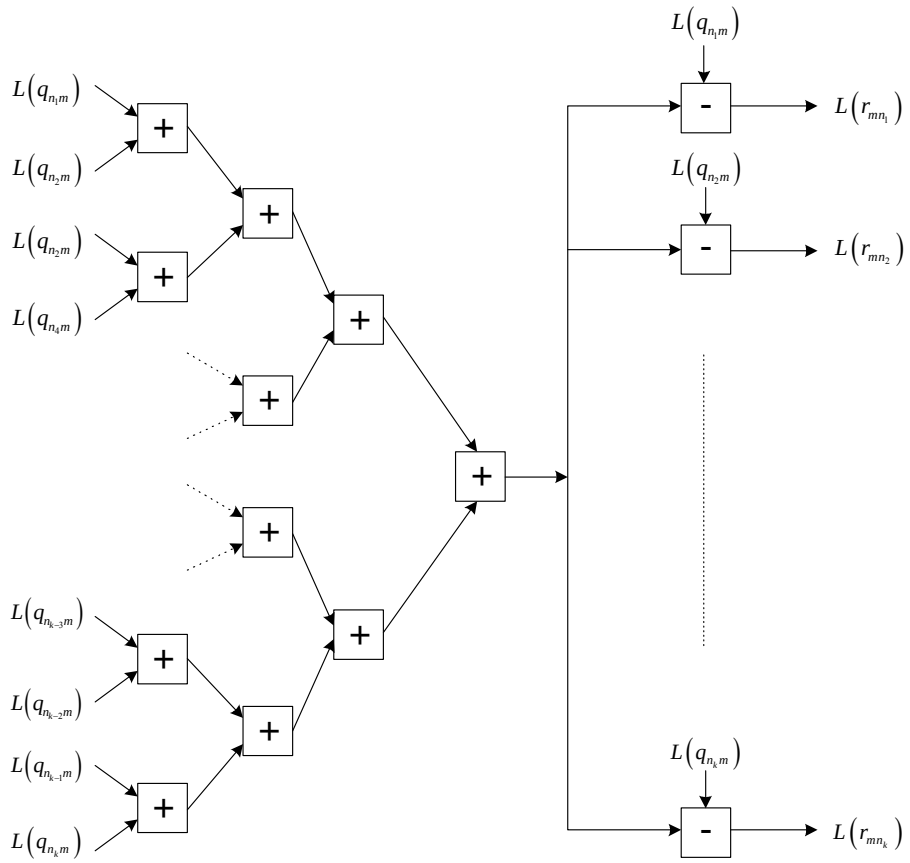


Figura 4-11 Configuração paralela para o cálculo das mensagens que cada CN_m envia para os BN's a ele ligados.

Definindo a função,

$$h(x) = \ln\left|e^x - 1\right|, \tag{4.122}$$

então, a expressão (4.120) pode ser escrita na forma

$$L(r_{mn}) = h\left[L(q_{nm}) + L(S_m)\right] - h\left[L(q_{nm}) - L(S_m)\right] - L(S_m), \tag{4.123}$$

o que implica o cálculo da função $h(x)$ nos pontos $L(q_{nm}) + L(S_m)$ e $L(q_{nm}) - L(S_m)$. No entanto, a função $h(x)$ apresenta uma descontinuidade em 0 (ver figura 4-12) similar à função $\Phi(x)$. A forma mais simples de implementar esta função é recorrendo a uma aproximação linear por partes em que os factores multiplicadores são potências de 2 conforme é sugerido em [HEAD]. Neste caso, resultados experimentais [HEAD] demonstram que a diminuição de desempenho é praticamente desprezável face ao LSPA implementado sem qualquer aproximação.

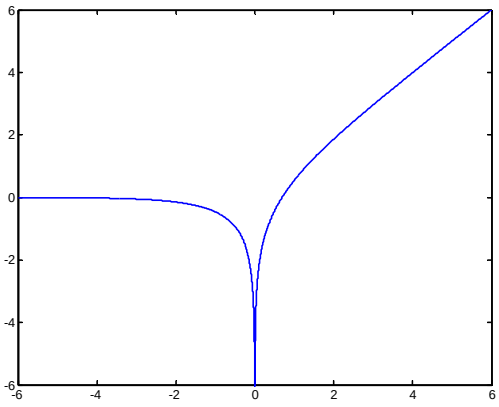


Figura 4-12 – Função $h(x)$.

A estrutura paralela da figura 4-10 serve de ponto de partida para uma simplificação do algoritmo LSPA proposta por Hu e Mittelholzer [HM], que se baseia no ordenamento estatístico das mensagens $L(q_{mn})$ recebidas pelo CN$_m$, em tudo semelhante à que é proposta por Guilloud, Boutillon e Danger [GBD].

O algoritmo MS-LSPA pode ser aplicado, tal como foi definido, às estruturas paralelas das figuras 4-10 ou 4-11. Designando por $n_{\min}$ e $n_{\sec\min}$, os índices do menor e do segundo menor valor $\left|L\left(q_{n_i m}\right)\right|$ recebido pelo CN$_m$, facilmente concluímos que a latência para determinar $\operatorname{sgn}(S_m)$, $\left|L\left(q_{n_{\min} m}\right)\right|$ e $\left|L\left(q_{n_{\sec\min} m}\right)\right|$ é $\log_2 k$, claramente inferior à versão série. Conhecidos estes valores podemos determinar, simultaneamente, as mensagens $L\left(r_{mn_i}\right)$ que o CN$_m$ envia para cada um dos BN's a ele ligados. De acordo com (4.110) temos,

$$L\left(r_{mn_i}\right)=\begin{cases}\operatorname{sgn}\left[L(S_m)\right]\operatorname{sgn}\left[L\left(q_{n_i m}\right)\right]\left|L\left(q_{n_{\min} m}\right)\right| & \Leftarrow \quad n_i \neq n_{\min} \\ \operatorname{sgn}\left[L(S_m)\right]\operatorname{sgn}\left[L\left(q_{n_i m}\right)\right]\left|L\left(q_{n_{\sec\min} m}\right)\right| & \Leftarrow \quad n_i = n_{\min}\end{cases} \tag{4.124}$$

A determinação de $\left|L\left(q_{n_{\min} m}\right)\right|$ e $\left|L\left(q_{n_{\sec\min} m}\right)\right|$ obriga a um ordenamento dos $\left|L\left(q_{n_i m}\right)\right|$ recebidos, com $i=1,\cdots,k$. Após a sua determinação, todas as restantes mensagens recebidas são consideradas como fidedignas, ou seja, $\left|L\left(q_{n_i m}\right)\right|=\infty$ se $n_i \neq n_{\min}$ e $n_i \neq n_{\sec\min}$. Como só é importante o seu sinal na determinação das mensagens $L\left(r_{mn_i}\right)$, então, a estrutura da figura 4-11 produz precisamente à sua saída as mensagens do

algoritmo MS-LSPA[22]. Esta aproximação considerando apenas os dois menores valores, conduz, no entanto, a uma diminuição de desempenho face ao algoritmo LSPA.

A ideia proposta por Hu e Mittelholzer [HM], [GBD], é considerar não apenas os 2 menores valores, mas sim, os λ menores $\left|L\left(q_{n_i m}\right)\right|$, com $\lambda = 2, \cdots, k$, classificando as restantes $k - \lambda$ mensagens recebidas como perfeitamente fidedignas. Neste sentido, a árvore da figura 4-10 pode ser decomposta em duas sub-árvores. A primeira cujas folhas são os λ menores $\left|L\left(q_{n_i m}\right)\right|$ e a segunda cujas folhas são as restantes mensagens consideradas fidedignas, tal como se representa na figura 4-13.

Tendo em conta que as folhas da sub-árvore direita são as $k - \lambda$ mensagens mais fidedignas cujo módulo é $+\infty$, logo para calcular $L\left(S_m''\right)$ é apenas necessário efectuar multiplicações de sinal (∞ é o elemento neutro da operação $\boxplus$, então, $\left|L\left(S_m''\right)\right| = \infty$). À sub-árvore esquerda formada pelas λ mensagens menos fidedignas, poderá ser aplicada o algoritmo LSPA ou qualquer uma das suas simplificações com vista à obtenção de $L\left(S_m'\right)$. Operando sobre a árvore equilibrada[23] da figura 4-13 esta operação poderá ser executada com uma latência de $\log_2 \lambda$ em vez de $\log_2 k$ (tempo que é gasto a percorrer a árvore equilibrada da figura 4-11 ao aplicar o algoritmo LSPA).

---

[22] Note que ∞ é o elemento neutro da operação $\boxplus$ e, como tal, também o será para a operação $\boxminus$.

[23] No sentido de que os ramos da árvore são quase equilibrados. Evitar a interpretação à letra do conceito da teoria da computação.

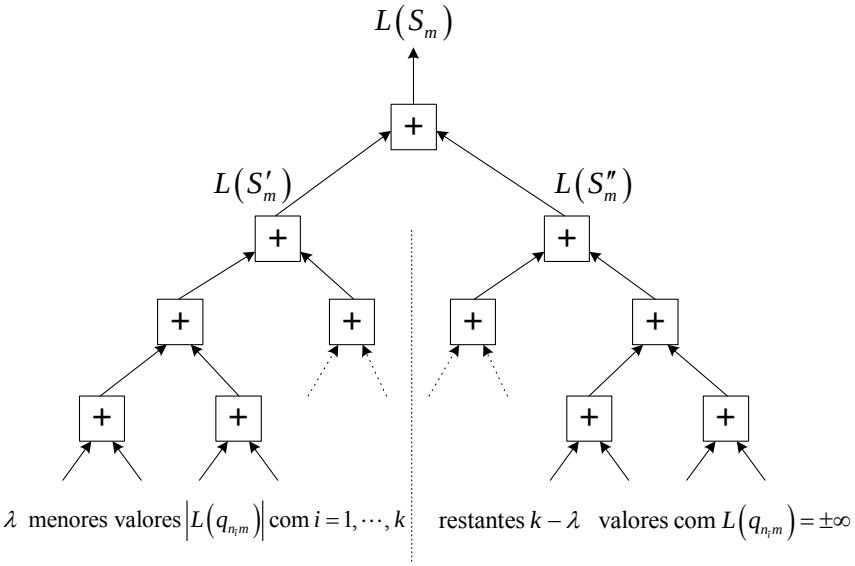


Figura 4-13 – Árvore equilibrada em que as λ mensagens menos fidedignas são separadas das restantes $k - \lambda$ mensagens mais fidedignas.

Desta forma é conseguido um compromisso entre latência, eficiência computacional e desempenho. Basta considerar $\lambda = 3$ ou $\lambda = 4$, para conseguir um desempenho muito próximo do algoritmo LSPA. De facto, a diminuição acentuada do desempenho do algoritmo MS-LSPA face ao algoritmo LSPA, sobretudo para códigos LDPC de maior comprimento, deve-se à pobre estimação das mensagens $L(r_{mn_i})$ enviadas pelos CN's (usamos apenas os 2 menores valores no seu cálculo). Ao considerarmos $\lambda > 2$ estamos a melhorar consideravelmente a estimação das mensagens $L(r_{mn_i})$ enviadas pelos CN's, face ao algoritmo MS-LSPA e, daí o aumento de desempenho verificado [HM], [GBD].

De referir também que a latência vai depender de forma significativa do algoritmo de ordenamento estatístico dos $|L(q_{n_i m})|$. O algoritmo aqui descrito baseado na estrutura em árvore equilibrada da figura 4-10, pode ser também aplicado a uma estrutura em série como a da figura 4-8 mas, com o consequente aumento da latência.

Por último, para $\lambda = 2$ o algoritmo corresponde, essencialmente, ao MS-LSPA, com uma pequena diferença. Esta consiste no facto do módulo das mensagens enviadas pelo CN_m para todos os BN's n_i para os quais $n_i \neq n_{\min}$ e $n_i \neq n_{\sec\min}$ ser, neste caso, $|L(q_{n_{\min} m}) \boxplus L(q_{n_{\sec\min} m})|$, ao passo que no algoritmo MS-LSPA é $|L(q_{n_{\min} m})|$.

O algoritmo λ – Min LSPA, segundo a designação de [GBD] pode ser expresso por:

A11. Algoritmo λ-Min LSPA

Para todos os pares (BN_n, CN_m)*, ou seja, todos os pares* (m,n) *para os quais na matriz de teste de paridade* $\mathbf{H}$ *se tem* $h_{mn}=1$.

(0) Inicializacão:

$$L(q_{nm}) = L(c_n) = \frac{2y_n}{\sigma^2}$$

(1) Calcular a máxima verosimilhança logarítmica da mensagem que o CN_m *envia para o* BN_n*:*

$$L(r_{mn}) = \left(\prod_{n' \in N(m) \setminus n} \alpha_{n'm} \right) \Phi\left(\sum_{i=1}^{\lambda} \Phi(\beta_i) \right)$$

com

$$\alpha_{nm} = \operatorname{sgn}\left(L(q_{nm})\right),$$

$$\beta_{nm} = \left|L(q_{nm})\right| \text{ e } \beta_i \text{ o i-ésimo menor valor } \beta_{n'm} \text{ com } n' \in N(m) \setminus n$$

$$\Phi(x) = -\log\tanh\left(\frac{1}{2}x\right) = \log\left(\frac{e^x+1}{e^x-1}\right).$$

(2) Calcular máxima verosimilhança logarítmica das pseudo-probabilidades à posteriori:

$$L(Q_n) = L(c_n) + \sum_{m' \in M(n)} L(r_{m'n}).$$

(3) $\forall n$, *fazer:*

$$\hat{c}_n = \begin{cases} 1 & \Leftarrow \quad L(Q_n) < 0 \\ 0 & \Leftarrow \quad L(Q_n) > 0 \end{cases}$$

Se a palavra descodificada $\hat{\mathbf{c}}$ *verificar as equações de teste de paridade (*$\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*) ou o número máximo de iterações tiver sido atingido, então, parar. Caso contrário, voltar a (1).*

Em caso de paragem devolver: - Palavra descodificada $\hat{\mathbf{c}}$ *se* $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$*;*

- Erro se (# iterações = max_iterações).

4.5 GESTÃO PROBABILÍSTICA DO CÁLCULO DAS MENSAGENS

O algoritmo SPA e as suas simplificações, são algoritmos iterativos onde em cada passo do processamento, quer dos BN's quer dos CN's, cada nodo envia uma mensagem para cada um dos nodos a ele ligados. Este método de passagem de mensagens é

designado por *flooding schedule* [MB1], [KF]. Apesar do algoritmo SPA ser óptimo quando aplicado a códigos cujos TG não possuem ciclos, o mesmo não se passa quando aplicado a códigos cujos TG possuem ciclos[24]. Os ciclos introduzem realimentações positivas nas mensagens enviadas entre nodos. De facto, consideremos um código LDPC caracterizado por um dado TG e designemos por:

- g_i → giro do nodo i, ou seja, o comprimento do ciclo mais pequeno do TG que contém o nodo i;
- $g_{\max}$ e $g_{\min}$ → giro máximo e mínimo de todos os nodos que constituem o TG ($g_{\min}$ é o giro do código).

Dado um BN_n com giro $g_{\min}$, então, são necessárias $g_{\min}/2$ iterações do algoritmo SPA para que a mensagem enviada inicialmente pelo BN_n se propague de volta para esse BN. Assumindo que os bits das mensagens são estatisticamente independentes (canal de transmissão sem memória), então, até à iteração $g_{\min}/2$ as mensagens passadas entre cada par de nodos são óptimas, ou seja, não correlacionadas entre si. Os símbolos recebidos com base nos quais o algoritmo é inicializado e as mensagens recebidas até essa iteração por cada nodo são independentes. No entanto, a partir da iteração $g_{\min}/2$, as mensagens recebidas pelo BN_n, ou por qualquer outro BN com giro $g_{\min}$, passam a ser dependentes das mensagens enviadas inicialmente por esses nodos, colocando em causa a convergência do algoritmo para as verdadeiras probabilidades à *posteriori*. No entanto, para qualquer outro BN com giro $g > g_{\min}$, as mensagens enviadas continuavam a ser óptimas até à iteração $g/2$.

Para solucionar este problema Mao e Banihashemi [MB1] sugerem uma gestão temporal do cálculo das mensagens, como forma de melhorar o desempenho do algoritmo SPA sem qualquer aumento da sua complexidade. Assim, é proposto sincronizar os instantes de tempo em que os diferentes BN's enviam para os CN's a eles ligados mensagens não "óptimas". Por exemplo, o BN_n anteriormente referido, com giro $g_{\min}$, não deverá actualizar as mensagens que envia a partir da iteração $g_{\min}/2$, enviando sempre as mesmas mensagens a cada um dos BN's a ele ligados até à iteração $g_{\max}/2$, altura em que deverá ser retomado o processo de actualização das mensagens a enviar

[24] Gráficos de Tanner sem ciclos não representam bons códigos [ETV] em virtude de estes possuírem, nesse caso, uma baixa distância mínima.

em cada iteração. O mesmo se passa para qualquer BN_v com giro $g_{\min} < g_v < g_{\max}$, em que entre a iteração $g_v/2$ e $g_{\max}/2$ as mensagens enviadas pelo BN_v não são actualizadas.

Esta estratégia apresenta duas vantagens [MB1]. Por um lado é maximizado o número de iterações do algoritmo SPA em que apenas as mensagens "óptimas" circulam ao longo do TG e, por outro lado, na iteração $g_{\max}/2$ quando é retomado para cada BN_n o processo de actualização da mensagem a enviar em cada iteração, a mensagem recebida pelo BN_n contém muito mais informação óptima do que conteria na iteração $g_n/2$.

Existem várias formas de implementar uma estratégia semelhante à descrita. Por exemplo, uma gestão probabilística do cálculo das mensagens, conforme é sugerido por Mao e Banihashemi em [MB1]. Neste caso, a cada BN_n é associada uma probabilidade, $p_n = g_n/g_{\max}$, de em cada iteração do algoritmo SPA a mensagem enviada pelo BN_n ser actualizada, com excepção da primeira iteração em que é seguido o algoritmo SPA na sua forma original. Resultados experimentais [MB1] evidenciam uma melhoria significativa com este método relativamente ao algoritmo SPA. O número de iterações realizadas na descodificação de uma palavra de código para um dado SNR do canal é praticamente igual ao número de iterações que seriam realizadas pelo algoritmo SPA. No entanto, o número de cálculos realizados é em muitos dos casos inferior ao do algoritmo SPA na medida que nem todas as mensagens enviadas por cada BN são calculadas em cada iteração. De facto, sendo E o número de ramos do TG do código, então, no algoritmo SPA são calculadas em cada iteração $2E$ mensagens, ao passo que com o método probabilístico anterior apenas são calculadas $E + \sum_n j\, g_n / g_{\max}$ com $j = \#M(n)$.

CAPÍTULO 5

DESCODIFICAÇÃO ITERATIVA – RESULTADOS EXPERIMENTAIS

Neste capítulo apresentamos resultados experimentais obtidos para alguns dos algoritmos apresentados no capítulo 4, procurando realizar uma comparação crítica segundo o ponto de vista do desempenho e da complexidade computacional.

Será também proposto para o algoritmo LSPA um novo método de normalização das mensagens enviadas dos CN's para os BN's. Os resultados obtidos revelam-se promissores comparativamente ao algoritmo SPA segundo Gallager [Gal1], [Gal2].

Começaremos, no entanto, por fazer referência às diversas medidas usadas para exprimir o desempenho de um código, bem como, a alguns aspectos importantes a ter em conta na comparação de códigos com diferentes taxas de informação.

5.1 ANÁLISE DE DESEMPENHO E COMPARAÇÃO DE CÓDIGOS

5.1.1 SISTEMAS DE COMUNICAÇÃO E DE ARMAZENAMENTO DIGITAIS

A codificação de canal assume um papel fundamental nos sistemas de comunicação actuais. De forma abstracta, um sistema deste tipo pode ser representado por um diagrama de bloco similar ao representado na figura 5-1:

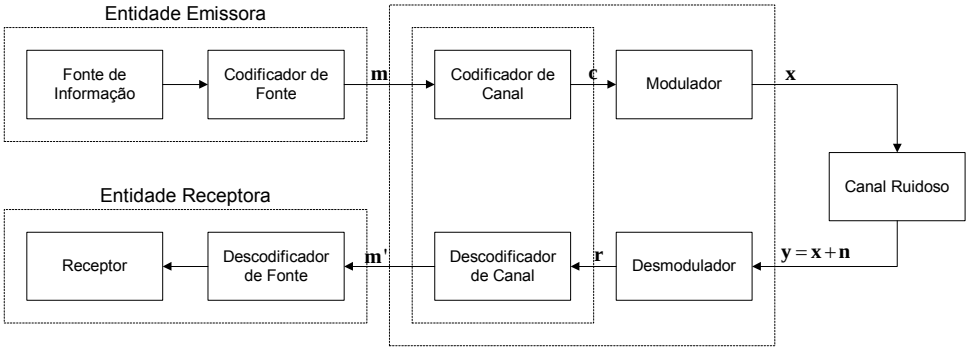


Figura 5-1 – Componentes básicos de um sistema de comunicação digital.

A forma como se relacionam os diversos blocos torna claro o método como é avaliado o desempenho de um código de correcção de erros e como podem ser comparados códigos com diferentes taxas de informação.

A fonte de informação discreta emite símbolos que são codificados pelo codificador de fonte, de acordo com um dado alfabeto código. O codificador de fonte procura remover toda a redundância estatística presente nos símbolos emitidos pela fonte reduzindo, desta forma, o débito simbólico r_s imposto ao sistema transmissor.

O codificador de canal, por sua vez, introduz redundância (de forma controlada) nas mensagens a transmitir por forma a aumentar a sua imunidade ao ruído do canal. Assim, considerando o caso de um código de bloco binário (n,k), às mensagens, $\mathbf{m}$, de comprimento k, este faz corresponder palavras de código, $\mathbf{c}$, de comprimento n, pelo que o débito simbólico à saída passa a ser, $r_c = r_s/R$, com $R = k/n$, a taxa de informação do código.

Os símbolos produzidos pelo codificador de canal são convertidos pelo modulador em sinais que possam ser transmitidos de forma eficiente através do canal de comunicação. A escolha do esquema de modulação é, normalmente, baseada num conjunto de restrições, como sejam, a potência máxima possível de transmitir, a largura de banda disponível, entre outros. Estes sinais são transmitidos através do canal sofrendo, regra geral, diversos efeitos, como sejam, distorção e atenuação.

O desmodulador opera sobre o sinal recebido, $\mathbf{y}$, realizando a operação inversa ao do modulador, fornecendo uma palavra, $\mathbf{r}$, ao descodificador de canal. No caso de um

sistema tipo FEC este tenta detectar e corrigir os erros da palavra recebida, fornecendo à sua saída uma estimativa, $\mathbf{m}'$, da mensagem transmitida[25].

5.1.2 MEDIDAS DE DESEMPENHO

São várias as medidas usadas para exprimir o desempenho de um sistema de transmissão digital, entre as quais se destacam a taxa de mensagens erradas (MER) e a taxa de erros (BER) à saída do descodificador de canal. Se a definição de MER é perfeitamente clara, considerando-se a existência de erro sempre que a mensagem descodificada $\mathbf{m}' \neq \mathbf{m}$ (ver figura 5-1), já a definição de BER não é muito clara [Wick], [Bos].

A dúvida surge quando procuramos saber se a taxa de erros em causa diz respeito aos bits da palavra de código descodificada $\mathbf{c}'$ ou, simplesmente, aos bits de mensagem $\mathbf{m}'$ que correspondem a essa palavra de código. Se muitos autores, aquando da apresentação de resultados relativos ao estudo de desempenho de um dado código, não clarificam devidamente a que BER se referem, é pratica comum considerar a taxa de erros entre os bits de informação (ou mensagem).

Por exemplo, a análise do desempenho de um código binário (n, k), não sistemático com um comprimento longo, descodificado com o recurso a uma tabela (correspondência entre palavras de código, $\mathbf{c}$, e mensagens, $\mathbf{m}$), é um estudo extremamente moroso. Em alternativa é usual considerar-se que se a palavra recebida contém x erros, então, o número médio de bits de mensagem errados é $x \times R$ e, nesse caso,

$$\mathrm{BER}_{\text{bits informação}} = \mathrm{BER}_{\text{bits código}}, \tag{5.1}$$

o que simplifica o cálculo do BER.

Se o BER e o MER representam duas medidas muito utilizadas, outras medidas existem, nomeadamente, a análise sobre o tipo de erros ocorridos no processo de descodificação.

---

[25] Um sistema que corrige e detecta os erros no receptor é designado por FEC (*forward error control*). Um sistema que detecta os erros no receptor e pede a retransmissão da informação errada ao emissor é designado por ARQ (*Automatic-Repeat-Request*). Este sistema requer a utilização de um canal de comunicação nos dois sentidos.

Nem todos os erros ocorridos durante a transmissão são detectados. Referimos no capítulo 2 que no caso de um código linear, usando descodificação por síndroma, só havia detecção de erros se o síndroma $\mathbf{s} = \mathbf{r} \times \mathbf{H}^T$ fosse diferente do vector nulo. Assim, num dado esquema de descodificação poderão ocorrer quatro tipos de situações:

- A palavra recebida, $\mathbf{r}$, é a palavra de código que foi transmitida ou que o descodificador de canal corrige correctamente e, logo, não há erros;
- A palavra recebida $\mathbf{r}$ é uma palavra de código diferente da que foi transmitida, pelo que este erro não é detectado na descodificação;
- A palavra recebida $\mathbf{r}$ não é uma palavra de código mas o descodificador de canal corrige-a erradamente.
- A palavra recebida $\mathbf{r}$ não é uma palavra de código e o descodificador de canal não a consegue corrigir. Nesta situação existe detecção dos erros ocorridos.

Assim, outras medidas importantes de desempenho são a taxa de mensagens erradas não detectadas e a taxa de erros não detectados.

5.1.3 COMPARAÇÃO ENTRE CÓDIGOS

Definidas que foram as medidas de desempenho, o problema que se coloca é de como comparar de forma justa o desempenho de códigos com diferentes dimensões e taxas de informação.

O uso de um código de canal com taxa de informação, $R = k/n$, tem duas consequências [Wick], [Bos]. Em primeiro lugar, o débito simbólico de transmissão aumenta passando a ser, $r_c = r_s / R$, com r_s e r_c o débito à entrada e saída do codificador de canal, respectivamente. Em segundo lugar, a energia de transmissão aumenta para nE_s em vez de kE_s, com E_s a energia transmitida por símbolo. De forma a comparar o desempenho de diferentes esquemas de codificação, a energia do sinal passa a ser expressa em termos da energia enviada por bit de informação, E_b, com

$$E_b = \frac{E_s}{R}. \tag{5.2}$$

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

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

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

5.1.4 GANHO DE CODIFICAÇÃO

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

Fixando a energia transmitida por bit E_b, a utilização de um código de taxa R provoca um aumento da taxa de transmissão, r_s, e uma diminuição da energia enviada por símbolo, E_s, ou seja, uma diminuição da SNR à saída do canal. Como consequência, verifica-se o aumento da taxa de erros da sequência recebida, em comparação com a que seria obtida se não fosse usado qualquer esquema de codificação. No entanto, a redundância introduzida com o código de canal, permite que após a descodificação muitos dos erros sejam corrigidos, garantindo um desempenho superior ao sistema sem codificação, como pode ser visualizado na figura 5-2.

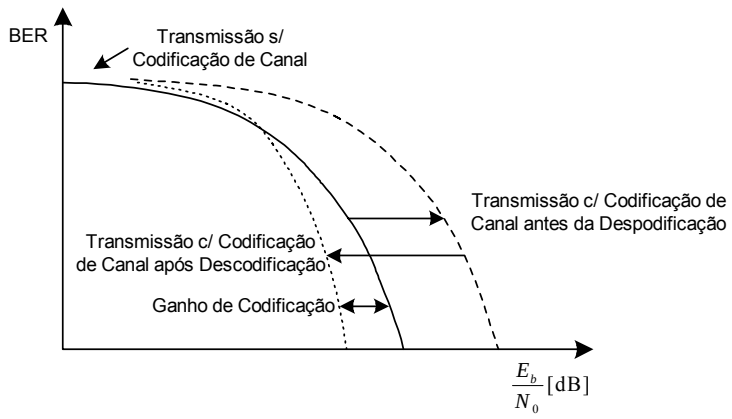


Figura 5-2 – Conceito de Ganho de Codificação.

5.2 RESULTADOS EXPERIMENTAIS

5.2.1 CÓDIGOS ANALISADOS E CONDIÇÕES DE SIMULAÇÃO

No estudo realizado foram utilizados três códigos LDPC de diferentes dimensões e taxas de informação. Escolhemos os seguintes códigos cujos resultados são reportados na literatura:

- LDPC 96.2A3.565 (designado no texto por código α). Código irregular obtido pelo método construção 2A, com comprimento de bloco, $n = 96$, taxa de informação, $R = 1/3$, e caracterizado por um giro máximo igual a 8. A distribuição dos graus de cada BN e CN, e do giro dos BN's encontra-se representada na figura 5-3. A matriz de teste de paridade, bem como, uma análise do seu desempenho está disponível em [Mac3].

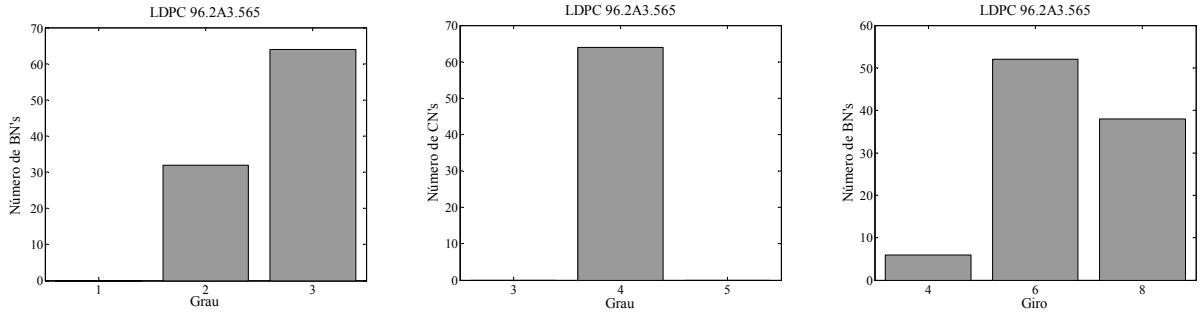


Figura 5-3 – Distribuição dos graus dos BN's e CN's e dos giros dos BN's para o código α.

- LDPC 252.252.3.252 (designado no texto por código β). Código regular com comprimento de bloco, $n = 504$, e taxa de informação, $R = 1/2$, possuindo cada BN grau 3 (ver figura 5-4). Este código encontra-se também disponível em [Mac3] e o seu desempenho foi reportado em [CF1] e [CF2].

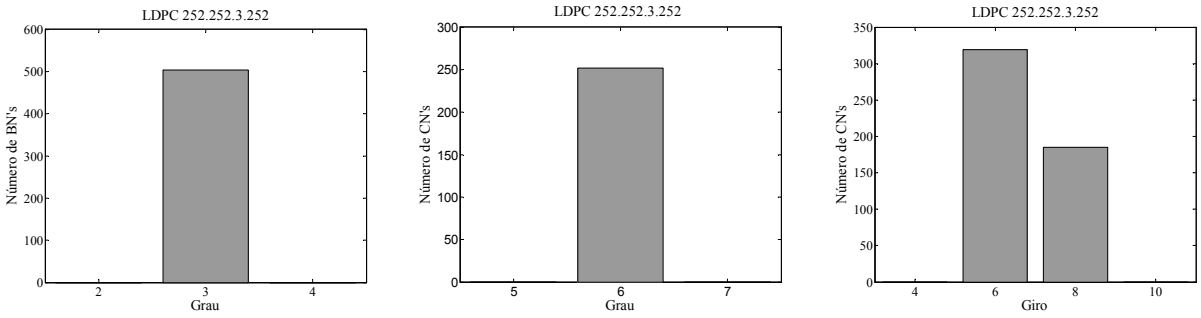


Figura 5-4 – Distribuição dos graus dos BN's e CN's e dos giros dos BN's para o código β.

- LDPC 1268.456.2A.1 (designado no texto por código γ). Código irregular de dimensões $(1268, 456)$, obtido segundo o método de construção 2A. Foi projectado para

o transporte de células ATM num projecto industrial. O seu desempenho encontra-se reportado em [MB1] e [MB2] e foi fornecido pelos autores.

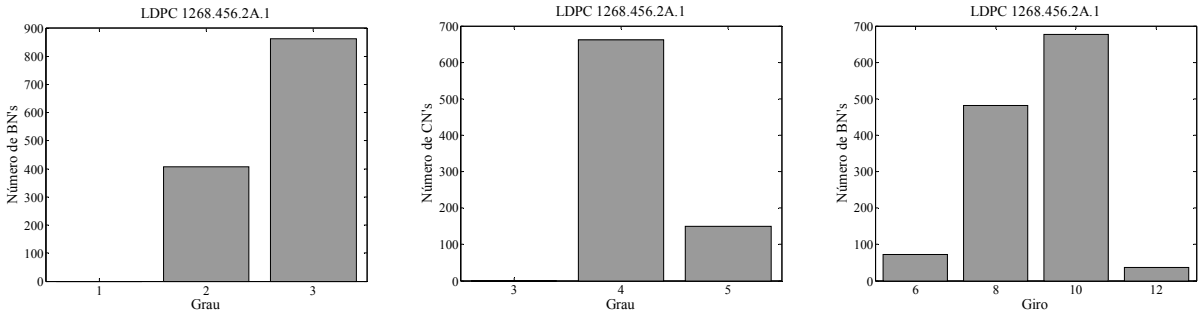


Figura 5-5 – Distribuição dos graus dos BN's e CN's e dos giros dos BN's para o código γ.

A opção pelo código γ deveu-se ao facto de ser um código mais longo, com uma distribuição de giros menos uniforme e indicado para testar o método de gestão probabilística, proposto por Mao e Banihashemi [MB1] para calcular as mensagens enviadas dos BN's para os CN's no algoritmo SPA. Pretendíamos, desta forma, confirmar os bons resultados apresentados pelos autores.

Escolhemos o código β devido ao facto de este se encontrar reportado em vários artigos de referência na área [CF1], [CF2]. Pretendíamos desta forma validar o nosso trabalho ao reproduzir os resultados apresentados em [CF1], [CF2].

Por último, pretendíamos averiguar o desempenho dos vários algoritmos estudados para códigos mais curtos. Esta foi a razão pela qual escolhemos o código α.

Condições de Simulação

Todo o código desenvolvido foi escrito em C. À semelhança de Mao e Banihashemi [MB1] utilizámos os geradores de erros do *Numerical Recipes in C* [PTVF] para produzir ruído AWGN. Neste caso, e sem perda de generalidade, optámos por testar apenas palavras de código nulas, em virtude de os códigos LDPC serem lineares e os algoritmos de descodificação estudados tomarem em conta a simetria do canal AWGN no cálculo das suas probabilidades de inicialização. Desta forma, evitámos a geração aleatória de mensagens, a sua codificação e posterior extracção das palavras descodificadas, o que tornaria qualquer simulação realizada extremamente morosa.

Todos os algoritmos estudados obedeceram exactamente às mesmas condições de simulação (para cada código em particular), ou seja, o mesmo número de palavras testadas, corrompidas pela mesma sequência de erros e, ainda, o mesmo número máximo de iterações admissíveis.

Assim, o número de palavras testadas para cada um dos códigos referidos foi escolhido de forma a garantir, após descodificação, um número de palavras erradas superior a 100, para toda a gama de valores de SNR (por forma a obter valores de BER entre 10^{-1} e 10^{-6}). Já a escolha do número máximo de iterações resultou de um compromisso entre o valor a partir do qual não havia melhoria aparente do desempenho do algoritmo SPA (para o menor valor de SNR estudado) e um número que não tornasse o processo de descodificação demasiado moroso.

As condições de simulação encontram-se apresentadas na tabela 5-1.

	Número de Palavras de Código Testadas	Número Máximo de Iterações	Gama de Valores de E_b/N_0 Analisada
Código α	10^6	100	1 dB - 5 dB
Código β	10^6	200	0.5 dB - 3.5 dB
Código γ	10^7	100	1 dB - 3 dB

Tabela 5-1 – Condições de simulação para os códigos α, β e γ.

5.2.2 ALGORITMOS SPA, MS-LSPA E MS-LSPA NORMALIZADO

Entre as várias simplificações ao algoritmo SPA, abordadas na secção 4.4.4, optámos por estudar apenas os algoritmos MS-LSPA e MS-LSPA normalizado segundo o método proposto por Chen e Fossorier [CF1], [CF2]. Tal decisão, ficou a dever-se ao facto de os resultados reportados em [CF1], [CF2], mostrarem que os algoritmos APP-LSPA e UMP-LSPA apresentam uma diminuição muito acentuada do desempenho (uma taxa de erros quase duas ordens de grandeza superior para valores de SNR mais elevados) relativamente ao SPA, o que os torna relativamente pouco atractivos como algoritmos eficientes para a descodificação de códigos LDPC.

SPA versus LSPA

Os resultados de simulação que apresentamos para o algoritmo SPA, bem como, as comparações feitas com os restantes algoritmos do ponto de vista da latência, dizem respeito à implementação do mesmo na forma original proposta por Gallager e não no domínio logarítmico (algoritmo LSPA). Tal escolha ficou a dever-se, em primeiro lugar, ao facto de a latência do algoritmo SPA ser muito inferior (cerca de 10 vezes) à do algoritmo LSPA (mesmo usando o algoritmo frente e verso no cálculo das mensagens

$L(r_{mn})$), ao contrário do que seria de esperar. Uma análise do algoritmo LSPA revelou que a grande maioria do tempo de processamento era gasto a efectuar a operação $\boxplus$ (ver equação (4.4)), devido às funções logaritmo e exponencial presentes no cálculo de $g(x)$ (ver equação (4.5)), ao contrário do que sucedia para o SPA, em que cálculo das mensagens $r_{mn}(x)$, baseado em soma de produtos, era realizado de forma bastante eficiente.

O segundo motivo para optarmos pelo algoritmo SPA em detrimento do LSPA deveu-se ao desempenho deste último se revelar inferior ao do SPA, como pode ser observado na figura 5-6.

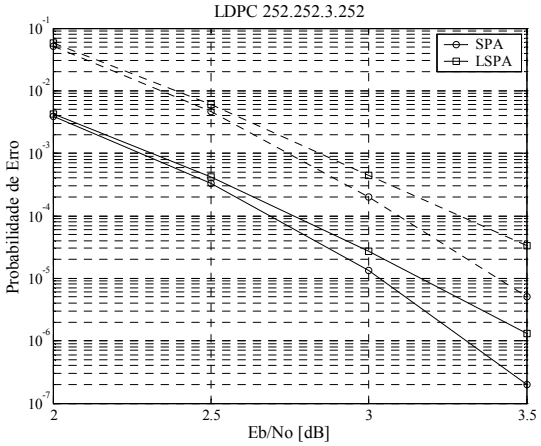


Figura 5-6 – Desempenho expresso em termos de BER (–) e MER (···), dos algoritmos SPA e LSPA para o código β.

Apesar de matematicamente equivalentes, pudemos verificar no caso do código β, que para valores de E_b/N_0 superiores a 2 dB, o algoritmo LSPA apresentava uma diminuição progressiva do desempenho face ao algoritmo SPA. Tal é motivado por erros de precisão aritmética. De facto, ambos os algoritmos foram implementados em C usando reais, com precisão simples, para guardar as mensagens trocadas entre nodos do TG. Tal decisão revelou-se acertada para o algoritmo SPA, na medida em que as mensagens são valores de probabilidades (logo na gama $[0;1]$). Já para o algoritmo LSPA, devido ao facto de as mensagens serem máximas verosimilhanças logarítmicas (valores entre $]-\infty;+\infty[$) e de para SNR elevados a incerteza relativa aos símbolos recebidos ser muito menor (o que dá origem a valores absolutos das mensagens enviadas entre nodos bastante elevados), podem ocorrer erros de ultrapassagem de gama

dinâmica no cálculo das mensagens $L(r_{mn})$ e $L(q_{nm})$ (ver (4.6) e (4.7)), com a consequente diminuição do desempenho.

Factores de normalização do algoritmo MS-LSPA

O factor de normalização, θ, para o algoritmo MS-LSPA, foi determinado experimentalmente com base no método proposto por Chen e Fossorier [CF1], [CF2]. Assim, de acordo com o procedimento descrito na secção 4.4.4 foram obtidos os gráficos da figura 5-7.

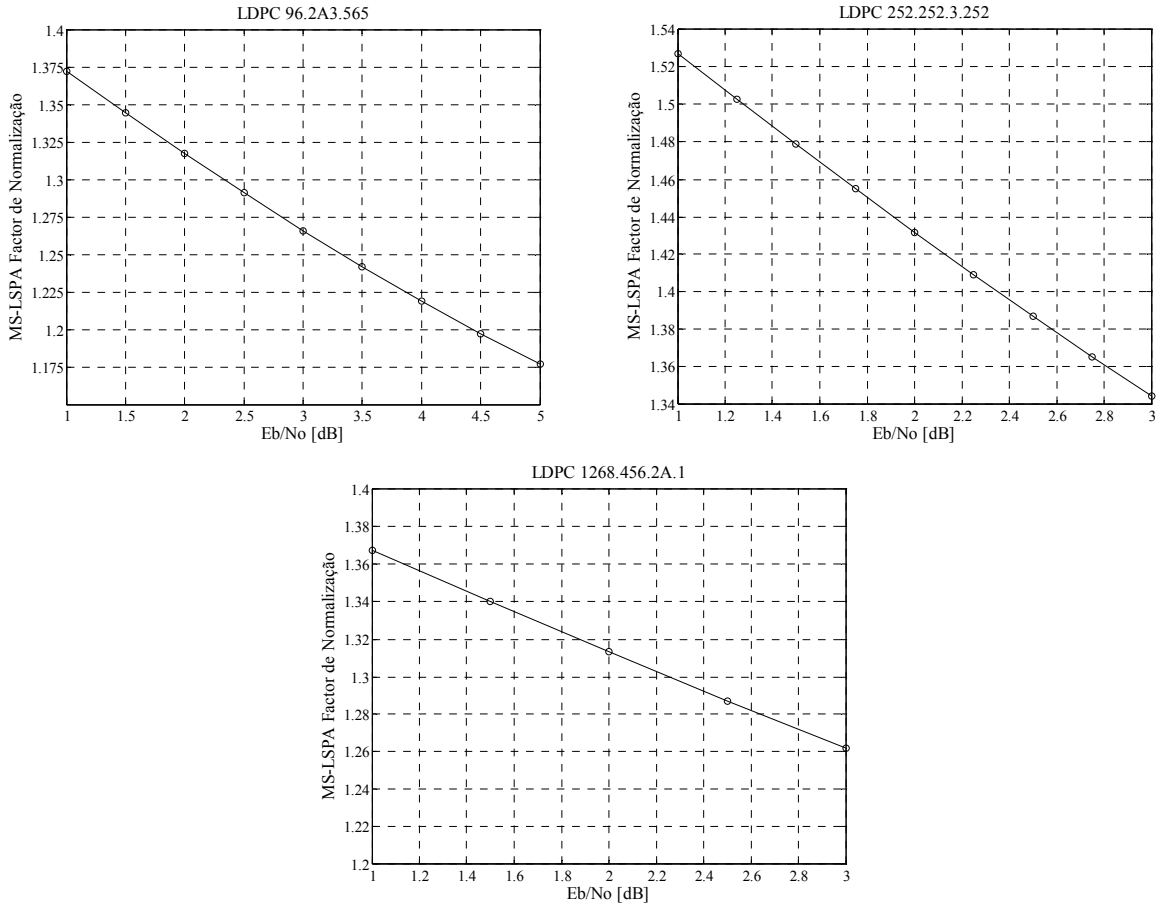


Figura 5-7 – Factores de normalização para o algoritmo MS-LSPA, obtidos experimentalmente para os códigos α, β e γ.

Para cada um dos códigos, o valor de θ escolhido correspondeu à SNR do canal para o qual o desempenho do algoritmo SPA expresso em termos de BER, pertencia ao intervalo $\left[10^{-4},10^{-3}\right]$. Os valores escolhidos foram pois:

	Factor de Normalização (θ)
Código α	1.22
Código β	1.4
Código γ	1.3

Tabela 5-2 – Factores de normalização escolhidos para o algoritmo MS-LSPA.

Desempenho

Na figura 5-8 apresenta-se o desempenho dos algoritmos SPA, MS-LSPA e MS-LSPA Normalizado (com os factores de normalização da tabela 5-2), expresso em termos de BER (linha contínua) e de MER (tracejado). Apresenta-se, ainda, para o caso do código α a percentagem de bit errados (linha contínua) e de mensagens erradas (tracejado) não detectadas.

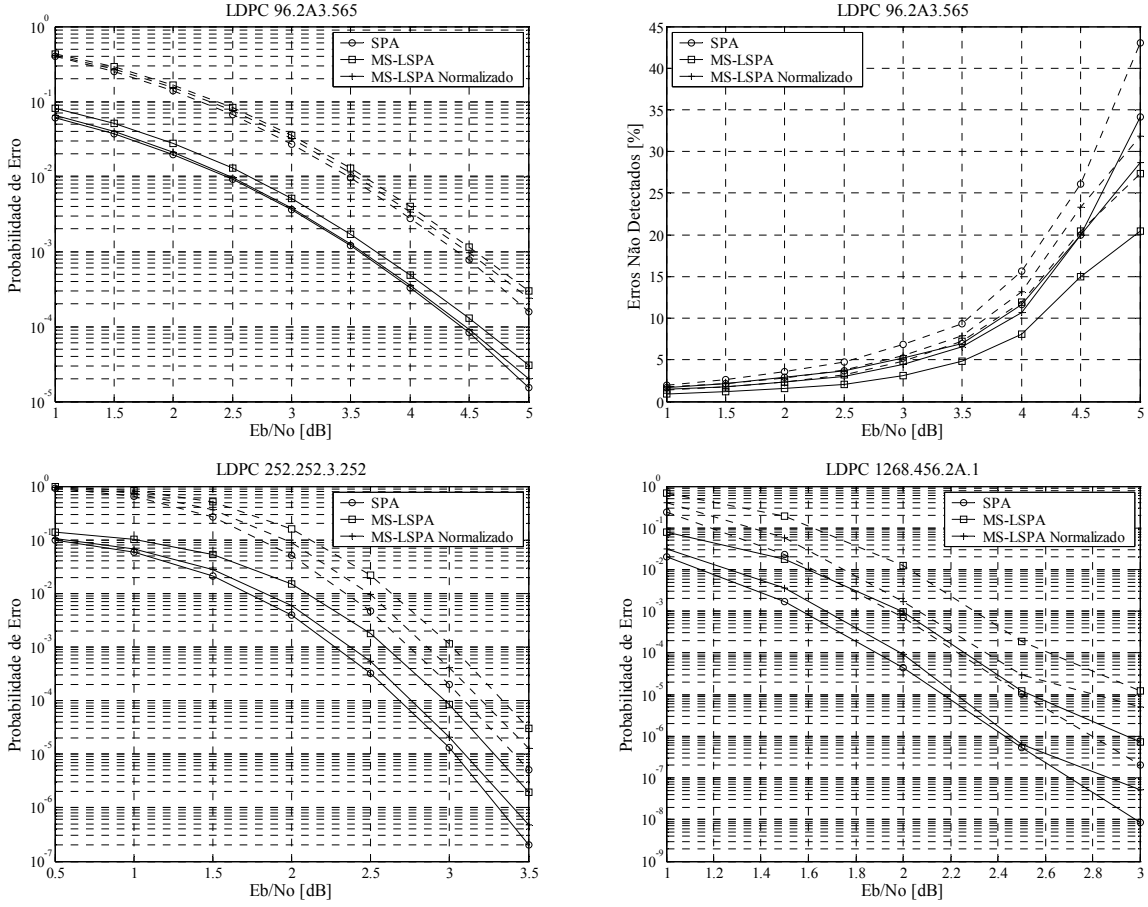


Figura 5-8 – Desempenho expresso em termos de BER (–) e MER (···), dos algoritmos SPA, MS-LSPA e MS-LSPA normalizado para os códigos α, β e γ, e em termos de percentagem de bits errados (–) e de mensagens erradas (···) não detectadas para o código α.

Os resultados obtidos para o código β coincidiram com os apresentados por Chen e Fossorier em [CF1] e [CF2] o que nos permitiu dissipar qualquer dúvida sobre a correcção da implementação dos algoritmos.

MS-LSPA

A latência do algoritmo MS-LSPA revelou-se inferior à do SPA, tal como era de esperar. No entanto, observámos também, uma diminuição significativa do desempenho, para todos os códigos testados. Tomando por referência um BER de 10^{-4} essa diminuição é de cerca de 0.125 dB para o código α, e de 0.3 dB e 0.35 dB para os códigos β e γ, respectivamente. No caso dos dois últimos, essa diminuição de desempenho corresponde para o mesmo SNR a um BER cerca de 10 vezes superior.

A diminuição do desempenho tem como origem a sobrestimação das mensagens, $L(r_{mn})$ (calculadas segundo (4.106)), enviadas dos CN's para os BN's. Não é pois de estranhar que esta diminuição seja função da distribuição dos graus dos CN's. Dado um CN_m de grau k, o erro de estimação cometido no cálculo das mensagens por ele enviadas, é tanto maior quanto maior for k, na medida em que são desprezadas um maior número de contribuições provenientes dos BN's a ele ligados. Esta é uma das razões pelas quais podemos observar uma diminuição de desempenho para o código β idêntica à que é observada para γ.

A análise dos resultados obtidos, complementados pelos reportados na literatura [CF1], [CF2], [HM], [HEAD], permitiram-nos concluir que a diminuição de desempenho é, também, função do número de CN's do código. Códigos com maior número de CN's apresentam uma maior degradação do desempenho, na medida em que, a acumulação dos erros torna-se mais significativa. Esta é a razão pela qual a diminuição do desempenho foi maior para o código γ.

MS-LSPA Normalizado

Com o algoritmo MS-LSPA normalizado observa-se uma melhoria significativa do desempenho relativamente ao MS-LSPA, com ganhos de codificação aproximados de 0.15 dB, 0.275 dB e 0.25 dB para os códigos α, β e γ respectivamente. A realização de apenas mais duas multiplicações por CN em cada iteração não representou um aumento significativo da latência do algoritmo, tendo-se revelado bastante compensatória do ponto de vista de melhoria do desempenho. Em qualquer dos casos, o algoritmo MS-LSPA normalizado apresenta, ainda assim, um desempenho inferior (menos de

0.1 dB) ao algoritmo SPA, sendo que essa diminuição é, à semelhança do MS-LSPA, mais significativa para códigos com maior número de CN's.

Taxa de erros não detectados

A taxa de erros não detectados, é desprezável, mesmo nula, para os códigos β e γ, confirmando os resultados reportados em [LS], [Lech]. Os erros não detectáveis são praticamente desprezáveis para códigos com comprimento de bloco mais longos (superior a 1000), uma vez que estes tendem a possuir uma distância mínima mais elevada. No entanto, para o código α podemos verificar na figura 5-8, que os erros não detectados constituem uma grande percentagem dos erros ocorridos, para valores de SNR mais elevados. Como é baixa a distância mínima do código, facilmente o algoritmo converge para uma palavra de código diferente da que foi transmitida. Esta situação é bastante grave do ponto de vista da descodificação conforme foi referido em 4.3.3. Os algoritmos MS-LSPA e MS-LSPA normalizado apresentam um pior desempenho em relação ao algoritmo SPA, com um aumento de cerca de 10% dos erros não detectados.

Número de Iterações

O número médio de iterações dos algoritmos SPA, MS-LSPA e MS-LSPA normalizado, bem como, o seu desvio padrão encontra-se representada na figura 5-9.

Uma análise do número médio de iterações realizado pelo algoritmo SPA, permite concluir que são adequados os valores estipulados para o número máximo de iterações (ver tabela 5-1) para cada um dos códigos. Apenas o código β e para valores de E_b/N_0 inferiores a 1.5 dB, apresenta um valor muito próximo do máximo. Eventualmente, considerando um limite mais elevado, poderíamos ter uma melhoria de desempenho do algoritmo para esses valores de SNR.

Para todos os algoritmos podemos observar que o número médio de iterações e o seu desvio padrão são da mesma ordem de grandeza e que ambos decrescem consideravelmente com o aumento do SNR do canal.

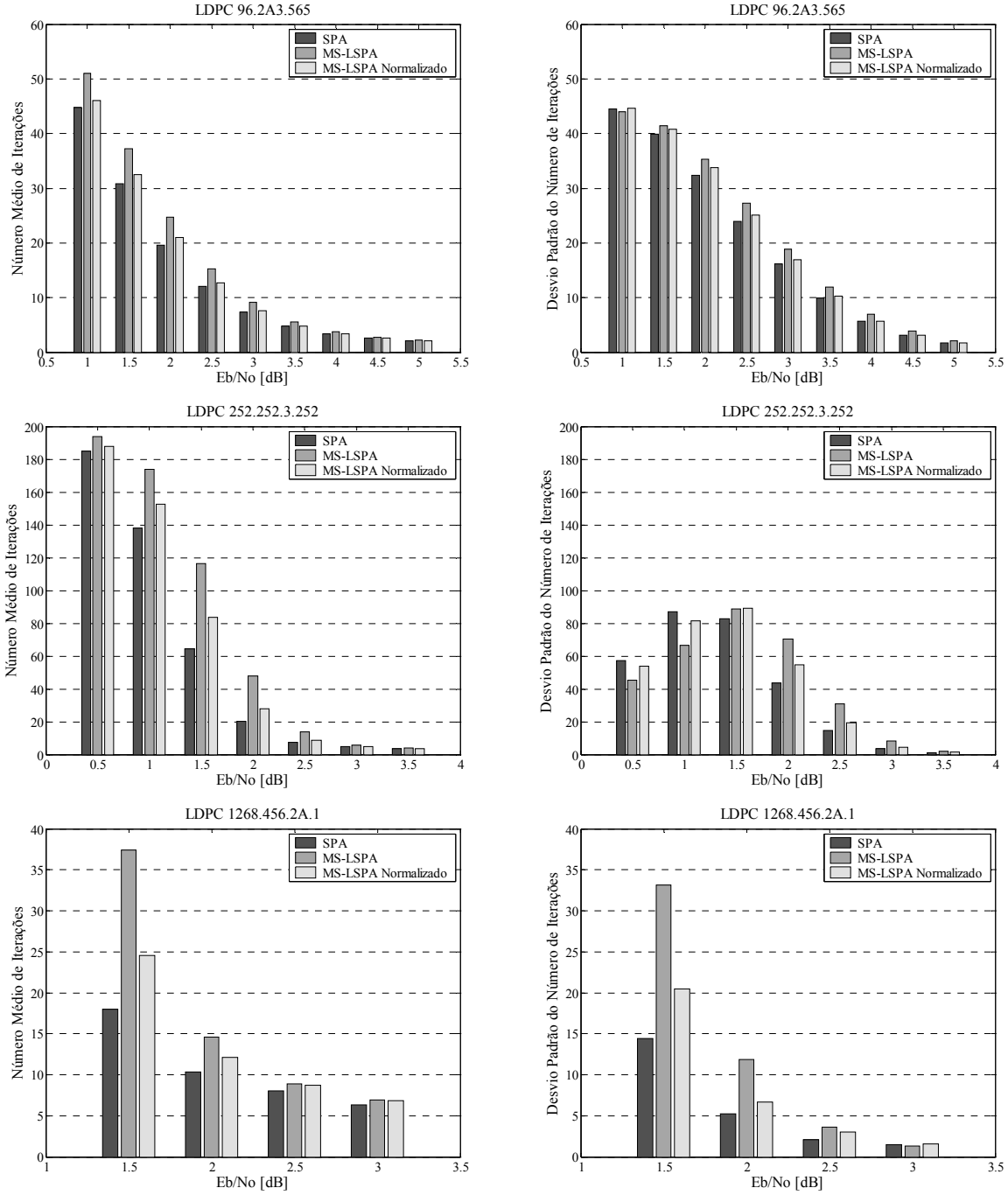


Figura 5-9 – Média e desvio padrão do número de iterações dos algoritmos SPA, MS-LSPA e MS-LSPA normalizado para os códigos α, β e γ.

A comparação dos resultados entre os algoritmos permite concluir que em todas as situações, o algoritmo MS-LSPA apresenta um número médio de iterações para descodificar uma palavra superior ao do algoritmo SPA. O motivo é o mesmo pelo qual o algoritmo MS-LSPA apresenta pior desempenho que o SPA, ou seja, má estimação das mensagens $L(r_{mn})$ enviadas dos CN's para os BN's, o que leva a que seja necessário

realizar um maior número de iterações para o algoritmo convergir. Este aumento é, como seria de esperar, mais notório para códigos com maior número de CN's. Podemos também constatar que para o código γ, esse aumento é sensivelmente o dobro para SNR mais baixos. Podemos então concluir que outro factor importante é a regularidade do código. De facto, sendo este código irregular, no algoritmo SPA os BN's de maior grau convergem mais rapidamente [LS] (devido ao maior número de contribuições recebidas). No algoritmo MS-LSPA, os erros de estimação cometidos no cálculo das mensagens $L(r_{mn})$ recebidas por cada BN, fazem com que a convergência dos BN's de maior grau seja muito mais lenta com o consequente aumento do número médio de iterações.

O aumento do número médio de iterações é compensado, em parte, pela redução do tempo de cada iteração devido ao muito menor número de operações aritméticas realizadas, o que faz com que este apresente, normalmente, uma latência inferior ao SPA (função do próprio código).

A normalização do MS-LSPA reduz consideravelmente o número médio de iterações, ainda assim superior ao algoritmo SPA (ver figura 5-9). A latência deste algoritmo é ligeiramente superior ao do MS-LSPA, na medida em que é necessário realizar mais duas multiplicações por cada CN (em parte compensado pelo menor número de iterações necessárias para descodificar uma palavra). Isto faz do algoritmo MS-LSPA normalizado uma boa alternativa ao SPA do ponto de vista do desempenho versus complexidade.

5.2.3 GESTÃO PROBABILÍSTICA

No estudo realizado procurámos analisar o método de melhoria de desempenho do algoritmo SPA, proposto por Mao e Banihashemi em [MB1] baseado na gestão das mensagens enviadas dos BN's para os CN's conforme apresentado na secção 4.5. Procurávamos, para tal, reproduzir os bons resultados apresentados em [MB1] e [YHB] para o código γ.

Independentemente do algoritmo de descodificação utilizado, o objectivo do método é minimizar as realimentações positivas existentes devido aos BN's possuírem diferentes giros. Era nossa convicção que o método deveria apresentar melhorias de desempenho, semelhantes às que eram reportadas para o algoritmo SPA, quando aplicado a outros algoritmos, nomeadamente, o MS-LSPA.

SPA

O desempenho do algoritmo SPA na sua forma original, com gestão probabilística (SPA-GP) e com gestão não probabilística (SPA-GNP) encontra-se representada na figura 5-10 para os códigos estudados.

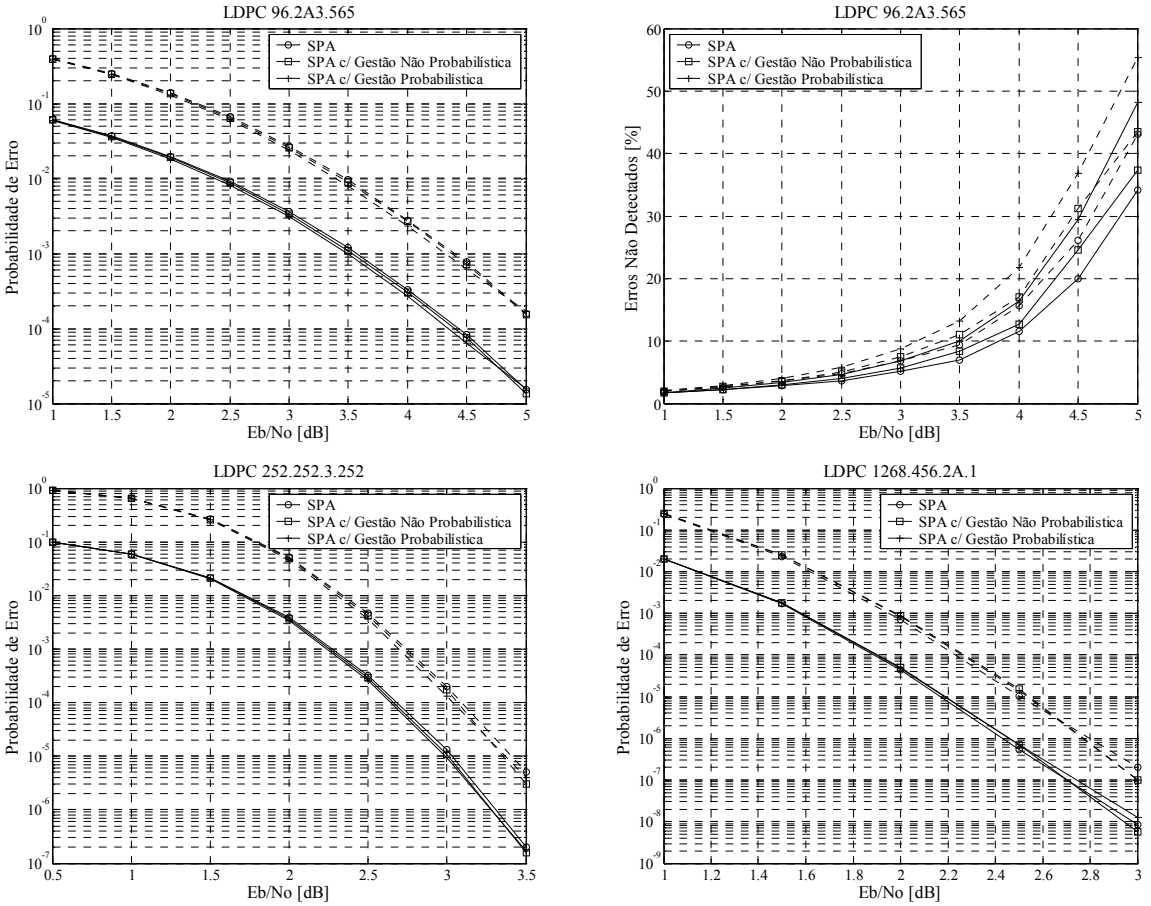


Figura 5-10 – Desempenho expresso em termos de BER (–) e MER (···), dos algoritmos SPA, SPA com gestão probabilística e gestão não probabilística para os códigos α, β e γ, e em termos de percentagem de bits errados (–) e de mensagens erradas (···) não detectadas para o código α.

Uma observação dos gráficos permite concluir que o algoritmo SPA-GP apresenta, regra geral, melhor desempenho do que o SPA-GNP. No entanto, os resultados obtidos são no mínimo intrigantes, quando comparamos o desempenho destes dois algoritmos com o SPA. Para o código α, tendo por referência um BER de 10^{-4}, observa-se, para a melhor das abordagens (algoritmo SPA-GP), um ganho de codificação de 0.09 dB. Já para o código β, tendo por referência um BER ainda mais baixo de 10^{-5}, esse ganho é de apenas 0.06 dB. Para o código γ há mesmo uma diminuição do desempenho do

algoritmo SPA-GP relativamente ao SPA, ao contrário dos 2 dB de ganho reportados por Mao e Banihashemi em [MB1] para um BER de 10^{-6}.

No sentido de esclarecer tal diferença contactámos um dos autores, por forma a averiguar se o problema se devia a uma má interpretação do método [MB1], ou ao gerador de ruído AWGN. Dos contactos estabelecidos chegámos à conclusão que a interpretação do algoritmo está correcta. A realização de novos testes, utilizando o gerador de ruído que nos foi indicado, confirmou os resultados já antes obtidos. Estes factos permitem-nos afirmar que os resultados reportados em [MB1] e [YHB] não são fiáveis.

Uma explicação possível é o facto de cada BN do TG do código, estar incluído em ciclos de diferentes comprimentos. Num BN_n de giro g_n, o congelamento das mensagens por ele enviadas, desde a iteração $g_n/2$ até iteração $g_{\max}/2$, impede que as contribuições recebidas dos BN's mais distantes (incluídos em ciclos mais longos que passam pelo BN_n) não sejam por ele propagadas aos CN's a ele ligados, o que poderia, de certa forma, contribuir para a convergência do algoritmo. Esta é também uma das razões apontadas para o facto de o método de gestão probabilística apresentar um melhor desempenho do que a gestão não probabilística.

Quando comparados do ponto de vista do número de iterações realizadas os algoritmos SPA-GP e SPA-GNP são praticamente equivalentes ao SPA (ver figura 5-11), com a vantagem de serem realizados um menor número de operações por iteração. A latência do algoritmo SPA-GP é, no entanto, muito superior ao SPA e SPA-GNP, na medida em que é necessário gerar por iteração e por cada BN um valor aleatório com o objectivo de decidir se esse BN retém ou não as mensagens a enviar.

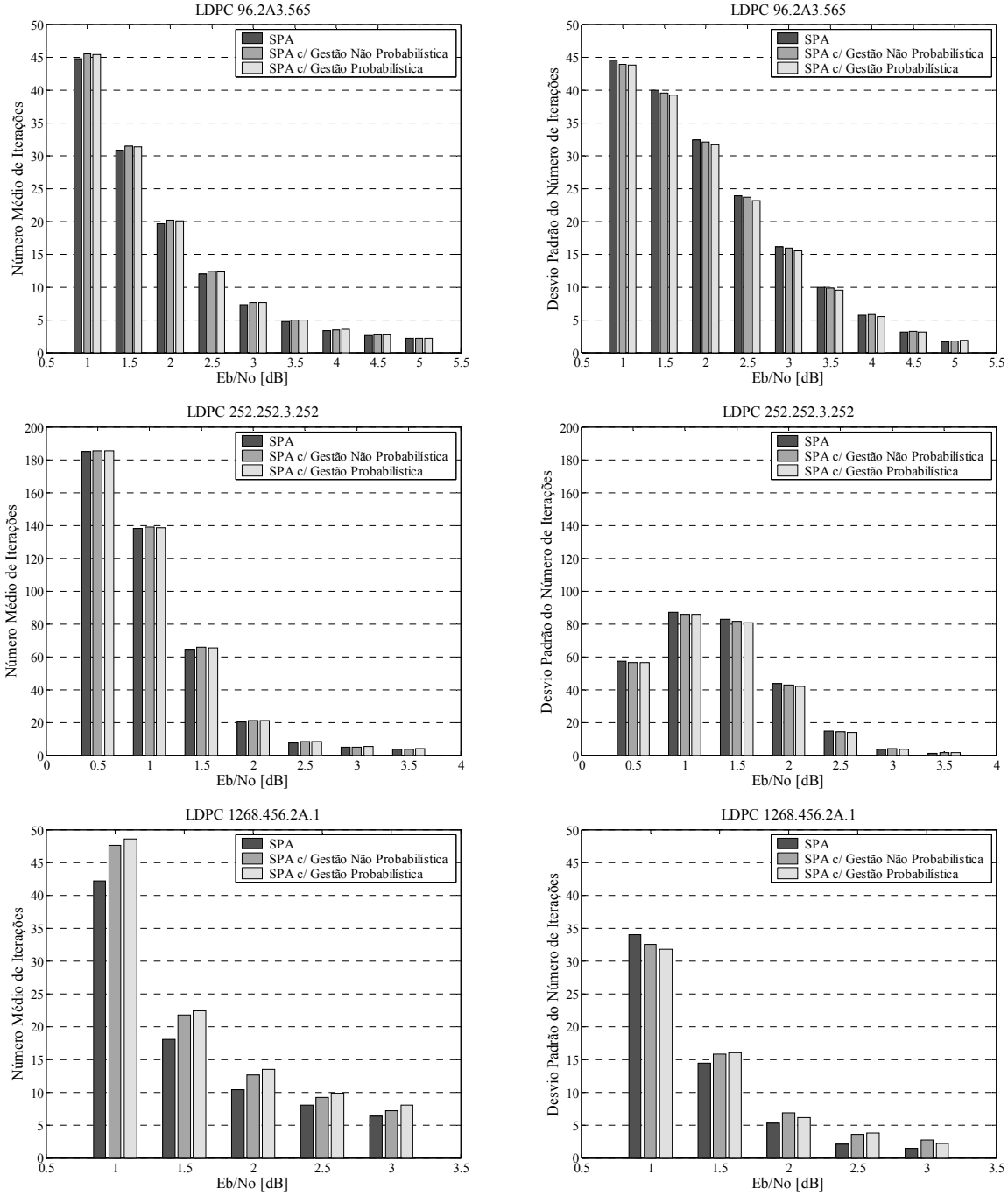


Figura 5-11 – Média e desvio padrão do número de iterações dos algoritmos SPA, SPA com gestão probabilística e não probabilística para os códigos α, β e γ.

MS-LSPA

A aplicação ao algoritmo MS-LSPA, do método de gestão das mensagens enviadas dos BN´s para os CN's, produziu resultados idênticos aos obtidos para o algoritmo SPA, conforme se pode observar nas figuras 5-12 e 5-13. Tais resultados, vêm reforçar a ideia de que o método proposto não é eficiente.

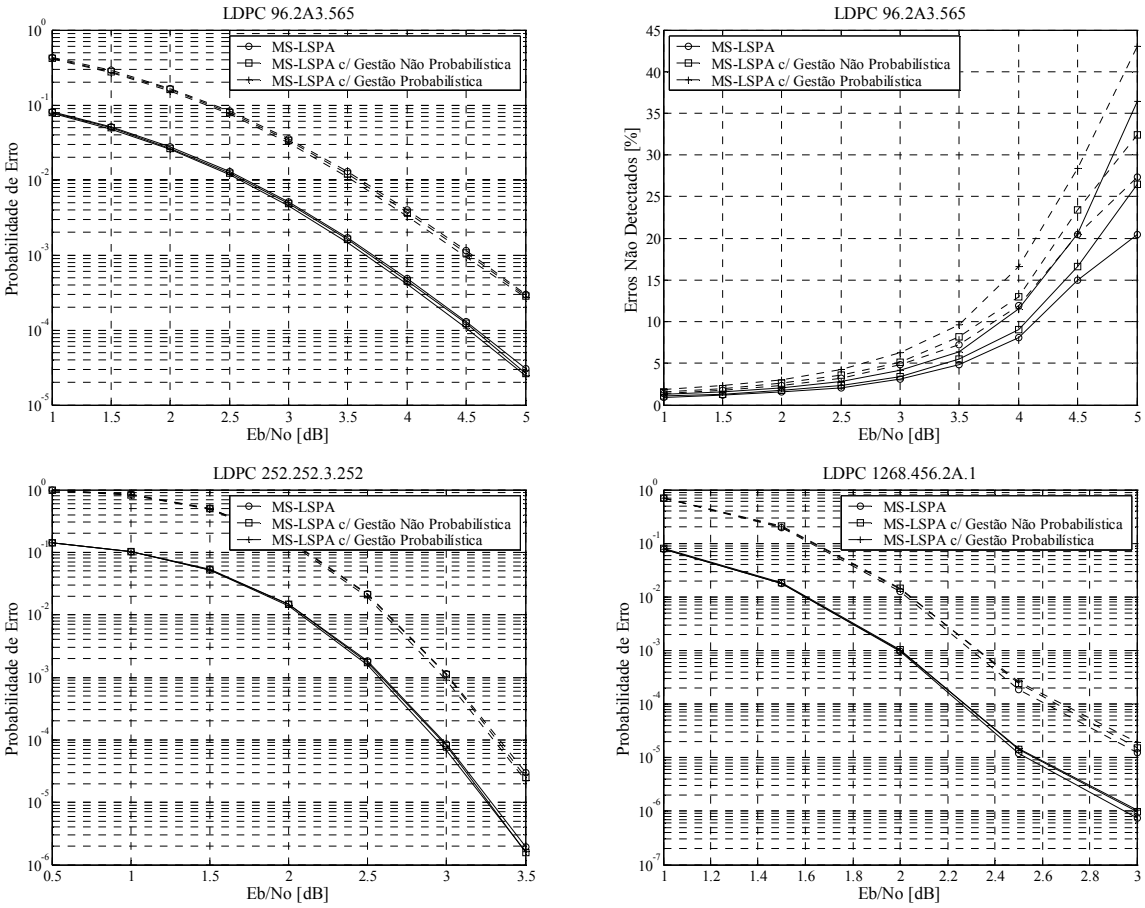


Figura 5-12 – Desempenho expresso em termos de BER (–) e MER (···), dos algoritmos MS-LSPA, MS-LSPA com gestão probabilística e gestão não probabilística para os códigos α, β e γ, e em termos de percentagem de bits errados (–) e de mensagens erradas (···) não detectadas para o código α.

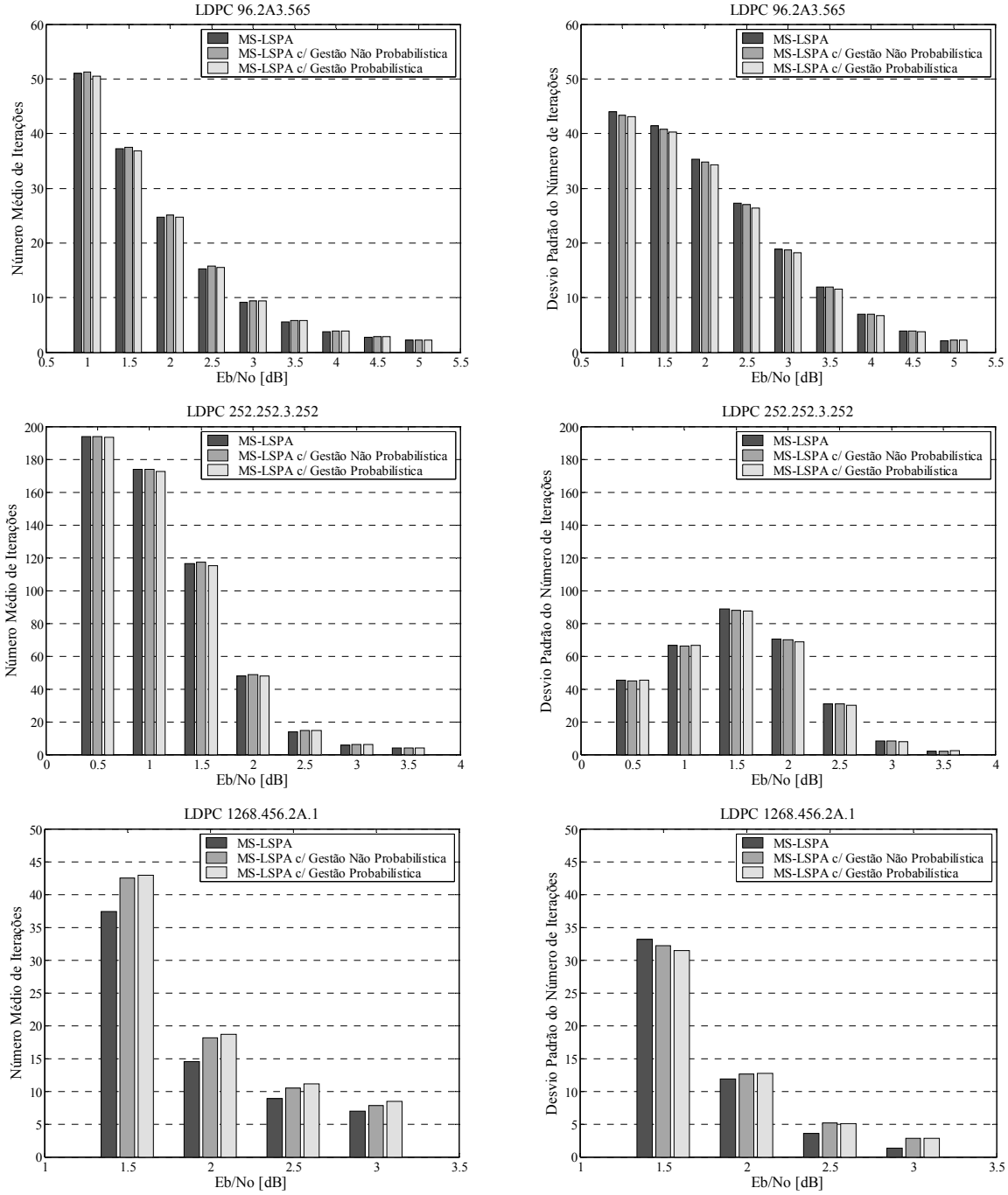


Figura 5-13 – Média e desvio padrão do número de iterações dos algoritmos MS-LSPA, MS-LSPA com gestão probabilística e não probabilística para os códigos α, β e γ.

5.3 ALGORITMO LSPA NORMALIZADO

Mencionámos já por várias vezes o facto de que os ciclos do TG de um código introduzem uma realimentação positiva nas mensagens enviadas entre nodos, pelo que o desempenho do algoritmo SPA não é óptimo.

Algumas modificações do algoritmo SPA e LSPA têm sido propostas no sentido de melhorar o seu desempenho. Explorámos, por exemplo, na secção anterior um método baseado na gestão do envio das mensagens dos BN's para os CN's [MB1], sem resultados positivos. Fossorier propôs também um método que combina o algoritmo SPA com uma descodificação estatística ordenada [Foss]. Os resultados reportados revelam uma melhoria significativa relativamente ao desempenho do SPA conseguida, no entanto, à custa de um aumento considerável da complexidade computacional do algoritmo de descodificação.

5.3.1 NORMALIZAÇÃO

O trabalho desenvolvido por He, Sun e Wang [HSW] despertou o nosso interesse. Os autores propuseram a implementação do algoritmo LSPA usando quantificação segundo a lei-μ para representar as mensagens enviadas entre nodos. Os resultados reportados revelam que com apenas 8 bits de quantificação existe uma melhoria do desempenho do algoritmo. A perda de informação resultante do processo de quantificação não uniforme (considerando um dado número mínimo de bits), ao invés de provocar uma degradação do desempenho do algoritmo, contribui para uma melhoria do mesmo. Desta forma, os erros de má estimação das mensagens (devido à realimentação) enviadas entre nodos são atenuados.

Estes resultados alertaram-nos para o facto de os erros de estimação, serem, regra geral, por excesso em virtude de a realimentação ser positiva, e que provoca um aumento incorrecto da máxima verosimilhança logarítmica (logo diminuição da incerteza – ver figura 5-14) das mensagens enviadas.

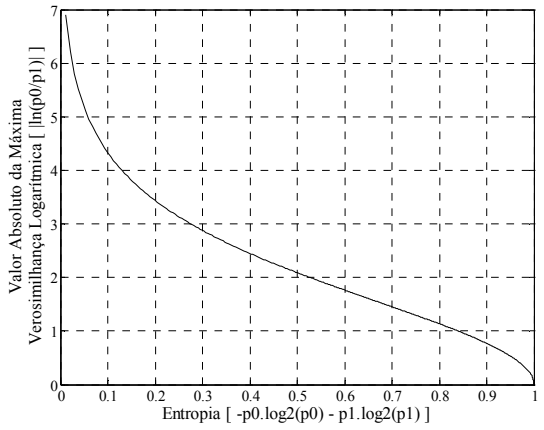


Figura 5-14 – Máxima verosimilhança logarítmica como função monótona decrescente da entropia.

Tal conhecimento levou-nos a propor para o algoritmo LSPA, a aplicação de um factor de normalização às mensagens, $L(r_{mn})$, enviadas dos CN's para os BN's, por forma a compensar a sobrestimação das mesmas. Assim, as mensagens, $L(r_{mn})$ (calculadas segundo um dos métodos indicados nas secções 4.4.3 e 4.4.4) são multiplicadas por uma constante positiva $\theta < 1$, ou seja,

$$L'(r_{mn}) = \theta \times L(r_{mn}). \tag{5.8}$$

O método proposto é similar ao descrito por Chen e Fossorier [CF1], [CF2] para a normalização do algoritmo MS-LSPA, que produz uma melhoria significativa do desempenho do algoritmo.

Esta proposta vai no encontro do recente trabalho apresentado por Yazdani, Hemati e Banihashemi [YHB] que também propõem a normalização do algoritmo LSPA, mas das mensagens, $L(q_{nm})$, enviadas dos BN's para os CN's.

A escolha recaiu nas mensagens enviadas dos CN's para os BN's, na medida em que o grau dos CN's é superior ao dos BN's e, como tal, o erro de sobrestimação aí cometido é (na nossa opinião) mais gravoso. Outra razão reside no facto do factor de normalização quando aplicado às mensagens $L(q_{nm})$ (conforme propõem Yazdani, Hemati e Banihashemi) afectar o valor da probabilidade à posteriori $L(c_n)$ de inicialização do algoritmo que intervém no seu cálculo (ver equação 4.71).

O método proposto pode também ser aplicado ao algoritmo SPA onde o factor de correcção, θ, corresponde a um potência. As mensagens normalizadas enviadas dos CN's para os BN's são dadas por,

$$r'_{mn}(x) = \left[r_{mn}(x) \right]^{\theta}, \tag{5.9}$$

com $x \in \{0,1\}$. Esta forma de implementação do método proposto resulta, no entanto, num aumento significativo da complexidade computacional do algoritmo SPA, ao contrário do que acontece para o LSPA. As duas abordagens são matematicamente equivalentes.

5.3.2 DESEMPENHO DO ALGORITMO SPA NORMALIZADO

Referimos na secção 5.2.2 que a implementação do algoritmo LSPA apresentava um desempenho inferior ao SPA devido a erros de precisão aritmética. Por forma a evitarmos esses erros e avaliarmos com alguma justiça o método proposto, optámos pela sua implementação segundo (5.9), apesar do consequente aumento da complexidade computacional.

O factor de normalização para cada código foi determinado experimentalmente, fazendo uma pesquisa exaustiva. Verificámos que o factor a considerar é função do código. Foi escolhido o valor que para a gama de valores de SNR estudados produz uma melhoria mais significativa do desempenho. Na figura 5-15 apresentamos a sua variação para os códigos β e γ.

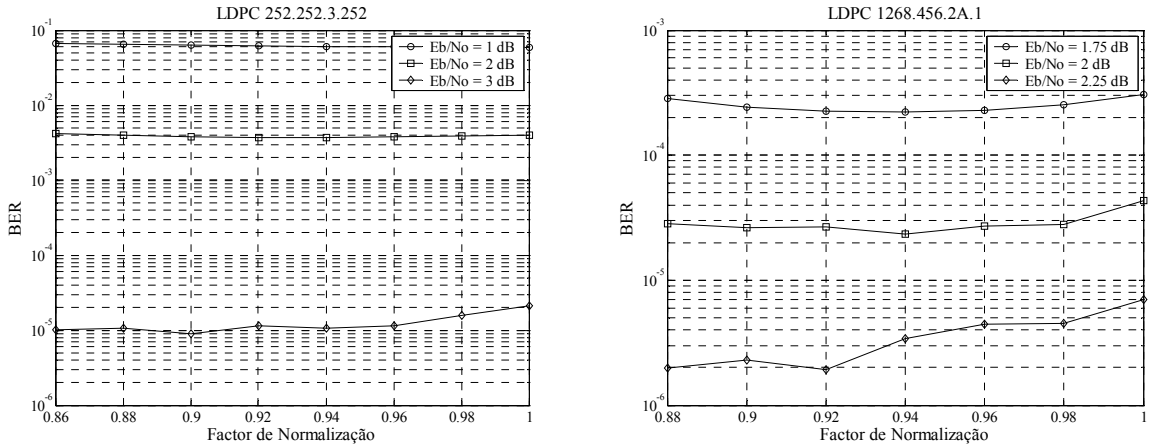


Figura 5-15 – Desempenho algoritmo SPA normalizado para os códigos β e γ, considerando vários factores de normalização e de E_b/N_0.

O desempenho dos algoritmos SPA e SPA normalizado encontra-se representado na figura 5-16, para os códigos α, β e γ, em que os factores de normalização utilizados são os documentados na tabela 5-3.

	Factor de Normalização (θ)
Código α	0.93
Código β	0.9
Código γ	0.92

Tabela 5-3 – Factores de normalização escolhidos para o algoritmo SPA.

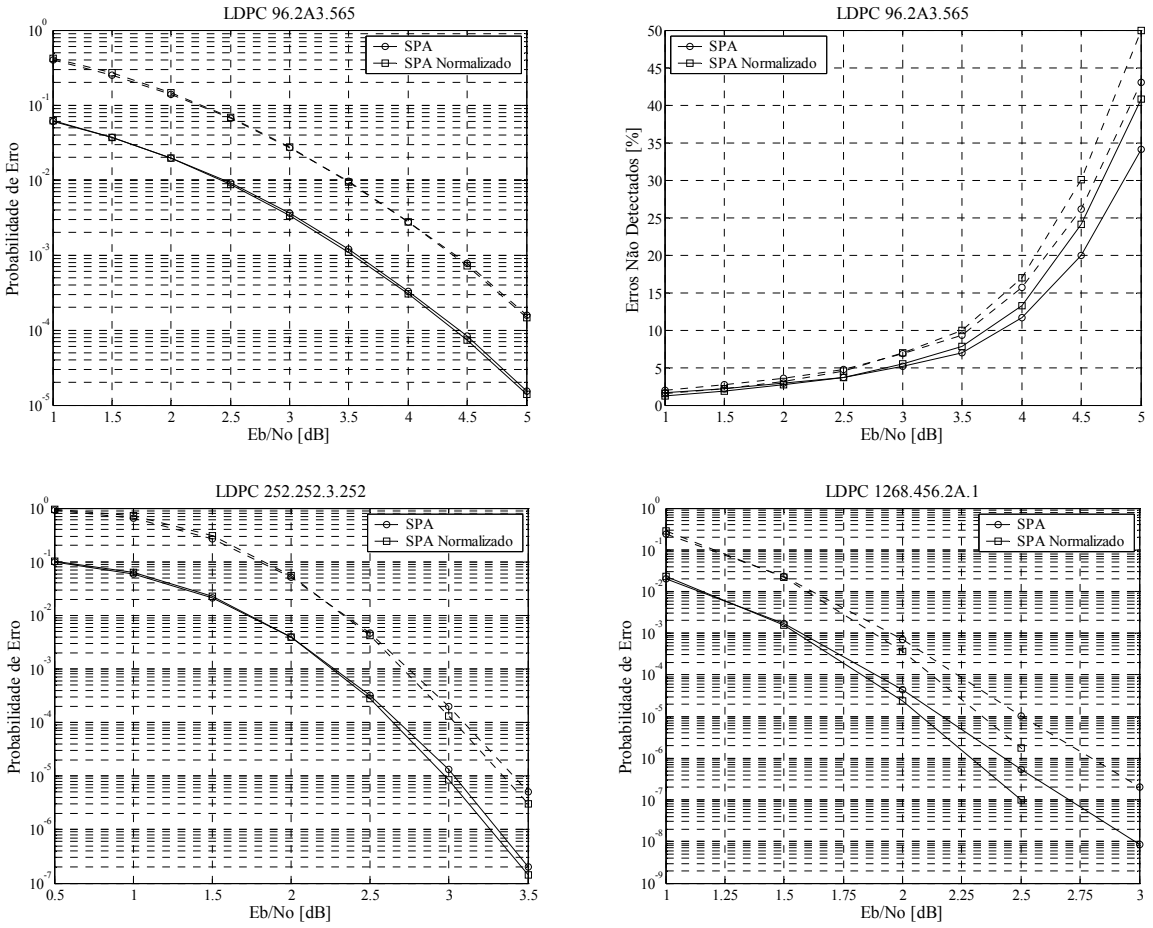


Figura 5-16 – Desempenho expresso em termos de BER (–) e MER (···), dos algoritmos SPA, SPA normalizado para os códigos α, β e γ, e em termos de percentagem de bits errados (–) e de mensagens erradas (···) não detectadas para o código α.

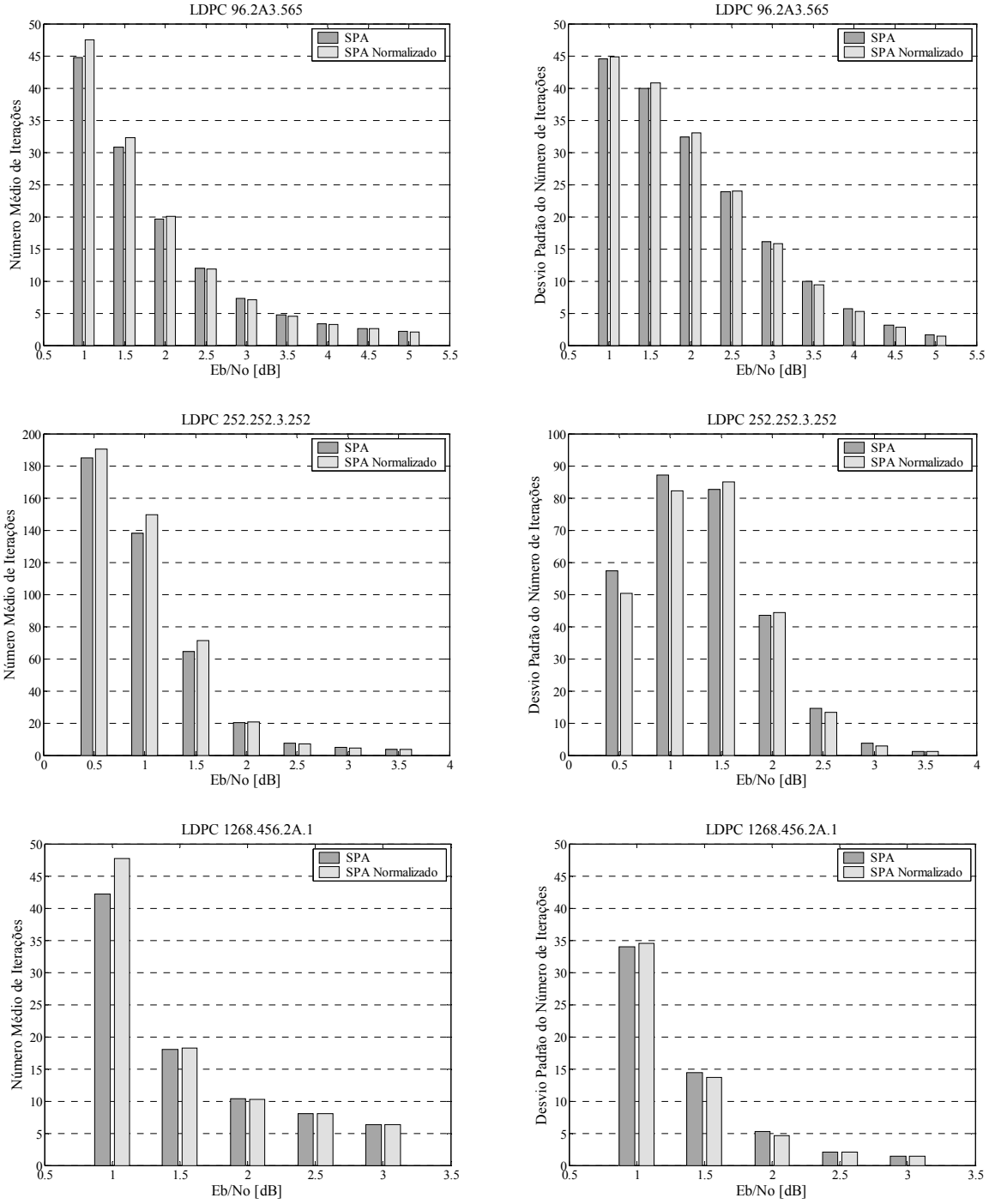


Figura 5-17 – Média e desvio padrão do número de iterações dos algoritmos SPA, SPA normalizado para os códigos α, β e γ.

Uma análise dos resultados obtidos permite constatar que o método proposto representa um ganho efectivo, na medida em que o algoritmo SPA normalizado apresentou sempre um melhor desempenho do que o SPA convencional.

Esse ganho é, no entanto, marginal para códigos com comprimentos de bloco mais pequenos. De facto, para o código α o ganho é praticamente nulo. Com o código β, tendo por referência um BER de 10^{-5}, o ganho de codificação é já aproximadamente 0.08 dB.

Para códigos mais longos o aumento é notório. De facto, no caso do código γ, tomando por referência um BER de 10^{-7}, verificamos que o ganho de codificação é de 0.2 dB. Para um E_b/N_0 de 3 dB, o algoritmo SPA normalizado corrige mesmo todos os erros ocorridos.

No que diz respeito à complexidade computacional do novo método, em termos da média e desvio padrão do número médio de iterações, verificamos pela análise dos gráficos da figura 5-17 que é praticamente igual.

Em resumo, o método proposto consegue ganhos de desempenho significativos relativamente aos resultados reportados na literatura (para códigos mais longos), com o mesmo número médio de iterações. Para o algoritmo LSPA (matematicamente equivalente ao SPA) o peso adicional é de apenas uma multiplicação por cada mensagem enviada pelos CN's.

CAPÍTULO 6

CONCLUSÕES E TRABALHO FUTURO

Neste capítulo apresentamos um resumo das conclusões tiradas no decurso deste trabalho, fazendo uma análise crítica dos resultados experimentais obtidos, tendo por base todo o conhecimento teórico reportado no capítulo 4.

Indicaremos ainda algumas sugestões de trabalho futuro, que pretendemos vir a explorar.

6.1 CONCLUSÕES

Simplificações do algoritmo SPA

- O Algoritmo MS-LSPA apresenta uma diminuição não desprezável de desempenho comparativamente ao algoritmo SPA. Esta fica a dever-se à má estimação das mensagens enviadas dos CN's para os BN's, e como tal, é mais gravosa para códigos com maior número de CN's.

- A diminuição de desempenho do algoritmo MS-LSPA pode ser aceite quando a principal restrição do sistema é a sua latência. Mesmo necessitando, em média, de um maior número de iterações para descodificar uma palavra de código, devido à mais lenta convergência do algoritmo (má estimação das mensagens enviadas dos CN's para os BN's), o reduzido número de operações envolvidas

nos cálculos tornam-no num algoritmo rápido comparativamente aos outros estudados.

- O algoritmo MS-LSPA normalizado é uma boa opção para o dilema desempenho (SPA) versus latência (MS-LSPA). Com apenas mais 2 multiplicações realizadas por CN, o aumento da latência do algoritmo face ao MS-LSPA é pouco significativo, com consideráveis ganhos de desempenho, próximos do algoritmo SPA.

Melhorias ao algoritmo SPA

- A existência de ciclos no TG contribui para uma má estimação das mensagens enviadas entre nodos devido às realimentações positivas presentes no cálculo das mesmas. Considerando a implementação do algoritmo SPA no domínio logarítmico, é possível compensar a má estimação das mensagens enviadas dos CN' para os BN's à custa de um factor de normalização positivo, $\theta < 1$, função do código. Essa melhoria é mais notória para códigos com maior número de CN's em que a acumulação dos erros de estimação será maior. Nos códigos mais longos tendem a existir um maior número de BN's com giros longos, pelo que o factor de normalização introduzido permite compensar de forma mais eficaz a realimentação positiva que ocorre mais tardiamente para cada BN.

- O método de normalização proposto para o algoritmo LSPA não afecta de forma significativa a sua latência, sendo o número médio de iterações sensivelmente o mesmo do algoritmo LSPA.

- O método proposto por Mao e Banihashemi em [MB1], baseado numa gestão das mensagens enviadas dos BN's para os CN's, quando aplicado ao algoritmo SPA não apresenta bons resultados ao contrário do reportado.

6.2 TRABALHO FUTURO

A menor latência do algoritmo SPA comparativamente ao algoritmo LSPA observada nas simulações efectuadas permite-nos pensar que é possível explorar a implementação eficiente do algoritmo SPA na forma convencional em DSP's, possuidores de instruções optimizadas para a realização de somas de produtos.

A melhoria de desempenho observada com o método de normalização das mensagens enviadas dos CN's para os BN's, proposto para o algoritmo LSPA, bem como, as recentes propostas de normalização das mensagens enviadas dos BN's para os CN's propostas por Yazdani, Hemati e Banihashemi [YHB], permite-nos sugerir a combinação dos dois métodos com factores criteriosamente escolhidos para cada código. Certamente, este caminho irá produzir resultados ainda melhores.

Será também de explorar e generalizar o método de quantificação proposto por He, Sun e Wang [HSW], não só por ir ao encontro das restrições que, normalmente, são colocadas em termos do número de bits disponíveis para a representação dos valores numéricos, bem como, o facto do esquema de quantificação proposto produzir uma melhoria do desempenho do algoritmo SPA.

ANEXO A

LEMAS

Lema L1

Considere uma sequência de m *dígitos binários estatisticamente independentes* $\mathbf{a}=(a_1,a_2,\cdots,a_m)$ *em que* $p_k=\Pr(a_k=1)$. *A probabilidade de* **a** *conter um número par de 1's é*

$$\frac{1}{2}+\frac{1}{2}\prod_{k=1}^{m}(1-2p_k) \tag{L1.1}$$

e a probabilidade de conter um número impar de 1's é

$$\frac{1}{2}-\frac{1}{2}\prod_{k=1}^{m}(1-2p_k). \tag{L1.2}$$

Demonstração:

A prova é feita por indução. Seja $z_l=a_1\oplus a_2\oplus\cdots\oplus a_l$ a soma módulo 2 dos primeiros l que constituem a palavra **a**.

- Para $m=2$ temos:

$$\begin{aligned}\Pr(par)&=\Pr(z_2=0)=\Pr(a_1\oplus a_2=0)=\\&=p_1p_2+(1-p_1)(1-p_2)=\frac{1}{2}+\frac{1}{2}(1-2p_1)(1-2p_2)\end{aligned} \tag{L1.3}$$

- Assumindo que a equação (L1.1) é verdadeira para $m=L-1$ vamos verificar se ainda é válida para $m=L$. Assim, usando (L1.3) temos,

$$\Pr(z_L=0)=\Pr(z_{L-1}\oplus a_L=0)=\frac{1}{2}+\frac{1}{2}\big(1-2\Pr(z_{L-1}=1)\big)(1-2p_L). \tag{L1.4}$$

Como assumimos que a equação (L1.1) é verdadeira para $m=L-1$ vem:

$$\begin{aligned}\Pr\left(z_L=0\right)&=\frac{1}{2}+\frac{1}{2}\left(1-2\times\left(\frac{1}{2}-\frac{1}{2}\prod_{k=1}^{L-1}\left(1-2p_k\right)\right)\right)\left(1-2p_L\right)=\\&=\frac{1}{2}+\frac{1}{2}\prod_{k=1}^{L}\left(1-2p_k\right)\end{aligned}\tag{L1.5}$$

e logo

$$\Pr\left(impar\right)=1-\Pr\left(par\right)=1-\left(\frac{1}{2}+\frac{1}{2}\prod_{k=1}^{L}\left(1-2p_k\right)\right)=\frac{1}{2}-\frac{1}{2}\prod_{k=1}^{L}\left(1-2p_k\right).\tag{L1.6}$$

(c. q. d.)

Lema L2

Considere a transmissão através de um canal AWGN de palavras binárias $\mathbf{c}$*, cujos bits são modelados BPSK, com* $x_i=\left(-1\right)^{c_i}$*. Os símbolos recebidos,* $y_i=x_i+n_i$ *com* n_i *AWGN são de média nula e variância* σ^2*. Admitindo que* $\Pr\left(c_i=0\right)=\Pr\left(c_i=1\right)=1/2$ *então,*

$$\Pr\left(x_i=x\middle|y\right)=\frac{1}{1+e^{-\frac{2yx}{\sigma^2}}}.\tag{L2.1}$$

Demonstração:

Por aplicação do teorema de Bayes, sabemos que:

$$\begin{aligned}\Pr\left(x_i=x\middle|y\right)&=\frac{p\left(y\middle|x_i=x\right)\Pr\left(x_i=x\right)}{p\left(y\right)}\\&=\frac{p\left(y\middle|x_i=x\right)\Pr\left(x_i=x\right)}{p\left(y\middle|x_i=-1\right)\Pr\left(x_i=-1\right)+p\left(y\middle|x_i=1\right)\Pr\left(x_i=1\right)}.\end{aligned}\tag{L2.2}$$

Para um canal Gaussiano a função densidade de probabilidade é

$$p\left(y\middle|x_i=x\right)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{\left(y-x\right)^2}{2\sigma^2}}.\tag{L2.3}$$

Por outro lado, atendendo a que os bits transmitidos são modelados BPSK, com $x_i=\left(-1\right)^{c_i}$ e que $\Pr\left(c_i=0\right)=\Pr\left(c_i=1\right)=1/2$ vem

$$\Pr\left(x_i=-1\right)=\Pr\left(x_i=1\right)=\frac{1}{2}.\tag{L2.4}$$

Logo, por substituição de (L2.3) e (L2.4) em (L2.2) obtemos:

$$\Pr\left(x_i = x \middle| y\right) = \frac{\frac{1}{2}\times\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(y-x)^2}{2\sigma^2}}}{\frac{1}{2}\times\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(y+1)^2}{2\sigma^2}}+\frac{1}{2}\times\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(y-1)^2}{2\sigma^2}}}$$

$$= \frac{e^{-\frac{(y-x)^2}{2\sigma^2}}}{e^{-\frac{(y+1)^2}{2\sigma^2}}+e^{-\frac{(y-1)^2}{2\sigma^2}}}$$

$$= \frac{e^{-\frac{y^2}{2\sigma^2}}e^{+\frac{2xy}{2\sigma^2}}e^{-\frac{x^2}{2\sigma^2}}}{e^{-\frac{y^2}{2\sigma^2}}e^{-\frac{2y}{2\sigma^2}}e^{-\frac{1}{2\sigma^2}}+e^{-\frac{y^2}{2\sigma^2}}e^{+\frac{2y}{2\sigma^2}}e^{-\frac{1}{2\sigma^2}}} \quad . \qquad \text{(L2.5)}$$

Como $x^2 = 1$ vem, finalmente,

$$\Pr\left(x_i = x \middle| y\right) = \frac{e^{+\frac{xy}{\sigma^2}}}{e^{-\frac{y}{\sigma^2}}+e^{\frac{y}{\sigma^2}}} = \frac{1}{e^{-\frac{y(1+x)}{\sigma^2}}+e^{\frac{y(1-x)}{\sigma^2}}}$$

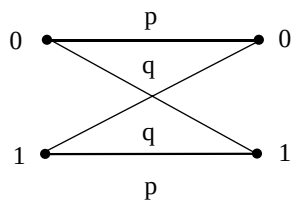
$$= \begin{cases} \dfrac{1}{1+e^{\frac{2y}{\sigma^2}}} & \Leftarrow \quad x=-1 \\ \dfrac{1}{1+e^{-\frac{2y}{\sigma^2}}} & \Leftarrow \quad x=+1 \end{cases}$$

$$= \frac{1}{1+e^{-\frac{2yx}{\sigma^2}}} \quad . \qquad \text{(L2.6)}$$

(c. q. d.)

Lema L3

Seja x um bit transmitido através de um canal BSC, caracterizado por uma probabilidade de acerto p e de erro $q = 1 - p$,



BSC

e y o bit recebido. Admitindo que $\Pr(x=0) = \Pr(x=1) = 1/2$ então:

$$\Pr\left(x = 0 \middle| y\right) = p^{1-y}q^{y}, \qquad \text{(L3.1)}$$

$$\Pr\left(x = 1 \middle| y\right) = p^{y}q^{1-y}. \qquad \text{(L3.2)}$$

Demonstração:

Por aplicação do teorema de Bayes, sabemos que:

$$\Pr\left(x=a\middle|y\right)=\frac{\Pr\left(y\middle|x=a\right)\Pr\left(x=a\right)}{\Pr\left(y\right)}=$$

$$=\frac{\Pr\left(y\middle|x=a\right)\Pr\left(x=a\right)}{\Pr\left(y\middle|x=0\right)\Pr\left(x=0\right)+\Pr\left(y\middle|x=1\right)\Pr\left(x=1\right)}. \qquad \text{(L3.3)}$$

Consideremos então o caso em que $a=0$ (para o caso em que $a=1$ a prova far-se-á de forma análoga). Assim, admitindo que $\Pr\left(x=0\right)=\Pr\left(x=1\right)=1/2$, (L3.3) vem

$$\Pr\left(x=0\middle|y\right)=\frac{\frac{1}{2}\times\Pr\left(y\middle|x=0\right)}{\frac{1}{2}\times\Pr\left(y\middle|x=0\right)+\frac{1}{2}\times\Pr\left(y\middle|x=1\right)}$$

$$=\frac{\Pr\left(y\middle|x=0\right)}{\Pr\left(y\middle|x=0\right)+\Pr\left(y\middle|x=1\right)}$$

$$=\begin{cases}\dfrac{p}{p+q} & \Leftarrow \quad y=0\\ \dfrac{q}{q+p} & \Leftarrow \quad y=1\end{cases}$$

$$=p^{1-y}q^{y} \qquad . \qquad \text{(L3.4)}$$

(c. q. d.)

Lema L4

Seja $x_1, x_2, \cdots, x_n$, com $n\geq 2$, um conjunto de variáveis aleatórias binárias e seja y_n a variável aleatória resultante da soma módulo 2 de todos as variáveis x_i, com $i=1,\cdots,n$. A máxima verosimilhança de y_n é função das máximas verosimilhanças das variáveis aleatórias x_i com

$$\mathrm{mv}\left(y_n\right)=\frac{1+\prod_{i=1}^{n}\left(\frac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)}{1-\prod_{i=1}^{n}\left(\frac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)}. \qquad \text{(L4.1)}$$

Esta verifica também a seguinte relação recursiva,

$$\mathrm{mv}\left(y_n\right)=\frac{\mathrm{mv}\left(y_{n-1}\right)\mathrm{mv}\left(x_n\right)+1}{\mathrm{mv}\left(y_{n-1}\right)+\mathrm{mv}\left(x_n\right)}, \qquad \text{(L4.2)}$$

em que a máxima verosimilhança de uma variável aleatória x é definida por

$$\mathrm{mv}(x)=\frac{\Pr(x=0)}{\Pr(x=1)}. \tag{L4.3}$$

Demonstração:

Comecemos por provar em primeiro lugar a expressão (L4.2). Assim, vamos considerar apenas a soma de duas variáveis aleatórias x_1 e x_2 com $y=x_1\oplus x_2$. A máxima verosimilhança da variável aleatória y é

$$\begin{aligned}\mathrm{mv}(y) &= \frac{\Pr(y=0)}{\Pr(y=1)}=\frac{\Pr(x_1=0)\Pr(x_2=0)+\Pr(x_1=1)\Pr(x_2=1)}{\Pr(x_1=0)\Pr(x_2=1)+\Pr(x_1=1)\Pr(x_2=0)}= \\ &= \frac{\dfrac{\Pr(x_1=0)}{\Pr(x_1=1)}\dfrac{\Pr(x_2=0)}{\Pr(x_2=1)}+1}{\dfrac{\Pr(x_1=0)}{\Pr(x_1=1)}+\dfrac{\Pr(x_2=0)}{\Pr(x_2=1)}}=\frac{\mathrm{mv}(x_1)\mathrm{mv}(x_2)+1}{\mathrm{mv}(x_1)+\mathrm{mv}(x_2)}\end{aligned}\quad , \tag{L4.4}$$

portanto, função das máximas verosimilhanças de x_1 e x_2.

Da mesma forma, se conclui que o resultado anterior pode ser aplicado de forma recursiva. De facto, seja $y_n=x_1\oplus x_2\oplus\cdots\oplus x_n$ com $n>2$. Podemos simplesmente escrever que $y_n=y_{n-1}\oplus x_n$ e, como tal, temos que:

$$\mathrm{mv}(y_n)=\frac{\mathrm{mv}(y_{n-1})\mathrm{mv}(x_n)+1}{\mathrm{mv}(y_{n-1})+\mathrm{mv}(x_n)} \tag{L4.5}$$

tal como queríamos demonstrar.

Tendo em conta o resultado anterior provemos por indução a expressão (L4.1). Comecemos por considerar $n=2$. Por (L4.1) vem:

$$\begin{aligned}\mathrm{mv}(y_2) &= \frac{1+\prod_{i=1}^{2}\left(\dfrac{\mathrm{mv}(x_i)-1}{\mathrm{mv}(x_i)+1}\right)}{1-\prod_{i=1}^{2}\left(\dfrac{\mathrm{mv}(x_i)-1}{\mathrm{mv}(x_i)+1}\right)}=\frac{1+\left(\dfrac{\mathrm{mv}(x_1)-1}{\mathrm{mv}(x_1)+1}\right)\left(\dfrac{\mathrm{mv}(x_2)-1}{\mathrm{mv}(x_2)+1}\right)}{1-\left(\dfrac{\mathrm{mv}(x_1)-1}{\mathrm{mv}(x_1)+1}\right)\left(\dfrac{\mathrm{mv}(x_2)-1}{\mathrm{mv}(x_2)+1}\right)}= \\ &= \frac{\left[\mathrm{mv}(x_1)+1\right]\left[\mathrm{mv}(x_2)+1\right]+\left[\mathrm{mv}(x_1)-1\right]\left[\mathrm{mv}(x_2)-1\right]}{\left[\mathrm{mv}(x_1)+1\right]\left[\mathrm{mv}(x_2)+1\right]-\left[\mathrm{mv}(x_1)-1\right]\left[\mathrm{mv}(x_2)-1\right]}= \\ &= \frac{\mathrm{mv}(x_1)\mathrm{mv}(x_2)+1}{\mathrm{mv}(x_1)+\mathrm{mv}(x_2)}\end{aligned}\quad . \tag{L4.6}$$

Este valor é exactamente o mesmo da equação (L4.4), provando a validade da expressão para $n=2$.

Assumindo que a equação (L4.1) é verdadeira para y_n vamos verificar se ainda é válida para y_{n+1} . Assim:

$$
\begin{aligned}
\mathrm{mv}\left(y_{n+1}\right) &= \mathrm{mv}\left(y_n \oplus x_{n+1}\right) \\
&= \frac{\mathrm{mv}\left(y_n\right)\times \mathrm{mv}\left(x_{n+1}\right)+1}{\mathrm{mv}\left(y_n\right)+\mathrm{mv}\left(x_{n+1}\right)} \\
&= \frac{\dfrac{1+\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)}{1-\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)}\times \mathrm{mv}\left(x_{n+1}\right)+1}{\dfrac{1+\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)}{1-\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)}+\mathrm{mv}\left(x_{n+1}\right)} \\
&= \frac{\mathrm{mv}\left(x_{n+1}\right)+\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)\times \mathrm{mv}\left(x_{n+1}\right)+1-\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)}{1+\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)+\mathrm{mv}\left(x_{n+1}\right)-\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)\times \mathrm{mv}\left(x_{n+1}\right)} \\
&= \frac{\left[\mathrm{mv}\left(x_{n+1}\right)+1\right]+\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)\times\left[\mathrm{mv}\left(x_{n+1}\right)-1\right]}{\left[\mathrm{mv}\left(x_{n+1}\right)+1\right]-\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)\times\left[\mathrm{mv}\left(x_{n+1}\right)-1\right]} \\
&= \frac{1+\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)\times\dfrac{\left[\mathrm{mv}\left(x_{n+1}\right)-1\right]}{\left[\mathrm{mv}\left(x_{n+1}\right)+1\right]}}{1-\prod_{i=1}^{n}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)\times\dfrac{\left[\mathrm{mv}\left(x_{n+1}\right)-1\right]}{\left[\mathrm{mv}\left(x_{n+1}\right)+1\right]}} \\
&= \frac{1+\prod_{i=1}^{n+1}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)}{1-\prod_{i=1}^{n+1}\left(\dfrac{\mathrm{mv}\left(x_i\right)-1}{\mathrm{mv}\left(x_i\right)+1}\right)} \qquad . \qquad \text{(L4.7)}
\end{aligned}
$$

(c. q. d.)

Lema L5

Seja $x_1, x_2, \cdots, x_n$, *com* $n \geq 2$, *um conjunto de variáveis aleatórias binárias e seja* y_n *a variável aleatória resultante da soma módulo 2 de todos as variáveis* x_i, *com* $i = 1, \cdots, n$. *A máxima verosimilhança logarítmica de* y_n *é função das máximas verosimilhanças das variáveis aleatórias* x_i *com,*

$$\mathrm{mvl}\left(y_n\right)=\ln\left(\frac{1+\prod_{i=1}^{n}\left(\frac{\mathrm{e}^{\mathrm{mvl}\left(x_i\right)}-1}{\mathrm{e}^{\mathrm{mvl}\left(x_i\right)}+1}\right)}{1-\prod_{i=1}^{n}\left(\frac{\mathrm{e}^{\mathrm{mvl}\left(x_i\right)}-1}{\mathrm{e}^{\mathrm{mvl}\left(x_i\right)}+1}\right)}\right). \tag{L5.1}$$

Esta verifica também a seguinte relação recursiva

$$\mathrm{mvl}\left(y_n\right)=\ln\left(\frac{\mathrm{e}^{\mathrm{mvl}\left(y_{n-1}\right)+\mathrm{mvl}\left(x_n\right)}+1}{\mathrm{e}^{\mathrm{mvl}\left(y_{n-1}\right)}+\mathrm{e}^{\mathrm{mvl}\left(x_n\right)}}\right) \tag{L5.2}$$

em que a máxima verosimilhança logarítmica de uma variável aleatória x é definida por

$$\mathrm{mvl}\left(x\right)=\ln\left(\frac{\Pr\left(x=0\right)}{\Pr\left(x=1\right)}\right). \tag{L5.3}$$

Demonstração:

A demonstração do lema resulta apenas, e só, da aplicação da função logaritmo natural às equações (L4.1) e (L4.2) do lema L4. De facto, a partir da definição da função máxima verosimilhança logarítmica temos

$$\mathrm{mvl}\left(x\right)=\ln\left[\mathrm{mv}\left(x\right)\right]. \tag{L5.4}$$

Como tal, aplicando a função logaritmo natural à equação (L4.1) vamos obter,

$$\begin{aligned}\mathrm{mvl}\left(y_n\right)&=\ln\left[\mathrm{mv}\left(y_n\right)\right]\\&=\ln\left(\frac{1+\prod_{i=1}^{n}\left(\frac{\mathrm{e}^{\ln\left[\mathrm{mv}\left(x_i\right)\right]}-1}{\mathrm{e}^{\ln\left[\mathrm{mv}\left(x_i\right)\right]}+1}\right)}{1-\prod_{i=1}^{n}\left(\frac{\mathrm{e}^{\ln\left[\mathrm{mv}\left(x_i\right)\right]}-1}{\mathrm{e}^{\ln\left[\mathrm{mv}\left(x_i\right)\right]}+1}\right)}\right)\\&=\ln\left(\frac{1+\prod_{i=1}^{n}\left(\frac{\mathrm{e}^{\mathrm{mvl}\left(x_i\right)}-1}{\mathrm{e}^{\mathrm{mvl}\left(x_i\right)}+1}\right)}{1-\prod_{i=1}^{n}\left(\frac{\mathrm{e}^{\mathrm{mvl}\left(x_i\right)}-1}{\mathrm{e}^{\mathrm{mvl}\left(x_i\right)}+1}\right)}\right).\end{aligned} \tag{L5.5}$$

De forma idêntica, por aplicação da função logaritmo natural à equação (L4.2) vamos obter,

$$\begin{aligned}\mathrm{mvl}\left(y_n\right)&=\ln\left[\mathrm{mv}\left(y_n\right)\right]\\&=\ln\left[\frac{\mathrm{mv}\left(y_{n-1}\right)\mathrm{mv}\left(x_n\right)+1}{\mathrm{mv}\left(y_{n-1}\right)+\mathrm{mv}\left(x_n\right)}\right]\\&=\ln\left[\frac{\mathrm{e}^{\ln\left[\mathrm{mv}\left(y_{n-1}\right)\right]}\mathrm{e}^{\ln\left[\mathrm{mv}\left(x_n\right)\right]}+1}{\mathrm{e}^{\ln\left[\mathrm{mv}\left(y_{n-1}\right)\right]}+\mathrm{e}^{\ln\left[\mathrm{mv}\left(x_n\right)\right]}}\right]\\&=\ln\left[\frac{\mathrm{e}^{\mathrm{mvl}\left(y_{n-1}\right)}\mathrm{e}^{\mathrm{mvl}\left(x_n\right)}+1}{\mathrm{e}^{\mathrm{mvl}\left(y_{n-1}\right)}+\mathrm{e}^{\mathrm{mvl}\left(x_n\right)}}\right].\end{aligned} \tag{L5.6}$$

(c. q. d.)

Lema L6

Dados dois números reais λ e μ então

$$2\tanh^{-1}\left(\tanh\left(\lambda/2\right)\tanh\left(\mu/2\right)\right)=\log\left(\frac{1+e^{\lambda+\mu}}{e^{\lambda}+e^{\mu}}\right). \tag{L6.1}$$

Demonstração:

Sabendo que:

$$\tanh\left(\frac{x}{2}\right)=\frac{e^{x}-1}{e^{x}+1} \tag{L6.2}$$

e que

$$\tanh^{-1}\left(x\right)=\frac{1}{2}\ln\left(\frac{1+x}{1-x}\right) \tag{L6.3}$$

podemos desenvolver a expressão (L6.1):

$$\begin{aligned}2\tanh^{-1}\left(\tanh\left(\lambda/2\right)\tanh\left(\mu/2\right)\right)&=\log\left(\frac{1+\tanh\left(\lambda/2\right)\tanh\left(\mu/2\right)}{1-\tanh\left(\lambda/2\right)\tanh\left(\mu/2\right)}\right)=\\&=\log\left(\frac{1+\dfrac{e^{\lambda}-1}{e^{\lambda}+1}\dfrac{e^{\mu}-1}{e^{\mu}+1}}{1-\dfrac{e^{\lambda}-1}{e^{\lambda}+1}\dfrac{e^{\mu}-1}{e^{\mu}+1}}\right)=\\&=\log\left(\frac{e^{\lambda+\mu}+e^{\lambda}+e^{\mu}+1+e^{\lambda+\mu}-e^{\lambda}-e^{\mu}+1}{e^{\lambda+\mu}+e^{\lambda}+e^{\mu}+1-e^{\lambda+\mu}+e^{\lambda}+e^{\mu}-1}\right)=\\&=\log\left(\frac{1+e^{\lambda+\mu}}{e^{\lambda}+e^{\mu}}\right)\end{aligned}. \tag{L6.4}$$

(c. q. d.)

Lema L7

Dados dois números reais x e y então

$$\log_{a}\left(a^{x}+a^{y}\right)=\max\left(x,y\right)+\log_{a}\left(1+a^{-\left|x-y\right|}\right). \tag{L7.1}$$

Demonstração:

Consideremos dois números reais x e y quaisquer. Então, duas situações poderão acontecer: $x>y$ ou $x<y$. Para cada um dos casos vem

$$\log_a\left(a^x + a^y\right) = \begin{cases} \log_a\left[a^x\left(1+a^{y-x}\right)\right] & \Leftarrow \quad x > y \\ \log_a\left[a^y\left(1+a^{x-y}\right)\right] & \Leftarrow \quad x < y \end{cases}$$
$$= \begin{cases} x + \log_a\left(1+a^{y-x}\right) & \Leftarrow \quad x > y \\ y + \log_a\left(1+a^{x-y}\right) & \Leftarrow \quad x < y \end{cases}$$
$$= \max(x, y) + \log_a\left(1+a^{-|x-y|}\right) \quad . \qquad \text{(L7.2)}$$

(c. q. d.)

Lema L8

Dados dois números reais x e y então

$$\max(0, x+y) - \max(x, y) = \operatorname{sgn}(x)\operatorname{sgn}(y)\min\left(|x|, |y|\right). \qquad \text{(L8.1)}$$

Demonstração:

A demonstração do lema em causa assenta na enumeração de todos os casos possíveis. Assim se:

- $x > 0$ e $y > 0$

$$\max(0, x+y) - \max(x, y) = \begin{cases} x+y-x & \Leftarrow \quad x > y \\ x+y-y & \Leftarrow \quad x < y \end{cases}$$
$$= \begin{cases} y & \Leftarrow \quad x > y \\ x & \Leftarrow \quad x < y \end{cases} \quad , \qquad \text{(L8.2)}$$

$$\operatorname{sgn}(x)\operatorname{sgn}(y)\min\left(|x|, |y|\right) = \begin{cases} 1\times 1\times |y| & \Leftarrow \quad x > y \\ 1\times 1\times |x| & \Leftarrow \quad x < y \end{cases}$$
$$= \begin{cases} y & \Leftarrow \quad x > y \\ x & \Leftarrow \quad x < y \end{cases} \quad , \qquad \text{(L8.3)}$$

logo a igualdade é válida para $x > 0$ e $y > 0$.

- $x < 0$ e $y < 0$

$$\max(0, x+y) - \max(x, y) = \begin{cases} 0-x & \Leftarrow \quad x > y \\ 0-y & \Leftarrow \quad x < y \end{cases}$$
$$= \begin{cases} |x| & \Leftarrow \quad x > y \\ |y| & \Leftarrow \quad x < y \end{cases} \quad , \qquad \text{(L8.4)}$$

$$\operatorname{sgn}(x)\operatorname{sgn}(y)\min\left(|x|, |y|\right) = \begin{cases} (-1)\times(-1)\times|x| & \Leftarrow \quad x > y \\ (-1)\times(-1)\times|y| & \Leftarrow \quad x < y \end{cases}$$
$$= \begin{cases} |x| & \Leftarrow \quad x > y \\ |y| & \Leftarrow \quad x < y \end{cases} \quad , \qquad \text{(L8.5)}$$

logo a igualdade é válida para $x<0$ e $y<0$.

- $x>0$ e $y<0$

$$\max(0,x+y)-\max(x,y)=\begin{cases}x+y-x & \Leftarrow \quad |x|>|y| \\ 0-x & \Leftarrow \quad |x|<|y|\end{cases}$$
$$=\begin{cases}-|y| & \Leftarrow \quad |x|>|y| \\ -|x| & \Leftarrow \quad |x|<|y|\end{cases}, \tag{L8.6}$$

$$\operatorname{sgn}(x)\operatorname{sgn}(y)\min(|x|,|y|)=\begin{cases}1\times(-1)\times|y| & \Leftarrow \quad |x|>|y| \\ 1\times(-1)\times|x| & \Leftarrow \quad |x|<|y|\end{cases}$$
$$==\begin{cases}-|y| & \Leftarrow \quad |x|>|y| \\ -|x| & \Leftarrow \quad |x|<|y|\end{cases}, \tag{L8.7}$$

logo a igualdade é válida para $x>0$ e $y<0$.

- $x<0$ e $y>0$

$$\max(0,x+y)-\max(x,y)=\begin{cases}0-y & \Leftarrow \quad |x|>|y| \\ x+y-y & \Leftarrow \quad |x|<|y|\end{cases}$$
$$=\begin{cases}-|y| & \Leftarrow \quad |x|>|y| \\ -|x| & \Leftarrow \quad |x|<|y|\end{cases}, \tag{L8.8}$$

$$\operatorname{sgn}(x)\operatorname{sgn}(y)\min(|x|,|y|)=\begin{cases}1\times(-1)\times|y| & \Leftarrow \quad |x|>|y| \\ 1\times(-1)\times|x| & \Leftarrow \quad |x|<|y|\end{cases}$$
$$==\begin{cases}-|y| & \Leftarrow \quad |x|>|y| \\ -|x| & \Leftarrow \quad |x|<|y|\end{cases}, \tag{L8.9}$$

logo a igualdade é válida para $x<0$ e $y>0$.

Conclui-se portanto que a igualdade (L8.1) é válida para $\forall\ x, y$.

(c. q. d.)

BIBLIOGRAFIA

[BGT] C. Berrou, A. Glavieux e P. Thitimajshimi, "Near Shannon Limit Error-Correcting Coding and Decoding", *Proceedings ICC'93*, Genéve, Suíça, pp. 1064-70, Maio 1993.

[BNK] T. Bhatt, K. Narayanan, N. Kehtarnavaz, "Fixed Point DSP Implementation of Low-Density Parity Check Codes", *Proceedings of the 9th IEEE DSP Workshop*, Hunt, Texas, Outubro 2000.

[Bos] Martin Bossert, *Channel Coding for Telecommunications*, John Wiley & Sons, 1999.

[BVJD] C. Berrou, S. Vaton, M. Jézéquel e C. Douillard, "Computing the Monimum Distance of Linear Codes by the Error Impulse Method", *IEEE GLOBECOM '02*, Taipei, Taiwan, Novembro 2002.

[Carl] A. B. Carlson, *Communication Systems: An Introduction to Signals and Noise in Electrical Communication*, 3ª Edição, McGraw-Hill, 1986.

[CF1] J. Chen e M. P. C. Fossorier, "Near Optimum Universal Belief Propagation Based Decoding of Low-Density Parity Check Codes", *IEEE Transactions on Communications*, vol. 50, nº 3, pp. 406-414, Março 2002.

[CF2] J. Chen e M. P. C. Fossorier, "Decoding Low-Density Parity Check Codes with Normalized APP-Based Algorithm", *IEEE GLOBECOM '01*, vol. 2, pp. 1026-1030, 2001.

[CFRU] S. Chung, G.Forney, T. Richardson e R. Urbanke, "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit", *IEEE Communications Letters*, vol. 5, nº 2, pp. 58-60, Fevereiro 2001.

[CT] Thomas M. Cover, Joy A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.

[EMD] E. Eleftheriou, T. Mittelholzer e A. Dholakia, "Reduced-Complexity Decoding Algorithm for Low-Density Parity-Check Codes", *Electronics Letters*, vol. 37, nº 2, pp. 102-104, Janeiro 2001.

[ETV] T. Etzion, A. Trachtenberg e A. Vardy, "Which Codes Have Cycle-Free Tanner Graphs", *IEEE Transactions on Information Theory*, vol. 45, nº 6, pp. 2173-2181, Setembro 1999.

[FMI] M. P. C. Fossorier, M. Mihaljevic e H. Imai, "Reduced Complexity Iterative Decoding of Low-Density Parity Check Codes Based on Belief Propagation", *IEEE Transactions on Communications*, vol. 47, nº 5, pp. 673-690, Maio 1999.

[Forn] G. D. Forney, Jr., "On Iterative Decoding and the Two-Way Algorithm", *Proceedings of International Symposium on Turbo Codes and Related Topics*, Brest, França. pp. 12-25, Setembro 1997.

[Foss] Marc P. C. Fossorier, "Iterative Reliability-Based Decoding of Low-Density Parity Check Codes", *IEEE Journal on Selected Areas in Communications*, vol. 19, nº 5, pp. 908-917, Maio 2001.

[Gal1] R. G. Gallager, "Low-Density Parity-Check Codes", *IRE Transactions Information Theory*, vol. IT-8, pp. 21-28, Janeiro 1962.

[Gal2] R. G. Gallager, *Low-Density Parity-Check Codes.* , Cambridge, MA: MIT Press, 1963.

[Gal3] R. G. Gallager, *Information Theory and Reliable Communication*, John Wiley and Sons, 1968.

[GBD] F. Guillod, E. Boutillon e J. L. Danger, "λ-Min Decoding Algorithm of Regular and Irregular LDPC Codes", *3rd International Symposium on Turbo Codes & Related Topics*, Brest, France, pp 451-454, Setembro, 2003.

[HEAD] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold e A. Dholakia, "Efficient Implementations of the Sum-Product Algorithm for Decoding LDPC Codes", *IEEE GLOBECOM '01*, vol. 2, 2001.

[HM] X.-Y. Hu e T. Mittelholzer, "An Ordered-Statistics-Based Approximation of the Sum-Product Algorithm", *ITS 2002, Internacional Telecommunications Symposium*, pp. 205-210, Setembro 2002.

[HMO] J. Hagenauer, M. Morez e E. Offer, "Analog Turbo-Networks in VLSI: The Next Step in Turbo Decoding and Equalization", *Proceedings 2nd Intl. Sypmp. Turbo Codes & Related Topics*, Brest, France, pp. 209-218, Setembro 2000.

[HSW] Y. He, S. Sun e X. Wang, "Fast Decoding of LDPC Codes Using Quantisation", *Electronics Letters*, vol. 38, nº 4, Fevereiro 2002.

[JW] S. Johnson e S. R. Weller, "Quasi-cyclic LDPC codes from difference families", *3rd Australian Communications Theory Workshop*, Canberra, February 4-5, 2002.

[KF] F. R. Kschischang e B. J. Frey, "Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models", *IEEE Journal on Selected Areas in Communications*, vol. 16, nº 2, pp. 219-230, Fevereiro 1998.

[KFL] F. R. Kschischang, B. J. Frey e H. Loeliger, "Factor Graphs and the Sum-Product Algorithm", *IEEE Transactions on Information Theory*, vol. 47, nº 2, pp. 498-519, Fevereiro 2001.

[KLF] Y. Kou, S. Lin e 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 2001.

[Ksch] F. R. Kschischang, "Codes Defined on Graphs", *IEEE Communications Magazine*, pp. 118-125, Agosto 2003.

[Lech] G. Lechner, "Convergence of Sum-Product Algorithm for Finite Length Low-Density Parity-Chech Codes", *Winter School on Coding and Information Theory*, Monte Verità, Suiça, Fevereiro 2003.

[LLHT] H.-A Loeliger, F. Lustenberger, M. Helfenstein e F. Tarkov, "Probability Propagation and Decoding in Analog VLSI", *IEEE Transactions on Information Theory*, vol. 47, pp. 837-843, Fevereiro 2001.

[LLWP] W. K. Leung, W. L. Lee, A. Wu e L.Ping, "Efficient implementation technique of LDPC decoder", *Electronic Letters*, vol. 37, nº 20, pp. 1231-1232, 27 de Setembro 2001.

[LMSS1] M. Luby, M. Mitzenmacher, M. Shokrollahi e D. Spielman, "Analysis of Low Density Codes and Improved Designs Using Irregular Graphs", *Proceedings of the 30th ACM Symposium on Theory of Computing (STOC)*, Maio 1998.

[LMSS2] M. Luby, M. Mitzenmacher, M. Shokrollahi e D. Spielman, "Improved Low-Density Parity-Check Codes Using Irregular Graphs and Belief Propagation", *Proceedings of the 1998 IEEE International Symposium on Information Theory*, Agosto 1998.

[LS] G. Lechner e J. Sayir, "On the Convergence of Log-Likelihood Values in Iterative Decoding", *Mini-Workshop on Topics in Information Theory*, Essen, Alemanha, Setembro 2002.

[Mac1] D. J. C. Mackay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2003.

[Mac2] D. J. C. Mackay, "Good Error-Correcting Codes based on Very Sparse Matrices", *IEEE Transactions on Information Theory*, vol. 45, pp. 399-431, Março 1999.

[Mac3] Sítio de Internet: *http://www.inference.phy.cam.ac.uk/mackay/codes/data.html*

[MB1] Y. Mao e A. H. Banihashemi, "Decoding Low-Density Parity-Check Codes With Probabilistic Scheduling", *IEEE Communication Letters*, vol. 5, nº 10, pp. 414-416, Outubro 2001.

[MB2] Y. Mao e A. H. Banihashemi, "A Heuristic Search for Good Low-Density Parity-Check Codes at Short Block Lenghts", *IEEE ICC 2001*, Filândia, Junho 2001.

[MN1] D. J. C. Mackay e R. M. Neal, "Near Shannon limit performance of low density parity check codes", *Electronics Letters*, vol. 33, nº 6, pp 457-458, Março 1997.

[MN2] D. J. C. Mackay e R. M. Neal, "Good Codes based on Very Sparse Matrices", *Cryptography and Coding. 5th IMA Conference ed. Colin Boyd, Lecture Notes in Computer Science nº* 1025, pp.100-111, Springer, Berlim, 1995.

[MW] J. A. McGowan e R. C. Williamson, "Loop Removal from LDPC Codes", *IEEE Information Theory Workshop 2003*, Paris, 2003.

[Nay] A. Nayagam, "Codes on Graphs: Behavioural Realizations and the Sum-Product Algorithm", Fevereiro 2001.

[NB] A. Nouh e A. H. Banihashemi, "Bootstrap Decoding of Low-Density Parity Check Codes", *IEEE Communications Letters*, vol. 6, nº 9, pp. 391-392, Setembro 2002.

[PL] L.Ping e W. K. Leung, "Decoding Low Density Parity Check Codes with Finite Quantization Bits", *IEEE Communications Letters*, vol. 4, nº 2, pp. 62-64, Fevereiro 2000.

[PN] A. Prabhakar e K.R. Narayanan, "Pseudorandom Construction of LDPC codes using Linear Congruential Sequences", *IEEE Transactions on Communications*, vol. 50, nº 9, pp. 1389-1396, Setembro 2002.

[PTVF] W. H. Press, S. A. Teukolsky, W. T. Vetterling e B. P. Flannery, *Numerical Recipes in C - The Art of Scientific Computing Second Edition*, Cambridge University Press, 2002.

[Rich] T. Richardsom, "The Renaissance of Gallager's Low-Density Parity-Check Codes", *IEEE Communications Magazine*, pp. 126-130, Agosto de 2003.

[RSU] T. Richardson, M. Shokrollahi e R. Urbanke, "Design of Capacity Approaching Irregular Low-Density Parity-Check Codes", *IEEE Transactions on Information Theory*, vol. 47, nº 2, pp. 619-637, Fevereiro 2001.

[RU] T. Richardson e R. Urbanke, "The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding", *IEEE Transactions on Information Theory*, vol. 47, nº 2, pp. 599-618, Fevereiro 2001.

[Ryan] W. Ryan, "An Introduction to Low-Density Parity-Check Codes", University of Arizona, Abril 2001.

[Sal] Carlos Salema, *Feixes Hertzianos*, Colecção Ensino da Ciência e Tecnologia, IST-Press, 1998.

[Shan] C. E. Shannon, "A mathematical theory of communication", *Bell Systems Technical Journal*, 27: 397-423, 623-656, 162, 1948.

[Tan1] R. M. Tanner, "A recursive approach to low complexity codes", *IEEE Transactions on Information Theory*, vol. IT-27, nº 5, pp. 533-547, Setembro 1981.

[VO] A. J. Viterbi, J. K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, 1979.

[Wel1] R. B. Wells, *Applied Coding and Information Theory for Engineers*, Prentice Hall Information and System Science Series, Series Editor, 1999.

[Weld] E. J. Weldon, "Difference-Set Cyclic Codes", *Bell Systems Technical Journal*, vol. 45, pp. 1045-1055, Setembro 1996.

[Wick] Stephen B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice Hall, 1995.

[YHB] M. Yazdani, S. Hemati e A. Banihashemi, "Improving Belief Propagation on Graphs With Cycles", *IEEE Communications Letters*, vol. 8, nº 1, pp 57-59, Janeiro 2004.

[YNA] E. Yeo, B. Nicolić e V. Anantharam, "Architectures and Implementations of Low-Density Parity Check Decoding Algorithms", *invited paper at IEEE International Midwest Symposium on Circuits and Systems*, Agosto, 2002.

[Zar] Robert H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, John Wiley & Sons, 2002.

[ZWP] Tong Zhang, Zhongfeng Wang e Keshab K. Parhi, "On Finite Implementation of Low Density Parity Check Codes Decoder", *Proceedings of ISCAS*, Sidney, Australia, Maio de 2001.