SBrT 2000 - XVIII Simpósio Brasileiro de Telecomunicações, 3 a 6 de Setembro, 2000, Gramado-RS

ALGORITMO H-BCJR DE BUSCA REDUZIDA PARA CÓDIGOS TURBO

H. Freire e J. Portugheis

DECOM - FEEC - UNICAMP, CP 6101, 13083-970 Campinas, SP- Brasil
e-mail: <hfreire, jaime>@decom.fee.unicamp.br

SUMÁRIO

Recentemente foram propostas duas modificações do algoritmo BCJR utilizado na decodificação de códigos turbo. Estas modificações têm como objetivo uma busca reduzida de estados na treliça dos códigos. Uma delas, o algoritmo M, limita o atraso máximo da decodificação turbo. A outra, o algoritmo T, apresenta um atraso médio bastante reduzido. Neste artigo, apresentamos um algoritmo de busca reduzida, H-BCJR, que simultaneamente apresenta um atraso médio comparável ao algoritmo T e uma limitação no atraso máximo. Quando utilizado na decodificação turbo, apresenta um bom compromisso entre desempenho e atraso máximo.

1 INTRODUÇÃO

Códigos Turbo foram propostos por Berrou, Glavieux e Thitimajshima em 1993 [1]. Eles demonstraram, através de resultados de simulações, que uma relação sinal-ruído de $0,7$ dB era suficiente para garantir uma probabilidade de erro de bit de 10^{-5} através do uso de um código de taxa $1/2$ num canal AWGN. O resultado foi surpreendente, pois antes não se havia chegado tão próximo da capacidade de canal (que para esta taxa é de $0,0$ dB) com uma complexidade de codificação/decodificação relativamente baixa.

Na decodificação turbo, dois decodificadores MAP (Máximo *a Posteriori*) trocam informação entre si e procedem de forma iterativa para refinar as estimativas dos bits de informação. Cada decodificador calcula probabilidades relativas a estados da treliça derivando probabilidades *a posteriori* para cada bit de informação, as quais são passadas como informação extrínseca para o outro decodificador. Para canais sem memória, a implementação de um decodificador MAP pode ser obtida através da utilização do algoritmo BCJR [2].

Entretanto, o algoritmo BCJR é bastante intensivo computacionalmente. Esforços foram feitos no sentido de simplificar o algoritmo, sem penalizar significativamente seu desempenho. Franz e Anderson propuseram mecanismos de busca reduzida para o algoritmo BCJR [3]. Uma das técnicas, o algoritmo M, baseia-se na manutenção dos M estados mais significativos da treliça. A outra técnica, o algoritmo T, elimina a computação para os estados da treliça cujas probabilidades caiam abaixo de um dado limiar.

No presente trabalho propõe-se um outro método de busca reduzida, o algoritmo H, que pode ser visto como um método híbrido, pois além de possuir um limiar para decisão, também limita o número máximo de operações. O algoritmo BCJR armazena as probabilidades *a posteriori* dos estados e as utiliza para calcular o conjunto de probabilidades de transição de estados de cada seção da treliça. Isto permite obter-se uma estimativa para cada bit de informação. Para um ruído não muito intenso, um caminho se destaca na treliça, caminho este associado aos estados com maior probabilidade, enquanto os demais estados apresentam probabilidades muito pequenas. Quando a probabilidade de um estado for suficientemente pequena, ela pode ser igualada a zero, evitando-se a computação associada a ela, sem afetar significativamente o desempenho do algoritmo. O objetivo passa a ser definir critérios para eliminar os estados pouco significativos. A seção 2 irá revisar o algoritmo BCJR, a seção 3 tratará dos algoritmos de busca reduzida M-BCJR e T-BCJR e a seção 4 descreverá uma decodificação turbo. Na seção 5, o algoritmo H-BCJR será apresentado. Em seguida, resultados de simulação do desempenho dos algoritmos considerados serão mostrados na seção 6. Finalmente, conclusões serão apresentadas na seção 7.

2 ALGORITMO BCJR

O algoritmo BCJR gera dois conjuntos de vetores, $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$, referentes às probabilidades de estados, calculados de forma recursiva ao longo da seqüência de símbolos recebidos do canal $Y_1^\tau = (Y_1, \ldots, Y_\tau)$. Em seguida, esses dois conjuntos de vetores são utilizados para calcular-se as probabilidades associadas às transições entre estados e, conseqüentemente, as probabilidades dos bits de informação. O uso de notação matricial favorece a exposição do algoritmo.

O algoritmo BCJR permite obter a probabilidade de que o codificador tenha estado no estado i no instante t, dado o conjunto de símbolos recebidos do canal, ou seja,

$$P[S_t = i \mid Y_1^\tau] \tag{1}$$

Para codificadores não-recursivos, estas probabilidades são suficientes para obter-se as probabilidades dos bits de informação. Para codificadores recursivos, isso não é válido, é necessário obter a probabilidade de transição de estados, a fim de que se obtenha a probabilidade da entrada correspondente. Ou seja, é necessário calcular

$$P[S_{t-1} = i;\ S_t = j \mid Y_1^\tau] \tag{2}$$

a probabilidade de que tenha ocorrido uma transição do estado i para o estado j entre os instantes $t-1$ e t, dada a seqüência recebida do canal. Na verdade, o algoritmo encontra a probabilidade conjunta, $\sigma_t(i,j) = P[S_{t-1} = i;\ S_t = j;\ Y_1^\tau]$. A equação (2) nada mais é que $\boldsymbol{\sigma}_t$ escalada por uma constante, $P[Y_1^\tau]$, a probabilidade da seqüência recebida. Para se obter a probabilidade de transição de estados, define-se a matriz Γ_t, cujos elementos são

$$\Gamma_t(i,j) = P[S_t = j;\ Y_t \mid S_{t-1} = i] \tag{3}$$

Esta matriz armazena a probabilidade de uma transição para o estado j e observação Y_t, dado o estado anterior i. É esta matriz que leva em conta as probabilidades de transição do canal e é nela que devem entrar informações *a priori* sobre os dados. Há uma matriz Γ_t para cada seção da treliça. Finalmente, precisamos definir os conjuntos de vetores $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$, vetores referentes aos passos recursivos direto e reverso, cujos elementos são

$$\alpha_t(i) \quad = \quad P[S_t = i;\ Y_1^t],\ 1 \leqslant t \leqslant \tau \tag{4}$$

$$\beta_t(j) \quad = \quad P[S_t = i;\ Y_1^t],\ \ 1 \leqslant t \leqslant \tau - 1 \tag{5}$$

Os passos do algoritmo BCJR são:

1. Inicializar os vetores $\boldsymbol{\alpha}_0$ e $\boldsymbol{\beta}_\tau$.

2. Calcular o conjunto de vetores $\boldsymbol{\alpha}_t$, a partir do procedimento recursivo direto

$$\alpha_t = \alpha_{t-1}\Gamma_t,\ t = 1, ..., \tau. \tag{6}$$

3. Calcular o conjunto de vetores $\boldsymbol{\beta}_t$, a partir do procedimento recursivo reverso

$$\beta_t = \Gamma_t \beta_{t+1},\ \ t = \tau - 1, ..., 1. \tag{7}$$

4. Obter o conjunto de probabilidades de transição $\boldsymbol{\sigma}_t$ a partir de

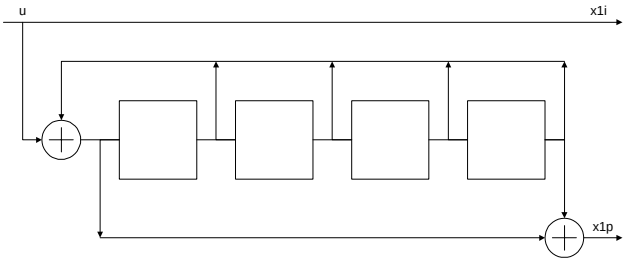

Figura 1: Codificador RSC, taxa 1/2, $m = 4$.

$$\sigma_t(i,j) = \alpha_{t-1}(i)\Gamma_t(i,j)\beta_t(j) \tag{8}$$

A soma de todos os elementos de um vetor $\boldsymbol{\sigma}_t$ resulta $P[Y_1^\tau]$ e normalizando (8) por esse fator pode-se obter a probabilidade de uma dada transição condicionada à seqüência Y_1^τ recebida. Uma vez calculado o conjunto de vetores $\boldsymbol{\sigma}_t$, as probabilidades de bit de informação em cada seção da treliça podem ser obtidas através do somatório de todos os $\sigma_t(i,j)$ cuja transição corresponda ao bit de entrada considerado, escalados por $P[Y_1^\tau]$. Ou seja, para obter-se $P[u_t = 0]$, a probabilidade do bit de informação u_t ter sido zero, deve-se somar todos os elementos de $\boldsymbol{\sigma}_t$ cujas transições i,j correspondam ao bit zero. $P[u_t = 1]$ será o complemento daquela probabilidade. Enfim, deve-se ressaltar que é importante normalizar também cada $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$ à medida que eles são calculados, pois o processo recursivo - um produto de probabilidades - faz com que os elementos desses vetores sejam reduzidos até valores desprezíveis.

3 ALGORITMOS DE BUSCA REDUZIDA

Os algoritmos de busca reduzida exploram o fato de que boa parte das componentes dos vetores $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$ possui valores pouco significativos quando comparados com a componente de valor máximo. Sendo assim, esses elementos poderiam ser retirados dos cálculos, simplificando o número de operações sem perdas expressivas em desempenho. O objetivo dos algoritmos de busca reduzida é definir de forma adequada quais estados serão utilizados nos cálculos.

O algoritmo M-BCJR preserva os M estados com maior probabilidade em cada seção da treliça. Sendo assim, o procedimento recursivo que produz o vetor $\boldsymbol{\alpha}_t$ (6) utiliza apenas os M maiores elementos do vetor $\boldsymbol{\alpha}_{t-1}$, os demais são igualados a zero. O mesmo procedimento pode ser aplicado no passo recursivo reverso, mas uma vez que os elementos de $\boldsymbol{\sigma}_t$ são produtos de $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$ (8), basta calcular os $\boldsymbol{\beta}_t$ apenas nas regiões da treliça onde existam os α_t correspondentes. O M-BCJR possibilita limitar o número máximo de operações, mas apresenta um desempenho pobre comparado ao BCJR e como também

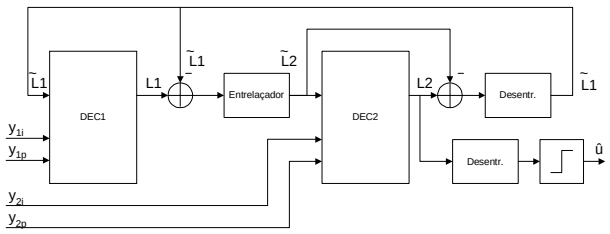

Figura 2: Estrutura do decodificador turbo.

em comparação com os algoritmos de busca reduzida a serem descritos a seguir.

No algoritmo T-BCJR, os elementos de $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$ são igualados a zero quando caírem abaixo de um certo limiar. Em cada seção da treliça, o vetor $\boldsymbol{\alpha}_t$correspondente é normalizado. Para calcular o vetor $\boldsymbol{\alpha}_t$ em (6), utiliza-se apenas os elementos de $\boldsymbol{\alpha}_{t-1}$ que estiverem acima do limiar. Assim como no algoritmo M-BCJR, a computação dos $\boldsymbol{\beta}_t$ só precisa ser feita nas regiões da treliça onde existirem estados sobreviventes. O esquema T-BCJR é bastante simples e apresenta um desempenho próximo ao do BCJR, com a vantagem de executar um número médio de operações muito menor. O maior problema desse algoritmo é que não há controle sobre o número máximo de operações executadas. Apesar do número médio de operações ser reduzido, pode haver casos em que todos os elementos de $\boldsymbol{\alpha}$ caiam acima do limiar, logo o decodificador estará operando com a complexidade de um BCJR.

4 DECODIFICAÇÃO TURBO

Nesta seção será descrita a decodificação turbo para um código de taxa 1/4. Um código de taxa 1/4 é obtido utilizando-se dois codificadores idênticos de taxa 1/2 em paralelo. O primeiro codificador recebe a palavra u_t e o segundo recebe a versão entrelaçada de u_t. O diagrama do codificador utilizado, do tipo RSC (codificador sistemático recursivo), está mostrado na figura 1 [4]. O decodificador turbo está apresentado na figura 2. Na figura, DEC j representa um decodificador MAP.

Numa iteração ocorre a seguinte seqüência de eventos. Dec1 recebe as seqüências do canal y_{1i} e y_{1p}, bem como a informação *a priori* $\tilde{L}1$, o logaritmo da razão de verossimilhança sobre o bit de informação. Na primeira iteração, $\tilde{L}1$ é igual a zero. A saída de Dec1, $L1$, é o logaritmo da razão de verossimilhança das probabilidades *a posteriori*, $P[u_t \mid Y_1^T]$, $u_t = 0, 1$. Entretanto, não é este valor que é passado para Dec2, e sim a diferença $\tilde{L}2 = L1 - \tilde{L}1$, chamada informação extrínseca. Este novo valor é usado como informação *a priori* para Dec2. O fato de se eliminar o termo $\tilde{L}1$, referente à informação *a priori* para Dec1, deve-se à necessidade de passar para Dec2 apenas a nova informação obtida. $L1$ carrega muita informação das iterações anteriores, enquanto $\tilde{L}2$ enfatiza a contribuição da última decodificação de Dec1. Dec2 recebe ainda as seqüências do canal y_{2i} e y_{2p}. Dec2 calcula novas informações *a posteriori*, $L2$. Para a próxima iteração, é calculado o novo valor *a priori* para Dec1, $\tilde{L}1 = L2 - \tilde{L}2$. De forma análoga, é necessário eliminar a informação referente à etapa anterior. Ao final de todas as iterações, os bits decodificados são obtidos a partir de uma decisão sobre $L2$.

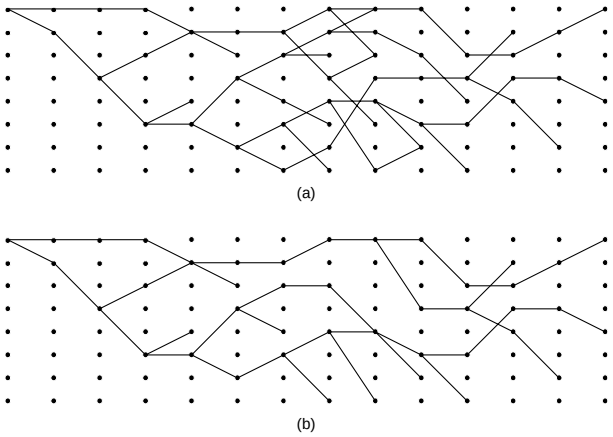

Figura 3: Estados sobreviventes em algoritmos de busca reduzida. (a) Treliça para T-BCJR. (b) Treliça para H-BCJR com $M = 4$.

5 O ALGORITMO H-BCJR

Os algoritmos de busca reduzida se baseiam na redução dos cálculos que envolvem os vetores α e β, referentes às probabilidades de estados, através da introdução de zeros nas componentes menos significativas. Um critério para se determinar as componentes menos significativas é inserido entre os passos 1 e 2 do algoritmo BCJR (seção 2). O algoritmo H-BCJR baseia-se na introdução simultânea dos critérios de limiar (T) e de número máximo de estados sobreviventes (M). As componentes que estiverem abaixo de um limiar serão igualadas a zero. Compara-se então o número de componentes não-nulas com um número máximo pré-fixado de estados sobreviventes, M. Doravante denominaremos M de teto. Se o número de componentes não-nulas for inferior ao teto, procede-se normalmente com os passos seguintes do algoritmo BCJR. Caso contrário, os M maiores valores formam o conjunto de estados sobreviventes. Vale a pena ressaltar que a complexidade adicional de escolher as M maiores componentes é desprezível quando comparada com a redução no número de operações do algoritmo BCJR original.

A figura 3 mostra a variação do número de estados sobreviventes para os algoritmos de busca reduzida T-BCJR e H-BCJR. Uma linha conectando dois nós indica que o nó à esquerda foi utilizado para calcular o nó à direita, e que este sobreviveu. Da figura fica claro que, ao contrário do T-BCJR, o H-BCJR

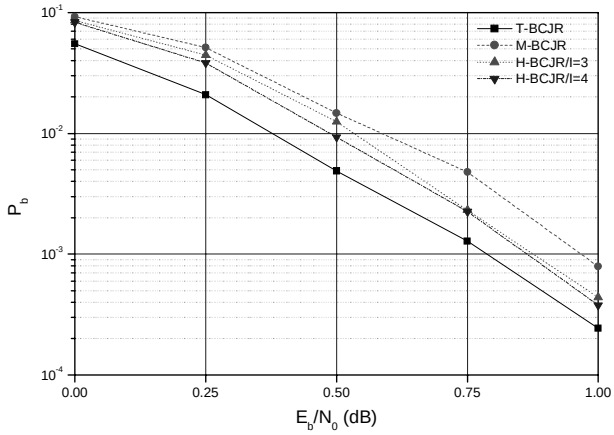

Figura 4: Probabilidade de erro de bit (P_b) para os vários algoritmos de busca reduzida.

possui um limite para o número de estados sobreviventes.

Na decodificação turbo, cada iteração traz um ganho adicional menor do que a iteração anterior (lei dos retornos diminuídos). Isto significa que existe um número de iterações a partir do qual o ganho adicional passa a ser desprezível. Dizemos então que o algoritmo atingiu a saturação. A iteração em que a saturação é atingida depende do comprimento do entrelaçador, N. Quanto maior for N, maior será o número de iterações necessárias para atingir a saturação. Como cada iteração corresponde a duas decodificações MAP através do algoritmo BCJR, a complexidade da decodificação turbo poderá tornar-se extremamente elevada.

Fazendo-se uma análise do número médio de estados sobreviventes do algoritmo T-BCJR em função do número de iterações, observa-se que existe uma dada iteração I a partir da qual o número médio de estados não varia significativamente. Baseado nisso, propõe-se que a partir desta iteração I o algoritmo H-BCJR seja utilizado nos decodificadores MAP. Sendo assim, garantimos também uma redução do número máximo de operações da decodificação turbo.

6 RESULTADOS DE SIMULAÇÃO

Todos os resultados foram obtidos para um código turbo de taxa 1/4 cujos codificadores componentes possuem taxa 1/2, memória $m = 4$ e entrelaçador de comprimento $N = 1024$. O diagrama do codificador componente é apresentado na figura 1. Todos os resultados foram obtidos para oito iterações do decodificador turbo.

A figura 4 apresenta o desempenho do algoritmo H-BCJR, para um limiar 10^{-5} e teto igual a 12. A figura mostra resultados para $I = 3$ e $I = 4$. Como referência, apresentamos também o desempenho dos algoritmos M-BCJR para $M = 12$ e T-BCJR para um limiar 10^{-5}. Na região de maior E_b/N_0 a diferença de desempenho entre $I = 3$ e $I = 4$ é desprezível. Resultados

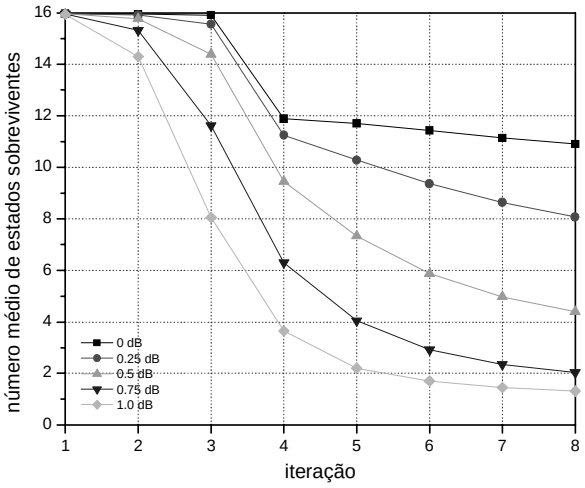

Figura 5: Decodificação Turbo H-BCJR/$I = 3$. Número médio de estados sobreviventes em cada iteração, para limiar 10^{-5} e teto 12. Total de 8 iterações.

de simulação foram obtidos para valores menores de I, mas a degradação de desempenho foi significativa, aproximando-se da degradação mostrada pelo M-BCJR, $M = 12$. Valores maiores de I não produzem um bom compromisso entre desempenho e redução no número máximo de operações.

Além do desempenho, estamos também interessados em observar a redução no número médio de iterações para o algoritmo H-BCJR. A figura 5 mostra o número médio de estados preservados em cada iteração para diversos valores de E_b/N_0. Os resultados foram obtidos para $I = 3$ e um teto $M = 12$. Para valores de P_b desejada em torno de 10^{-3} ($E_b/N_0 = 0,75\ dB$) o número médio de estados sobreviventes se aproxima de dois. Este comportamento pode também ser observado quando o algoritmo T-BCJR é utilizado. Portanto, podemos concluir que a utilização do algoritmo H-BCJR mantém o número médio de estados sobreviventes comparável ao do T-BCJR e apresenta controle do número máximo de operações característico do algoritmo M-BCJR.

7 CONCLUSÕES

Foi apresentado um novo algoritmo de busca reduzida para um decodificador turbo. Este algoritmo explora o fato de que, durante a execução do algoritmo BCJR, os valores de $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$ variam bastante em uma dada seção da treliça, e apenas alguns destes valores são significantes para a determinação do desempenho. O algoritmo H-BCJR estabelece um critério de seleção para os $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$ a serem preservados de forma a reduzir simultaneamente as complexidades média e máxima de decodificação. A complexidade máxima está associada ao atraso máximo de decodificação, um parâmetro a ser considerado em diversos enlaces de um sistema de comunicações.

Uma investigação de outros algoritmos de busca reduzida com melhor compromisso entre desempenho e número máximo de operações está em desenvolvimento.

REFERÊNCIAS

[1] C. Berrou, A. Glavieux e P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes", in Proc. IEEE ICC'93, Geneva, Switzerland, May 1993, pp. 1740-1745.

[2] L. R. Bahl, J. Cocke, F. Jelinek, e J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", IEEE Trans. Inform. Theory, vol IT-20, Mar. 1974, pp. 284-287.

[3] V. Franz e J. B. Anderson, "Concatenated decoding with a reduced-search BCJR algorithm", IEEE Journal Selec. Areas Commun., vol 16, Feb. 1998, pp. 186-195.

[4] D. Divsalar and F. Pollara, "Turbo codes for deep-space communications", TDA Progress Report 42-120, JPL, Feb. 1995, pp. 29-39.