Revista de Engenharia e Pesquisa Aplicada, Volume 2, Número 1, 2016

Estudo sobre decodificação iterativa usando códigos de treliça

Souza, I. M. M.
Escola Politécnica de Pernambuco
Universidade de Pernambuco
50.720-001 - Recife, Brasil
igor.poli@yahoo.com.br

Alcoforado, M. L. M. G.
Escola Politécnica de Pernambuco
Universidade de Pernambuco
50.720-001 - Recife, Brasil
mariadelourdesalcoforado@yahoo.com.br

Resumo *Este artigo mostra a implementacao do algoritmo BCJR para utilizacao em codigos de bloco lineares, com e sem o uso da decodificacao iterativa. Sao feitas simulacoes computacionais, e geradas curvas de desempenho, relacionando valores das probabilidadesde erro com os de relacao sinal-ruido.*

Abstract *This paper shows the implementation of the BCJRalgorithm for linear block codes using using iterativedecoding or not. The simulation results are made and the curves obtained, relating bit error probability values and signal- to-noise ratio.*

1 Introdução

Um dos grandes desafios da Engenharia de Telecomunicações e recuperar informações que num processo de transmissão ou armazenamento tenham sofrido algum tipo de alteração. Existe a necessidade de garantir a integridade dos dados enviados através de algum tipo de canal, e para tanto, todo sistema de transmissão digital precisa de algum tipo de técnica de correção de erros.

Os sistemas de comunicação representados pelo modelo clássico[1] consideram que há apenas um remetente querendo se comunicar, através de um canal ruidoso, com apenas um destinatário. A Figura 1.1 ilustra o modelo de comunicação clássico, no qual são indicados apenas a fonte de informação, o codificador de canal, o canal digital, o decodificador de canal e o destino. Neste artigo, a fonte de informação utilizada e sem memória, e emite símbolos binários e equiprováveis. As mensagens devem ser enviadas a um destino através de canal contaminado com ruído aditivo gaussiano branco (RAGB).[2]

O objetivo deste artigo e o estudo, a avaliação e a implementação de sistemas codificados para simulação em presença de ruído aditivo gaussiano branco. Na seção 2, e apresentada uma técnica de geração da treliça para códigos de bloco lineares a partir da matriz de verificação de paridade. A seção 3 versa o tratamento matemático necessário para decodificação de códigos de bloco usando o algoritmo de decodificação de Bahl et al BCJR[3]. A seção 4 traz a decodificação iterativa para uso em códigos de bloco lineares. A seção 5 ilustra os resultados das simulações através de curvas de desempenho geradas e finalmente na seção 6 a conclusão e comentários finais.

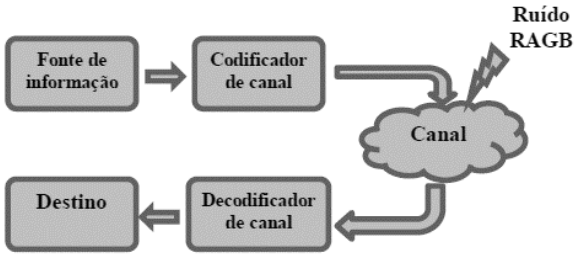

Figura 1.1 – Sistema de Comunicação ponto a ponto.

2 Treliça para códigos de blocos lineares

Seja C um código de bloco linear binário com parâmetros (n,k) , isto e os vetores correspondentes as mensagens são denotados por $\vec{u} = \{u_1,\ u_2, \dots, u_k\}$ as correspondentes palavras código por $\vec{v} = \{v_1,\ v_2, \dots, v_n\}$. A treliça correspondente para o código de bloco C e definida a partir da matriz de verificação de paridade. Seja **H** a matriz de verificação de paridade do código de bloco linear C(n,k) e $\overrightarrow{h_1}, i = 1,2, \dots, n$ os vetores coluna da matriz **H.** Os estados da treliça podem ser definidos como $S_i\ i = 0,1, \dots, n$, em que:

$$S_0 = \vec{0}\,,$$
$$S_i = S_{i\text{-}1} + v_i \vec{h}_i,\quad i = 1,2,\dots,n\,. \qquad (2.1)$$

O estado atual S_i é sempre função do estado anterior S_{i-1}, bem como do bit v_i. A partir de (2,1) é possível encontrar a treliça correspondente ao código de bloco.

Conforme ilustrado no exemplo 2.1

Exemplo 2.1: A treliça encontrada a partir da matriz verificação de paridade **H**:

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

representando o código de Hamming[4] C(7,4), esta ilustrada na Figura 2.1. Onde cada estado possível e mostrado na base decimal. Por exemplo: O estado (001) está representado como 1.

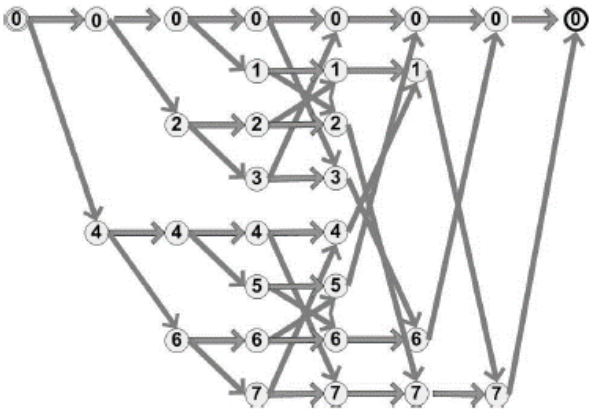

Figura 2.1. Treliça obtida a partir da matriz de verificação de paridade do código $C(7,4)$.

3 Algoritmo BCJR

O algoritmo proposto por Bahl et al (BCJR) pode ser aplicado a qualquer código que tenha uma treliça associada[4].

Seja $\vec{v} = \{v_1, v_2, \dots, v_k\}$, a entrada para um canal RAGB. A sequência de saída do canal e denotada por $\vec{r} = \{r_1, r_2, \dots, r_n\}$. A variável aleatória r_i, i = 1,...,n é definida como $r_i = (1 - 2v_i) + q_i$, onde q_i representa amostras de ruídos independentes com variância σ^2 e média 0 (zero). O algoritmo BCJR calcula a razão de log-verossimilhança, $\Lambda(v_i)$, associada a cada símbolo de palavra-código, v_i como mostrado na equação (3.1).

Em que $P\{v_i = 1|\vec{r}\}$ e $P\{v_i = 0|\vec{r}\}$ é a probabilidade a *posteriori* dos bits que compõe a palavra-código $\vec{v}$.

$$\Lambda(v_i) = \ln \frac{P\{v_i = 1 \mid \vec{r}\}}{P\{v_i = 0 \mid \vec{r}\}}. \qquad (3.1)$$

Após artifícios matemáticos e S_i e o conjunto de estados do instante i da treliça, (3.1) pode ser reescrita como:

$$\Lambda(v_i) = \ln \frac{\sum_{m \in S_i} \sum_{m' \in S_{i-1}} \gamma_1(r_i)\alpha_{i-1}(m')\beta_i(m)}{\sum_{m \in S_i} \sum_{m' \in S_{i-1}} \gamma_{-1}(r_i)\alpha_{i-1}(m')\beta_i(m)}, \qquad (3.2)$$

em que $\alpha_i(m)$, i = 0,1, ..., n, pode ser calculado de maneira recursiva por meio de:

$$\alpha_i(m) = \sum_{j \in \{-1,1\}} \sum_{m' \in S_{i-1}} \alpha_{i-1}(m)\gamma_j(r_i).$$

As condições de contorno para $\beta_i(m)$ são:

$$\beta_n(m) = \begin{cases} 1, m = 0 \\ 0, m \neq 0. \end{cases}$$

Considerando β_j^i como conjunto das transições de estrado de *m'* para *m*, as probabilidades de transição de canal contaminado com ruído aditivo gaussiano branco e das probabilidades de transição da treliça do codificador:

$$\gamma_j(r_t) = \begin{cases} P_i(j)e^{\left(\frac{(r_i - j)^2}{2\sigma^2}\right)}; \ m', m \in B_i^j \\ 0; \ outros, \end{cases} \qquad (3.7)$$

em que $P_i(j)$ representa a probabilidade *a priori* $P\{v_i = j\}$ A igualdade (3.2) pode ser reescrita da seguinte forma:

$$\Lambda(v_i) = \ln \frac{\sum_{m \in S_i} \sum_{m' \in S_{i-1}} P_i(1)e^{\left(\frac{(r_i - 1)^2}{2\sigma^2}\right)} \alpha_{i-1}(m')\beta_i(m)}{\sum_{m \in S_i} \sum_{m' \in S_{i-1}} P_i(0)e^{\left(\frac{(r_i + 1)^2}{2\sigma^2}\right)} \alpha_{i-1}(m')\beta_i(m)}. \qquad (3.8)$$

O algoritmo BCJR, portanto segue da seguinte forma:

- Cálculo de valores $\gamma_i(r_t)$:
- Calcular recursivamente $\alpha_i(m)$ *e* $\beta_i(m)$
- Cálculo dos $\Lambda(v_i)$

4 Decodificação iterativa

4.1 O codificador

Para que possa ser efetuada a decodificação iterativa e necessário que o codificador para o código * C tenha a estrutura bidimensional[5] ilustrada na Figura 4.1. O código de bloco * C e formado tal que cada palavracodigo seja um arranjo de n_1 colunas e n_2 linhas onde cada linha e uma palavra código em C^- , código de bloco sistemático com parâmetros (n_1, k_1) , e cada coluna e uma palavra código em $C^|$, código de bloco sistemático com parâmetros (n_2 , k_2) . Na figura 4.1 P^- *e* $P^|$ representam os bits da paridade de C^- *e* $C^|$, respectivamente. Os k_1k_2 os símbolos binários localizados na quina superior esquerda são símbolos de informação. Os $(n_1 - n_2)/k_2$ símbolos binários localizados na quina superior direita são calculados a partir das regras de paridade de C^- e os $(n_2 - n_2)/k_1$ símbolos binários localizados na quinta inferior esquerda são calculados a partir de regras de paridade de $C^|$.

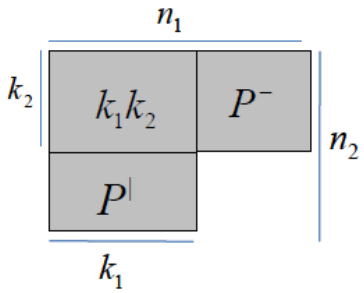

Figura 4.1. Estrutura de um código bidimensional C^* com $C^-(n_1,k_1)$ e $C^|(n_2,k_2)$ como códigos componentes.

4.2 O decodificador turbo

O decodificador turbo utiliza o princípio da decodificação iterativa [2] e consiste de dois componentes, DEC1 e DEC2, concatenados em série. DEC1 e responsável pela decodificação das k_2 linhas ou seja, o código componente C^-, e DEC2, das $k_1$colunas, isto é, o código componente $C^|$. Esse esquema é ilustrado na figura 4.2. Supõe-se que os dois decodificadores usam o algoritmo BCJR.

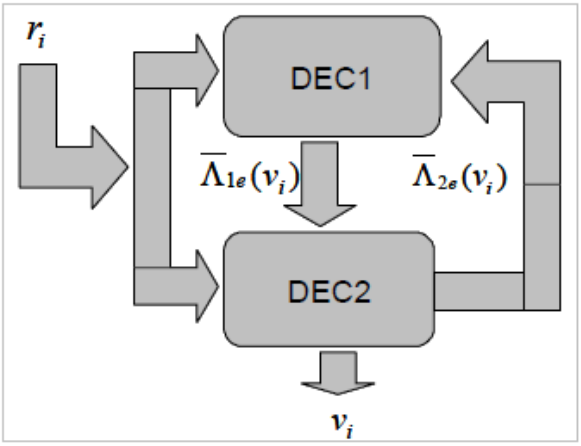

Figura 4.2. Esquema da decodificação iterativa.

Reescrevendo a equação (3.8), para calcular a razão de Log-Verossimelhança de DEC1 tem-se:

$$\Lambda_1(v_i) = \ln\frac{P_i^1(1)}{P_i^1(0)} + \frac{2v_i}{\sigma^2} + \Lambda_{1e}(v_i)\,, \qquad (4.1)$$

sendo que:

$$\Lambda_{1e}(v_i) = \ln\frac{\sum_m^{m\in S_t}\sum_{m'}^{m'\in S_{i-1}}\alpha_{i-1}(m')\beta_i(m)}{\sum_m^{m\in S_t}\sum_{m'}^{m'\in S_{i-1}}\alpha_{i-1}(m')\beta_i(m)} \qquad (4.2)$$

e o expoente "1" em $P_i^1(1)$ e $P_i^1(0)$ representa as probabilidades $P_i(1)$ e $P_i(0)$ correspondentes a DEC1.

Para primeira iteração $ln\frac{P_i^1(1)}{P_i^1(0)\{v_i=0\}} = 0$, já que os símbolos emitidos pela fonte de informação são equiprováveis.

A sequência recebida por DEC2 esta correlacionada com a saída suave do primeiro decodificador $\Lambda_1(v_i)$. Portanto, a contribuição devida a r_i deve ser retirada de $\Lambda_1(v_i)$ para eliminar essa correlação. Como $\Lambda_{1e}(v_i)$ não depende r_i esse termo pode ser usado como probabilidade a priori para decodificação do segundo estágio.

$$\Lambda_{1e}(v_i) = \ln\frac{P_i^2(1)}{P_i^2(0)} \quad . \qquad (4.3)$$

O expoente "2" em P_i^2 (1) e P_i^2 (0) representa as probabilidades $P_i(1)$ e $P_i(0)$ correspondentes a DEC2.

Utilizando (4.3) e a igualdade $P_i^2(1) = 1 - P_i^2(0)$, as probabilidades a priori de DEC2 podem ser escritas da seguinte forma:

$$P_i^2(1) = \frac{\exp(\Lambda_{1e}(v_i))}{1+\exp(\Lambda_{1e}(v_i))},$$

$$P^2{}_i(0) = \frac{1}{1+\exp(\Lambda_{1e}(v_i))}.$$

$$\text{Logo: } \ln\frac{P^2\{v_i=1\}}{P^2\{v_i=0\}} = \ln e^{(\Lambda_{1e}(v_i))} = \Lambda_{1e}(v_i)\,.$$

Então:

$$\Lambda_2(v_i) = \Lambda_{1e}(v_i) + \frac{2v_i}{\sigma^2} + \Lambda_{2e}(v_i)\,. \qquad (4.5)$$

Para as demais iterações, a probabilidade a priori de DEC1 será atualizada com uma equação análoga a (4.5). Logo temos:

$$\Lambda_1(v_i) = \Lambda_{2e}(v_i) + \frac{2v_i}{\sigma^2} + \Lambda_{1e}(v_i)$$

5 Resultados das Simulações

Foram considerados para efeito de simulações computacionais a decodificação utilizando o algoritmo BCJR

para os casos utilizando os códigos de Hamming C(15,11) , C(31,26) e C(63,57) conforme ilustrado na figura 5.1.

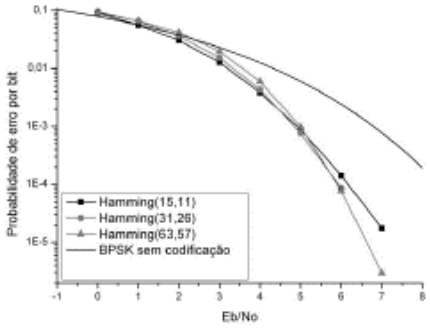

Figura 5.1. Desempenho do BCJR na decodificação de alguns códigos de Hamming.

Para decodificação iterativa foram realizadas duas simulações com 4 iterações cada. Para compor C^*, na primeira simulação, foram utilizados dois códigos de Hamming $C^{|}$=C^{-}=C(15,11). Os resultados obtidos nessa simulação podem ser vistos na figura 5.2.

Já para outra simulação, foram utilizados dois códigos de Hamming $C^{|} = C^{-}$= C(31,26) para compor C^*. Também foram realizadas 4 iterações e o desempenho dessa decodificação pode ser observado na figura 5.3.

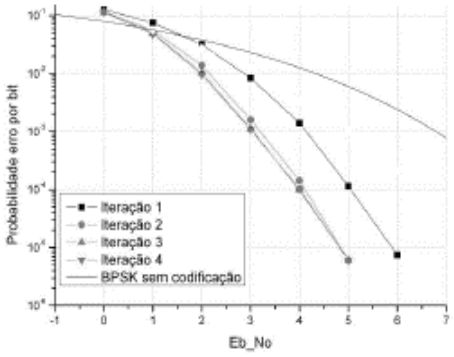

Figura 5.2. Desempenho da decodificação iterativa com $C^{-} = C^{|} = C(15,11)$.

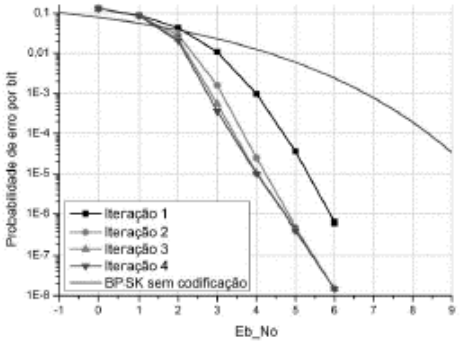

Figura 5.3. Desempenho da decodificação iterativa com $C^{-} = C^{|} = C(31,26)$.

6 Conclusões

Como esperado, e possível notar um ganho de aproximadamente 2,5dB para probabilidade de erro por bit de 10^{-3}, quando compara-se o sistema codificado com o sistema sem codificação.

As curvas obtidas para decodificação iterativa, figura 5.2 e figura 5.3, mostram uma melhora de desempenho a cada nova iteração. A partir da terceira iteração a probabilidade de erro por bit passa a não melhorar significativamente. E possível perceber um ganho de aproximadamente 1dB para probabilidade de erro por bit de 10^{-4} quando comparadas as curvas relacionadas a primeira e segunda iteração, em ambos os casos.

Referências

[1] SHANNON, C.E. A Mathematical Theory of Communication. Bell Syst. Tec. Jour., v. 27, July 1948,pp.379-423, 623-656.

[2] ALCOFORADO, M. L. M. G. Codificacao Iterativa para o Canal Aditivo com Dois Usuarios Binarios. Recife, Brasil, 2005. Tese (Doutorado em Engenharia Eletrica)- Departamento de Eletronica e Sistemas, UFPE.

[3] BAHL, L. R.; COCKE, J.; JELINEK, F.; RAVIV, J. Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate. IEEE Trans. Inform. Theory, v. IT-20, March 1974, pp.284-287.

[4] LIN, S.; COSTELLO JR., D. Error Control Coding: Fundamentals and Applications, Prentice-Hall Inc., 2004.

[5] HAGENAUER, J. Iterative Decoding of Binary Block and Convolutional Codes. IEEE Trans. Inform. Theory, v. 42, No 2, March 1996, pp.429-445.