Descodificação iterativa de códigos LDPC por transferência de mensagens em grafos de factores

© Sílvio A. Abrantes
Departamento de Engenharia Electrotécnica e de Computadores
Faculdade de Engenharia, Universidade do Porto
Porto, Portugal
sam@fe.up.pt

Julho de 2005

Resumo: Os códigos turbo e LDPC, os melhores códigos correctores de erros da actualidade, partilham em comum o facto de usarem métodos de descodificação iterativa que recorrem a algoritmos de natureza idêntica. Em particular os códigos LDPC são habitualmente caracterizados por grafos bipartidos de Tanner e grafos de factores e são descodificados com um algoritmo muito genérico de transferência de mensagens entre nós dos grafos, o algoritmo da soma-e-produto, de que o algoritmo BCJR, usado nos turbo-códigos, é um caso particular. Neste tutorial é apresentado o algoritmo da soma-e-produto e duas simplificações (os algoritmos "max-product" e "min-sum") e a sua utilização é ilustrada através de vários exemplos detalhados.

Palavras-chave: códigos LDPC, algoritmo da soma-e-produto, algoritmos de transferência de mensagens, códigos em grafos, grafos de Tanner, grafos de factores.

1. Objectivo

Este texto é um tutorial de apresentação da descodificação iterativa por transferência de mensagens em grafos de Tanner, método normalmente usado na descodificação de códigos LDPC. São abordados os algoritmos da soma-e-produto, "max-product" e "min-sum" e a exposição serve-se de exemplos numéricos para ilustrar os procedimentos. O texto destina-se a todos os que desejem compreender o funcionamento dos algoritmos que, através de grafos, descodificam códigos correctores de erros de uma forma iterativa muito eficiente.

A estrutura do tutorial é a seguinte: na Secção 2 é feita a introdução do tema, o que inclui referências não só aos códigos LDPC mas também aos grafos de Tanner e de factores; na Secção 3 é explicado o funcionamento dos algoritmos de transferência de mensagens em grafos de factores, mostrando-se como se calculam as mensagens com o *algoritmo da soma-e-produto*, assumam aquelas o papel de estimativas de probabilidades condicionais ou sejam logaritmos de razões de verosimilhanças (LLR); os algoritmos de cálculo aproximado "max-product" e "min-sum" são também expostos na Secção 3; a Secção 4 apresenta um exemplo numérico detalhado de descodificação iterativa de um código de blocos através do algoritmo da soma-e-produto; o tutorial prossegue com um Apêndice e termina com a lista de referências.

2. Introdução aos códigos LDPC e aos grafos de Tanner

A Fig. 1 apresenta o diagrama de blocos de um sistema genérico de codificação e descodificação de canal. Seja $\mathbf{u} = u_1 u_2 \ldots u_k$ uma sequência de k bits que, aplicada a um codificador binário de taxa k/n, produz a palavra de código $\mathbf{x} = x_1 x_2 \ldots x_n$. Esta, depois de atravessar o canal de comunicações, é transformada na sequência $\mathbf{y} = y_1 y_2 \ldots y_n$ recebida

pelo descodificador. É tarefa deste estimar convenientemente a palavra de código **x** ou a sequência binária original, **u**.

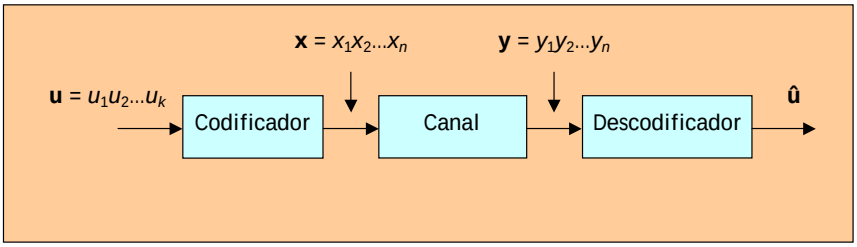


Fig. 1 Diagrama de blocos de um sistema genérico de codificação/descodificação.

Neste texto vamos ver como descodificar a sequência **y** quando usamos códigos LDPC. Em particular, vamos ver como o fazer usando grafos bipartidos, ou de Tanner. Antes de começar, contudo, temos de saber o que são códigos LDPC, grafos de Tanner e… LLR a posteriori.

2.1. Breve introdução aos códigos LDPC

Os códigos LDPC (de "Low Density Parity Check") foram inventados por Robert Gallager, que os apresentou pela primeira vez em 1962 [1] e depois na sua tese de doutoramento em 1963 [2]. Apesar de Gallager também ter proposto o método (iterativo) de descodificação, a verdade é que os códigos LDPC eram demasiado "avançados" para a tecnologia da época e ficaram adormecidos e deixados ao abandono durante cerca de trinta anos. A letargia terminaria em 1993 com o aparecimento dos seus despertadores, os turbo-códigos [3].

Um código LDPC é um código de blocos binário caracterizado por uma matriz de verificação de paridade **H** esparsa, isto é, uma matriz em que o número de "uns" é pequeno em relação ao número de "zeros". Isso quer dizer que o número de variáveis envolvidas em cada equação de paridade é pequeno. Na verdade, como as linhas de **H** representam as equações de paridade e as colunas representam os bits da palavra de código, o elemento da linha *j* e coluna *i* só é 1 se o bit x_i fizer parte da *j*-ésima equação de paridade. A Fig. 2 apresenta um exemplo de matriz **H** de um código LDPC em que existem seis bits 1 em cada linha e três bits 1 em cada coluna. A um código em que, como neste, o número de "uns" por coluna ou por linha é constante chama-se código LDPC regular, caso contrário diz-se… irregular!

Actualmente os códigos LDPC são, juntamente com os códigos turbo, os códigos correctores de erros com melhor desempenho (em particular os códigos LDPC irregulares) porque são os que melhor se aproximam do inultrapassável limite de Shannon. Na prática os códigos LDPC têm comprimentos muito longos, com valores de *n* da ordem das dezenas de milhar ou mesmo mais (um milhão, por exemplo!).

É usual descrever os códigos LDPC através de grafos de Tanner.

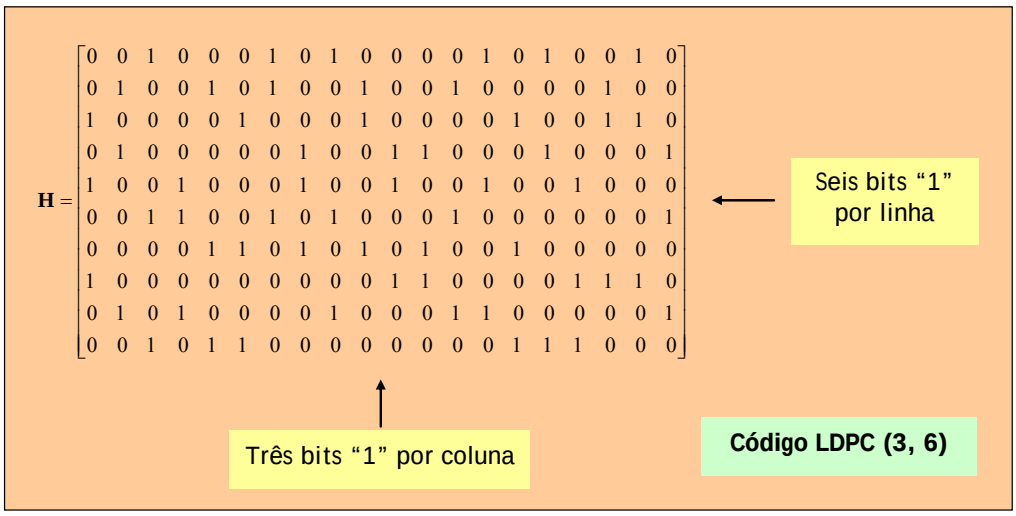


Fig. 2 Exemplo de matriz **H** de um código LDPC regular.

2.2. Grafos de Tanner e grafos de factores

Os grafos de Tanner são devidos a R. Michael Tanner, que os inventou em 1981 para caracterizar e descodificar códigos correctores de erros como BCH e LDPC [4].

Os grafos de Tanner são grafos *bipartidos*. Com isto quer-se dizer que o grafo está dividido em duas regiões distintas, cada uma com seu tipo de nó, ligadas por linhas que unem nós de tipo diferente. Supondo que queremos representar um código de blocos (n,k), num dos lados do grafo são colocados n nós de variáveis, habitualmente representados por círculos, e no outro são colocados $n-k$ nós de controlo, ou de paridade, habitualmente representados por quadrados (veja-se a Fig. 3). Ao número de linhas que confluem num dado nó chama-se *grau* do nó e o número total de linhas do grafo é igual ao número de "uns" da matriz **H** do código. Nos grafos de Tanner cada nó de variável está associado a um bit de código e cada nó de paridade está associado a uma equação de paridade (e como tal, apenas está ligado às variáveis dessa equação). Sendo assim, podemos encarar cada nó de paridade como uma função com dois valores possíveis: se a equação de paridade respectiva for satisfeita vale 1, se não o for vale 0.

Consideremos então um código linear binário de Hamming (n, k) = (7, 4) com a matriz de verificação de paridade **H** da Fig. 3, para a qual as equações de paridade são

$$x_1 \oplus x_2 \oplus x_3 \oplus x_5 = 0$$
$$x_2 \oplus x_3 \oplus x_4 \oplus x_6 = 0$$
$$x_3 \oplus x_4 \oplus x_5 \oplus x_7 = 0$$

sendo $\mathbf{x} = x_1, x_2, \ldots, x_7$ uma palavra de código. A mesma Fig. 3 mostra o correspondente grafo de Tanner.

Neste exemplo temos três funções de quatro variáveis[1], $f_1(x_1, x_2, x_3, x_5)$, $f_2(x_2, x_3, x_4, x_6)$ e $f_3(x_3, x_4, x_5, x_7)$, funções que, como se acabou de dizer, valem 1 ou 0 consoante as

[1] Usando a terminologia dos grafos bipartidos: os nós de paridade têm todos grau 4. Já os nós de variáveis têm graus diversos: por exemplo, a variável x_3 tem grau 3 e a variável x_4 tem grau 2.

equações de paridade do código são ou não satisfeitas, ou seja, f_1, f_2 e f_3 são funções binárias *indicadoras de pertença* ao código. Portanto, uma dada palavra de n bits pertence ao código se todas as funções indicadoras forem iguais a 1 e não pertence se pelo menos uma das funções for nula. A sequência 1101000, por exemplo, não é uma palavra de código pois nem todas as funções associadas aos nós de paridade valem 1:

$$\begin{aligned} 1+1+0+0=0 &\Rightarrow f_1=1 \\ 1+0+1+0=0 &\Rightarrow f_2=1 \\ 0+1+0+0=1 &\Rightarrow f_3=0 \end{aligned}$$

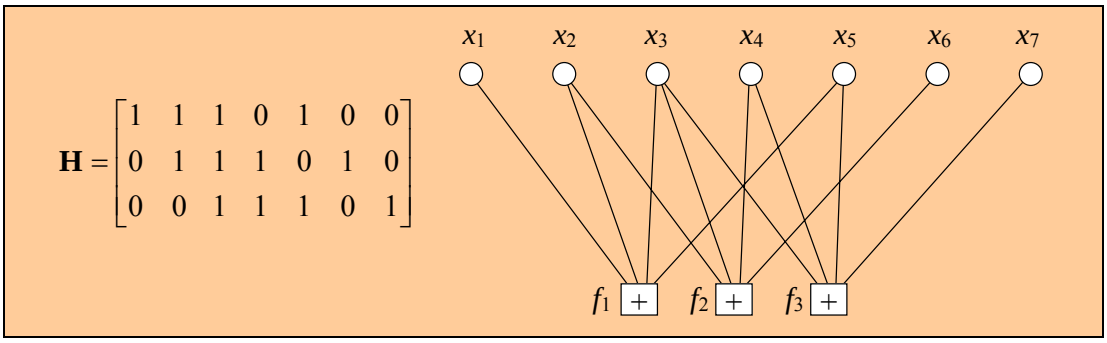


Fig. 3 A matriz de paridade de um código de Hamming (7,4) e o grafo de Tanner associado.

Poderemos escrever então:

$$f_1(x_1,x_2,x_3,x_5)=\delta(x_1\oplus x_2\oplus x_3\oplus x_5)=\begin{cases}1 & x_1\oplus x_2\oplus x_3\oplus x_5=0\\ 0 & x_1\oplus x_2\oplus x_3\oplus x_5\neq 0\end{cases} \quad (1)$$

$$f_2(x_2,x_3,x_4,x_6)=\delta(x_2\oplus x_3\oplus x_4\oplus x_6) \quad (2)$$

$$f_3(x_3,x_4,x_5,x_7)=\delta(x_3\oplus x_4\oplus x_5\oplus x_7) \quad (3)$$

Nas expressões atrás $\delta(x)$ representa o delta de Kronecker, $\delta(x)=\begin{cases}1 & x=0\\ 0 & x\neq 0\end{cases}$.

Na aplicação que nos interessa, a descodificação, os grafos de Tanner são usados para transferir *mensagens* entre nós através das linhas que os unem. Ora é bem de ver que quanto mais linhas o grafo possuir mais mensagens são transferidas e mais cálculos são necessários. Aqui está a razão pela qual os códigos LDPC, com o seu número reduzido de "uns" em **H**, estão particularmente bem talhados para a descodificação baseada em grafos bipartidos.

2.2.1. Dos grafos de Tanner aos grafos de factores

Embora de modo breve, é conveniente fazer uma referência aos grafos de factores.

Suponhamos que uma dada função, dita *função global*, é igual ao produto de *funções locais* de várias variáveis, como $g(x_1,x_2,x_3,x_4)=h_1(x_1,x_3)h_2(x_1,x_4)h_3(x_2,x_3,x_4)$. Um grafo de factores é um grafo bipartido que serve para visualizar a factorização da função

global indicando que variáveis são argumentos de que funções locais. Um exemplo, referente à função $g(x_1, x_2, x_3, x_4)$, é apresentado na Fig. 4.

Do ponto de vista dos grafos de factores o grafo de Tanner do código de Hamming da Fig. 3 representa o produto das três funções indicadoras f_1, f_2 e f_3, $f_1(x_1, x_2, x_3, x_5)\, f_2(x_2, x_3, x_4, x_6)\, f_3(x_3, x_4, x_5, x_7)$, de valor 1 ou 0 consoante a sequência de sete bits $\mathbf{x} = x_1, x_2, \ldots, x_7$ pertença ou não ao código.

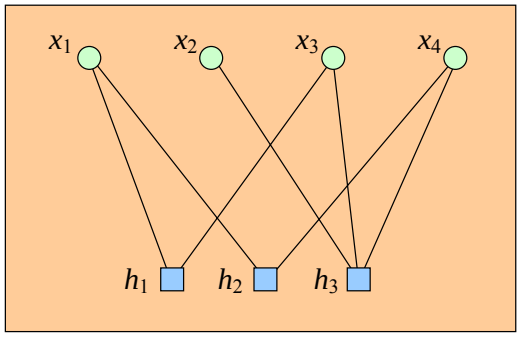
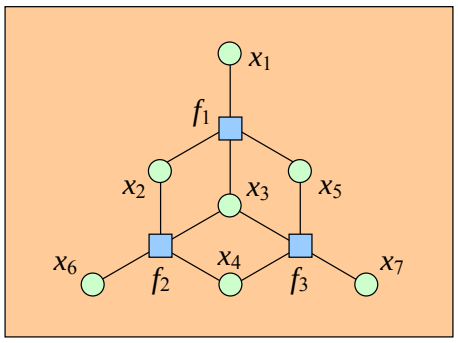


Fig. 4 O grafo de factores de $g(x_1, x_2, x_3, x_4) = h_1(x_1, x_3) h_2(x_1, x_4) h_3(x_2, x_3, x_4)$.

Cronologicamente primeiro surgiram os grafos bipartidos de Tanner. Depois, por volta de 1996, Wiberg e outros modificaram-nos por forma a conterem variáveis de estado [5] [6] [7]. Mais tarde ainda surgiram os grafos de factores [8] [9], generalizações dos grafos "tipo" Wiberg que se aplicam a funções factorizáveis como a de cima. O algoritmo genérico de cálculo em todos esses grafos é denominado *algoritmo da soma-e-produto* e vai ocupar-nos bastante em breve.

Exemplo 1

Os grafos de factores prestam-se a composições gráficas interessantes. Voltemos ao código de Hamming (7,4) da Fig. 3. O seu grafo de Tanner desenhado como grafo de factores apresenta uma atraente simetria:



2.2.2. Grafo de factores associado a um canal binário simétrico

Na Fig. 5 é apresentado o diagrama de probabilidades de transição de um canal binário simétrico sem memória (ou canal BSC[2]) com probabilidade de erro p, ou seja, com $p(y \mid x) = \begin{cases} 1-p & y = x \\ p & y \neq x \end{cases}$ ou, equivalentemente, $p(y \mid x) = \begin{cases} py + (1-p)(1-y) & x = 0 \\ p(1-y) + (1-p)y & x = 1 \end{cases}$.

Suponhamos que a palavra de código $\mathbf{x} = x_1, x_2, \ldots, x_n$ é transmitida através do canal, que a transforma na sequência binária $\mathbf{y} = y_1, y_2, \ldots, y_n$.

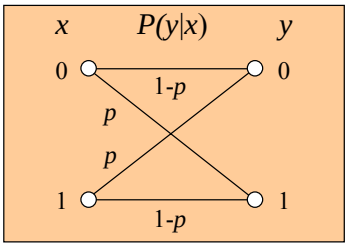


Fig. 5 Um canal binário simétrico com probabilidade de erro p.

Como o canal BSC não tem memória as n transmissões de símbolos são independentes e portanto a probabilidade condicional conjunta a priori $p(\mathbf{y}|\mathbf{x})$ é igual ao produto das probabilidades condicionais individuais $p(y_i|x_i)$:

$$p(\mathbf{y} \mid \mathbf{x}) = p(y_1, y_2, \ldots, y_n \mid x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(y_i \mid x_i) .$$

Sendo um produto de factores esta probabilidade condicional conjunta pode ser representada por um grafo de factores em que cada nó de paridade representa a função probabilidade condicional a priori $p(y_i|x_i)$. Ora, atendendo a que os valores y_i recebidos à saída do canal não são, de facto, variáveis mas sim constantes cada nó de paridade desse grafo apenas depende de uma única variável, x_i. Com $n = 7$ o grafo de factores do canal BSC é o apresentado na Fig. 6.

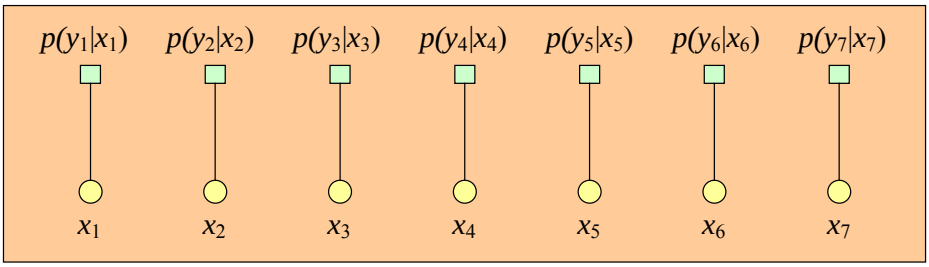


Fig. 6 O grafo de factores de $p(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{7} p(y_i \mid x_i)$.

2.2.3. Grafo de factores associado a um canal AWGN

O grafo de factores de um canal afectado de ruído gaussiano branco aditivo – o conhecido canal AWGN[3] – é semelhante ao grafo de factores de um canal BSC. Tal

[2] BSC: "Binary Symmetric Channel"

como este, o canal AWGN não tem memória e a probabilidade condicional a priori de uma dada sequência é igual ao produto das probabilidades condicionais individuais. Mas agora as probabilidades discretas dão lugar à função densidade de probabilidade (fdp) do ruído gaussiano, de média nula e variância σ^2,

$$f(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-y^2/2\sigma^2}$$

isto é, $\mathcal{N}(0,\sigma^2)$, e a sequência binária de valores lógicos 0 e 1 é convertida na sequência antipodal $x_i = \pm 1$. Assim, com ruído gaussiano cada um destes valores, -1 e +1, fica associado a uma fdp gaussiana com a mesma variância σ^2 e de média -1 e +1, respectivamente (designemo-las por $f_-(y)$ e $f_+(y)$, como se mostra na Fig. 7). Se na saída do canal recebermos a sequência de valores reais y_i as probabilidades condicionais de que precisamos são calculadas para os dois valores possíveis de x_i: $p(y_i \mid x_i = -1) = f_-(y_i)$ e $p(y_i \mid x_i = +1) = f_+(y_i)$. É preciso, contudo, normalizar estes valores para que a sua soma seja unitária, isto é,

$$p(y_i \mid x_i = -1) = \frac{f_-(y_i)}{f_-(y_i) + f_+(y_i)} = \frac{1}{1 + \exp\left(2y_i/\sigma^2\right)}$$

$$p(y_i \mid x_i = +1) = \frac{f_+(y_i)}{f_-(y_i) + f_+(y_i)} = \frac{1}{1 + \exp\left(-2y_i/\sigma^2\right)}$$

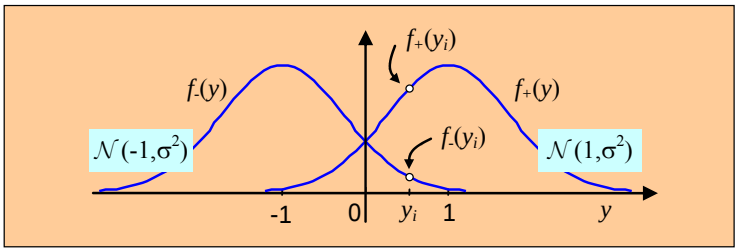


Fig. 7 As funções densidade de probabilidade gaussianas associadas aos símbolos -1 e +1.

Estes valores normalizados desempenham o mesmo papel que os valores de $p(y_i|x_i)$ do canal binário simétrico, de modo que usaremos $p(y_i|x_i)$ sem perda de generalidade.

2.2.4. Grafo de factores de um canal BSC em série com um descodificador de canal

Juntando os grafos de factores de $p(\mathbf{y}|\mathbf{x})$ e do (des)codificador (7, 4) da Fig. 3 obtemos o grafo de factores global[4] da Fig. 8. Os nós p_i estão "pendurados" e têm apenas um

[3] AWGN: "Additive White Gaussian Noise".
[4] No grafo passámos a designar os nós de paridade do canal por p_i.

vizinho, x_i. Dado que o grafo de factores pode ser desenhado como árvore diz-se que os nós p_i (ou a linha que une p_i a x_i) são *nós-folha* (ou *nós-linha*). Por vezes os nós p_i (os quadrados) são omitidos.

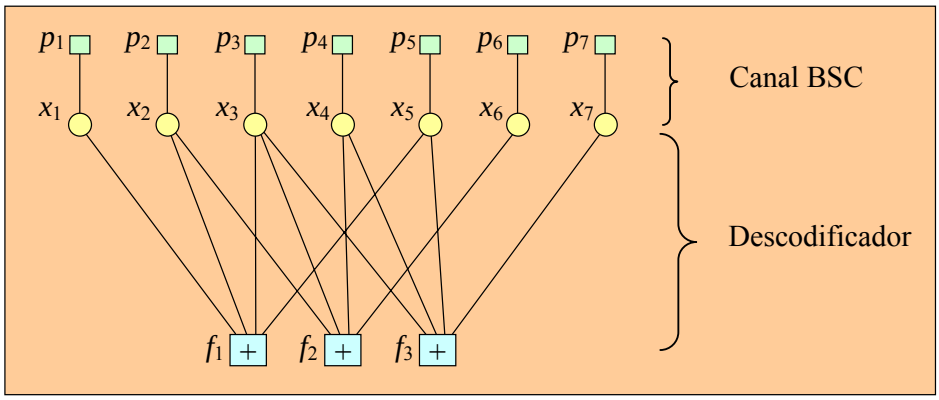


Fig. 8 O grafo de factores global de canal BSC + descodificador.

2.3. Probabilidades e LLR a posteriori

É habitual estimar a sequência **x** a partir da sequência recebida **y** usando um de dois métodos: estimação de máxima verosimilhança (ou estimação ML, de "Maximum Likelihood") e estimação da máxima probabilidade a posteriori (ou estimação MAP, de "Maximum A Posteriori"). O primeiro método calcula a probabilidade condicional a priori $p(\mathbf{y} \mid x_i)$ – para determinar a sequência $\hat{\mathbf{x}}$ que maximiza $p(\mathbf{y} \mid \mathbf{x})$ – e o segundo calcula a probabilidade condicional a posteriori $p(x_i \mid \mathbf{y}) = p(\mathbf{y} \mid x_i)\dfrac{p(x_i)}{p(\mathbf{y})}$ (também designada por APP[5]) para determinar o símbolo x_i. O método que nos interessa é o método de estimação MAP. Queremos, portanto, calcular a probabilidade de se ter enviado o bit x_i dado que se recebeu a sequência **y**. Além desta probabilidade APP também nos vai interessar usar o logaritmo da razão de probabilidades $\dfrac{p(x_i = 1 \mid \mathbf{y})}{p(x_i = 0 \mid \mathbf{y})}$,

$$L(x_i \mid \mathbf{y}) = \ln \frac{p(x_i = 1 \mid \mathbf{y})}{p(x_i = 0 \mid \mathbf{y})},$$

que designaremos por *LLR a posteriori*[6].

O critério de decisão final sobre o bit x_i é o habitual: se a probabilidade APP ou a razão LLR estiverem acima de um determinado limiar de decisão escolhemos $x_i = 1$, se estiverem abaixo escolhemos $x_i = 0$. O limiar de decisão é 0,5 no caso da APP e 0 no caso da LLR. É costume exprimir o critério desta maneira:

$$L(x_i \mid \mathbf{y}) \underset{0}{\overset{1}{\gtrless}} 0$$

[5] APP: "A Posteriori Probability"
[6] LLR: "Log-Likelihood Ratio"

Portanto, se a LLR for positiva escolhemos o bit 1 e se for negativa escolhemos o bit 0.

As probabilidades e as LLR a posteriori podem ser calculadas de diversas maneiras. Em codificação turbo, por exemplo, é normal usar o algoritmo BCJR [10]. Com códigos LDPC é costume recorrer a grafos (de Tanner e de factores) e a algoritmos de transferência de mensagens entre os seus nós. O que contêm as mensagens? Estimativas de probabilidades ou quantidades com elas relacionadas, como a LLR. Vamos ver como se faz.

3. Transferência de mensagens entre nós de grafos

Os algoritmos de descodificação baseados em grafos são designados genericamente por *algoritmos de transferência de mensagens*. O nome que efectivamente tomam depende um pouco do contexto: enquanto que na Teoria da Codificação e na presença de grafos de factores se prefere a designação *algoritmo da soma-e-produto* ("sum-product algorithm", ou SP), em redes bayesianas a comunidade da Inteligência Artificial usa mais o termo "belief propagation algorithm" (BP). O que aconteceu é que, por vias e em aplicações muito distintas, ambas as comunidades chegaram a um algoritmo que a outra desconhecia mas que o tempo veio a provar que, afinal, era o mesmo[7]. O algoritmo BP foi formulado por J. Pearl em 1988 [11] e chama-se assim porque propaga "crenças" (probabilidades) de uns nós de grafos para os outros. Mas já Gallager, mais de vinte anos antes, utilizara um método iterativo semelhante no seu trabalho pioneiro sobre os códigos LDPC. Tanner generalizou-o em 1981 com o algoritmo SP [4]. Mais tarde, já depois do aparecimento dos turbo-códigos em 1993, o algoritmo da soma-e-produto foi aplicado com sucesso aos grafos de factores, do que resultou a sua formulação mais genérica [7]. Na prática, chamemos-lhe nós BP ou SP, estes algoritmos de transferência de mensagens são algoritmos SISO ("Soft Input-Soft Output") pois a entrada e a saída – probabilidades – são valores reais.

Os algoritmos SP e BP são tão gerais que, sem o suspeitarmos, muitos métodos, algoritmos e técnicas conhecidos há décadas não são senão… casos particulares daqueles. A lista é grande e inclui, entre outros, os algoritmos de descodificação de Gallager, Viterbi e BCJR (estes últimos usados em treliças de códigos convolucionais, e não só…) e a descodificação iterativa de turbo-códigos (com o algoritmo BCJR) [12]. E até os filtros de Kalman não escapam [7] [9].

Consideremos então um grafo de Tanner com nós de variáveis x_i e nós de controlo f_j a trocar mensagens deste género: a variável x_i **q**uestiona f_j sobre as "opiniões" que os outros x_i têm dela e f_j **r**esponde-lhe com as "opiniões" que conhece. Tais perguntas (questões) e respostas reflectem-se na notação que vamos usar:

$q_{ij}(x_i)$ – mensagem do nó de variável x_i para o nó de paridade f_j

$r_{ji}(x_i)$ – mensagem do nó de paridade f_j para o nó de variável x_i

Estas mensagens, ou probabilidades, são função de um único argumento (a variável x_i) quer circulem num sentido quer no outro. A mensagem de x_i para f_j representa a probabilidade de x_i ter um certo valor, dado o valor observado dessa variável (a

[7] Porém, Kschischang e outros [7] [8] consideram que o algoritmo BP é um caso particular do "seu" algoritmo SP.

quantidade $p(y_i|x_i)$) e dados os valores que recebeu dos outros nós de paridade, o que, se x_i for binária, é escrito como

$$q_{ij}(x_i) = \begin{bmatrix} q_{ij}(x_i = 0) \\ q_{ij}(x_i = 1) \end{bmatrix} = \\ = \begin{bmatrix} p(x_i = 0 | \sim \{f_j\}, \mathbf{y}) \\ p(x_i = 1 | \sim \{f_j\}, \mathbf{y}) \end{bmatrix}$$

Na expressão anterior[8] $\sim \{f_j\}$ representa o conjunto das funções (ou nós de paridade) ligadas a x_i excepto f_j.

De igual modo, a mensagem de f_j para x_i é a probabilidade de x_i ter um certo valor e a equação de paridade f_j ser satisfeita, dado que se recebeu **y**,

$$r_{ji}(x_i) = \begin{bmatrix} r_{ji}(0) \\ r_{ji}(1) \end{bmatrix} = \\ = \begin{bmatrix} p(x_i = 0, f_j(\,) = 1 | \mathbf{y}) \\ p(x_i = 1, f_j(\,) = 1 | \mathbf{y}) \end{bmatrix}.$$

Se a variável puder tomar M valores discretos as mensagens também têm M valores, um por cada valor possível de x_i:

$$\begin{bmatrix} q_{ij}(0) \\ q_{ij}(1) \\ \vdots \\ q_{ij}(M-1) \end{bmatrix} \qquad \begin{bmatrix} r_{ji}(0) \\ r_{ji}(1) \\ \vdots \\ r_{ji}(M-1) \end{bmatrix}$$

Neste texto só iremos considerar variáveis binárias (bits, portanto) pois os códigos LDPC mais vulgares são binários.

Exemplo 2

Observe-se a Fig. 9. O que significa a mensagem $r_{11}(x_1)$? De acordo com o exposto atrás $\begin{bmatrix} r_{11}(0) \\ r_{11}(1) \end{bmatrix} = \begin{bmatrix} p(x_1 = 0, f_1(x_1, x_2, x_3, x_5) = 1 | \mathbf{y}) \\ p(x_1 = 1, f_1(x_1, x_2, x_3, x_5) = 1 | \mathbf{y}) \end{bmatrix}$. Mas escrever $f_1(x_1, x_2, x_3, x_5) = 1$ é o mesmo que escrever $x_1 \oplus x_2 \oplus x_3 \oplus x_5 = 0$. Portanto,

[8] Note-se que bastaria considerar apenas um dos dois valores da mensagem dado que conhecendo um conhecemos o outro (pois a soma dos dois elementos do vector é 1).

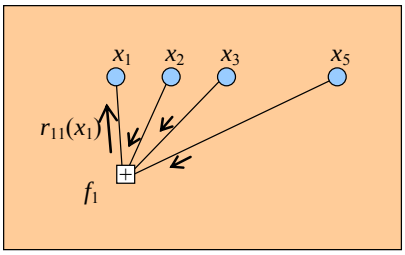


Fig. 9 A mensagem $r_{11}(x_1)$

$$\begin{bmatrix} r_{11}(0) \\ r_{11}(1) \end{bmatrix} = \begin{bmatrix} p(x_1 = 0, x_1 \oplus x_2 \oplus x_3 \oplus x_5 = 0 \,|\, \mathbf{y}) \\ p(x_1 = 1, x_1 \oplus x_2 \oplus x_3 \oplus x_5 = 0 \,|\, \mathbf{y}) \end{bmatrix} = \\ = \begin{bmatrix} p(x_2 \oplus x_3 \oplus x_5 = 0 \,|\, \mathbf{y}) \\ p(x_2 \oplus x_3 \oplus x_5 = 1 \,|\, \mathbf{y}) \end{bmatrix}$$

A soma dos três bits x_2, x_3 e x_5 é nula em quatro situações mutuamente exclusivas e o mesmo se passa com a soma igual a 1:

Soma igual a 0	Soma igual a 1
$x_2 = 0$, $x_3 = 0$, $x_5 = 0$	$x_2 = 0$, $x_3 = 0$, $x_5 = 1$
$x_2 = 0$, $x_3 = 1$, $x_5 = 1$	$x_2 = 0$, $x_3 = 1$, $x_5 = 0$
$x_2 = 1$, $x_3 = 0$, $x_5 = 1$	$x_2 = 1$, $x_3 = 0$, $x_5 = 0$
$x_2 = 1$, $x_3 = 1$, $x_5 = 0$	$x_2 = 1$, $x_3 = 1$, $x_5 = 1$

Logo,

$$r_{11}(0) = p(x_2 = 0, x_3 = 0, x_5 = 0 \,|\, \mathbf{y}) + p(x_2 = 0, x_3 = 1, x_5 = 1 \,|\, \mathbf{y}) + \\ + p(x_2 = 1, x_3 = 0, x_5 = 1 \,|\, \mathbf{y}) + p(x_2 = 1, x_3 = 1, x_5 = 0 \,|\, \mathbf{y})$$

$$r_{11}(1) = p(x_2 = 0, x_3 = 0, x_5 = 1 \,|\, \mathbf{y}) + p(x_2 = 0, x_3 = 1, x_5 = 0 \,|\, \mathbf{y}) + \\ + p(x_2 = 1, x_3 = 0, x_5 = 0 \,|\, \mathbf{y}) + p(x_2 = 1, x_3 = 1, x_5 = 1 \,|\, \mathbf{y})$$

Quais são os valores das mensagens e como é que elas são transferidas? Tudo começa com os valores observados no canal, aqueles que são entregues ao descodificador.

Suponhamos, por exemplo, que o nó da variável x_i está ligado a três nós de paridade e recebe do canal a probabilidade fixa p_i, como está na Fig. 10 à esquerda. No início de todo o processo de transferências a variável difunde p_i pelos três nós de paridade de forma igual – ou seja, não precisamos de nada mais que os valores de p_i como ponto de partida do algoritmo. Depois, cada um dos nós de paridade envia a x_i uma nova mensagem que tem em conta exclusivamente a "opinião" (as estimativas) sobre x_i que acabou de receber de outros nós de variáveis (não mostrados na figura).

O recebimento das mensagens pelos nós de variáveis termina a primeira iteração de cálculos, que prosseguem e se repetem daqui para a frente com transferências dos nós de variáveis para os nós de paridade e vice-versa, de um lado para o outro… até se atingir um determinado número pré-estabelecido de iterações ou quando já houver "certezas" suficientes sobre a idoneidade das mensagens em circulação (por exemplo, verificando se a estimativa final $\hat{\mathbf{x}}$ satisfaz $\hat{\mathbf{x}}\mathbf{H}^T = \mathbf{0}$).

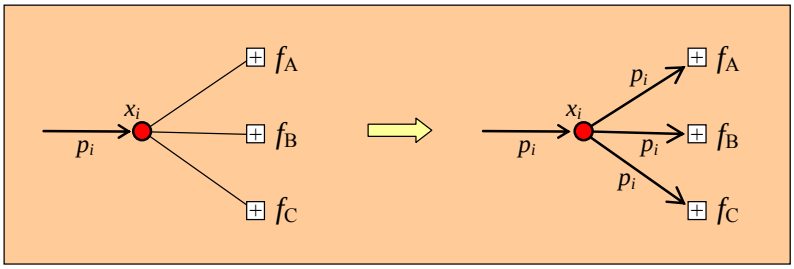


Fig. 10 Início da difusão de mensagens dos nós de variáveis para os nós de paridade.

Para exemplificar com um caso concreto consideremos o grafo de Tanner da Fig. 8 e os vários passos da Fig. 11 relativamente à variável x_2.

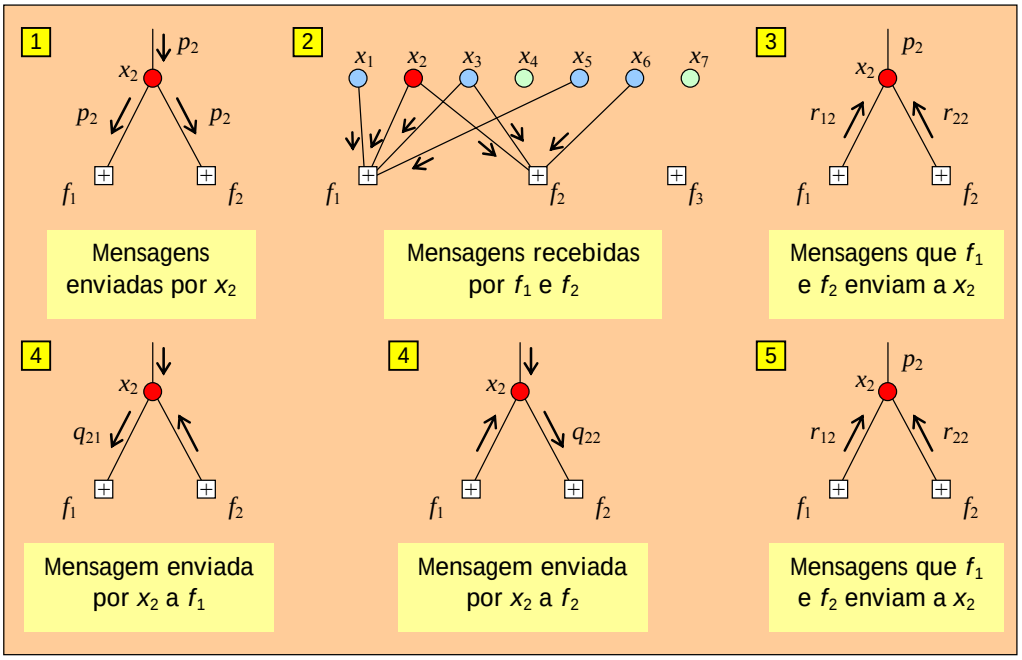


Fig. 11 Exemplo de transferência de mensagens entre os nós de grafos bipartidos.

1. No início a variável x_2 espalha pelos nós de paridade f_1 e f_2 a mensagem a priori (isto é, a estimativa de probabilidade) que recebeu do canal, p_2.

2. Mas o nó f_1 também recebeu estimativas de x_1, x_3 e x_5 e o nó f_2 também recebeu estimativas de x_3 e x_6.

3. Cada um dos nós f_1 e f_2 processa as mensagens que **não** vieram de x_2 e envia a esta o resultado (r_{12} e r_{22}, respectivamente).

4. O processo agora repete-se com as mensagens a circular das variáveis para os nós de paridade e destes para as variáveis: por exemplo e voltando à variável x_2, esta envia a f_1 uma mensagem q_{21} baseada no que **os outros** nós de paridade, p_2 e f_2, lhe enviaram[9], e envia a f_2 uma outra mensagem, q_{22}, esta baseada no que recebeu de p_2 e f_1.

5. Os nós f_1 e f_2 enviam a x_2 novas mensagens mais refinadas.

[9] Note-se que cada nó de paridade p_i envia a x_i sempre a mesma mensagem em todas as iterações.

A Fig. 12 mostra, por adaptação da matriz **H** da Fig. 3, quais as mensagens envolvidas num dado cálculo [13].

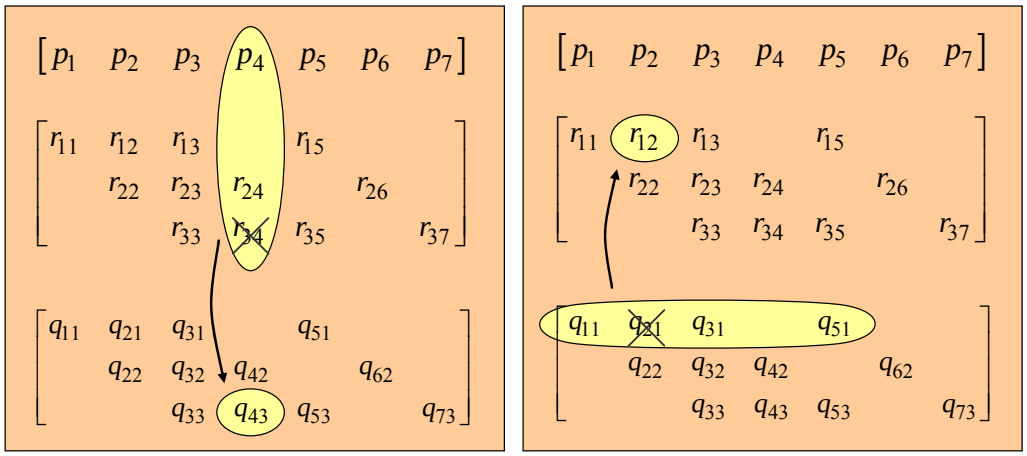


Fig. 12 Indicação das mensagens envolvidas nos cálculos através da adaptação da matriz **H**.

Convém notar que termos como "mensagens", "probabilidades" e "estimativas de probabilidades" são usados praticamente como sinónimos, consoante o contexto.

3.1. Regras de actualização de mensagens probabilísticas

No algoritmo da soma-e-produto as mensagens são calculadas de acordo com as seguintes expressões:

Mensagens com variáveis genéricas

- **Do nó de variável x_i para o nó de paridade f_j:**

$$q_{ij}(x_i) = K_{ij} \prod_{j' \neq j} r_{j'i}(x_i) \tag{4}$$

- **Do nó de paridade f_j para o nó de variável x_i:**

$$r_{ji}(x_i) = \sum_{i' \neq i} \left[f_j(x_1, x_2, \ldots) \prod_{i' \neq i} q_{i'j}(x_i) \right] \tag{5}$$

K_{ij} é um factor de normalização com valor tal que $q_{ij}(0) + q_{ij}(1) + \ldots q_{ij}(M-1) = 1$ e a função de várias variáveis $f_j(x_1,x_2,\ldots)$ é, como se viu, uma função indicadora de pertença que só toma dois valores, 0 e 1. Na Eq. (5) as somas e os produtos justificam o nome do algoritmo SP.

A Eq. (4) diz-nos que a mensagem que uma variável transmite **a um** dos nós de paridade a que está ligada é igual ao produto, devidamente normalizado, das mensagens que acabou de receber **dos outros** nós de paridade (incluindo o canal). A Eq. (5) diz-nos que a mensagem que um nó de paridade envia **a um** dos nós de variáveis a que está ligado é igual à soma dos produtos das mensagens que o nó acabou de receber **das outras** variáveis e que satisfazem a equação de paridade. As mensagens que circulam

em ambos os sentidos são mensagens exteriores ou *extrínsecas* à variável em causa, e isto é muito importante.

A estimativa final da probabilidade a posteriori $p(x_i|\mathbf{y})$ produzida por cada nó de variável x_i é igual ao produto normalizado de todas as mensagens recebidas dos nós de paridade, incluindo a mensagem a priori do canal:

$$p(x_i \mid \mathbf{y}) = K_i \prod_j r_{ji}(x_i)$$

Tal como antes, o factor de normalização K_i tem um valor tal que $\sum_{k=0}^{M-1} p(x_i = k \mid \mathbf{y}) = 1$.

3.1.1. Mensagens com variáveis binárias

Se as variáveis forem binárias as expressões das mensagens tornam-se mais simples, especialmente as enviadas pelos nós de paridade:

Mensagens com variáveis binárias

- **Do nó de variável x_i para o nó de paridade f_j:**

$$\begin{aligned} q_{ij}(0) &= K_{ij}(1-p_i)\prod_{j'\neq j} r_{j'i}(0) \\ q_{ij}(1) &= K_{ij}\,p_i \prod_{j'\neq j} r_{j'i}(1) \end{aligned} \qquad (6)$$

- **Do nó de paridade f_j para o nó de variável x_i:**

$$\begin{aligned} r_{ji}(0) &= \frac{1}{2}\left\{1+\prod_{i'\neq i}\left[1-2q_{i'j}(1)\right]\right\} \\ r_{ji}(1) &= 1-r_{ji}(0) \end{aligned} \qquad (7)$$

O factor K_{ij} é tal que $q_{ij}(0) + q_{ij}(1) = 1$. Em cima tornou-se explícita a presença de $p(y_i|x_i = 1) = p_i$ e $p(y_i|x_i = 0) = 1 - p_i$ (o que não acontecia na Eq. (4)). Alternativamente as mensagens dos nós de paridade podem ser escritas numa forma mais compacta como

$$r_{ji}(x_i) = \frac{1}{2}\left\{1+(-1)^{x_i}\prod_{i'\neq i}\left[1-2q_{i'j}(1)\right]\right\}$$

Como $q_{ij}(0) + q_{ij}(1) = 1$ então

$$K_{ij}(1-p_i)\prod_{j'\neq j} r_{j'i}(0) + K_{ij}\,p_i \prod_{j'\neq j} r_{j'i}(1) = 1$$

donde se tira que

$$K_{ij} = \frac{1}{p_i \prod_{j' \neq j} r_{j'i}(1) + (1-p_i) \prod_{j' \neq j} (1 - r_{j'i}(1))} \tag{8}$$

Daqui para a frente consideraremos apenas variáveis binárias e mensagens de um único elemento[10] com a estimativa de $p(x_i = 1|\mathbf{y})$. Isso permite-nos simplificar a notação mais um pouco se escrevermos r_{ji} e q_{ij} em vez de $r_{ji}(1)$ e $q_{ij}(1)$ e se deixarmos de usar $r_{ji}(0)$ e $q_{ij}(0)$ por serem redundantes devido à normalização. Assim passaremos a usar as seguintes equações de actualização:

Mensagens com variáveis binárias (expressões mais simples)

- **Do nó de variável x_i para o nó de paridade f_j:**

$$q_{ij} = K_{ij} p_i \prod_{j' \neq j} r_{j'i} \quad (\text{com } K_{ij} = \frac{1}{p_i \prod_{j' \neq j} r_{j'i} + (1-p_i) \prod_{j' \neq j} (1 - r_{j'i})}) \tag{9}$$

- **Do nó de paridade f_j para o nó de variável binária x_i (expressão mais simples):**

$$1 - 2r_{ji} = \prod_{i' \neq i} (1 - 2q_{i'j}) \quad \text{ou} \quad r_{ji} = \frac{1}{2}\left[1 - \prod_{i' \neq i} (1 - 2q_{i'j})\right] \tag{10}$$

Exemplo 3

Consideremos os nós de grau 4 da Fig. 13. O nó de variável v_3 envia ao nó de paridade f_1 a mensagem

$$q_{31} = K_{31} p_3 r_{33} r_{43} \qquad (\text{sendo } K_{31} = \frac{1}{p_3 r_{33} r_{43} + (1-p_3)(1-r_{33})(1-r_{43})}).$$

As mensagens para os outros nós de paridades seriam calculadas da mesma maneira:

$$q_{33} = \frac{p_3 r_{13} r_{43}}{p_3 r_{13} r_{43} + (1-p_3)(1-r_{13})(1-r_{43})}$$

$$q_{34} = \frac{p_3 r_{13} r_{33}}{p_3 r_{13} r_{33} + (1-p_3)(1-r_{13})(1-r_{33})}$$

Por sua vez o nó de paridade f_2 envia ao nó da variável v_2 a mensagem r_{22} obtida de

$$1 - 2r_{22} = (1 - 2q_{12})(1 - 2q_{32})(1 - 2q_{42}),$$

a qual também pode ser calculada pela expressão alternativa

[10] O Apêndice fica reservado para os pormenores dos cálculos com mensagens de dois ou mais valores.

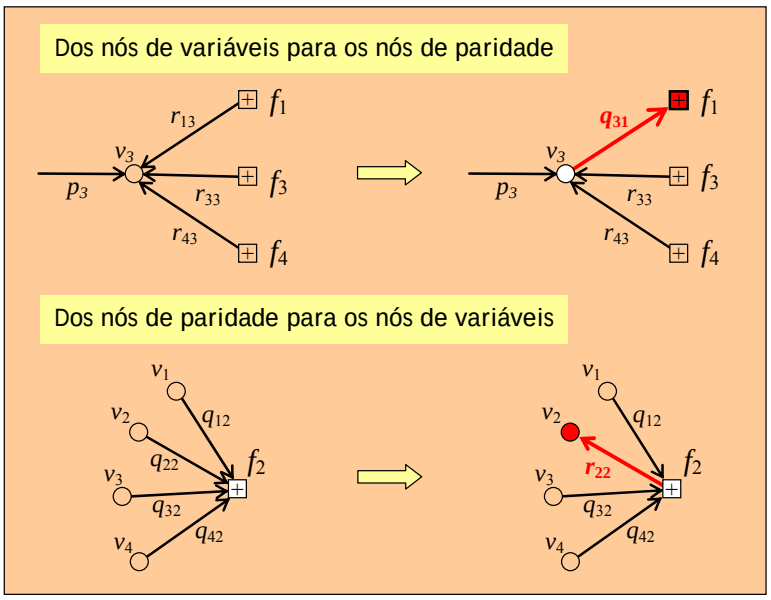


Fig. 13 Mensagens entre nós de grafos bipartidos.

$$r_{22} = q_{12}(1-q_{32})(1-q_{42}) + q_{32}(1-q_{12})(1-q_{42}) + q_{42}(1-q_{12})(1-q_{32}) + q_{12}q_{32}q_{42} \qquad (11)$$

De igual modo o mesmo nó de paridade envia para os nós das variáveis v_1, v_3 e v_4 mensagens com as probabilidades estimadas r_{21}, r_{23} e r_{24}, respectivamente, as quais se calculam de acordo com

$$1-2r_{21} = (1-2q_{22})(1-2q_{32})(1-2q_{42})$$
$$1-2r_{23} = (1-2q_{12})(1-2q_{22})(1-2q_{42})$$
$$1-2r_{24} = (1-2q_{12})(1-2q_{22})(1-2q_{32})$$

3.1.2. Por que é que é necessário um factor de normalização?

Sem normalização as estimativas de probabilidades que as variáveis enviariam aos nós de paridade seriam sempre inferiores às que deles tinham recebido dado estas serem sempre inferiores a um. Por exemplo, se uma variável de grau 4 recebesse os valores 0,6, 0,9 e 0,8 de três nós de paridade a estimativa que iria enviar para o quarto nó seria, sem normalização, igual ao produto $0,6 \times 0,9 \times 0,8 = 0,432$.

Suponhamos um caso simples de uma variável x_1 de grau 3 que recebe as mensagens r_{11} e r_{21} e vai disso informar o nó de paridade 3 enviando-lhe a mensagem

$$q_{13} = K_{13} r_{11} r_{21}.$$

Ora deverá também ser $1-q_{13} = K_{13}(1-r_{11})(1-r_{21})$. Combinando as duas expressões concluímos que o factor de normalização a usar é

$$K_{13} = \frac{1}{r_{11}r_{21} + (1-r_{11})(1-r_{21})},$$

como já sabíamos e está na Eq. (8). No exemplo precedente a mensagem da variável de grau 4 deveria, portanto, ter valor

$$\frac{0,6\times 0,9\times 0,8}{0,6\times 0,9\times 0,8+0,4\times 0,1\times 0,2}=0,98\,,$$

quantidade que é muito mais razoável que a não normalizada 0,432.

Exemplo 4

Mensagens de nós de um dado grau expressas à custa de mensagens de "graus" inferiores

As mensagens dos nós de um dado grau podem ser expressas à custa de mensagens de nós de grau inferior. Comecemos pela mensagem de um nó de variável de grau 3 (ver Fig. 14a). Se tiver recebido as mensagens *a* e *b* de dois nós de paridade, o nó de variável envia para o terceiro nó de paridade uma mensagem que é função de *a* e *b* (e a que chamaremos VAR(*a*,*b*)):

Grau 3: $$VAR(a,b)=\frac{ab}{ab+(1-a)(1-b)}\,.$$

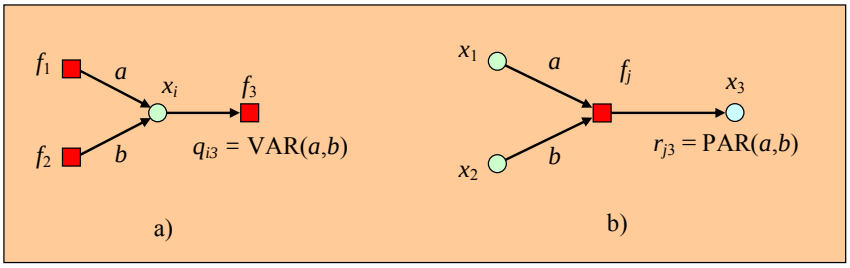


Fig. 14 As funções VAR e PAR em nós de grau 3.

Consideremos agora um nó de variável de grau 4 ao qual chegam as mensagens *a*, *b* e *c*. A mensagem que ele transmite para o quarto nó de paridade é

$$VAR(a,b,c)=\frac{abc}{abc+(1-a)(1-b)(1-c)}\,.$$

Dividindo o numerador e o denominador por $ab+(1-a)(1-b)$ e fazendo $\alpha=\frac{ab}{ab+(1-a)(1-b)}$ (e, portanto, $1-\alpha=\frac{(1-a)(1-b)}{ab+(1-a)(1-b)}$) obtemos

$$VAR(a,b,c)=\frac{\alpha c}{\alpha c+(1-\alpha)(1-c)}=VAR(\alpha,c)\,.$$

Mas $\alpha = VAR(a,b)$, ou seja, $VAR(a,b,c) = VAR(c,VAR(a,b))$. Logo, estamos a exprimir a mensagem de "grau" 4 à custa da mensagem de "grau" 3. Generalizando para nós de grau *n*:

Grau n: $VAR(f_1, f_2, \ldots, f_{n-1}) = VAR(f_1, VAR(f_2, \ldots, f_{n-1}))$

Quanto aos nós de paridade passa-se algo de semelhante:

Grau 3: $PAR(a,b) = a(1-b) + b(1-a)$

Grau n: $PAR(x_1, x_2, \ldots, x_{n-1}) = PAR(x_1, PAR(x_2, \ldots, x_{n-1}))$

Vejamos como se chega a estas conclusões. Em nós de grau 3 (ver Fig. 14b) a mensagem (isto é, a função PAR(*a*,*b*)) é obtida através da Eq. (10):

$$PAR(a,b) = \frac{1}{2}\left[1-(1-2a)(1-2b)\right] = a - ab + b - ab = \\ = a(1-b) + b(1-a)$$

E se o nó de paridade tiver, por exemplo, grau 4 (ver Fig. 15)? A mensagem que envia é

$$PAR(a,b,c) = \frac{1}{2}\left[1-(1-2a)(1-2b)(1-2c)\right].$$

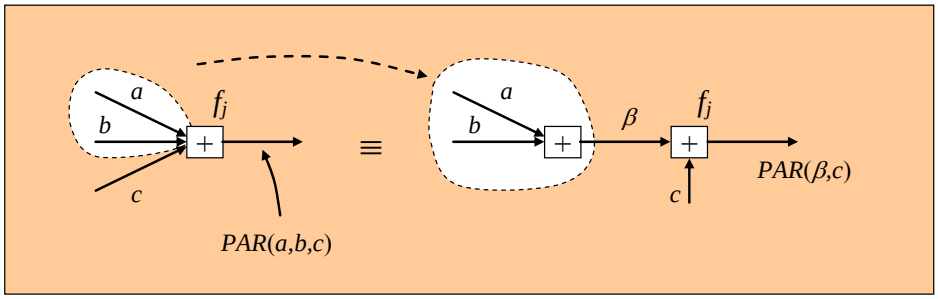


Fig. 15 A função PAR em nós de paridade de grau 4.

Mas, fazendo $(1-2a)(1-2b) = 1-2\beta$ (isto é, $\beta = PAR(a,b)$), então

$$PAR(a,b,c) = \frac{1}{2}\left[1-(1-2\beta)(1-2c)\right] = PAR(\beta,c) = \\ = PAR(c, PAR(a,b))$$

Mais uma vez conseguimos exprimir a mensagem de um dado "grau" à custa da mensagem de um "grau" inferior.

3.1.3. Simplificação: o algoritmo "max-product"

Regressemos ao nó de paridade de grau 3 da Fig. 14b. A mensagem transmitida resulta de um produto (cf. Eq. (10)) que, como se viu, vale $r_{j3} = PAR(a,b) = a(1-b)+b(1-a)$. Uma simplificação do algoritmo substitui este cálculo pelo máximo das duas parcelas: $r_{j3} \approx \max\big(a(1-b), b(1-a)\big)$. Com nós de paridade de grau superior as multiplicações da Eq. (10) são simplificadas de maneira idêntica. Como? Lembrando-nos da Eq. (11) tomamos apenas o valor máximo dos diversos produtos envolvidos,

$$r_{ji} \approx \max\big(a(1-b)(1-c)\ldots, b(1-a)(1-c)\ldots, c(1-a)(1-b)\ldots, abc\ldots\big). \qquad (12)$$

A este algoritmo simplificado, em que as mensagens dos nós de variáveis resultam de um produto (Eq. (9)) e as mensagens dos nós de paridade são um máximo, dá-se o nome de… algoritmo "max-product", evidentemente.

3.1.4. Ciclos e aproximações

As equações de actualização de mensagens apresentadas produzem estimativas exactas das probabilidades a posteriori ao fim da primeira (e única) iteração se o grafo de Tanner não contiver ciclos [7]. Em grafos com ciclos, uma situação muito vulgar, o procedimento de cálculo é iterativo e as estimativas das probabilidades deixam de ser exactas para passarem a ser apenas aproximações, que são tanto melhores quanto mais compridos forem os ciclos. A convergência do algoritmo deixa de estar garantida mas mesmo assim ele funciona muito bem (admiravelmente bem, até…).

O código da Fig. 3 contém ciclos: por exemplo, $x_2 \to f_1 \to x_3 \to f_2 \to x_2$ é um ciclo de comprimento 4 e $x_2 \to f_1 \to x_3 \to f_3 \to x_4 \to f_2 \to x_2$ é um ciclo de comprimento 6. Como em grafos bipartidos os nós de uma dada classe (variáveis ou paridade) não estão ligados entre si, os ciclos, se existirem, têm sempre comprimento par (no mínimo 4).

3.2. Regras de actualização de mensagens logarítmicas

Até agora considerámos que as mensagens representam estimativas de probabilidades. Porém, quando as variáveis são binárias muitas vezes é preferível fazer circular mensagens que representem quocientes de probabilidades ou o seu logaritmo (LLR). Nesse caso as equações de actualização de mensagens (Eqs. (6) e (7)) terão de ser adequadamente modificadas. Consideremos então as seguintes definições:

$$\ln\frac{p(y_i \mid x_i = 1)}{p(y_i \mid x_i = 0)} = \ln\frac{p_i}{1-p_i} = L(p_i) \quad \text{(canais binários simétricos)}$$

$$L(r_{ij}) = \ln\frac{r_{ij}(1)}{r_{ij}(0)} = \ln\frac{r_{ij}}{1-r_{ij}}$$

$$L(q_{ij}) = \ln\frac{q_{ij}(1)}{q_{ij}(0)} = \ln\frac{q_{ij}}{1-q_{ij}}$$

Partindo da Eq. (6) facilmente obtemos a expressão das mensagens emitidas pelos nós de variáveis:

$$L(q_{ij}) = \ln \frac{K_{ij}\, p_i \prod_{j' \neq j} r_{j'i}(1)}{K_{ij}(1-p_i) \prod_{j' \neq j} r_{j'i}(0)} =$$

$$= \ln \frac{p_i}{1-p_i} + \sum_{j' \neq j} \ln \frac{r_{j'i}(1)}{r_{j'i}(0)}$$

Ou seja, $L(q_{ij}) = L(p_i) + \sum_{j' \neq j} L(r_{j'i})$. Substituímos, portanto, multiplicações por somas, o que é bastante mais simples. Quanto à mensagem dos nós de paridade não é tão fácil chegar à expressão final. Começamos com a Eq. (7) e escrevemos

$$L(r_{ji}) = \ln \frac{r_{ji}(1)}{r_{ji}(0)} = \ln \frac{1 - \prod_{i' \neq i} \left(1 - 2q_{i'j}\right)}{1 + \prod_{i' \neq i} \left(1 - 2q_{i'j}\right)}.$$

Neste ponto temos de introduzir a função tangente hiperbólica $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$ (quem diria?!), a partir da qual podemos obter as seguintes igualdades:

$$\tanh(x/2) = \frac{e^x - 1}{e^x + 1} \qquad\qquad \tanh^{-1}(x) = \frac{1}{2} \ln \frac{1+x}{1-x}$$

Agora, sendo por definição $L(q_{ij}) = \ln \dfrac{q_{ij}}{1-q_{ij}}$ então também é $q_{ij} = \dfrac{e^{L(q_{ij})}}{1 + e^{L(q_{ij})}}$, donde se tira que

$$1 - 2q_{ij} = 1 - 2\frac{e^{L(q_{ij})}}{1 + e^{L(q_{ij})}} =$$

$$= \frac{1 - e^{L(q_{ij})}}{1 + e^{L(q_{ij})}} = -\tanh\left(\frac{L(q_{ij})}{2}\right)$$

Como a tangente hiperbólica é uma função ímpar então também podemos escrever

$$1 - 2q_{ij} = \tanh\left(-\frac{L(q_{ij})}{2}\right) =$$

$$= \Phi\left(L(q_{ij})\right)$$

onde definimos a função auxiliar $\Phi(x) = \tanh\left(-x/2\right)$.

Estamos prontos para retomar a expressão da LLR $L(r_{ji})$ e obter finalmente

$$L(r_{ji}) = \ln \frac{1-\prod_{i'\neq i}\left(1-2q_{i'j}\right)}{1+\prod_{i'\neq i}\left(1-2q_{i'j}\right)} = \ln \frac{1-\prod_{i'\neq i}\tanh\left(-L(q_{i'j})/2\right)}{1+\prod_{i'\neq i}\tanh\left(-L(q_{i'j})/2\right)} =$$

$$= 2\tanh^{-1}\left(-\prod_{i'\neq i}\tanh\left(-L(q_{i'j})/2\right)\right)$$

e também a expressão equivalente $L(r_{ji}) = \Phi^{-1}\left(\prod_{i'\neq i}\Phi\left(L(q_{i'j})\right)\right)$. Em resumo:

Mensagens logarítmicas

- **Do nó de variável x_i para o nó de paridade f_j:**

$$L(q_{ij}) = L(p_i) + \sum_{j'\neq j} L(r_{j'i}) \tag{13}$$

- **Do nó de paridade f_j para o nó de variável x_i:**

$$L(r_{ji}) = 2\tanh^{-1}\left[-\prod_{i'\neq i}\tanh\left(-L(q_{i'j})/2\right)\right] =$$
$$= \Phi^{-1}\left[\prod_{i'\neq i}\Phi\left(L(q_{i'j})\right)\right] \tag{14}$$

Nota 1: A Eq. (14) pode escrever-se de uma maneira mais elegante e até mais memorizável:

$$\Phi\left(L(r_{ji})\right) = \prod_{i'\neq i}\Phi\left(L(q_{i'j})\right)$$

Nota 2: Se tivéssemos definido $L(x) = \ln\frac{1-x}{x}$ em vez de $L(x) = \ln\frac{x}{1-x}$ a Eq. (14) escrever-se-ia com mais simplicidade:

$$L(r_{ji}) = 2\tanh^{-1}\left[\prod_{i'\neq i}\tanh\left(L(q_{i'j})/2\right)\right] \qquad \text{ou} \qquad \tanh\left(L(r_{ji})/2\right) = \prod_{i'\neq i}\tanh\left(L(q_{i'j})/2\right).$$

No fim de todas as iterações calcula-se a LLR final somando todas as LLR que confluem na variável:

$$L(x_i|\mathbf{y}) = L(q_i) =$$
$$= L(p_i) + \sum_j L(r_{ji})$$

A expressão anterior mostra que a LLR a posteriori é igual à soma de uma componente fixa intrínseca à variável, $L(p_i)$, com uma componente que lhe é extrínseca e resultou da transferência de mensagens, $\sum_j L(r_{ji})$.

3.2.1. Canais com ruído AWGN

As expressões anteriores são válidas para outros canais além daquele que temos considerado até aqui (o canal binário simétrico): há só que usar o valor adequado da LLR a priori $\ln\frac{p(y_i \mid x_i = 1)}{p(y_i \mid x_i = 0)}$ ou $\ln\frac{p(y_i \mid x_i = +1)}{p(y_i \mid x_i = -1)}$. Assim, se se enviar uma sequência binária $x_i = \pm 1$ através de um canal AWGN com ruído de média nula e variância σ^2 e se se receber a sequência de valores reais y_i, a LLR a priori , isto é, aquilo de que necessitamos para iniciar os cálculos, vale

$$\begin{aligned} L(p_i) &= \ln\frac{p(y_i \mid x_i = +1)}{p(y_i \mid x_i = -1)} = \\ &= \ln\frac{\left(1/\sqrt{2\pi\sigma^2}\right)\exp\left[-(y_i - 1)^2/2\sigma^2\right]}{\left(1/\sqrt{2\pi\sigma^2}\right)\exp\left[-(y_i + 1)^2/2\sigma^2\right]} = \\ &= \frac{2y_i}{\sigma^2} \end{aligned}$$

O ruído à saída de um filtro adaptado no receptor tem variância $\sigma^2 = \frac{N_0}{2E_c}$, em que $N_0/2$ é a densidade espectral de potência do ruído e $E_c = R_c E_b$ é a energia de cada bit codificado (sendo E_b a energia de cada bit antes da codificação e $R_c = k/n$ a taxa do código). Portanto,

$$L(p_i) = 4\frac{E_c}{N_0}y_i = 4\frac{R_c E_b}{N_0}y_i .$$

Agora é só usar este valor nas equações de actualização.

3.2.2. Simplificações: o algoritmo "min-sum"

As razões LLR são usadas para simplificar cálculos, nomeadamente para substituir multiplicações por somas. Ora, se bem que tenhamos substituído multiplicações por adições no cálculo das mensagens de variáveis, as multiplicações subsistem no cálculo das mensagens dos nós de paridade. Porém, não por muito tempo! Assim, definindo uma nova função auxiliar, $\Psi(x) = -\ln\left[\tanh\left(x/2\right)\right]$, $x > 0$ (ver Fig. 16), e sabendo que qualquer número real y é igual ao produto do seu módulo pelo seu sinal, $y = \mathrm{sgn}(y)|y|$, a Eq. (14) pode escrever-se

$$L(r_{ji}) = (-1)^{d_j}\left(\prod_{i' \neq i}\mathrm{sgn}\left[L(q_{i'j})\right]\right)\Psi\left(\sum_{i' \neq i}\Psi\left(\left|L(q_{i'j})\right|\right)\right) \qquad (15)$$

em que d_j é o grau do nó. Acontece que, como se vê na Fig. 16, Ψ é uma função positiva fortemente decrescente pelo que o somatório das funções Ψ é aproximadamente igual ao termo dominante, correspondente ao menor dos $|L(q_{ji})|$ envolvidos. Além disso, a

função Ψ é igual à sua inversa, $\Psi^{-1}(x) = \Psi(x)$, isto é, $\Psi(\Psi(x)) = x$. Logo, a expressão acima simplifica-se em

$$L(r_{ji}) \approx (-1)^{d_j} \left(\prod_{i' \neq i} \operatorname{sgn}\left[L(q_{i'j}) \right] \right) \Psi\left(\max_{i' \neq i} \left(\Psi\left(\left| L(q_{i'j}) \right| \right) \right) \right) =$$

$$= (-1)^{d_j} \left(\prod_{i' \neq i} \operatorname{sgn}\left[L(q_{i'j}) \right] \right) \min_{i' \neq i} \left(\left| L(q_{i'j}) \right| \right)$$

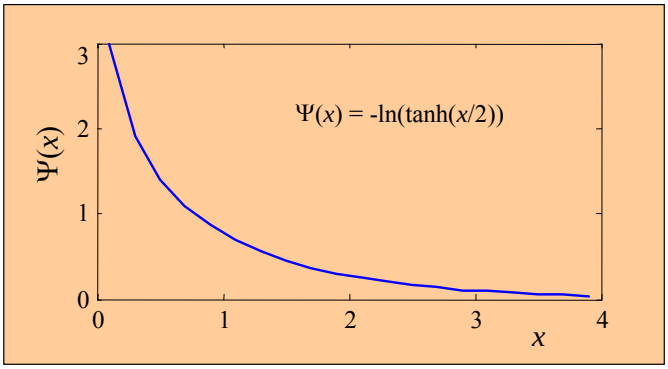


Fig. 16 A função auxiliar $\Psi(x) = -\ln\left[\tanh\left(x/2\right)\right]$.

A esta forma aproximada (mas bastante mais simples) de cálculo dá-se o nome de… "min-sum algorithm":

Algoritmo "min-sum"

- **Do nó de variável x_i para o nó de paridade f_j:**

$$L(q_{ij}) = L(p_i) + \sum_{j' \neq j} L(r_{j'i}) \tag{16}$$

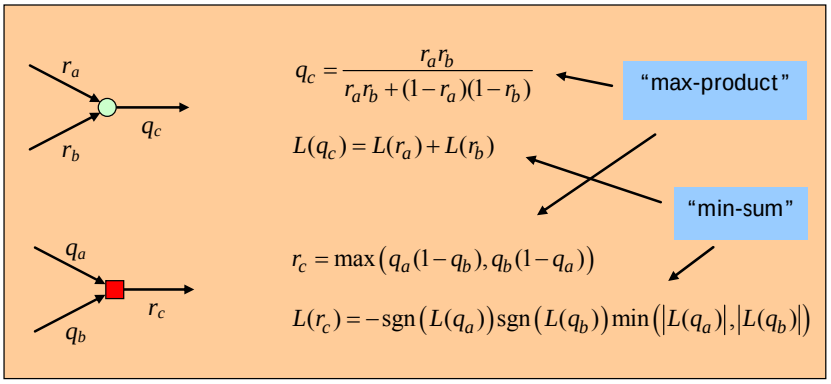
- **Do nó de paridade f_j para o nó de variável x_i:**

$$L(r_{ji}) = (-1)^{d_j} \left(\prod_{i' \neq i} \operatorname{sgn}\left[L(q_{i'j}) \right] \right) \min_{i' \neq i} \left(\left| L(q_{i'j}) \right| \right) \tag{17}$$

Quer este algoritmo quer o algoritmo "max-product" só diferem do algoritmo SP "normal" no modo de cálculo, aproximado, das mensagens dos nós de paridade. As mensagens dos nós de variáveis, essas, são calculadas da maneira "normal".

Exemplo 5

A figura seguinte apresenta as expressões de actualização de nós com os algoritmos simplificados "max-product" (com probabilidades) e "min-sum" (com LLRs) no caso de nós de grau 3.



4. Exemplo numérico de aplicação

Vamos então a um exemplo numérico considerando um canal binário simétrico (BSC) e o codificador[11] (7,4) da Fig. 3. A compreensão do exemplo tornará fácil a aplicação do algoritmo da soma-e-produto a outros tipos de canal, como o canal de ruído AWGN.

Suponhamos que a palavra de código **x** = 1 1 0 0 0 1 0 foi transmitida através de um canal BSC com $p = 0{,}1$, que a transformou na sequência **y** = 1 1 0 1 0 1 0. Ou seja, o quarto bit está errado (mas o descodificador ainda não sabe!…). Como é que poderemos corrigir o bit errado usando transferência de mensagens ao longo das linhas do grafo de Tanner do codificador?

Como se mostra na Fig. 17, cada nó-folha p_i envia do canal para o seu argumento x_i uma mensagem fixa igual ao valor da respectiva probabilidade a priori p_i:

$$p(y_i \mid x_i = 1) = p_i \,.$$

Por exemplo, x_1 recebe a informação a priori $p(y_1 = 1 \mid x_1 = 1) = 0{,}9$ e x_3 recebe $p(y_3 = 0 \mid x_3 = 1) = 0{,}1$.

Por sua vez cada variável x_i difunde a mensagem que recebeu do canal pelos nós de paridade aos quais está ligada. Assim, por exemplo,

$$q_{21} = q_{22} = \\ = p_2 = 0{,}9 \qquad\qquad q_{51} = q_{53} = \\ = p_5 = 0{,}1$$

A Fig. 17 mostra os valores das primeiras mensagens recebidas pelos nós de paridade. Que mensagens vão eles agora enviar aos nós de variáveis? Consideremos f_1, por exemplo. Este nó recebeu estimativas de x_1, x_2, x_3 e x_5. Então para x_1 enviará uma estimativa baseada nas estimativas que recebeu das outras variáveis (x_2, x_3 e x_5), para x_2 enviará uma estimativa que se baseia nas que recebeu de x_1, x_3 e x_5, e para x_3 e x_5 fará de modo idêntico. Em particular, a mensagem que x_1 recebe de f_1 é dada, de acordo com a Eq. (10), por

[11] Este código não é, de facto, um código LDPC mas para o caso é irrelevante.

$$r_{11} = \frac{1}{2}\left[1 - \prod_{i\in\{2,3,5\}} (1-2q_{i1})\right] =$$
$$= \frac{1}{2}\left[1-(1-2q_{21})(1-2q_{31})(1-2q_{51})\right] =$$
$$= 0,756$$

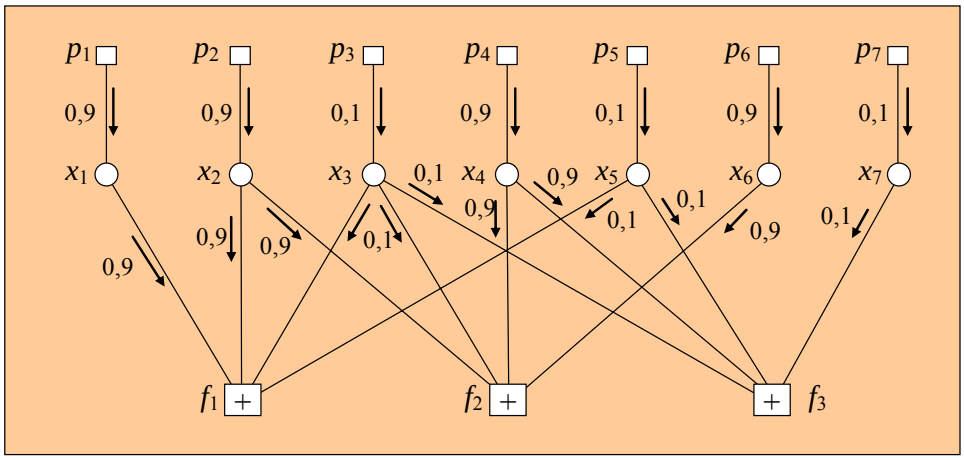


Fig. 17 Mensagens iniciais enviadas pelas variáveis aos nós de paridade f_j.

Identicamente obteríamos as mensagens de f_1 para as suas restantes variáveis,

$$r_{12} = 0,756 \qquad r_{13} = 0,244 \qquad r_{15} = 0,244,$$

e as mensagens de f_2 e f_3 também,

$$r_{22} = 0,244 \qquad r_{23} = 0,756 \qquad r_{24} = 0,244 \qquad r_{26} = 0,244$$

$$r_{33} = 0,756 \qquad r_{34} = 0,244 \qquad r_{35} = 0,756 \qquad r_{37} = 0,756,$$

como a Fig. 18 mostra. Em seguida cada variável envia aos nós de paridade o produto normalizado das mensagens que recebeu, como se ilustra na Fig. 19 para a variável x_2:

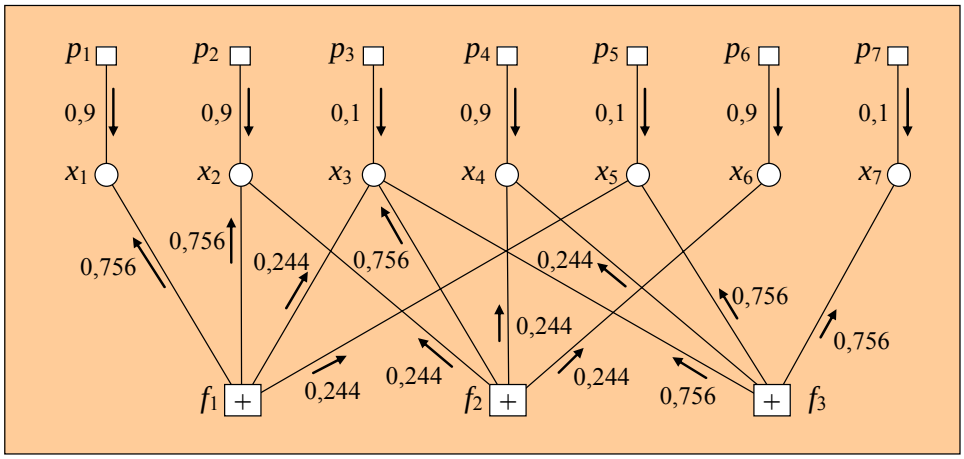


Fig. 18 Primeiras mensagens enviadas pelos nós de paridade aos nós de variáveis.

$$q_{21} = K_{21} p_2 r_{22} = \frac{p_2 r_{22}}{p_2 r_{22} + (1-p_2)(1-r_{22})} = \frac{0,9 \times 0,244}{0,2952} = 0,744$$

$$q_{22} = K_{22} p_2 r_{12} = \frac{p_2 r_{12}}{p_2 r_{12} + (1-p_2)(1-r_{12})} = \frac{0,680}{0,705} = 0,965$$

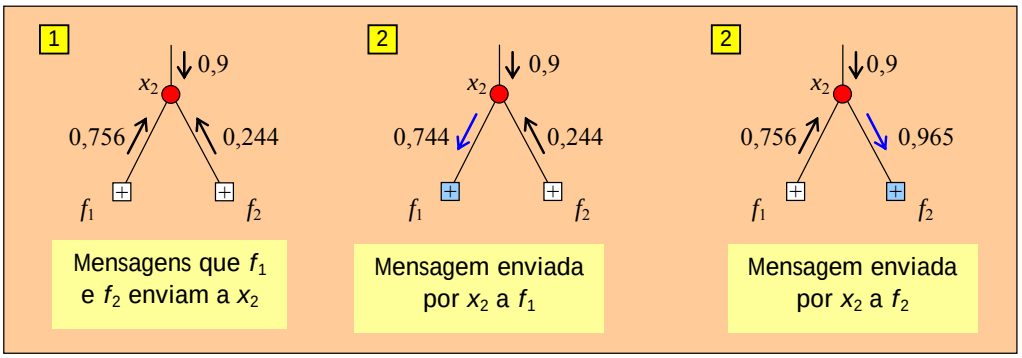


Fig. 19 Início da 2ª iteração: novas mensagens enviadas por x_2 aos nós de paridade f_1 e f_2.

No fim deste vai-e-vem de transferências a estimativa da probabilidade que cada nó de variável produz é igual ao produto normalizado de todas as estimativas que a ele confluem, incluindo a proveniente do canal. Embora só os valores finais interessem, a Tabela **1** mostra os valores da LLR a posteriori que foram sendo obtidos ao longo de cinco iterações, para nos apercebermos da sua evolução.

Tabela 1 – LLR a posteriori durante cinco iterações

Iteração nº	$L(x_1\|\mathbf{y})$	$L(x_2\|\mathbf{y})$	$L(x_3\|\mathbf{y})$	$L(x_4\|\mathbf{y})$	$L(x_5\|\mathbf{y})$	$L(x_6\|\mathbf{y})$	$L(x_7\|\mathbf{y})$
1	3,33	2,20	-1,07	-0,06	-2,20	1,07	-1,07
2	2,18	1,53	-1,06	-0,55	-1,53	1,44	-1,44
3	2,47	2,08	-1,89	0,21	-2,08	1,74	-1,74
4	2,73	2,06	-1,48	-0,39	-2,06	1,39	-1,39
5	2,44	1,95	-1,59	-0,42	-1,95	1,57	-1,57

Como é que estes valores foram obtidos? Vejamos um exemplo retomando a variável x_2: tendo as mensagens que lhe chegaram ao fim da primeira iteração sido $p_2 = 0,9$, $r_{12} = 0,756$ e $r_{22} = 0,244$ (ver Fig. 19) a estimativa $p(x_2 = 1|\mathbf{y})$ ao fim dessa iteração é igual ao produto normalizado de todas,

$$p(x_2 = 1 \,|\, \mathbf{y}) = \frac{0,9 \times 0,756 \times 0,244}{0,9 \times 0,756 \times 0,244 + 0,1 \times 0,244 \times 0,756} = 0,90$$

o que se traduz numa LLR igual a $L(x_2 \,|\, \mathbf{y}) = \ln \frac{p(x_2 = 1 \,|\, \mathbf{y})}{p(x_2 = 0 \,|\, \mathbf{y})} = \ln \frac{0,9}{0,1} = 2,20$, como está na Tabela, a que corresponderia uma estimativa $\hat{x}_2 = 1$ por $L(x_2|\mathbf{y})$ ser positiva.

Ao fim de 10 iterações a situação seria a retratada na Fig. 20. Nota-se bem que nos bastaria ter considerado apenas cinco iterações dada a constância de valores das LLR daí para a frente.

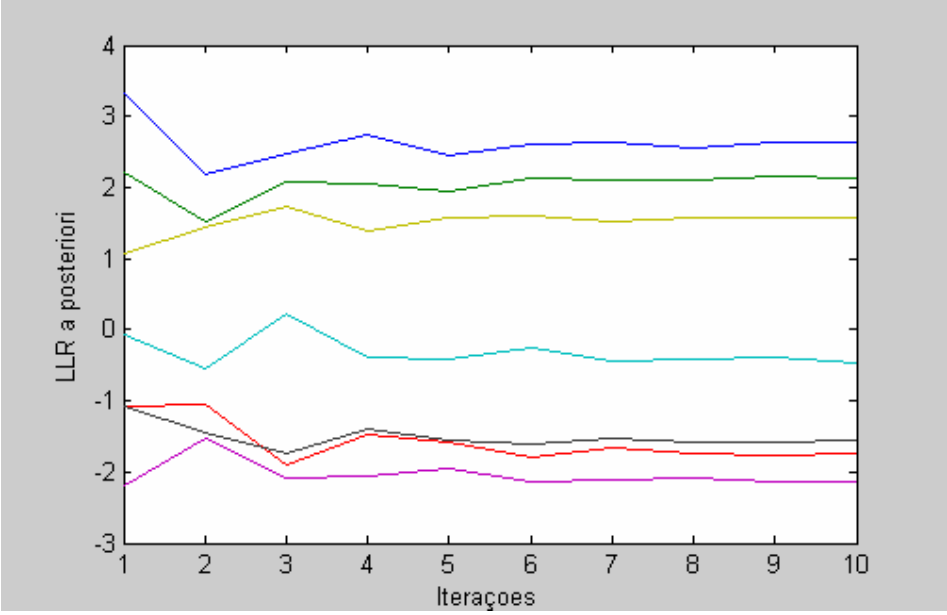


Fig. 20 As LLR a posteriori ao longo de 10 iterações.

A estimativa da sequência binária transmitida seria $\hat{\mathbf{x}} = \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_7 = 1100010$, exactamente igual a $\mathbf{x}$. A sequência original foi correctamente estimada, como queríamos…

Exemplo 6

O nó de paridade f_3 enviou a mensagem $r_{33} = 0{,}756$ ao nó de variável x_3 depois de ter recebido $q_{43} = 0{,}9$, $q_{53} = 0{,}1$ e $q_{73} = 0{,}1$. Qual é o valor aproximado de r_{33} calculado com o algoritmo "max-product"? É (cf. Eq. (12))

$$\begin{aligned} r_{33} &\approx \max\left[q_{43}(1-q_{53})(1-q_{73}), q_{53}(1-q_{43})(1-q_{73}), q_{73}(1-q_{43})(1-q_{53}), q_{43}q_{53}q_{73}\right] = \\ &= \max\left[0{,}9^3; 0{,}1^2 \times 0{,}9; 0{,}1^2 \times 0{,}9; 0{,}9 \times 0{,}1^2\right] = \\ &= 0{,}729 \end{aligned}$$

Exemplo 7

Mensagens logarítmicas

Naturalmente que a descodificação anterior poderia ter sido feita com mensagens LLR em vez de mensagens de probabilidades. Eis agora um pequeno exemplo de cálculo de mensagens LLR.

Vamos supor que já chegámos ao fim da primeira iteração, quando todas as variáveis acabaram de receber as mensagens LLR provenientes dos nós de paridade. Desta vez comecemos por colocar essas mensagens numa matriz de valores $L(r_{ji})$ estruturada à semelhança da Fig. 12, tal como está na Fig. 21, à esquerda. Em seguida os nós de variáveis enviam aos nós de paridade mensagens a colocar na matriz de valores $L(q_{ij})$ e que são calculadas somando LLRs de acordo com a Eq. (13). Isto completa a primeira metade da segunda iteração. Por exemplo,

$$L(q_{31}) = L(p_3) + L(r_{23}) + L(r_{33}) = \\ = -2,20 + 1,13 + 1,13 = 0,06$$

como está na parte direita da mesma figura. Na segunda metade da iteração estes valores de $L(q_{ij})$ dão origem às mensagens $L(r_{ji})$ da Fig. 22, calculadas através da Eq. (14). Por exemplo,

$$\Phi\left(L(r_{12})\right) = \Phi\left(L(q_{11})\right)\Phi\left(L(q_{31})\right)\Phi\left(L(q_{51})\right) = \\ = \Phi(2,20)\Phi(0,06)\Phi(-1,07)$$

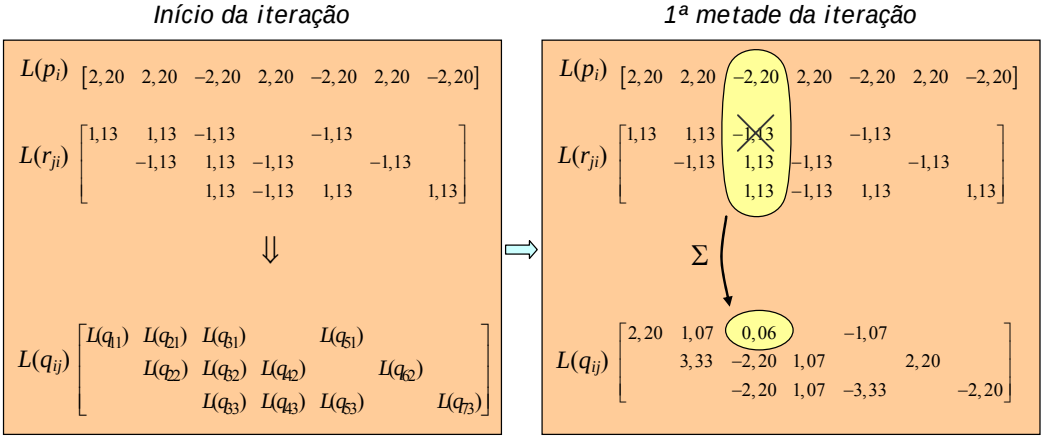


Fig. 21 Valores das mensagens logarítmicas no início da segunda iteração.

Como $\Phi(x) = \tanh(-x/2)$ então $\Phi(2,20) = -0,80$, $\Phi(0,06) = -0,03$ e $\Phi(-1,07) = 0,49$ pelo que $\Phi\left(L(r_{12})\right) = -0,08 \times (-0,03) \times 0,49 = 0,01$, donde se tira que $L(r_{12}) = \Phi^{-1}(0,01) = -0,02$. Os restantes elementos da matriz $L(r_{ji})$, apresentados na Fig. 22, à direita, são obtidos de maneira idêntica, o que completa a iteração.

Quanto valem as LLR a posteriori, por exemplo $L(x_5|\mathbf{y})$? Basta calcular

$$L(x_5|\mathbf{y}) = L(p_5) + L(r_{15}) + L(r_{35}) = \\ = -2,20 + 0,02 + 0,65 = -1,53$$

Se agora formos à terceira linha da sexta coluna da Tabela **1** o que é que encontramos? O mesmo, claro!

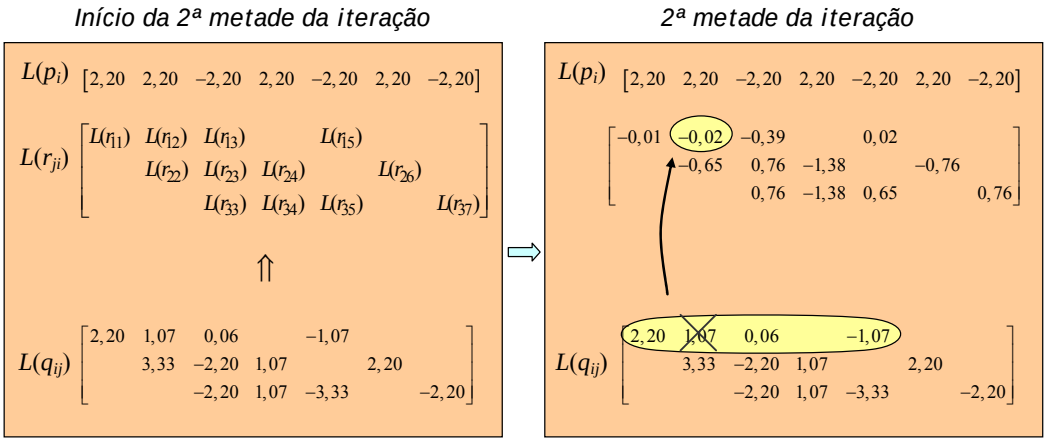


Fig. 22 Valores das mensagens logarítmicas no fim da segunda iteração.

E se usarmos o algoritmo "min-sum"? Quanto vale, por exemplo, $L(r_{12})$? De acordo com a Eq. (15) a mensagem que este nó de grau 4 envia vale

$$\begin{aligned} L(r_{12}) &= (-1)^4 \operatorname{sgn}\left[L(q_{11})\right]\operatorname{sgn}\left[L(q_{31})\right]\operatorname{sgn}\left[L(q_{51})\right]\Psi\left\{\Psi\left[\left|L(q_{11})\right|\right]+\Psi\left[\left|L(q_{31})\right|\right]+\Psi\left[\left|L(q_{51})\right|\right]\right\} = \\ &= \operatorname{sgn}(2,20)\operatorname{sgn}(0,06)\operatorname{sgn}(-1,07)\,\Psi\left[\Psi(2,20)+\Psi(0,06)+\Psi(1,07)\right] = \\ &= -\Psi(0,22+3,51+0,72) = -\Psi(4,45) = \\ &= -0,02 \end{aligned}$$

Este é o valor que consta da Fig. 22 (e tinha de constar pois a Eq. (15) é uma expressão exacta). Pelo contrário, o algoritmo "min-sum" (Eq. (17)) fornece-nos uma aproximação bastante grosseira:

$$\begin{aligned} L(r_{12}) &\approx (-1)^4 \operatorname{sgn}(2,20)\operatorname{sgn}(0,06)\operatorname{sgn}(-1,07)\min\left(2,20;0,06;1,07\right) = \\ &= -\min\left(2,20;0,06;1,07\right) = \\ &= -0,06 \end{aligned}$$

Apêndice
Exemplo numérico de mensagens vectoriais com canal BSC

Naturalmente que com mensagens vectoriais em vez de escalares são necessários mais cálculos para transferir as mensagens de uns nós para os outros. Se as variáveis forem binárias não há motivo, portanto, para não usarmos as expressões mais simples que conhecemos, as das Eqs. (9) e (10). Os dados do exemplo que se segue, com um código binário, são os mesmos do exemplo numérico atrás apresentado, porém, desta vez as mensagens vão ser calculadas como mensagens de dois elementos e não de um só. Isso é desnecessário, repete-se, mas o que se pretende é que, assim apresentado, o exemplo seja elucidativo relativamente a variáveis não binárias, especialmente no que respeita às mensagens dos nós de paridade.

Suponhamos, como antes, que a sequência **y** = 1 1 0 1 0 1 foi recebida pelo "nosso" descodificador (7,4) depois de ter atravessado um canal binário simétrico com $p = 0{,}1$. Vamos ver como proceder à transferência de mensagens de dois elementos.

Já sabemos que cada variável recebe do canal uma mensagem fixa (ver Fig. 23)

$$p(y_i \mid x_i) = \begin{cases} p(y_i \mid x_i = 0) = 1 - p_i \\ p(y_i \mid x_i = 1) = p_i \end{cases}$$

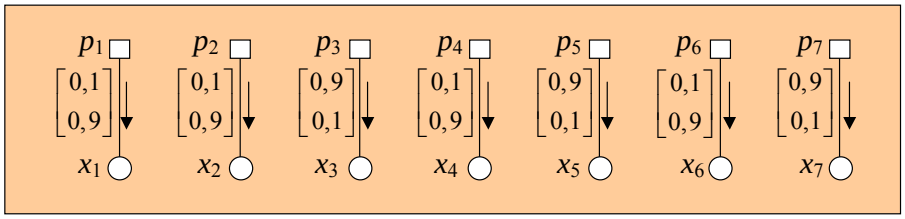


Fig. 23 Mensagens constantes que as variáveis x_i recebem do canal BSC.

Por exemplo,

$$p(y_1 = 1 \mid x_1) = \begin{bmatrix} p(y_1 = 1 \mid x_1 = 0) \\ p(y_1 = 1 \mid x_1 = 1) \end{bmatrix} = \begin{bmatrix} 0{,}1 \\ 0{,}9 \end{bmatrix}$$

Em seguida cada variável difunde a mensagem a priori que recebeu pelos seus nós de paridade. Assim, por exemplo,

$$q_{21}(x_2) = p(y_2 \mid x_2) = \begin{bmatrix} 0{,}1 & 0{,}9 \end{bmatrix}^T$$
$$q_{22}(x_2) = p(y_2 \mid x_2) = \begin{bmatrix} 0{,}1 & 0{,}9 \end{bmatrix}^T$$

Teremos, portanto, a situação global da Fig. 24, particularizada para o caso específico das mensagens iniciais difundidas por x_1, x_2 e x_7.

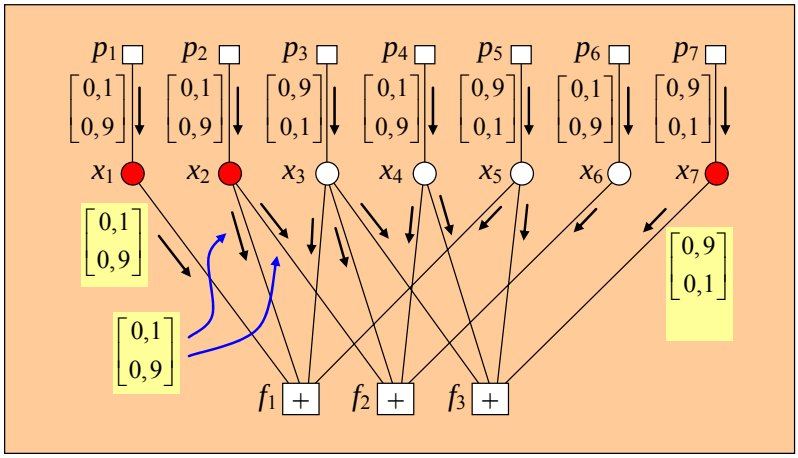


Fig. 24 Mensagens iniciais enviadas pelas variáveis x_i aos nós de paridade f_j.

Em seguida os nós de paridade vão enviar mensagens às variáveis. Por exemplo, f_1 envia a x_1 uma mensagem de dois elementos que envolve as mensagens que f_1 acabou

de receber de todas as suas variáveis excepto x_1. De acordo com a Eq. (5) essa mensagem é dada por

$$r_{11}(x_1)=\begin{bmatrix} r_{11}(x_1=0) \\ r_{11}(x_1=1) \end{bmatrix}=\sum_{\sim\{x_1\}}\left[\, f_1(x_1,x_2,x_3,x_5)q_{21}(x_2)q_{31}(x_3)q_{51}(x_5)\right]=$$
$$=\sum_{\{x_2\}}\sum_{\{x_3\}}\sum_{\{x_5\}} f_1(x_1,x_2,x_3,x_5)q_{21}(x_2)q_{31}(x_3)q_{51}(x_5)$$

onde a indicação $\sim\{x_1\}$ significa que a soma se estende aos valores de todas as variáveis excepto x_1. Como $f_1(x_1,x_2,x_3,x_5)=\delta(x_1\oplus x_2\oplus x_3\oplus x_5)$ (Cf. Eq. (1)), então

$$r_{11}(x_1)=\sum_{\{x_2\}}\sum_{\{x_3\}}\sum_{\{x_5\}}\delta(x_1+x_2+x_3+x_5)q_{21}(x_2)q_{31}(x_3)q_{51}(x_5)$$

Nos somatórios anteriores as somas estendem-se aos dois valores possíveis de x_2, x_3 e x_5, respectivamente, o que significa que para cada valor de x_1 vamos ter oito parcelas ao todo (uma para $x_2=0$, $x_3=0$, $x_5=0$, outra para $x_2=0$, $x_3=0$, $x_5=1$, etc.). E como nestes oito conjuntos de valores de x_2, x_3 e x_5 metade tem paridade par (a soma em módulo 2 das três variáveis é 0), e metade tem paridade ímpar (soma igual a 1), então quatro parcelas contêm $\delta(x_1)$ e quatro contêm $\delta(x_1+1)$. Por exemplo, uma das parcelas vai ser

$$\delta(x_1+0+1+1)q_{21}(0)q_{31}(1)q_{51}(1)=\delta(x_1)q_{21}(0)q_{31}(1)q_{51}(1)\,.$$

Assim, a expressão de $r_{11}(x_1)$ desenvolve-se em

$$\begin{aligned} r_{11}(x_1)&=\sum_{\sim\{x_1\}}\left[\delta(x_1+x_2+x_3+x_5)q_{21}(x_2)q_{31}(x_3)q_{51}(x_5)\right]= \\ &=\delta(x_1)[q_{21}(0)q_{31}(0)q_{51}(0)+q_{21}(0)q_{31}(1)q_{51}(1)+q_{21}(1)q_{31}(0)q_{51}(1)+q_{21}(1)q_{31}(1)q_{51}(0)]+ \\ &+\delta(x_1+1)[q_{21}(0)q_{31}(0)q_{51}(1)+q_{21}(0)q_{31}(1)q_{51}(0)+q_{21}(1)q_{31}(0)q_{51}(0)+q_{21}(1)q_{31}(1)q_{51}(1)] \end{aligned} \qquad (18)$$

Substituindo valores teríamos

$$\begin{aligned} r_{11}(x_1)&=\delta(x_1)(0{,}1\times 0{,}9^2+0{,}1^3+0{,}9^2\times 0{,}1+0{,}9\times 0{,}1\times 0{,}9) \\ &\quad+\delta(x_1+1)(0{,}1\times 0{,}9\times 0{,}1+0{,}1^2\times 0{,}9+0{,}9^3+0{,}9\times 0{,}1^2) \\ &=0{,}244\delta(x_1)+0{,}756\delta(x_1+1)=\begin{bmatrix} 0{,}244 \\ 0{,}756 \end{bmatrix} \end{aligned}$$

Identicamente obteríamos as mensagens de f_1 para os seus restantes argumentos:

$$r_{12}(x_2)=\begin{bmatrix} 0{,}244 \\ 0{,}756 \end{bmatrix} \qquad r_{13}(x_3)=\begin{bmatrix} 0{,}756 \\ 0{,}244 \end{bmatrix} \qquad r_{15}(x_5)=\begin{bmatrix} 0{,}756 \\ 0{,}244 \end{bmatrix}$$

A situação é ilustrada na Fig. 25. Da mesma maneira se obteriam as mensagens de f_2 e f_3 para os seus argumentos.

$$r_{22}(x_2)=\begin{bmatrix}0,756\\0,244\end{bmatrix} \qquad r_{23}(x_3)=\begin{bmatrix}0,244\\0,756\end{bmatrix} \qquad r_{24}(x_4)=\begin{bmatrix}0,756\\0,244\end{bmatrix} \qquad r_{26}(x_6)=\begin{bmatrix}0,756\\0,244\end{bmatrix}$$

$$r_{33}(x_3)=\begin{bmatrix}0,244\\0,756\end{bmatrix} \qquad r_{34}(x_4)=\begin{bmatrix}0,756\\0,244\end{bmatrix} \qquad r_{35}(x_5)=\begin{bmatrix}0,244\\0,756\end{bmatrix} \qquad r_{37}(x_7)=\begin{bmatrix}0,244\\0,756\end{bmatrix}$$

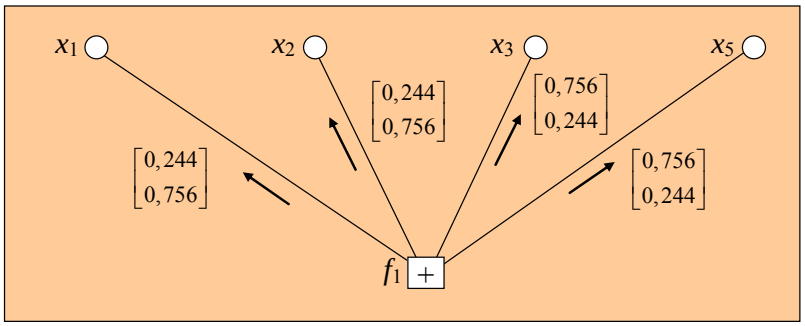


Fig. 25 Primeiras mensagens enviadas pelo nó de paridade f_1 aos nós de variáveis a que está ligado.

Uma pausa

Antes de prosseguir vamos parar um pouco para constatar que a Eq. (18) conduz ao mesmo resultado que obteríamos se usássemos a notação alternativa simplificada r_{ij} e q_{ij}. Se com essa equação quisermos determinar $r_{11}=r_{11}(x_1=1)$ deveremos considerar apenas a soma que multiplica $\delta(x_1+1)$ (porquê? Porque com $x_1=1$ é $\delta(x_1+1)=\delta(0)=1$ e $\delta(x_1)=\delta(1)=0$). Ora como $q_{ij}(0)=1-q_{ij}$ e $q_{ij}(1)=q_{ij}$ a Eq. (18) conduz-nos a

$$r_{11}=(1-q_{21})(1-q_{31})q_{51}+(1-q_{21})q_{31}(1-q_{51})+q_{21}(1-q_{31})(1-q_{51})+q_{21}q_{31}q_{51}$$

Ora compare esta expressão com a Eq. (11). Não nota semelhanças?

Voltemos às iterações… Tínhamos terminado a primeira e vamos agora iniciar a segunda. Nesta, cada variável x_i envia a cada nó de paridade uma nova mensagem calculada com base na estimativa inicial fixa $p(y_i|x_i)$ e nas diferentes mensagens que recebeu dos outros nós de paridade. Por exemplo, de x_2 para f_1 e f_2 vamos ter, respectivamente,

$$q_{21}(x_2)=K_{21}p(y_2\mid x_2)\bullet r_{22}(x_2)=K_{21}\begin{bmatrix}0,1\\0,9\end{bmatrix}\bullet\begin{bmatrix}0,756\\0,244\end{bmatrix}=K_{21}\begin{bmatrix}0,076\\0,220\end{bmatrix}$$

produto elemento-a-elemento

$$q_{22}(x_2)=K_{22}p(y_2\mid x_2)\bullet r_{12}(x_2)=K_{22}\begin{bmatrix}0,1\\0,9\end{bmatrix}\bullet\begin{bmatrix}0,244\\0,756\end{bmatrix}=K_{22}\begin{bmatrix}0,024\\0,680\end{bmatrix}$$

o que dá, depois de normalizar dividindo cada elemento pela soma dos dois,

$$q_{21}(x_2) = \begin{bmatrix} 0,256 \\ 0,744 \end{bmatrix}$$

$$q_{22}(x_2) = \begin{bmatrix} 0,035 \\ 0,965 \end{bmatrix}$$

como se mostra na Fig. 26. E o algoritmo prosseguiria, iteração após iteração…

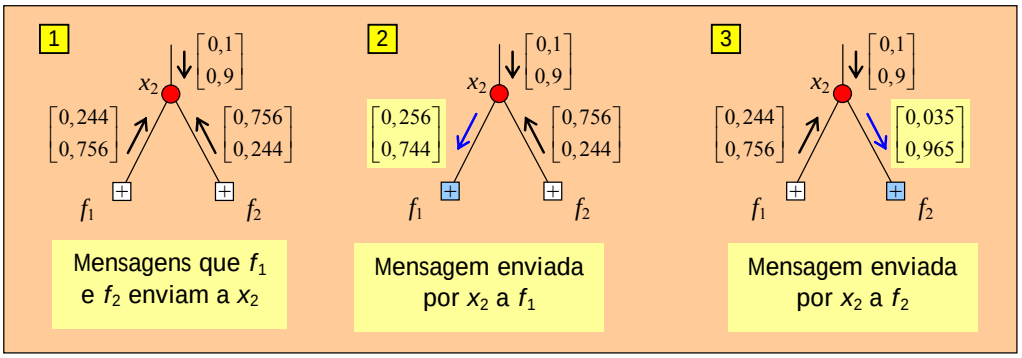


Fig. 26 Novas mensagens enviadas por x_2 aos nós de paridade f_1 e f_2.

No exemplo em que usámos mensagens escalares foi apresentada a Tabela **1** com os valores de LLR a posteriori obtidos durante cinco iterações. É claro que com as mensagens vectoriais de agora devemos obter exactamente os mesmos resultados. Vamos confirmá-lo. Por exemplo, vimos que ao fim da primeira iteração as mensagens

$$\begin{bmatrix} p(y_2 \mid x_2 = 0) \\ p(y_2 \mid x_2 = 1) \end{bmatrix} = \begin{bmatrix} 0,1 \\ 0,9 \end{bmatrix} \qquad r_{12}(x_2) = \begin{bmatrix} 0,244 \\ 0,756 \end{bmatrix} \qquad r_{22}(x_2) = \begin{bmatrix} 0,756 \\ 0,244 \end{bmatrix}$$

chegaram a x_2; então, para calcular a LLR a posteriori temos de primeiro calcular as probabilidades a posteriori

- multiplicando as três mensagens elemento-a-elemento,

$$\begin{bmatrix} 0,1 \times 0,244 \times 0,756 \\ 0,9 \times 0,756 \times 0,244 \end{bmatrix} = \begin{bmatrix} 0,018 \\ 0,166 \end{bmatrix},$$

- normalizando,

$$\begin{bmatrix} p(x_2 = 0 \mid \mathbf{y}) \\ p(x_2 = 1 \mid \mathbf{y}) \end{bmatrix} = \begin{bmatrix} 0,018/0,184 \\ 0,166/0,184 \end{bmatrix} = \begin{bmatrix} 0,10 \\ 0,90 \end{bmatrix}$$

e depois achar o logaritmo natural do quociente adequado,

$$L(x_2 \mid \mathbf{y}) = \ln \frac{p(x_2 = 1 \mid \mathbf{y})}{p(x_2 = 0 \mid \mathbf{y})} = \ln \frac{0,90}{0,10} = 2,20\,.$$

Não é este o valor que consta da Tabela **1**?

5. Referências

[1] R. G. Gallager, "Low-density parity-check codes", *IRE Transactions on Information Theory*, vol. IT-8, pp. 21-28, Janeiro de 1962.

[2] R. G. Gallager, *Low-density parity-check codes*, Monograph, *M.I.T. Press*, 1963. (Tese de Doutoramento)

[3] C. Berrou, A. Glavieux e P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding", *Proceedings of the IEEE International Conference on Communications* (ICC '93), Maio de 1993, pp. 1064-1070.

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

[5] N. Wiberg, H.-A. Loeliger e R. Kötter, "Codes and iterative decoding on general graphs", *European Transactions on Telecommunications*, vol. 6, pp. 513-525, Setembro/Outubro de 1995.

[6] N. Wiberg, "Codes and decoding on general graphs", Dissertação de Doutoramento, Linköping Univ., Linköping, Suécia, 1996.

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

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

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

[10] L. R. Bahl, J. Cocke, F. Jelinek e J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", *IEEE Trans. on Information Theory*, pp. 284-287, Março de 1974.

[11] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, Inc., 1988.

[12] R. J. McEliece, D. J. C. MacKay e J.-F. Cheng, "Turbo Decoding as an Instance of Pearl's "Belief Propagation" Algorithm", *IEEE Journal on Selected Areas In Communications*, Vol. 16, nº 2, pp.140-152, Fevereiro de 1998.

[13] J. Fan, "Lecture 2 on LDPC codes", Universidade de Stanford, EUA. Disponível *online* em http://www.stanford.edu/class/ee379b/class_reader/lecture2_4print.pdf. Acedido em 25-6-05.