Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação - FEEC
Departamento de Comunicações - DECOM

Algoritmos de Busca Reduzida para Decodificação Turbo

Autor:

Helder José Pacheco Freire

Orientador:

Prof. Dr. Jaime Portugheis

Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos para obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA.

Banca Examinadora:

Prof. Dr. Jaime Portugheis (Presidente)	DECOM - FEEC - UNICAMP
Prof. Dr. Marcelo E. Pellenz	PUC - Curitiba/PR
Prof. Dr. Celso de Almeida	DECOM - FEEC - UNICAMP
Prof. Dr. Renato Baldini	DECOM - FEEC - UNICAMP

Campinas, Março de 2003

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

F883a Freire, Helder José Pacheco
Algoritmos de busca reduzida para decodificação turbo / Helder José Pacheco Freire.--Campinas, SP: [s.n.], 2003.

Orientador: Jaime Portugheis.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Modulação digital. 2. Comunicações digitais. 3. Teoria da codificação. 4. Códigos de controle de erros (Teoria da informação). I. Portugheis, Jaime. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Este trabalho propõe algumas modificações que reduzem o esforço computacional de um esquema de decodificação turbo. Estas modificações simplificam o algoritmo BCJR, utilizado nos decodificadores, através do estabelecimento de um critério para a redução do número de estados calculados a cada estágio da treliça de decodificação. O desempenho das propostas é analisado através de resultados de simulação. A primeira proposta introduz simultaneamente dois critérios de busca reduzida de estados na treliça, objetivando limitar o esforço computacional máximo e a complexidade média. A avaliação é efetuada em canais AWGN e em canais com desvanecimento. São também propostos um esquema que apresenta um limiar variável para a definição dos estados sobreviventes e um outro que modifica o critério ao longo da treliça. Concluiu-se que as primeiras iterações da decodificação possuem uma influência decisiva no desempenho final. Para canais com desvanecimento, esta característica mostrou-se ainda mais evidente. Verificou-se que a aplicação simultânea de dois critérios de redução de esforço permite limitar a complexidade máxima, além de manter o desempenho em um patamar satisfatório.

Abstract

This dissertation proposes some modifications in the turbo decoder scheme in order to reduce computational effort. These modifications intend to simplify the BCJR algorithm, which is used in the decoders, through the use of a criterion for the reduction of the calculated states for each stage of the decoder trellis. The first proposal simultaneously introduces two reduced-search criteria for the calculated states in the trellis, with the objective of limiting both the maximum computational effort and the average complexity. The evaluation is done in AWGN channels and in fading channels. Two other schemes are also proposed, one presents an adaptive threshold for the choice of the surviving states and the other modifies the criterion as we advance through the trellis. We have concluded that the first decoding iterations have a decisive influence in the final performance. This characteristic is even more evident for fading channels. We have verified that the simultaneous application of two reduced-search criteria allows a greater limitation in maximum complexity while keeping good decoding performance.

A meus pais.

Agradecimentos

Ao professor Jaime Portugheis, sempre pronto a colaborar com valiosas sugestões

Aos professores Marcelo Pellenz, Celso de Almeida e Renato Baldini, que gentilmente atenderam ao convite para compor a banca

Aos amigos de jornada, tanto aos novos quanto aos de outras conquistas

E a todos aqueles que, direta ou indiretamente, contribuíram para a realização deste trabalho.

"O sábio modifica suas opiniões conforme os fatos

O tolo modifica os fatos para encaixá-los em suas crenças"

(*Anônimo*)

Sumário

Resumo iii

Abstract v

Agradecimentos ix

Lista de Figuras xvii

Lista de Acrônimos xix

1. Introdução 1

1.1 Sistema de Comunicações 1

1.2 Códigos Turbo 3

1.3 Proposta do Trabalho 5

1.4 Organização 6

2. Conceitos Básicos 7

2.1 Controle de Erros 7

2.2 Canal 8

2.2.1 Canais sem Memória 8

2.2.2 Canal AWGN 10

2.2.3 Capacidade de Canal 11
2.2.4 Canais com Desvanecimento 12
2.3 Codificação Convolucional 18
2.3.1 Análise no Domínio do Tempo 20
2.3.2 Análise no domínio da transformada 21
2.3.3 Representações Gráficas 22
2.4 Decodificação por Máxima Verossimilhança 25
2.4.1 Decisão para canais sem memória 25
2.4.2 Métrica de máxima verossimilhança 26
2.5 Algoritmo de Viterbi 27
2.6 Algoritmo BCJR 28
2.6.1 Visão Geral - Diagrama de Fluxo BCJR 29
2.6.2 Fonte de Sinal 31
2.6.3 Diagrama de Treliça 32
2.6.4 Probabilidades de Transição entre Estados 33

3. Códigos Turbo **37**

3.1 Visão Geral 37
3.2 Codificador 39
3.2.1 Introdução 39
3.2.2 Codificador Sistemático Recursivo 39
3.2.3 Sistema implementado 42
3.2.4 Terminação da Treliça 46
3.3 Entrelaçador 47
3.3.1 Distribuição de Pesos 48
3.4 Decodificação Turbo 49

3.4.1 Probabilidades *a posteriori* 50
3.4.2 Algoritmo MAP 53
3.4.3 Passos Iterativos 58
3.4.4 Algoritmos de Decisão Suave 59
3.4.5 Decodificador - Aspectos de Implementação 59

4. Algoritmos de Busca Reduzida **65**

4.1 Algoritmos de Busca Reduzida para BCJR 66
4.1.1 Por que não SOVA? 67
4.2 Algoritmo M 69
4.3 Algoritmo de Limiar (Algoritmo T) 69
4.4 Estados sobreviventes 71
4.4.1 Estabelecimento de um teto 74
4.5 Algoritmo Composto 74
4.6 Algoritmo de Limiar Variável 76
4.7 Algoritmo de Divisão de Treliça 78

5. Resultados de Simulação e Conclusões **81**

5.1 Resultados para o Algoritmo Composto 81
5.2 Resultados para Canais com Desvanecimento Rayleigh 84
5.3 Resultados para o Algoritmo de Limiar Variável 85
5.4 Resultados para o Algoritmo de Divisão de Treliça 86
5.5 Saturação do número de iterações 87
5.6 Comparação de Resultados 88
5.7 Conclusões e Perspectivas Futuras 90

Referências Bibliográficas **91**

Lista de Figuras

1.1 Diagrama de blocos de um sistema de comunicações. 2

1.2 Codificação de Canal. 3

2.1 Canal binário simétrico (BSC). 9

2.2 Modelo de canal AWGN. 10

2.3 Modelo de canal com desvanecimento. 16

2.4 Codificador convolucional genérico (n, k, m). 18

2.5 Exemplo de codificador convolucional. 19

2.6 Codificador de 4 estados. 23

2.7 Diagrama de treliça para codificador de 4 estados. 23

2.8 Diagrama de estados para o codificador da figura 2.6. 24

2.9 Diagrama de fluxo para o algoritmo BCJR. 30

2.10 Sistema de transmissão simplificado. 31

2.11 Exemplo de diagrama de treliça para uma fonte de Markov de 4 estados. 32

3.1 Sistema Codificador/Decodificador em paralelo. 38

3.2 Codificador recursivo. 40

3.3 Diagrama de estados para o codificador recursivo apresentado. 40

3.4 Sistema codificador em paralelo genérico utilizado em códigos turbo. 41

3.5 Codificador Sistemático Recursivo (RSC), taxa 1/2, $m = 4$. 43
3.6 Sistema codificador paralelo. 44
3.7 Módulo Codificador/Decodificador e canal. 45
3.8 Terminação da treliça. 46
3.9 Estrutura de um decodificador turbo. 49

4.1 Diagrama de fluxo de um decodificador genérico de busca reduzida. 68
4.2 Diagrama de fluxo do algoritmo M. 70
4.3 Diagrama de fluxo do algoritmo T. 72
4.4 Estados sobreviventes para cada seção da treliça. 73
4.5 Evolução dos estados sobreviventes em algoritmos de busca reduzida para uma mesma palavra recebida. 75
4.6 Diagrama de fluxo para Algoritmo de Limiar Variável. 77
4.7 Diagrama de fluxo de Algoritmo de Divisão de Treliça. 79

5.1 Probabilidade de erro de bit (P_b) para vários algoritmos de busca reduzida. 8 iterações. 82
5.2 Decodificação Turbo C-BCJR/I = 3. Número médio de estados sobreviventes para cada iteração. 83
5.3 Desempenho do Algoritmo Composto em canal com desvanecimento Rayleigh. . . 84
5.4 Desempenho do Algoritmo de Limiar Variável (LV) comparado ao BCJR. 85
5.5 Desempenho comparativo do Algoritmo de Divisão de Treliça (DT) e BCJR. . . . 86
5.6 Evolução do desempenho do Algoritmo DT ao longo de 8 iterações. 87
5.7 Número médio de estados sobreviventes para algoritmos de busca reduzida. 89
5.8 Evolução da probabilidade de erro de bit para algoritmos de busca reduzida. . . . 89

Lista de Acrônimos

APP Probabilidades *a posteriori*. *A posteriori probabilities*, p. 29.

ARQ Sistema de gerenciamento de transmissão de dados no qual a seqüência é retransmitida caso seja detectado erro no receptor. Do inglês *Automatic Repeat reQuest*, requisição de repetição automática, p. 8.

AWGN Ruído aditivo gaussiano branco. *Additive White Gaussian Noise*, p. 10.

BCJR Algoritmo para decodificação de códigos de bloco e convolucionais que minimiza a probabilidade de erro de símbolo de informação, denominado a partir das iniciais de seus autores [Bahl74], p. 4.

BPSK Esquema de modulação binária por chaveamento de fase. *Binary Phase Shift-Keying*, p. 17.

BSC Canal binário simétrico. *Binary Simmetric Channel*, p. 9.

C-BCJR Algoritmo composto utilizado para redução da complexidade computacional do BCJR, p. 75.

DMC Canal discreto sem memória. *Discrete Memoryless Channel*, p. 9.

DT Algoritmo de divisão de treliça para redução da complexidade computacional do BCJR, p. 86.

LLR Logaritmo da razão de verossimilhança. *Log-Likelihood Ratio*, p. 55.

LV Algoritmo de limiar variável para redução da complexidade computacional do BCJR, p. 85.

M-BCJR Algoritmo que limita o número máximo de estados na decodificação por BCJR, p. 4.

MAP Decodificador convolucional que maximiza probabilidades *a posteriori*. Máximo *a posteriori*, p. 4.

MLD Decodificador de máxima verossimilhança. *Maximum Likelihood Decoder*, p. 25.

PSK Esquema de modulação discreta por chaveamento de fase. *Phase Shift-Keying*, p. 11.

RSC Codificador sistemático recursivo. *Recursive Systematic Coder*, p. 41.

SOVA Algoritmo de Viterbi de saída suave. *Soft-Output Viterbi Algorithm*, p. 67.

T-BCJR Algoritmo de limiar para redução da complexidade computacional aplicado ao BCJR, p. 4.

Capítulo 1

Introdução

Grandes esforços têm sido feitos para se obter sistemas de transmissão mais confiáveis, mais robustos e de maior eficiência. Na busca desses objetivos, os sistemas digitais têm recebido enfoque especial. A integridade da seqüência de informação transmitida é garantida através de codificação. Entretanto, é preciso considerar a dicotomia desempenho/complexidade computacional. Neste trabalho são propostas algumas modificações que reduzem o esforço computacional de um esquema de decodificação turbo.

1.1 Sistema de Comunicações

Um sistema de comunicações digital e unidirecional está representado na figura 1.1. A Teoria da Informação é o ramo que abrange os assuntos referentes à codificação da informação, definindo seus limites fundamentais. A codificação de fonte trata da representação eficiente da informação no formato digital, por exemplo, eliminando redundâncias na seqüência. Os dados então seguem para o codificador de canal. A principal finalidade da codificação de canal é minimizar os efeitos do canal ruidoso sobre a seqüência de informação recebida de forma que a seqüência estimada no receptor seja idêntica à seqüência original transmitida. O canal

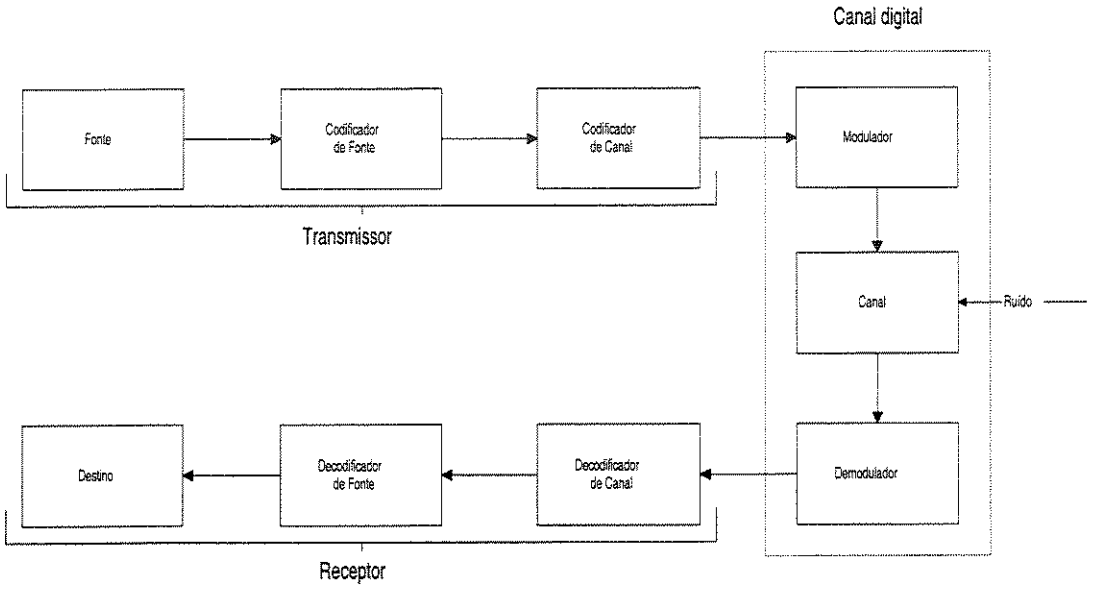

Figura 1.1: Diagrama de blocos de um sistema de comunicações.

propriamente dito é analógico, então os estágios modulador e demodulador são necessários para transformar os símbolos discretos da seqüência de informação em um sinal apropriado para transmissão no canal e vice-versa. O conjunto canal-modulador-demodulador pode ser tratado como um canal digital pelos demais blocos.

Este trabalho está centrado em um dos estágios que compõem um sistema de comunicações digital: a codificação de canal, que trata da questão da transmissão de informações através de um canal ruidoso. A codificação de canal consiste em transformar a seqüência binária de informação $\mathbf{u}$ em uma seqüência discreta codificada $\mathbf{v}$ chamada palavra código (figura 1.2). A seqüência $\mathbf{v}$ é geralmente uma seqüência binária, embora códigos não-binários sejam também amplamente utilizados. O canal ruidoso corrompe a seqüência transmitida $\mathbf{v}$, resultando em uma seqüência recebida $\mathbf{r} \neq \mathbf{v}$. A principal finalidade da codificação da canal é neutralizar os efeitos do canal ruidoso em um sistema de comunicações, de forma que a seqüência estimada $\hat{\mathbf{u}}$, reconstruída a partir da seqüência recebida $\mathbf{r}$, seja idêntica à seqüência de informação $\mathbf{u}$.

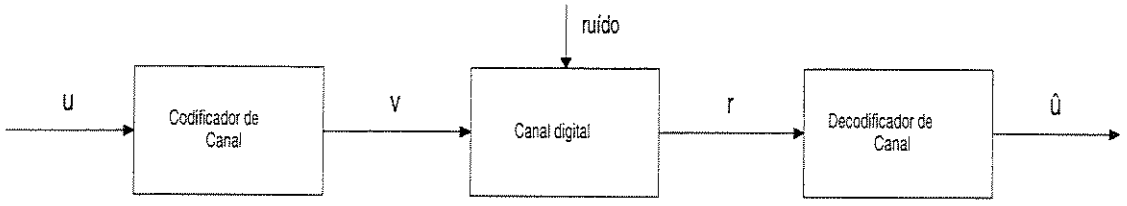

Figura 1.2: Codificação de Canal.

Desta forma, para que tenhamos $\hat{\mathbf{u}} = \mathbf{u}$ (em um número satisfatório de vezes) alguns aspectos de engenharia devem ser considerados. Um deles está ligado às características da informação a ser transmitida. Por exemplo, se $\mathbf{u}$ representa a digitalização de um quadro de voz em uma transmissão em tempo real, o retardo total de codificação e decodificação está limitado. Isto implica um limite no número máximo de operações por quadro de voz.

Por outro lado, o número médio de operações por quadro de voz está diretamente associado a dois aspectos adicionais distintos. O primeiro deles ocorre num sistema de transmissão no qual, sendo $\hat{\mathbf{u}} \neq \mathbf{u}$, a seqüência $\mathbf{u}$ pode ser retransmitida (sistemas ARQ), como por exemplo, em enlaces de uma rede de computadores. Neste caso, o retardo médio da rede é afetado pelo número médio de operações na decodificação. O segundo aspecto é o consumo de energia no receptor, ou seja, a redução no número médio de operações resulta num maior intervalo de recarga da bateria.

1.2 Códigos Turbo

Há várias maneiras de se implementar codificação de canal num sistema de comunicações. Códigos turbo são um esquema de codificação e decodificação que tem despertado interesse por permitir probabilidades de erro baixas para relações sinal-ruído próximas da capacidade de canal com complexidade de implementação razoavelmente pequena. Eles têm sido usados em comunicações de espaço profundo e estima-se que em breve estejam presentes em enlaces de voz

e outras aplicações usuais. Entretanto, o retardo e a complexidade computacional são fatores limitantes.

Os códigos turbo foram propostos por Berrou, Glavieux e Thitimajshima em 1993 [Berr93]. 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, conforme será apresentado na seção 2.2.3) 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, que minimiza a probabilidade de erro de símbolo de informação (o BCJR está descrito na seção 2.6). Entretanto, o algoritmo BCJR é computacionalmente intensivo. Esforços têm sido feitos no sentido de reduzir o número de operações sem penalizar o desempenho da decodificação turbo.

Duas variações do algoritmo BCJR foram aplicadas por Franz e Anderson [Franz98] na decodificação de códigos convolucionais concatenados em série e em paralelo. Ambas as variações propostas simplificam o algoritmo pela redução do número de estados calculados, chamados estados sobreviventes, por estágio da treliça de decodificação. Um dos algoritmos, o M-BCJR, é baseado na retenção dos M estados mais significativos como estados sobreviventes. O outro, o algoritmo T-BCJR, elimina o cálculo para os estados da treliça cujos valores de probabilidade associados sejam menores que um dado limiar T.

Frey e Kschischang propuseram em [Frey98] uma outra simplificação para a decodificação

turbo baseado na detecção prévia de certos símbolos que permitiriam reduzir a complexidade computacional das iterações de decodificação seguintes. Recentemente, Dasgupta e Narayanan mostraram em [Dasg01] que, caso expansão de *hardware* seja possível e aceitável, o uso do algoritmo T-BCJR em um esquema de decodificação paralela resulta em uma redução significativa no compromisso complexidade/retardo de decodificação sem haver sensível degradação de desempenho.

1.3 Proposta do Trabalho

Nesta dissertação propomos algumas modificações que reduzem o esforço computacional do algoritmo BCJR e analisamos o desempenho das mesmas através de resultados de simulação. As modificações propostas simplificam o algoritmo através da redução do número de estados calculados, chamados estados sobreviventes, por estágio da treliça de decodificação. Por simplicidade, cada modificação no algoritmo é tratada no texto como um algoritmo à parte.

Uma das propostas compreende uma variação do algoritmo BCJR que introduz simultaneamente dois critérios de busca reduzida de estados na treliça, sendo um critério que envolve um limiar para definir os estados sobreviventes e outro que retém apenas um número fixo de estados sobreviventes a cada seção da treliça. Este esquema permite reduzir não apenas o número médio de operações mas também impõe um limite no esforço computacional máximo. Esta proposta é avaliada em canais AWGN e em canais com desvanecimento.

É também proposto um algoritmo que apresenta um limiar variável para os estados sobreviventes, cujo valor depende do termo mais significativo em cada seção da treliça. Por fim, propõe-se um algoritmo que modifica o critério para definição dos estados sobreviventes ao longo da treliça, tratando de forma diferenciada os estados que compõem a parte inicial da treliça em relação aos da parte final.

1.4 Organização

O Capítulo 1 apresenta, de forma sucinta e geral, os temas a serem tratados, as motivações e a proposta do trabalho.

O Capítulo 2 apresenta diferentes modelos de canal, incluindo canais com desvanecimento, introduz os princípios da codificação com memória e mostra um algoritmo para decodificação que minimiza a probabilidade de erro de símbolo da fonte chamado algoritmo BCJR.

No Capítulo 3 são apresentados os fundamentos da codificação em paralelo, o papel do entrelaçador, as vantagens da decodificação iterativa e o sistema de codificação e decodificação utilizado no trabalho.

No Capítulo 4 são revistos alguns dos principais algoritmos para redução do esforço computacional. Novos algoritmos são propostos e comparados com os existentes, através de resultados de simulação.

E finalmente, no Capítulo 5 são apresentadas as conclusões e perspectivas para trabalhos futuros.

Capítulo 2

Conceitos Básicos

Neste capítulo serão abordados os conceitos básicos referentes ao trabalho desenvolvido. Haverá uma breve introdução sobre sistemas de comunicações digitais. Serão considerados diferentes modelos de canal e será discutida a codificação de canal com destaque para a codificação convolucional. Por fim, será apresentado um algoritmo de decodificação MAP (máximo *a posteriori*) que minimiza a probabilidade de erro de símbolo da fonte, denominado algoritmo BCJR, em contraste com os algoritmos que minimizam a probabilidade de erro de seqüência.

2.1 Controle de Erros

O controle de erros para um sistema de transmissão (ou gravação) semelhante ao da figura 1.1 pode ser obtido através da correção de erros direta, ou seja, pela aplicação de códigos corretores de erro que atuem automaticamente ao ser detectado erro no receptor. Como exemplo, temos os sistemas de comunicações de espaço profundo, nos quais um equipamento de codificação simples é colocado na nave e toda a operação de decodificação é feita em terra. Dessa forma, a operação de maior complexidade (decodificação) conta com mais recursos de *hardware* e maior tempo para realização, enquanto o transmissor fica encarregado apenas da operação mais simples

(codificação de canal).

No caso de um sistema de comunicações bidirecional, a abordagem mais simples para o controle de erros consiste na detecção de erros e retransmissão da seqüência de informação, método chamado ARQ (*automatic repeat request* - requisição automática de repetição). Em um sistema ARQ, ao ser detectado um erro no receptor, é enviada automaticamente uma requisição para que o transmissor repita a mensagem. O processo continua até que a mensagem seja recebida corretamente.

2.2 Canal

O canal de transmissão é o meio físico no qual a informação é transmitida. É necessário desenvolver um modelo para o canal de transmissão, pois nenhum canal é ideal. Canais reais apresentam resposta em freqüência não-uniforme. Canais reais apresentam ruídos e outras interferências que corrompem o sinal transmitido e causam erros na seqüência recebida.

Um estratégia para combater os erros causados pelo ruído é a codificação de canal. A codificação de canal acrescenta redundância de maneira controlada na seqüência de informação de forma que o receptor possa utilizar a informação adicional para restaurar uma seqüência corrompida.

2.2.1 Canais sem Memória

Os meios de transmissão são analógicos, portanto um sistema de comunicações digital necessita de um modulador/demodulador para transformar os símbolos discretos da seqüência de informação em um sinal apropriado para transmissão no canal e vice-versa. No nosso modelo, conforme a figura 1.1 podemos supor o modulador e demodulador como parte de um canal digital. Este canal composto apresenta uma entrada discreta e uma saída discreta, sendo caracterizado

pelo conjunto de entradas possíveis, pelo conjunto de saídas possíveis e pelas possibilidades condicionais de entrada-saída.

O modelo mais simples de canal digital é o canal binário simétrico ou BSC (*binary simmetric channel*). Suas entradas e saídas são seqüências binárias e o ruído causa erros estatisticamente independentes e com probabilidade p. Sejam X e Y as variáveis aleatórias representando os alfabetos de entrada e saída, respectivamente, então as probabilidades de transição de canal serão

$$P(Y = 0 \mid X = 1) = P(Y = 1 \mid X = 0) = p$$
$$P(Y = 0 \mid X = 0) = P(Y = 1 \mid X = 1) = 1 - p.$$

O diagrama de transição para este canal está representado na figura 2.1. A cada instante de tempo, cada bit de saída depende apenas do bit de entrada atual, por isso o canal é dito canal sem memória.

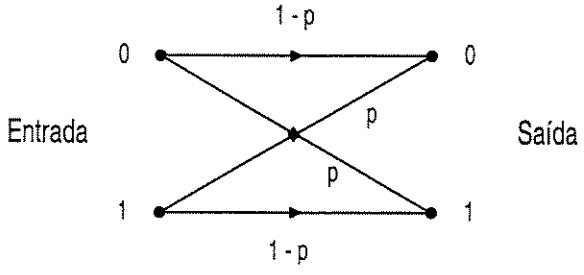

Figura 2.1: Canal binário simétrico (BSC).

O BSC é apenas um caso particular de um modelo mais abrangente chamado canal discreto sem memória, DMC (*discrete memoryless channel*). Um DMC é caracterizado por um alfabeto de entrada $X = \{x_0, x_1, ..., x_{q-1}\}$, um alfabeto de saída $Y = \{y_0, y_1, ..., y_{Q-1}\}$ e um conjunto de probabilidades condicionais (as probabilidades de transição) $P(Y = y_i \mid X = x_j) \equiv P(y_i \mid x_j)$, onde $i = 0, 1, ..., Q-1$ e $j = 0, 1, ..., q-1$. Dada uma seqüência de entrada $\mathbf{u}$ de n símbolos e a

correspondente seqüência de saída v, a probabilidade conjunta nada mais é que o produto das probabilidades condicionais, pois o canal não possui memória

$$P(Y_1 = y_1, ..., Y_n = y_n \mid X_1 = x_1, ..., X_n = x_n) = \prod_{k=1}^{n} P(Y = v_k \mid X = u_k).$$

2.2.2 Canal AWGN

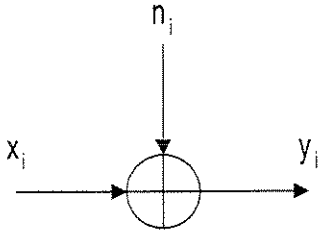

Figura 2.2: Modelo de canal AWGN.

Um modelo simples para o ruído é o ruído aditivo gaussiano branco AWGN (*additive white gaussian noise*). O modelo de canal equivalente possui entrada discreta e saída real. O modelo está representado na figura 2.2. Sua saída é dada por $y_i = x_i + n_i$. As n_i são variáveis aleatórias gaussianas de média zero e variância σ_i^2. Para um dado valor de x_i, segue-se que y_i possui distribuição gaussiana de média x_i e variância σ_i^2. As seqüências possuem comprimento N. As probabilidades condicionais são

$$p(y_i \mid x_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp[-\frac{(y_i - x_i)^2}{2\sigma_i^2}], \qquad i = 1, 2, ..., N$$

Uma vez que as variáveis n_i são não-correlacionadas e de distribuição gaussiana, elas são estatisticamente independentes. Portanto,

$$p(y_1, y_2, ..., y_N \mid x_1, x_2, ..., x_N) = \prod_{i=1}^{N} p(y_i \mid x_i).$$

2.2.3 Capacidade de Canal

A capacidade de canal indica a quantidade máxima de informação que se pode transmitir por um dado canal com probabilidade de erro tão pequena quanto se queira.

A capacidade de um canal ruidoso transmitir informação de forma confiável foi determinada por Shannon em 1948. O resultado, o Teorema da Codificação de Canal, diz que cada canal possui uma capacidade de canal C e, com decodificação apropriada, para qualquer taxa $R < C$ existem códigos de taxa R os quais atingem uma probabilidade de erro $P(E)$ arbitrariamente pequena.

A fórmula básica para cálculo de capacidade de canais AWGN limitados em faixa foi originalmente derivada por Shannon e é dada por:

$$C = W \log_2(1 + \frac{P_{med}}{W N_0}),$$

onde W é a largura de banda, P_{med} é a potência média do transmissor e N_0 é a densidade unilateral do ruído.

Percebe-se que a capacidade pode ser aumentada tanto pelo incremento na largura de banda W quanto pelo aumento da relação sinal-ruído $\frac{P_{med}}{W N_0}$.

A capacidade normalizada para a largura de banda W em função da relação sinal-ruído de bit, $\frac{E_b}{N_0}$, é

$$\frac{C}{W} = \log_2(1 + \frac{C}{W}\frac{E_b}{N_0}).$$

De onde vem

$$\begin{aligned} 2^{C/W} &= 1 + \frac{C}{W}\frac{E_b}{N_0} \\ \frac{E_b}{N_0} &= \frac{2^{C/W} - 1}{C/W}. \end{aligned} \tag{2.1}$$

Para taxa $r = \frac{1}{2}$ e modulação 4-PSK, a eficiência espectral $(\frac{C}{W})$ é 1 bit/símbolo. Substituindo em 2.1, temos:

$$\frac{E_b}{N_0} = 1 \Longrightarrow 0\ dB$$

válido para probabilidade de erro nula. Para probabilidades de erro finitas e pequenas é uma boa aproximação.

Da mesma expressão 2.1, considerando-se largura de banda infinita (fazendo $C/W \rightarrow 0$), temos

$$\frac{E_b}{N_0} = \lim_{C/W \rightarrow 0} \frac{2^{C/W} - 1}{C/W} = \ln 2 \Longrightarrow \frac{E_b}{N_0} = -1,6\ dB.$$

O valor $-1,6\ dB$ representa o limitante mínimo para a quantidade de energia por bit transmitido para que se possa ter comunicação confiável através de um canal AWGN.

2.2.4 Canais com Desvanecimento

2.2.4.1 Caracterização

O modelo mais simples de canal ruidoso, o canal AWGN, consiste em adicionar uma componente ruidosa ao sinal transmitido, onde a intensidade do ruído obedece a uma distribuição gaussiana. Neste modelo, o canal possui resposta ao impulso invariante no tempo.

Entretanto, um modelo mais sofisticado consideraria um canal com resposta ao impulso aleatoriamente variante no tempo. Esta caracterização permite modelar, por exemplo, a transmissão de sinais para comunicações de rádio na faixa de HF e a dispersão ionosférica na faixa de VHF. A resposta ao impulso nestes canais é conseqüência das variações dinâmicas nas características físicas das camadas ionosféricas, resultando no espalhamento do sinal e na dispersão do sinal em múltiplos percursos.

Sistemas de telefonia móvel também estão sujeitos a esse fenômeno, pois há um movimento relativo entre o transmissor e o receptor, além de inúmeros obstáculos que provocam reflexões e defasagens no sinal. O sinal recebido consiste de inúmeras réplicas do sinal transmitido com amplitudes distintas e com defasagens variantes no tempo. Devido à impossibilidade de um modelo determinístico, a melhor maneira de lidar com esse tipo de canal é utilizar uma abordagem

estatística.

2.2.4.2 Sinal recebido e resposta do canal

A resposta a impulso de um canal variante no tempo de múltiplos percursos pode ser descrita como um trem de impulsos, devido ao espalhamento temporal introduzido no sinal causado pelo meio de transmissão. Uma segunda característica deve-se a variações na estrutura do meio ao longo do tempo. Ou seja, se um pulso de características idênticas é enviado periodicamente, o sinal recebido apresentará variações ao longo do tempo na amplitude dos pulsos, nos atrasos entre os pulsos e no número de pulsos do sinal recebido. O sinal recebido correspondente ao sinal transmitido no tempo t_0 apresentará uma distorção diferente do sinal correspondente ao sinal transmitido no tempo t_1. Além disso, essas variações serão não-previsíveis para os usuários do canal.

No modelo matemático a ser utilizado para o canal será utilizada a representação complexa em banda básica dos sinais transmitido e recebido [Proak83]. Nesta representação, os sinais são transformados em uma forma complexa equivalente no qual a portadora é suprimida. Assim, o sistema resulta em um equivalente passa baixa, com o conteúdo de freqüência centrado em torno do zero. Então, o sinal transmitido pode ser representado por

$$s(t) = Re[s_l(t)e^{j2\pi f_c t}] \tag{2.2a}$$

onde $s_l(t)$ é a representação complexa em banda básica do sinal $s(t)$.

A transmissão ocorre através de um meio que permite múltiplos percursos. Associados a cada percurso há uma atenuação e um atraso variantes no tempo. Sendo assim, pode-se representar o sinal recebido na forma

$$r(t) = \sum_n \alpha_n(t)s(t - \tau_n(t)) \tag{2.3}$$

onde $\alpha_n(t)$ é a atenuação devido ao n-ésimo percurso e $\tau_n(t)$ é o atraso correspondente. Expandindo-se a representação de $s(t)$ para sua forma complexa (substituindo 2.2a em 2.3), temos

$$r(t) = Re\{[\sum \alpha_n(t)e^{-j2\pi f_c\tau_n(t)}s_l(t-\tau_n(t))]e^{j2\pi f_c t}\}. \tag{2.4}$$

Através da expressão complexa de $r(t)$ (2.4), obtém-se a representação em banda básica do sinal recebido

$$r_l(t) = \sum \alpha_n(t)e^{-j2\pi f_c\tau_n(t)}s_l(t-\tau_n(t)). \tag{2.5}$$

Uma vez que $r_l(t)$ é a resposta em banda básica do canal à entrada $s_l(t)$, então a resposta ao impulso do canal, variante no tempo, fica

$$c(\tau;t) = \sum \alpha_n(t)e^{-j2\pi f_c\tau_n(t)}\delta(\tau-\tau_n(t)) \tag{2.6}$$

onde $c(\tau;t)$ representa a resposta do canal no instante t devido a um impulso aplicado no instante $t-\tau$.

2.2.4.3 Representação fasorial do sinal recebido

A fim de caracterizar os efeitos do canal sobre um sinal transmitido, será analisado o envio de uma portadora não-modulada de freqüência f_c. Nesse caso, $s_l(t) = 1$, e o sinal recebido em banda básica (eq. 2.5) pode ser escrito como

$$\begin{aligned} r_l(t) &= \sum \alpha_n(t)e^{-j2\pi f_c\tau_n(t)} \\ &= \sum \alpha_n(t)e^{-j\theta_n(t)} \end{aligned} \tag{2.7}$$

onde $\theta_n(t) = 2\pi f_c\tau_n(t)$.

A expressão (2.7) permite interpretar o sinal recebido como a soma de fasores de amplitude $\alpha_n(t)$ e fase $\theta_n(t)$. Vale ressaltar que grandes variações dinâmicas no meio são necessárias para

que $\alpha_n(t)$ varie o suficiente para causar alterações significativas no sinal recebido. Por outro lado, variações da ordem de $1/f_c$ nos atrasos $\tau_n(t)$ associados aos diferentes percursos são suficientes para rotacionar a fase de $\theta_n(t)$ em 2π radianos. Uma vez que o valor de f_c é elevado, pequenas variações no meio de transmissão podem causar bruscas rotações de fase.

Portanto, o sinal recebido pode ser representado como uma soma de fasores que podem combinar-se de forma destrutiva, fazendo com que o sinal recebido seja muito pequeno, ou construtiva, aumentando sua amplitude. Estas variações de amplitude no sinal recebido neste modelo de canal, causadas pelas características variantes no tempo do meio de transmissão, fazem com que este tipo de canal seja denominado canal com desvanecimento.

Quando o número de percursos é grande, o que na prática é comum, o teorema central do limite pode ser aplicado e $r_l(t)$ pode ser modelado como um processo aleatório gaussiano complexo na variável t. Quando a resposta ao impulso $c(\tau;t)$ é também modelada como um processo aleatório gaussiano complexo de média nula, a envoltória $|c(\tau;t)|$ em qualquer instante t possui distribuição Rayleigh e a fase correspondente é uniformemente distribuída no intervalo $(-\pi,\pi)$. Neste caso, diz-se que o canal apresenta desvanecimento do tipo Rayleigh. Este modelo de canal é amplamente utilizado, pois incorpora fenômenos comuns que ocorrem em sistemas de telefonia móvel e em transmissões por dispersão ionosférica.

2.2.4.4 Modelo do canal com desvanecimento

Considera-se o canal como não-seletivo em freqüência. Sendo assim, pode-se utilizar para o canal o modelo representado na figura 2.3. Neste modelo há três processos aleatórios atuando no sinal transmitido, $F(t)$, $e^{j\theta(t)}$ e $n(t)$. O sinal recebido é

$$r(t) = F(t)e^{j\theta(t)}s(t) + n(t). \tag{2.8}$$

O processo de desvanecimento $F(t)$ introduz uma atenuação multiplicativa no sinal transmitido $s(t)$. $F(t)$ é considerado um processo de variação lenta, ou seja, o valor do processo

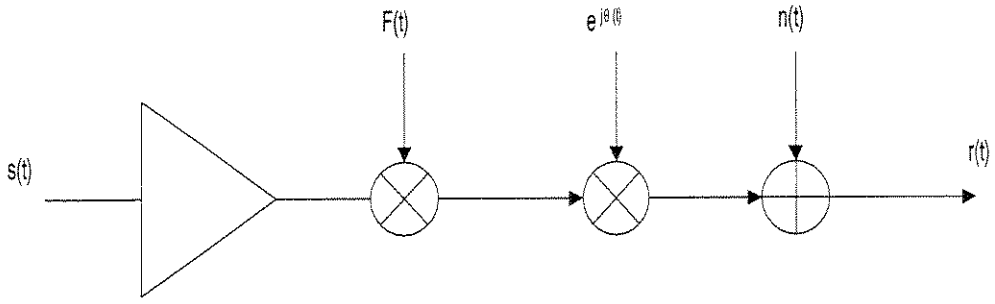

Figura 2.3: Modelo de canal com desvanecimento.

é constante durante o intervalo de tempo de observação do receptor. Portanto, $F(t)$ pode ser modelado como uma amostra de um processo gaussiano de média nula e variância σ_F^2.

F_k é uma amostra do processo de desvanecimento $F(t)$, representando uma amostra de uma gaussiana normalizada ($\sigma_F^2 = 1$) e de média nula.

O segundo processo do modelo é o processo de fase aleatória $e^{j\theta(t)}$, que introduz uma fase $\theta(t)$ uniformemente distribuída no intervalo $(-\pi, \pi)$. Assim como o processo de desvanecimento $F(t)$, $\theta(t)$ é considerado de variação lenta, isto é, para o intervalo $(kT \leq t \leq (k+N)T)$, seu valor é constante, $\theta(t) = \theta_k$.

O terceiro processo é o ruído aditivo gaussiano branco $n(t)$, provocado pelo ruído térmico inevitavelmente presente em qualquer sistema de comunicações. As componentes n_k representam as amostras de uma gaussiana complexa de média nula e variância definida por

$$\sigma_n^2 = \frac{1}{2}E[|n_k|^2] = \frac{N_0}{2}, \tag{2.9}$$

onde N_0 é a densidade unilateral do ruído. Vale ressaltar que as amostras do ruído gaussiano aditivo n_k apresentam variações de símbolo para símbolo.

2.2.4.5 Sinal recebido como combinação de processos aleatórios

A modulação considerada é do tipo BPSK. $s(t)$ é o sinal transmitido no intervalo de modulação $kT \leq t \leq (k+1)T$. Em sua representação complexa, a parcela do sinal referente à portadora foi suprimida, visto que não trazia nenhuma informação suplementar. A representação complexa caracteriza completamente o sinal.

A representação complexa do sinal $s(t)$ pode ser expressa como

$$s_k = Ae^{j\phi_k} \tag{2.10}$$

onde A é a amplitude do sinal e ϕ_k é a fase correspondente.

Uma vez que o processo de desvanecimento $F_k = \rho_k e^{j\Theta_k}$ está sendo considerado constante na janela de observação, bem como sua fase Θ_k, o sinal recebido $r(t)$ (equação 2.8) pode ser escrito como

$$r(t) = F_k e^{j\theta_k} s(t) + n(t) \tag{2.11}$$

onde θ_k é uma fase aleatória que o canal introduz, considerada constante no intervalo dado.

A equação 2.11 apresenta o sinal recebido através da influência no sinal $s(t)$ provocada pelos três processos aleatórios anteriormente descritos. Há uma diferença notável entre esta expressão e a correspondente ao sinal recebido para o caso de um canal AWGN, onde $r(t) = s(t) + n(t)$. O desvanecimento Rayleigh introduz um fator complexo que atua de forma multiplicativa no sinal transmitido $s(t)$, em contraposição ao ruído aditivo do modelo de canal AWGN.

2.3 Codificação Convolucional

A codificação convolucional é uma codificação de canal cuja saída apresenta memória. O codificador é baseado em registradores de deslocamento e a saída consiste de uma combinação linear da entrada com os bits no registrador. Devido à memória inerente ao registrador, os bits da seqüência de saída são relacionados uns com os outros. Um codificador convolucional genérico está representado na figura 2.4.

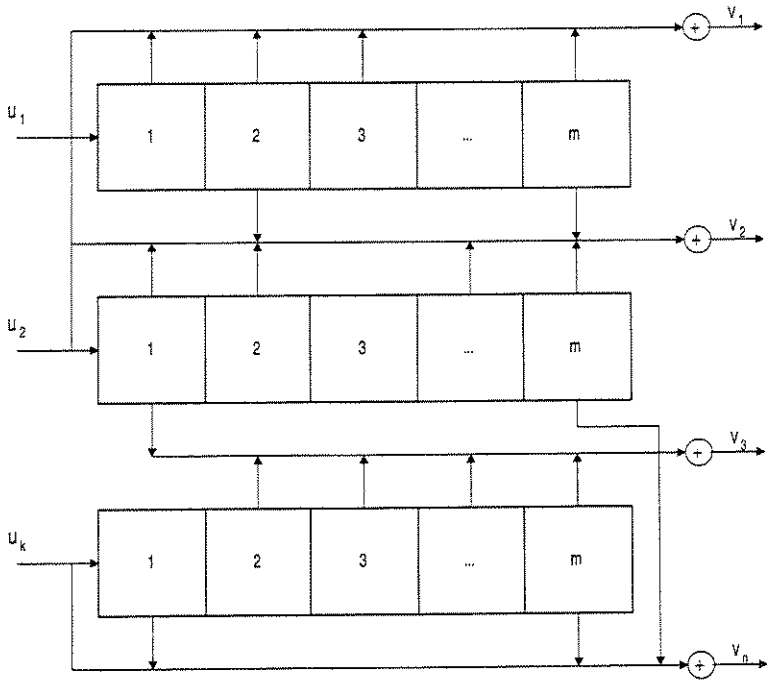

Figura 2.4: Codificador convolucional genérico (n, k, m).

A seqüência de informação $\mathbf{u}$ entra no codificador em blocos de k bits. No registrador, o conteúdo de cada estágio é transferido para o seguinte e assim sucessivamente. Combinações lineares dos bits formam os n bits de saída. A seqüência de saída $\mathbf{v}$ é formada por blocos de n bits. O codificador possui uma memória de comprimento m. Portanto, um codificador convolucional é caracterizado pela tripla (n, k, m). No exemplo da figura 2.5, $k = 1$, $n = 2$ e $m = 3$. Os

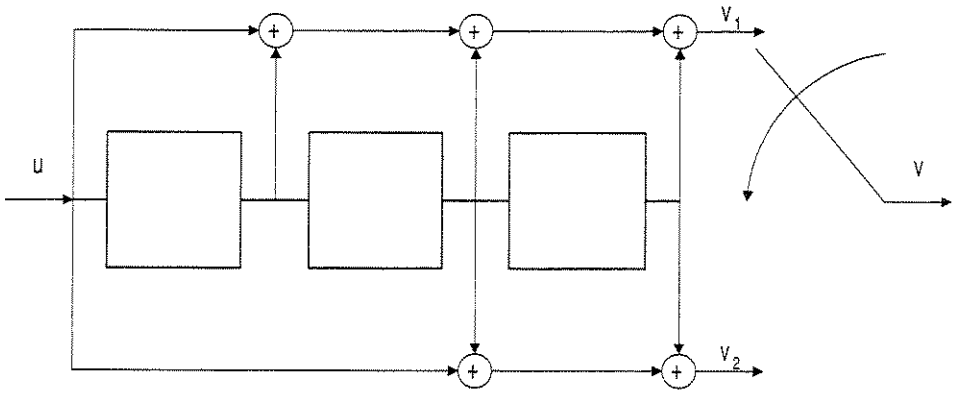

Figura 2.5: Exemplo de codificador convolucional, onde $k = 1$, $n = 2$ e $m = 3$.

somadores módulo 2 podem ser implementados com portas lógicas XOR (ou-exclusivo). Como a adição módulo 2 é uma operação linear, o codificador é um registrador de deslocamento linear.

Os k registradores não precisam ter todos o mesmo comprimento. Neste caso, define-se a ordem de memória m como o comprimento do maior registrador. Uma vez que cada bit de informação permanece no codificador por até $m + 1$ unidades de tempo, e durante cada instante pode afetar qualquer uma das n saídas (dependendo das conexões dos somadores) o número máximo de saídas que podem se afetadas por um único bit de informação é definido por $n_A = n(m + 1)$.

A razão $R = \frac{k}{n}$ é chamada taxa do código e dá uma medida da quantidade de redundância adicionada, pois são gerados n bits codificados para cada k bits de informação. Entretanto, uma seqüência de informação possui comprimento $k.L$ e a palavra código correspondente, comprimento $n(L + m)$, onde as $n.m$ saídas finais são geradas para que o registrador retorne ao estado nulo (limpar a memória do codificador). Então, a taxa do código de bloco seria $R = \frac{kL}{n(L+m)}$. Se $L \gg m$ (o modo de operação comum para códigos convolucionais), a taxa para códigos de bloco e convolucionais torna-se aproximadamente igual.

2.3.1 Análise no Domínio do Tempo

2.3.1.1 Seqüências geradoras

O comportamento no domínio do tempo de um codificador binário convolucional (de n saídas) pode ser definido em termos de suas n respostas ao impulso. Uma vez que o codificador é um sistema linear, qualquer seqüência de saída pode ser obtida pela convolução da seqüência de informação $\mathbf{u}$ com as respostas ao impulso do codificador. Estas podem ser obtidas entrando-se com a seqüência $\mathbf{u} = (1\ 0\ 0...)$, observando-se as seqüências de saída, $g^{(1)} = (g_0^{(1)}\ g_1^{(1)}\ ,...,\ g_m^{(1)})$ e $g^{(2)} = (g_0^{(2)}\ g_1^{(2)}\ ,...,\ g_m^{(2)})$. As respostas ao impulso $g^{(1)}$ e $g^{(2)}$ são chamadas seqüências geradoras do código. Para o codificador da figura 2.5, as seqüências geradoras são $g^{(1)} = (1111)$ e $g^{(2)} = (1011)$.

Então, as equações de codificação podem ser escritas como

$$\begin{aligned} v^{(1)} &= u * g^{(1)} \\ v^{(2)} &= u * g^{(2)} \end{aligned}$$

onde $*$ representa uma convolução discreta e as operações são módulo 2. A operação de convolução, uma vez expandida fica

$$v_t^{(j)} = \sum_{i=0}^{m} u_{t-i} g_i^{(j)} = u_{t-i} g_i^{(j)} + u_{t-i} g_i^{(j)} + ... + u_{t-i} g_i^{(j)}, \qquad j = 1, 2.$$

Após a codificação, as seqüências $v^{(1)}$ e $v^{(2)}$ são multiplexadas em uma única seqüência $\mathbf{v}$, chamada palavra código. A seqüência $\mathbf{v}$ é dada por $\mathbf{v} = (v_0^{(1)} v_0^{(2)},\ v_1^{(1)} v_1^{(2)},\ ...\ )$.

2.3.1.2 Representação matricial

As seqüências geradoras podem ser dispostas em notação matricial

$$\mathbf{G} = \begin{bmatrix} g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_2^{(2)} & \cdots & g_m^{(1)} g_m^{(2)} & & \\ & g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & \cdots & g_m^{(1)} g_m^{(2)} & \end{bmatrix}$$

na qual as áreas em branco são consideradas como zeros. Assim, a equação de codificação pode ser escrita em notação matricial como

$$\mathbf{v} = \mathbf{uG}$$

onde todas as operações são módulo 2. $\mathbf{G}$ é chamada matriz geradora do código. Cada linha de $\mathbf{G}$ é idêntica à anterior deslocada de n posições para a direita. A matriz geradora é essencialmente uma representação estática da operação de um codificador convolucional, cuja principal virtude está em sua dinâmica, na maneira em que sua saída é função da entrada e dos bits armazenados nos registradores.

2.3.2 Análise no domínio da transformada

2.3.2.1 Polinômio gerador

Em qualquer sistema linear, operações envolvendo convolução no domínio do tempo podem ser substituídas por operações envolvendo polinômios no domínio da transformada. Cada seqüência nas equações de codificação pode ser substituída por um polinômio correspondente. Na representação polinomial de uma seqüência binária, os coeficientes do polinômio são a própria seqüência. Ou seja, as seqüências geradoras $g^{(j)}$ do código servem de coeficientes para os polinômios geradores $g^{(j)}(D)$,

$$\begin{aligned} g^{(1)}(D) &= g_0^{(1)}D^0 + g_1^{(1)}D + g_2^{(1)}D^2 + ... + g_m^{(1)}D^m \\ g^{(2)}(D) &= g_0^{(2)}D^0 + g_1^{(2)}D + g_2^{(2)}D^2 + ... + g_m^{(2)}D^m. \end{aligned}$$

A determinação dos polinômios geradores também pode ser feita diretamente a partir do diagrama do circuito. Para o codificador da figura 2.5, os polinômios geradores são $g^{(1)}(D) = 1 + D + D^2 + D^3$ e $g^{(2)}(D) = 1 + D^2 + D^3$.

As equações de codificação são representadas por

$$\begin{aligned} v^{(1)}(D) &= u(D)g^{(1)}(D) \\ v^{(2)}(D) &= u(D)g^{(2)}(D) \end{aligned}$$

e a palavra código torna-se

$$v(D) = v^{(1)}(D^2) + Dv^{(2)}(D^2),$$

onde o operador D pode ser interpretado como o operador de atraso, cujo expoente indica o número de unidades de tempo que um bit está atrasado em relação ao bit inicial daquela seqüência.

2.3.3 Representações Gráficas

2.3.3.1 Treliça

As propriedades estruturais de códigos convolucionais podem ser representadas por diagrama de treliça ou diagrama de estados. Define-se estado de um codificador como o conteúdo de seus registradores. Os estados são denominados $S_0, S_1, ...S_{2^m-1}$. No diagrama de treliça, cada ponto representa um estado e cada ramo unindo dois pontos representa uma transição de estados e a saída correspondente.

Uma treliça possui $N + m$ níveis, onde N é o comprimento da seqüência de informação e m é a ordem de memória do código. Normalmente, o estado inicial é S_0, o estado (000...). Os primeiros $m - 1$ níveis correspondem à saída do codificador do estado inicial e os $m - 1$ últimos correspondem ao retorno a este mesmo estado. Nestes duas partes da treliça, nem todos os estados podem ser alcançados. Já na porção central da treliça (onde $m - 1 \leqslant t \leqslant N$) todos os estados são atingíveis. Note-se que esta porção central da treliça apresenta uma estrutura periódica fixa.

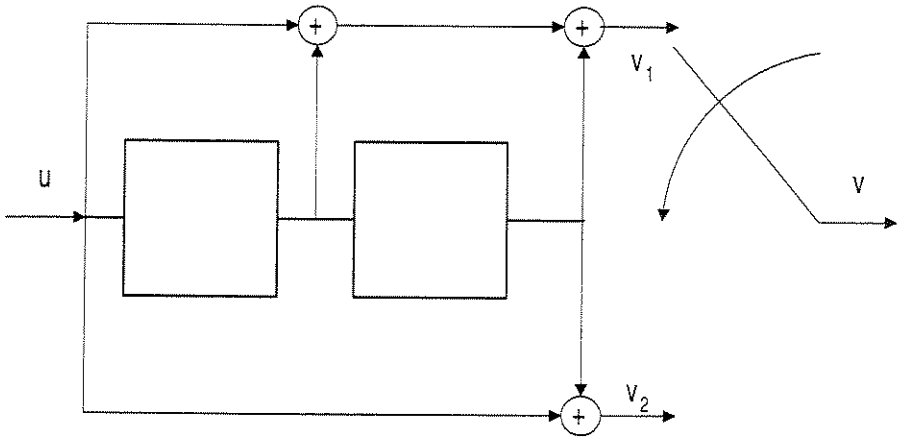

Figura 2.6: Codificador de 4 estados.

Para o codificador de 4 estados da figura 2.6, temos seu diagrama de treliça representado na figura 2.7. No item (a), a estrutura básica da treliça e em (b) a treliça para $t \geqslant 0$. Na estrutura em (a), chamada seção da treliça, temos à esquerda os nós que representam o estado atual do codificador. Os nós à direita representam o próximo estado. O ramo apresenta a entrada atual u e as duas saídas geradas, v_1 e v_2.

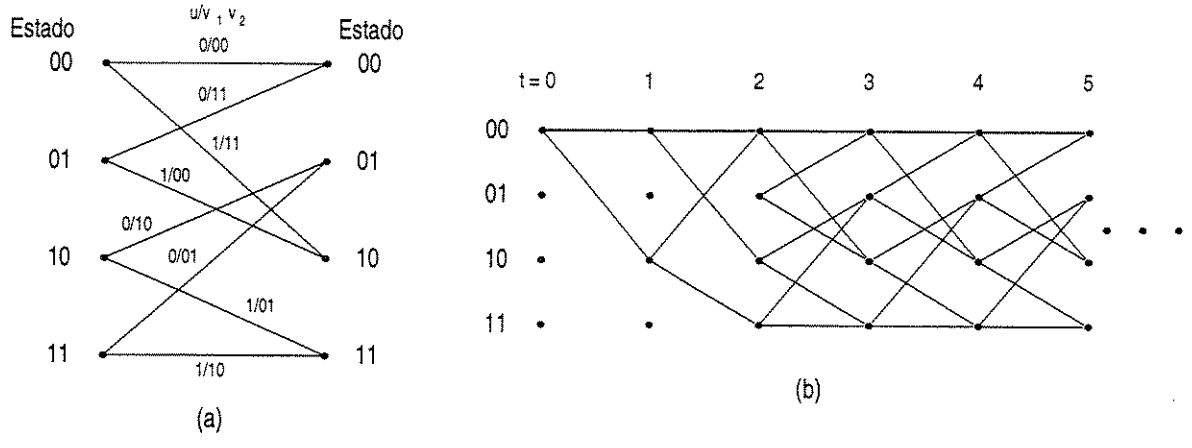

Figura 2.7: Diagrama de treliça para codificador de 4 estados.(a) Estrutura básica.(b) Diagrama para $t \geqslant 0$.

2.3.3.2 Diagrama de estados

Como o codificador convolucional é um circuito seqüencial, sua operação pode também ser descrita por meio de um diagrama de estados. Os círculos representam os estados e os ramos, as transições entre estados e a saída correspondente.

Pode-se obter o diagrama de estados diretamente do diagrama do codificador. Por exemplo, do codificador da figura 2.6 obtém-se o diagrama da figura 2.8. Outra forma de gerar o diagrama de estados é a partir de uma seção da treliça, colapsando-se os nós correspondentes da esquerda e da direita.

Conhecido o estado inicial do codificador, a palavra código correspondente a qualquer seqüência de entrada pode ser obtida através do diagrama de estados, seguindo o caminho determinado pelas entradas correspondentes. No nosso exemplo, assumindo-se que o codificador esteja inicialmente no estado $S_0 = 00$, a seqüência de entrada $u = (11010)$ irá gerar a saída $v = (11, 01, 01, 00, 10)$.

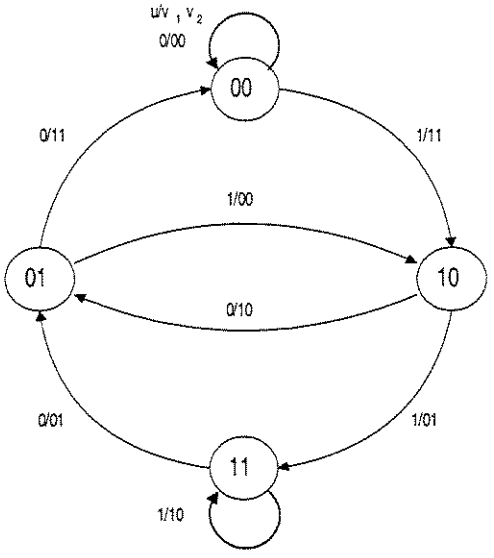

Figura 2.8: Diagrama de estados para o codificador da figura 2.6.

2.4 Decodificação por Máxima Verossimilhança

O decodificador deve produzir uma estimativa $\hat{\mathbf{u}}$ do sinal transmitido $\mathbf{u}$ baseado na sequência recebida $\mathbf{r}$. Como há uma correspondência de um para um entre a sequência de informação $\mathbf{u}$ e a palavra código $\mathbf{v}$, o decodificador trabalha com uma estimativa $\hat{\mathbf{v}}$ para $\mathbf{v}$. Portanto, $\hat{\mathbf{u}} = \mathbf{u}$ se e só se $\hat{\mathbf{v}} = \mathbf{v}$. Um erro de decodificação será cometido se, caso $\mathbf{v}$ tenha sido transmitida, tenhamos $\hat{\mathbf{v}} \neq \mathbf{v}$.

Um decodificador ótimo, isto é, que minimiza a probabilidade de erro $P(E)$, deve minimizar $P(\hat{\mathbf{v}} \neq \mathbf{v} \mid \mathbf{r})$ para todo $\mathbf{r}$, o que é equivalente a maximizar $P(\hat{\mathbf{v}} \neq \mathbf{v} \mid \mathbf{r}) = \frac{P(\mathbf{r} \mid \mathbf{v})P(\mathbf{v})}{P(\mathbf{r})}$. $P(\mathbf{r})$ independe da regra de decodificação, pois $\mathbf{r}$ é produzido anteriomente. Se todas as palavra código forem equiprováveis ($P(\mathbf{v})$ o mesmo para todo $\mathbf{v}$), basta maximizar $P(\mathbf{r} \mid \mathbf{v})$. Um decodificador que maximiza $P(\mathbf{r} \mid \mathbf{v})$ é chamado decodificador de máxima verossimilhança MLD (*maximum likelihood decoder*). Se as palavras código não forem equiprováveis, o MLD não é ótimo, mas como é difícil conhecer estas probabilidades com exatidão no receptor, o MLD é uma regra de decisão satisfatória.

No caso particular de um MLD decodificando num BSC, a estimativa escolhida é a palavra código $\mathbf{v}$ que minimiza a distância de Hamming entre $\mathbf{r}$ e $\mathbf{v}$, ou seja, é a palavra código que difere de $\mathbf{r}$ no menor número de posições. O MLD para o BSC é às vezes chamado de decodificador de distância mínima.

2.4.1 Decisão para canais sem memória

Seja uma seqüência de informação $\mathbf{u}$, de comprimento N, codificada na seqüência $\mathbf{v}$, de comprimento $\tau = n(N+m)$, e transmitida por um canal discreto sem memória DMC. A seqüência recebida é $\mathbf{r}$. O decodificador deve produzir uma estimativa $\hat{\mathbf{v}}$ para a palavra código $\mathbf{v}$ baseado na seqüência $\mathbf{r}$. O MLD deve escolher $\hat{\mathbf{v}}$ como a palavra código $\mathbf{v}$ que maximiza a função de

log-verossimilhança, $\log P(\mathbf{r} \mid \mathbf{v})$. Uma vez que, para um DMC (seção 2.2.1)

$$P(\mathbf{r} \mid \mathbf{v}) = \prod_{i=0}^{\tau-1} P(r_i \mid v_i) \tag{2.12}$$

segue que

$$\log P(\mathbf{r} \mid \mathbf{v}) = \sum_{i=0}^{\tau-1} \log P(r_i \mid v_i), \tag{2.13}$$

onde $P(r_i \mid v_i)$ são as probabilidades de transição de canal e supôs-se palavras código igualmente prováveis. Esta é a regra ótima para decisão sobre seqüências.

2.4.2 Métrica de máxima verossimilhança

A métrica utilizada no algoritmo de decisão por máxima verossimilhança é aquela que procura maximizar o valor da probabilidade de transição do canal $P(\mathbf{r} \mid \mathbf{v})$. Ou seja, para uma dada seqüência recebida $\mathbf{r}$, é a seqüência $\mathbf{v}$ que maximiza $P(\mathbf{r} \mid \mathbf{v})$.

Considere-se um canal AWGN e que o sistema estime com exatidão o estado do canal. Na seção 2.2.2 foi visto que a densidade de probabilidade de transição do canal é dada por:

$$P(\mathbf{r} \mid \mathbf{v}) = \left(\frac{1}{\sqrt{2\pi}\sigma_i}\right) \exp\left[-\frac{1}{2\sigma_i^2}(\mid \mathbf{r} - \mathbf{v} \mid^2)\right] \tag{2.14}$$

$$P(\mathbf{r} \mid \mathbf{v}) = \left(\frac{1}{\sqrt{2\pi}\sigma_i}\right) \exp[-\frac{1}{2\sigma_i^2}(\mid \mathbf{r} \mid^2 + \mid \mathbf{v} \mid^2 - 2\, \mathbf{r}^T\mathbf{v}\,)]. \tag{2.15}$$

Uma vez que o logaritmo é uma função monotonicamente crescente, maximizar $P(\mathbf{r} \mid \mathbf{v})$ em relação à seqüência $\mathbf{v}$ é equivalente a maximizar $\ln P(\mathbf{r} \mid \mathbf{v})$ sobre $\mathbf{v}$. A regra de decisão torna-se escolher a seqüência $\mathbf{v}$ que maximiza a métrica

$$v = \left(-\frac{1}{2\sigma_i^2}\right)(\mid \mathbf{r} \mid^2 + \mid \mathbf{v} \mid^2 - 2\, \mathbf{r}^T\mathbf{v}\,). \tag{2.16}$$

O primeiro e segundo termos da expressão 2.16 podem ser descartados, pois independem da seqüência $\mathbf{v}$ enviada (supondo modulação BPSK). O fator multiplicativo $-\frac{1}{2\sigma_i^2}$ também pode

ser descartado, pois é apenas um fator de escala, bem como o escalar -2 que multiplica $\mathbf{r}^T\mathbf{v}$. Portanto, a expressão da métrica torna-se, simplesmente,

$$\upsilon = \mathbf{r}^T\mathbf{v}. \tag{2.17}$$

Expandindo a expressão para os elementos do vetor, temos

$$\upsilon = \sum_{i=0}^{N-1} r_i v_i. \tag{2.18}$$

2.5 Algoritmo de Viterbi

Esta seção busca apenas introduzir o conceito do algoritmo de Viterbi, para contrastar com o algoritmo BCJR que será apresentado na seção seguinte. Uma descrição detalhada do algoritmo de Viterbi encontra-se em [Vitb67],[Vitb79].

Para cadeias de Markov de estados finitos, o algoritmo de Viterbi pode ser utilizado para encontrar a estimativa de máxima verossimilhança para uma seqüência de símbolos transmitidos.

Considere-se a transmissão uma seqüência de M palavras de k bits. O total de seqüências transmitidas possíveis é 2^{kM}. O receptor teria que avaliar as 2^{kM} seqüências possíveis para determinar a mais provável. Note-se que a complexidade do receptor cresce *exponencialmente* com o comprimento da seqüência, o que torna impraticável a decodificação de seqüências longas.

A principal atração do algoritmo de Viterbi está no fato de que sua complexidade de decodificação aumenta apenas *linearmente* com o comprimento da seqüência. Sua complexidade cresce exponencialmente com o número de estados, mas na ampla maioria das aplicações o "gargalo" está no comprimento da seqüência. Com isso, um receptor ótimo torna-se mais simples de implementar.

O algoritmo de Viterbi compreende uma regra iterativa de três passos para determinar a trajetória ótima ao longo da treliça que leva a um determinado estado em qualquer instante de tempo t. Seja $\mathbf{s}_t$ um estado particular do sistema no instante t e $d(\mathbf{s}_{t-1}, \mathbf{s}_t; y_t)$ a métrica associada

à transição do estado $\mathbf{s}_{t-1}$ para $\mathbf{s}_t$ (com valor observado y_t). Seja $D(\mathbf{s}_t)$ o custo associado à melhor trajetória que leva ao estado $\mathbf{s}_t$ e L o número de maneiras pelas qual a transição $\mathbf{s}_{t-1} \longrightarrow \mathbf{s}_t$ possa ocorrer. O algoritmo procura minimizar a função custo $D(\mathbf{s}_t)$. Os passos iterativos são:

1. Para cada valor de $\mathbf{s}_t$, determinar os L valores da métrica de transição $d(\mathbf{s}_{t-1}, \mathbf{s}_t; y_t)$ correspondentes aos L valores de $\mathbf{s}_{t-1}$ que possam levar a $\mathbf{s}_t$.

2. Para cada um dos possíveis L estados anteriores $\mathbf{s}_{t-1}$, adicionar a métrica $D(\mathbf{s}_{t-1})$, métrica da trajetória ótima até o valor $\mathbf{s}_{t-1}$, à métrica de transição $d(\mathbf{s}_{t-1}, \mathbf{s}_t; y_t)$ para o estado atual $\mathbf{s}_t$.

3. Para cada estado, identificar o caminho com a menor métrica como sendo o caminho sobrevivente A menor métrica é denominada $D(\mathbf{s}_t)$ e é armazenada, assim como os caminhos sobreviventes. $D(\mathbf{s}_t) = \min[D(\mathbf{s}_{t-1}) + d(\mathbf{s}_{t-1}, \mathbf{s}_t; y_t)]$.

2.6 Algoritmo BCJR

O problema de estimar-se as probabilidades *a posteriori* de estados e transições entre estados para canais sem memória foi abordado por Bahl *et alli* em um trabalho publicado em 1974 [Bahl74]. O artigo propôs um algoritmo para decodificação de códigos de bloco e convolucionais que minimiza a probabilidade de erro de símbolo de informação, o qual foi posteriormente denominado *algoritmo BCJR*, conforme as iniciais dos autores.

Na seção anterior foi apresentado um tradicional algoritmo de decodificação: o algoritmo de Viterbi. Este é um método de decodificação de máxima verossimilhança que minimiza a probabilidade de erro de seqüência para códigos convolucionais. Entretanto, o algoritmo de Viterbi não necessariamente minimiza a probabilidade de erro de símbolo (ou de bit). A proposta de Bahl *et alli* era um algoritmo de decodificação que, por construção, minimizasse

a probabilidade de erro de símbolo. Os autores partiram do problema de estimar-se as probabilidades *a posteriori* (APP) de estados e de transições entre estados de uma fonte de Markov, para então deduzir um algoritmo de decodificação para códigos lineares.

2.6.1 Visão Geral - Diagrama de Fluxo BCJR

A figura 2.9 mostra o diagrama de fluxo do algoritmo BCJR. Ele recebe do canal a versão ruidosa da palavra transmitida e tem como saída o vetor $\mathbf{APP_0}$ de probabilidades *a posteriori* de símbolo de informação.

O primeiro passo é atribuir os valores iniciais dos vetores $\boldsymbol{\alpha}_t$ e $\boldsymbol{\beta}_t$, vetores referentes aos passos recursivos direto e reverso, respectivamente. Estes vetores estão associados às probabilidades de estado da seqüência recebida. O passo seguinte é iniciar o laço recursivo direto. Os valores referentes aos bits recebidos no instante t são lidos do canal. O primeiro item a ser calculado é a matriz Gama, Γ, que refere-se às probabilidades de transição de canal e às transições no codificador. Alguns dos termos que compõem Γ são obtidos através de tabela de consulta, outros envolvem operações de ponto flutuante. De posse de Γ_t, o valor de Γ no instante t, pode-se atualizar $\boldsymbol{\alpha}_t$ através da equação $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1}\Gamma_t$. Este passo recursivo direto prossegue até que toda a seqüência recebida tenha sido processada, ou seja, até $t = \tau$. O comprimento τ da palavra recebida é dado pela soma do comprimento da seqüência de informação com os bits de terminação da treliça.

O laço reverso vem a seguir. A atualização de $\boldsymbol{\beta}_t$ é feita através equação $\boldsymbol{\beta}_t = \Gamma_t\boldsymbol{\beta}_{t+1}$. A matriz Γ_t, já calculada no laço anterior para obtenção de $\boldsymbol{\alpha}_t$, é novamente utilizada. Obtidos $\boldsymbol{\alpha}_t$ e $\boldsymbol{\beta}_t$, pode-se calcular $\boldsymbol{\sigma}_t$, q indica as probabilidades de transição na treliça. A probabilidade de uma transição do estado i para o estado j é dada por $\sigma_t(i) = \alpha_{t-1}(i)\Gamma_t(i,j)\beta_t(j)$.

Ao se obter os valores de $\boldsymbol{\sigma}_t$ para todo instante t, são também calculadas as probabilidades *a posteriori* do símbolo de entrada, u_t. Pondera-se então o somatório dos $\boldsymbol{\sigma}_t$ correspondentes a

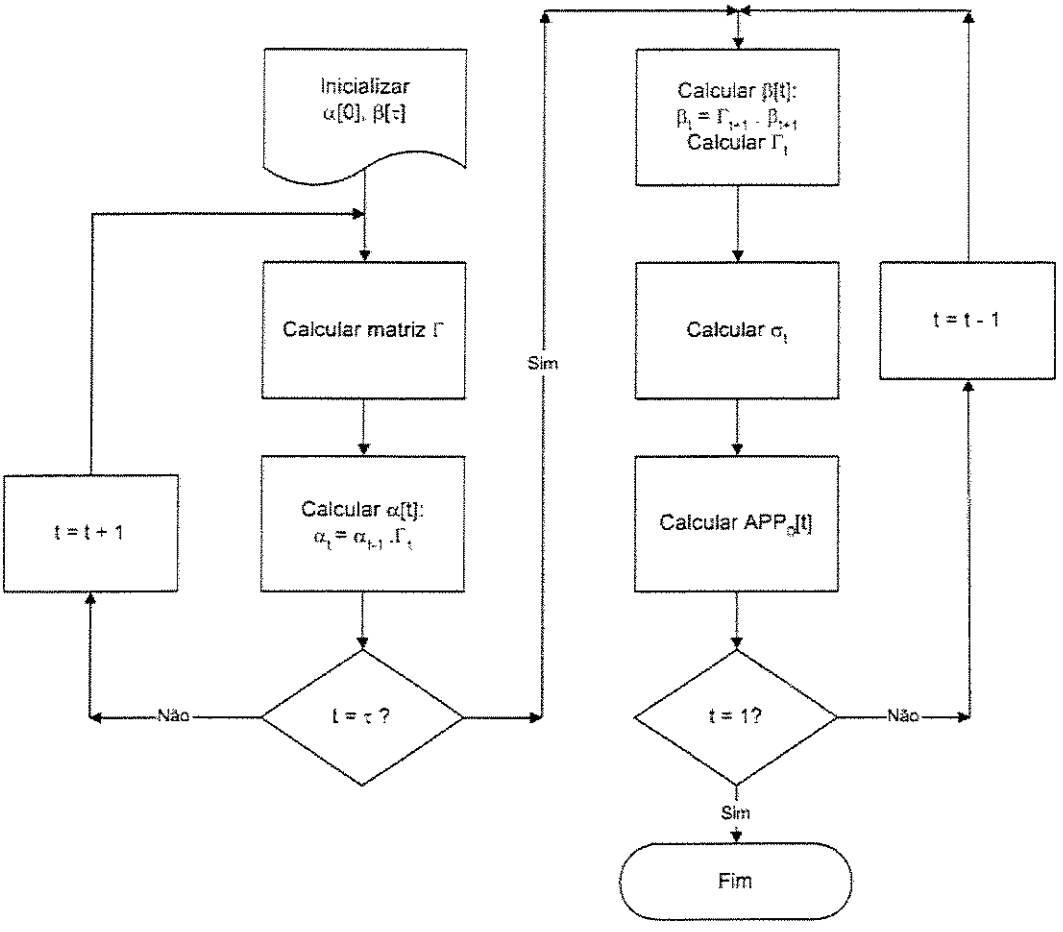

Figura 2.9: Diagrama de fluxo para o algoritmo BCJR.

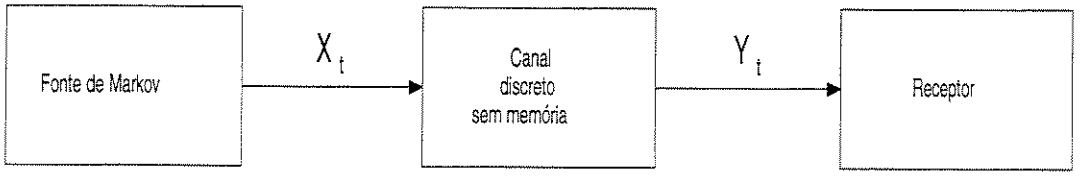

Figura 2.10: Sistema de transmissão simplificado.

transições causadas por entrada zero pelo somatório de todos os σ_t para obter-se a probabilidade de $u_t = 0$. Para obter-se a correspondente probabilidade de $u_t = 1$, basta considerar as transições causadas por entrada unitária.

2.6.2 Fonte de Sinal

Considere-se um sistema de transmissão simplificado (figura 2.10), composto por uma fonte de sinal, canal discreto sem memória e receptor (decodificador), no qual a fonte é um processo de Markov de tempo discreto e número de estados finito. Os M estados distintos da fonte de Markov são indexados pelo inteiro m, $m = 0, 1, ..., M-1$. O estado da fonte no instante de tempo t é denotado por S_t e sua saída por X_t. As transições de estado na fonte de Markov são indicadas pelas probabilidades de transição de fonte:

$$p_t(j \mid i) = P[S_t = j \mid S_{t-1} = i] \tag{2.19}$$

e a saída pelas probabilidades:

$$q_t(X \mid i, j) = P[X_t = X \mid S_{t-1} = i;\ S_t = j] \tag{2.20}$$

onde X pertence ao alfabeto de saída.

A fonte de Markov inicia no estado $S_0 = 0$, e produz uma sequência de saída $X_1^\tau = X_1, ..., X_\tau$, terminando no estado $S_\tau = 0$. X_1^τ é enviado por um canal sem memória ruidoso cuja saída é a sequência Y_1^τ, a sequência recebida, que é definida de maneira similar a X_1^τ como $Y_1^\tau = Y_1, ..., Y_\tau$.

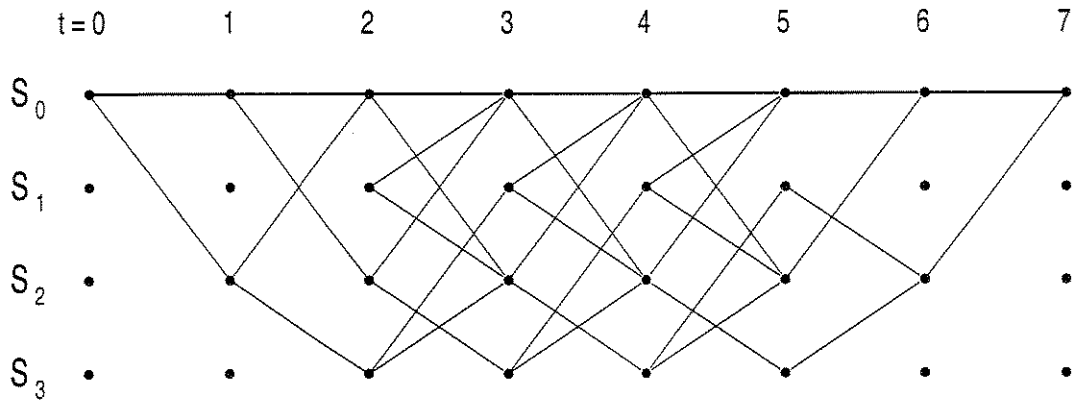

Figura 2.11: Exemplo de diagrama de treliça para uma fonte de Markov de 4 estados.

O objetivo do decodificador é examinar Y_1^τ e estimar as probabilidades *a posteriori* dos estados e transições da fonte de Markov, ou seja, as seguintes probabilidades condicionais:

$$P[S_t = i \mid Y_1^\tau] \tag{2.21}$$

$$P[S_{t-1} = i;\ S_t = j \mid Y_1^\tau] \tag{2.22}$$

2.6.3 Diagrama de Treliça

A sequência de estados de uma fonte de Markov variante ou invariante no tempo pode ser descrita através de um diagrama de treliça. Os nós representam os estados e os ramos representam as transições de probabilidade não-nula entre estados. O diagrama de treliça mostra a progressão no tempo das seqüências de estados (figura 2.11). Para cada sequência S_1^τ há um caminho único através do diagrama de treliça e vice-versa. Associada a cada nó da treliça temos a correspondente probabilidade *a posteriori* de estado e associada a cada ramo na treliça temos a correspondente probabilidade *a posteriori* de transição. O decodificador deve examinar a sequência recebida Y_1^τ e calcular essas probabilidades.

2.6.4 Probabilidades de Transição entre Estados

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 as probabilidades

$$P[S_t = i \mid Y_1^\tau] \tag{2.23}$$

ou seja, a probabilidade de que o codificador estava no estado i no instante t, dado o conjunto de símbolos recebidos do canal. Para codificadores não-recursivos, estas probabilidades são suficientes para obter-se as probabilidades dos bits de informação. Para codificadores recursivos, isso não é válido, sendo necessário obter-se a probabilidade de transição de estados, e desta a probabilidade da entrada correspondente. Ou seja, é necessário calcular

$$P[S_{t-1} = i;\ S_t = j \mid Y_1^\tau] \tag{2.24}$$

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 (2.24) nada mais é que $\boldsymbol{\sigma}_t$ escalonada 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

$$\Gamma_t(i,j) = P[S_t = j;\ Y_t \mid S_{t-1} = i] \tag{2.25}$$

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

$$\alpha_t(i) = P[S_t = i;\ Y_1^t],\ 1 \leqslant t \leqslant \tau \tag{2.26}$$

$$\beta_t(j) = P[Y_{t+1}^{\tau} \mid S_t = j],\ 1 \leqslant t \leqslant \tau - 1 \tag{2.27}$$

Os valores iniciais dos vetores $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$ estão associados ao estado inicial e ao estado final do codificador. Se este inicia e termina no estado 0, os vetores assumem, em $\boldsymbol{\alpha}_0$ e $\boldsymbol{\beta}_\tau$, os seguintes estados iniciais:

$$\alpha_0(0) = 1, \text{ e } \alpha_0(i) = 0,\ i \neq 0 \tag{2.28}$$

$$\beta_\tau(0) = 1, \text{ e } \beta_\tau(i) = 0,\ i \neq 0 \tag{2.29}$$

Os passos do algoritmo BCJR são:

1. Atribuir os valores iniciais aos vetores associados aos passos recursivos.

2. Obter a matriz de transição de estados Γ_t.

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

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1}\Gamma_t,\ t = 1, ..., \tau. \tag{2.30}$$

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

$$\boldsymbol{\beta}_t = \Gamma_t \boldsymbol{\beta}_{t+1},\ \ t = \tau - 1, ..., 1. \tag{2.31}$$

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

$$\sigma_t(i,j) = \alpha_{t-1}(i) \cdot \Gamma_t(i,j) \cdot \beta_t(j) \tag{2.32}$$

A soma de todos os elementos de um vetor $\boldsymbol{\sigma}_t$ resulta $P[Y_1^\tau]$ e normalizando (2.32) 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 $\boldsymbol{\sigma}_t(i,j)$ cuja transição corresponda ao bit de entrada considerado, escalonados 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.

Entretanto, o algoritmo BCJR requer grande capacidade de armazenamento e esforço computacional considerável. Todo o conjunto de vetores $\boldsymbol{\alpha}_t$ deve ser armazenado, o que requer aproximadamente $2^{km} \cdot \tau$ posições de memória, onde k é a dimensão do bloco de entrada, m é o comprimento da memória e τ é o comprimento da seqüência recebida. O custo de armazenagem cresce exponencialmente com o número de estados ($M = 2^m$) e linearmente com o comprimento da seqüência. O número de operações para se determinar $\boldsymbol{\alpha}_t$, para cada t, é $M \cdot 2^k$ produtos e $M \cdot 2^k$ adições.

Capítulo 3

Códigos Turbo

Ao final do capítulo anterior foi exposto o algoritmo BCJR, um algoritmo de decodificação que minimiza a probabilidade de erro de símbolo de informação. Neste capítulo serão apresentados os códigos turbo, um esquema de codificação/decodificação que pode chegar bem próximo da capacidade de canal. Decodificadores turbo atuam de forma iterativa e geralmente utilizam o algoritmo BCJR em seus decodificadores componentes.

3.1 Visão Geral

Um sistema codificador para códigos turbo é construído a partir de dois ou mais códigos simples. Estes códigos constituintes podem ser do tipo convolucional sistemático recursivo ou de bloco. Supondo dois codificadores componentes, a sequência de informação é processada pelo codificador 1, entrelaçada de alguma forma e enviada para o segundo codificador. Os bits são então transmitidos a uma dada taxa. Caso haja mais codificadores, entrelaçadores adicionais devem ser utilizados. Cada codificador adicional requer um entrelaçador a mais. Em geral, utilizam-se dois codificadores em paralelo (figura 3.1).

No receptor, de forma análoga, há dois decodificadores, cada um sendo responsável

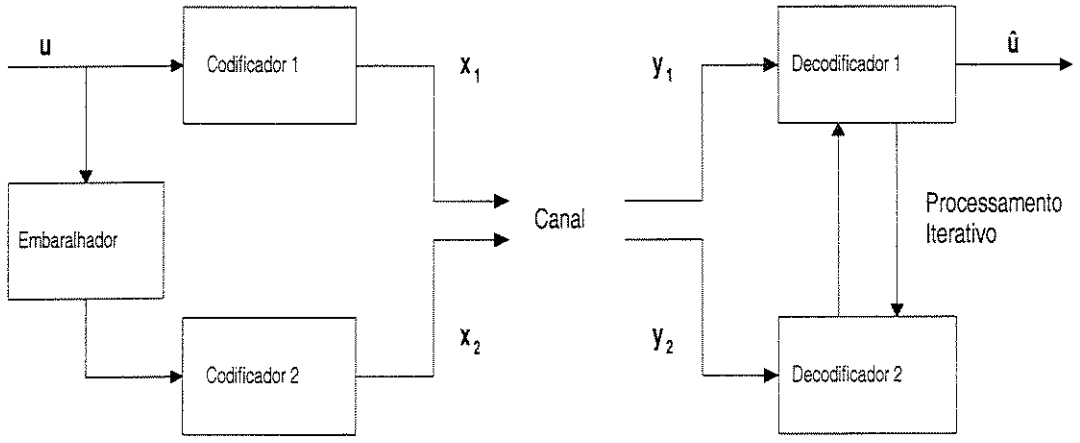

Figura 3.1: Sistema Codificador/Decodificador em paralelo.

pela decisão referente ao codificador correspondente. Entre os decodificadores, a operação de entrelaçamento é revertida.

Uma característica de destaque na decodificação turbo é que o processo de decodificação pode ser realizado mais de uma vez. Na primeira iteração, o segundo decodificador atualiza as probabilidades *a posteriori* de bits já feitas pelo primeiro decodificador. Na segunda iteração, o primeiro decodificador irá novamente atualizar estas probabilidades, a partir do que já foi realizado na etapa anterior, e assim por diante. Para que se atinja um desempenho satisfatório, podem ser necessárias várias iterações de decodificação. As operações de entrelaçamento/desentrelaçamento entre os dois decodificadores servem para dispersar os erros introduzidos pelo canal e para que cada decodificador trabalhe com sua própria versão da palavra recebida correspondente à sua respectiva palavra codificada.

Para que o processo iterativo trabalhe de forma otimizada, é necessário que os decodificadores componentes gerem como saída N métricas de decisão suave, correspondentes à verossimilhança de cada bit na sequência de informação. Além disso, devem receber como entrada N métricas de decisão suave. O cálculo da saída deve levar em conta, além da própria redundância do código, esta entrada de comprimento N. Esta capacidade de utilizar entradas de decisão suave (para

gerar saídas de decisão suave) é característica da decodificação turbo, e contribui decisivamente para seu bom desempenho. Entretanto, também contribui para aumentar a complexidade computacional do processo de decodificação.

O fato que a saída de um decodificador alimenta a entrada do próximo decodificador é de certa forma análogo ao papel de um turbocompressor em um motor de combustão interna. O turbocompressor utiliza a exaustão do motor (uma saída) para bombear ar para o cilindro, enriquecendo a "entrada". Desta analogia vem o nome dos códigos turbo.

3.2 Codificador

3.2.1 Introdução

Em geral, um sistema turbo pode utilizar como componentes tanto codificadores de bloco quanto convolucionais sistemáticos. Entretanto, devido ao modo de operação do decodificador, que realiza separadamente uma decodificação ótima dos codificadores componentes, os códigos convolucionais são mais vantajosos devido à existência de algoritmos de decodificação suave MAP. E enquanto códigos de bloco podem apresentar uma treliça com um grande número de estados, códigos convolucionais apresentam um diagrama de treliça simples e de fácil obtenção, além de um número de estados menor, fixado pela memória do codificador.

3.2.2 Codificador Sistemático Recursivo

Codificadores recursivos realimentam as saídas dos registradores na entrada. Um exemplo deste tipo de codificador está representado na figura 3.2. A entrada do primeiro registrador, m_0, é obtida por uma combinação linear da entrada do codificador u com as saídas dos demais registradores. Na figura, temos que m_0 é obtida por $m_0 = u + m_1 + m_2 + m_3$. A figura 3.3 mostra o diagrama de estados para o codificador recursivo apresentado.

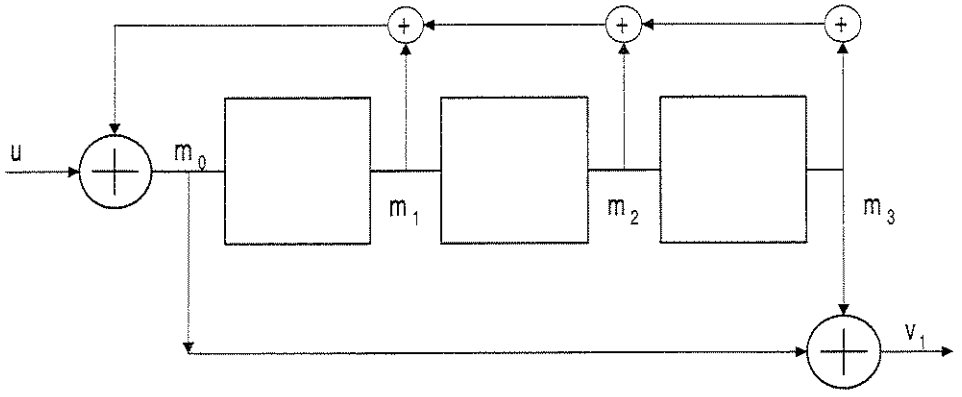

Figura 3.2: Codificador recursivo.

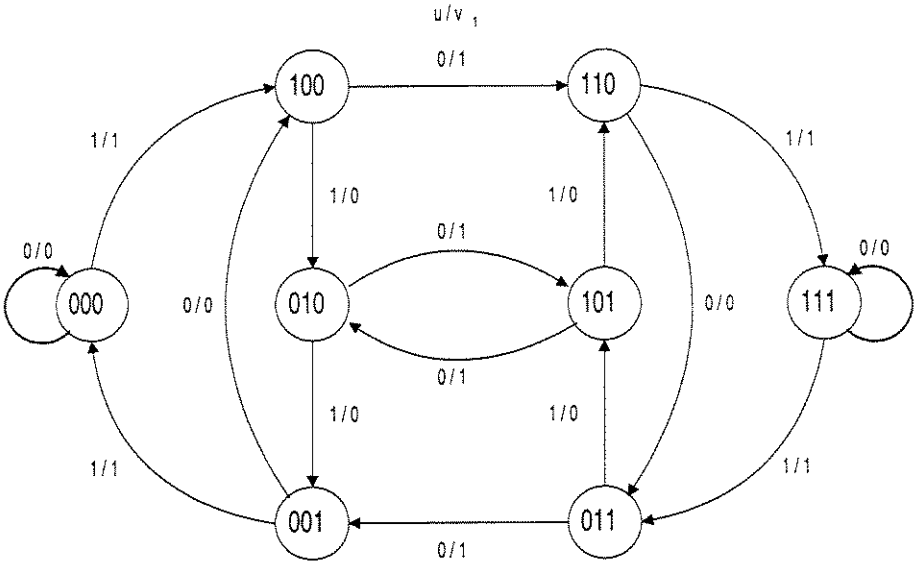

Figura 3.3: Diagrama de estados para o codificador recursivo da figura 3.2.

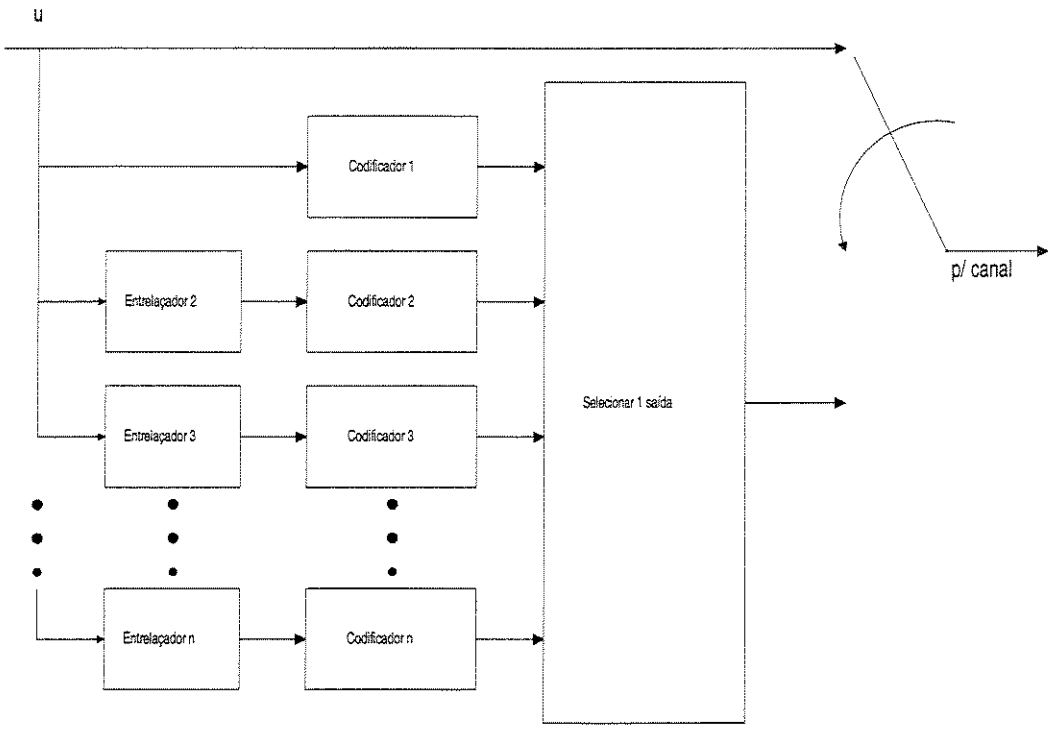

Figura 3.4: Sistema codificador em paralelo genérico utilizado em códigos turbo. Os codificadores são do tipo RSC.

O tipo de codificador convolucional mais utilizado em códigos turbo é o codificador sistemático recursivo, RSC (*recursive systematic encoder*). Há duas razões principais para a utilização de RSCs em sistemas turbo, em vez dos tradicionais codificadores diretos. A primeira razão é a saída sistemática, o que torna uma das saídas idêntica à entrada. Em um esquema genérico de código turbo, considerando-se todas as saídas dos codificadores componentes, as saídas sistemáticas dos codificadores são idênticas, a menos de uma permutação. Uma vez que o envio de todas essas saídas nada mais é que um código de repetição, pode-se simplesmente puncionar todas as saídas sistemáticas exceto uma, como na figura 3.4. A remessa de um pacote de dados mais compacto evita a expansão de largura de banda que resultaria do envio de tantas saídas sistemáticas.

A outra razão para utilizar-se RSCs deve-se a sua natureza recursiva e a seu papel no aumento da distância mínima do sistema. Um codificador direto que recebesse como entrada a sequência de informação ...00100... apresentaria uma resposta curta e com poucos bits 1, e a palavra codificada resultante teria peso pequeno. A fim de tirar vantagem dos múltiplos codificadores unidos por entrelaçadores para aumentar a distância mínima, faz-se necessário uma resposta ao impulso de peso maior possível. A resposta curta produzida por um codificador direto limita severamente a distância mínima, independentemente do trabalho do entrelaçador em dispersar os bits.

Já um codificador recursivo apresenta resposta ao impulso infinita. A sequência ...00100... não leva um RSC ao estado nulo, a resposta produzida teria peso infinito, a não ser pelo truncamento da treliça. Isto significa que uma sequência com apenas um bit 1 não é mais um caso limitante para a distância mínima de um sistema turbo. No pior caso, a sequência a ser entrelaçada irá conter dois bits 1, sendo possível então utilizar um entrelaçador que disperse os bits e aumente o peso das palavras codificadas, provocando um aumento na distância mínima, bem como no desempenho do código. Na seção 3.3 analisaremos como o espectro de pesos influencia o desempenho.

O diagrama de estados da figura 3.3 ilustra este comportamento. A sequência 100... faz com que o codificador circule pelos estados $100 - 110 - 011 - 001$ com saída 101010..., de peso infinito. A sequência de menor peso que leva o codificador para o estado nulo possui peso 2. A sequência 101 leva o codificador para o estado 111 com saída nula, enquanto a sequência 10001 o leva para o estado 000 com saída nula.

3.2.3 Sistema implementado

Para o presente trabalho, foi utilizado um sistema codificador composto por dois codificadores convolucionais recursivos idênticos concatenados em paralelo através de um

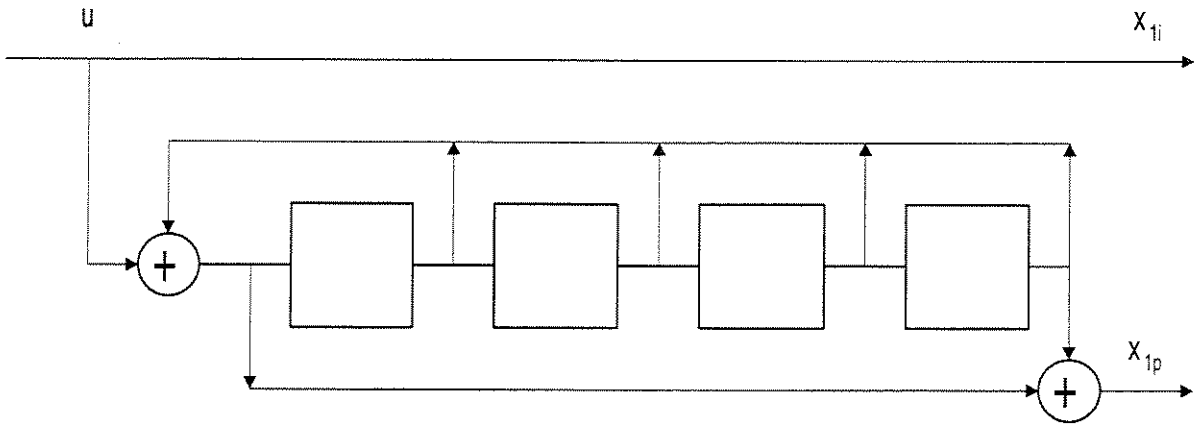

Figura 3.5: Codificador Sistemático Recursivo (RSC), taxa 1/2, $m = 4$.

entrelaçador aleatório. A figura 3.5 mostra um dos codificadores componentes, um RSC de taxa 1/2 e memória $m = 4$. Os somadores da realimentação foram omitidos para maior simplicidade.

Codificadores recursivos são geralmente descritos por uma razão entre polinômios geradores (g_a/g_b), que representam os polinômios de paridade e de realimentação. Esta razão é geralmente dada em notação octal. A descrição em termos de polinômios geradores foi apresentada na seção 2.3.2.

Por exemplo, o codificador implementado (figura 3.5) possui um polinômio de realimentação da forma $g_b(D) = 1D^0+1D+1D^2+1D^3+1D^4$, pois todas as ordens de memória estão presentes. A representação octal dos coeficientes do polinômio fica 37_8. O polinômio de paridade deste codificador é $g_a(D) = 1D^0+0D+0D^2+0D^3+1D^4$, cuja representação octal de seus coeficientes é 21_8. Então, este codificador pode ser descrito como um codificador recursivo de 16 estados com gerador (21/37).

A figura 3.6 mostra o sistema paralelo utilizado. O sistema apresenta taxa 1/4 e ordem de memória $m = 4$. O entrelaçador recebe a sequência de informação $\mathbf{u}$ e gera a sequência $\mathbf{u}'$ para o segundo codificador, entrelaçando aleatoriamente os bits de $\mathbf{u}$.

O primeiro codificador componente atua diretamente sobre a sequência de informação $\mathbf{u}$,

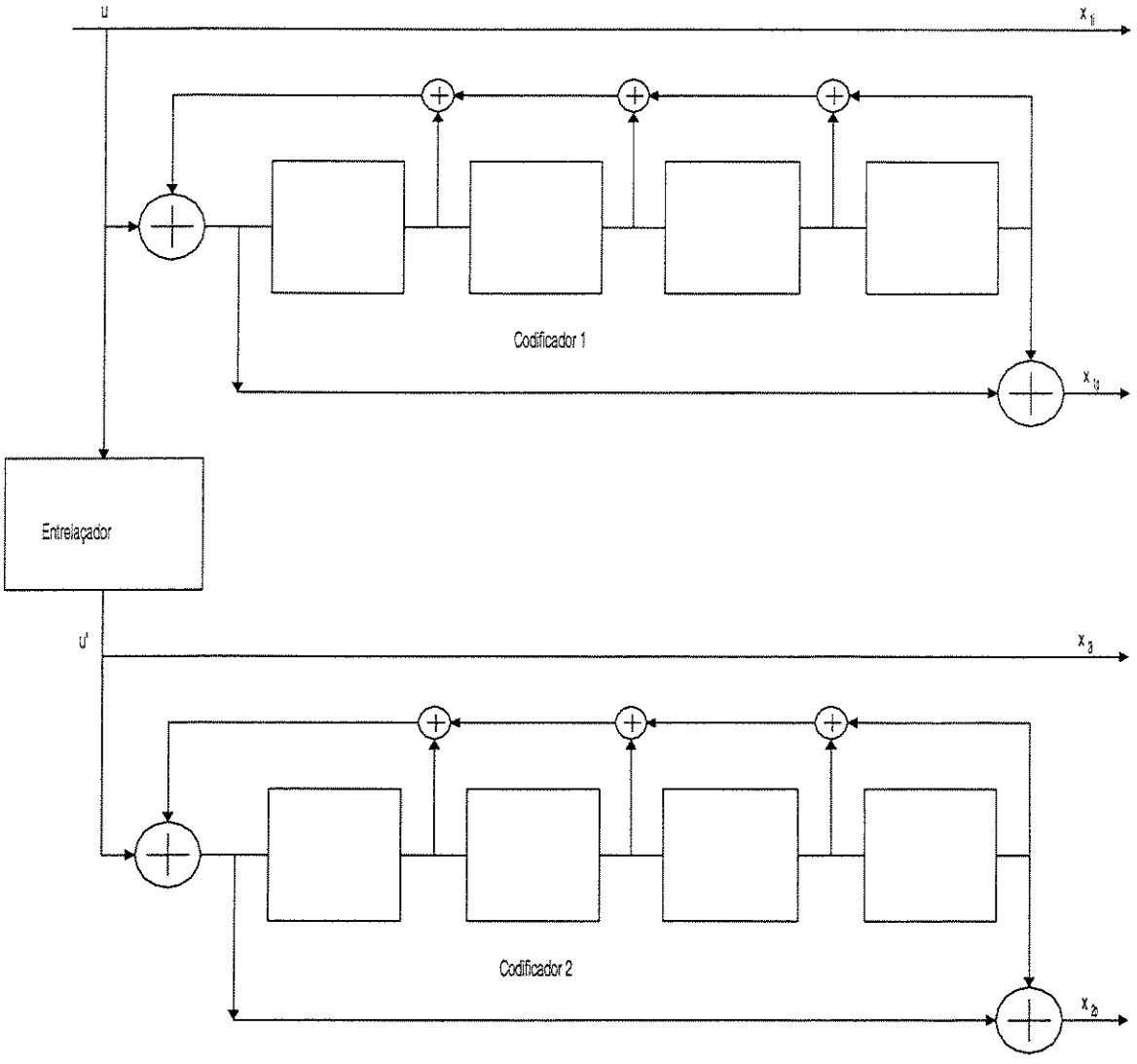

Figura 3.6: Sistema codificador paralelo.

de comprimento N, e produz as sequências de saída $\mathbf{x}_{1i}$ e $\mathbf{x}_{1p}$. O segundo codificador atua sobre a sequência reordenada $\mathbf{u}'$, gerada através de um enterlaçador de comprimento N, e produz as sequências de saída $\mathbf{x}_{2i}$ e $\mathbf{x}_{2p}$. Os codificadores utilizados não precisam ser idênticos. O entrelaçador realiza uma permutação dos N elementos da sequência $\mathbf{u}$.

Considerou-se um canal de transmissão AWGN (seção 2.2.2). O esquema está apresentado na figura 3.7. O diagrama referente ao codificador 2 é idêntico. n_{1i} e n_{1p} são amostras independentes de um processo gaussiano de média nula e variância unitária. ρ é a relação sinal-ruído (SNR), $\rho = \sqrt{2E_s/N_o} = \sqrt{2rE_b/N_o}$. $\mathbf{u}$ é a seqüência de informação, $\mathbf{x}$ é a seqüência transmitida pelo canal ($\mathbf{x}_{1i}$, $\mathbf{x}_{1p}$ e $\mathbf{x}_{2i}$, $\mathbf{x}_{2p}$ são as subseqüências referentes aos codificadores 1 e 2) e $\mathbf{y}$ é a seqüência recebida (subseqüências $\mathbf{y}_{1i}$, $\mathbf{y}_{1p}$ e $\mathbf{y}_{2i}$, $\mathbf{y}_{2p}$).

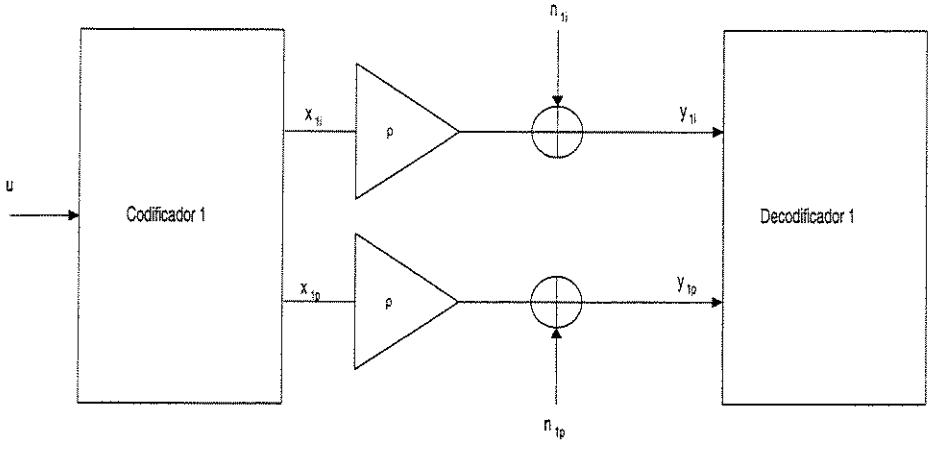

Figura 3.7: Módulo Codificador/Decodificador e canal.

3.2.4 Terminação da Treliça

A fim de terminar a treliça do código dado, levando-a de volta ao estado zero, não basta igualar a zero os m últimos bits de informação, pois os codificadores são recursivos. Logo, a sequência de final depende do do estado de cada codificador componente após os N bits de informação. O esquema proposto por Divsalar [Divs95] para solucionar isso consiste em, para cada codificador, utilizar o resultado do ou-exclusivo entre os bits presentes na memória do codificador como valor para o próximo bit de terminação. Assim, garante-se que o próximo bit será zero, pois ambas as parcelas da operação ou-exclusivo serão iguais. A figura 3.8 mostra como isso pode ser obtido. A chave fica na posição A pelos primeiros N ciclos do relógio e na posição B para os m ciclos restantes. O decodificador não assume conhecimento dos m bits finais. Portanto, o comprimento total da seqüência recebida é $\tau = N + m$. O mesmo mecanismo de terminação pode ser utilizado para codificadores em paralelo com taxas distintas.

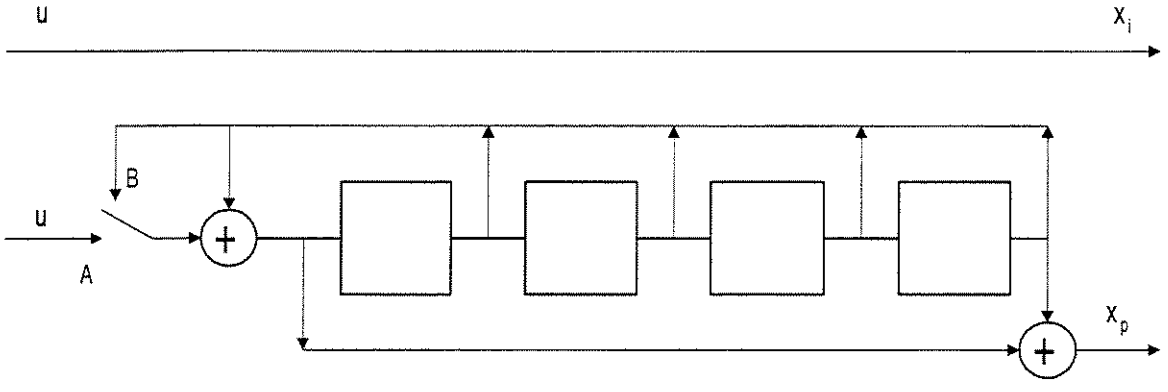

Figura 3.8: Terminação da treliça.

3.3 Entrelaçador

A escolha do entrelaçador é importante para o desempenho do código, pois afeta a distância mínima d e a distribuição de pesos do código. Visto que o codificador da figura 3.5 utiliza a mesma sequência de informação para gerar as palavras código x_1 e x_2, isso pode ocasionar palavras de peso pequeno. É necessário, portanto, escolher um entrelaçador de forma a evitar o emparelhamento de palavras código de peso baixo de um codificador componente com palavras código de peso baixo do outro. Além disso, o fato dos codificadores serem recursivos também colabora para melhorar a distribuição de pesos e aumentar a distância mínima, pois eles possuem resposta a impulso infinita. Por exemplo, a sequência de informação ...001000... produziria um resposta de peso infinito, não fosse pelo procedimento de truncar a treliça.

A distância mínima não é o único dado importante sobre um código, a distribuição de pesos é relevante, principalmente em relações sinal-ruído moderadas. Para códigos de comprimento pequeno, com $N = 16$, ainda é possível gerar a distribuição de pesos completa e investigar os efeitos da escolha do entrelaçador sobre a distribuição. Divsalar [Divs95] estudou o efeito da escolha do entrelaçador na distribuição de pesos das palavras código, comparando diversos tipos de entrelaçadores: o entrelaçador de bloco, o entrelaçador reverso, o entrelaçador aleatório e o sistema sem entrelaçador. O entrelaçador aleatório obteve o melhor resultado, em termos de distância mínima e distribuição de pesos, em relação aos demais.

Por outro lado, Herzberg [Herz98] propôs um entrelaçador de bloco reverso que pode atingir desempenho melhor que o entrelaçador aleatório para palavras de comprimentos curtos e valores de probabilidade de erro de bit (P_b) na faixa de 10^{-7}.

3.3.1 Distribuição de Pesos

Procederemos agora a uma breve análise da distribuição de pesos do código e o efeito causado pelo entrelaçador aleatório. Dado o espectro de distribuição de pesos do código, seja D_m, o coeficiente correspondente às palavras código com um peso total de m, definido como

$$D_m = \sum_{j=m-w} \frac{w}{N} A_{w,j}^{C_P} \tag{3.1}$$

onde $A_{w,j}^{C_P}$ é a função de enumeração de pesos do código C_P, w é peso da sequência de informação, j é o peso da palavra de paridade e N é o comprimento do entrelaçador. Utilizando-se o limitante da união, a probabilidade de erro de bit para a decodificação por máxima verossimilhança pode ser limitada superiormente por

$$P_b \leq \sum_m D_m Q(\frac{\sqrt{m}d}{2\sigma}) \tag{3.2}$$

onde d é a distância euclidiana entre sinais na constelação, σ^2 é a variância do ruído (AWGN) e a função $Q(x)$ é dada por $Q(x) = (\frac{1}{\sqrt{2\pi}}) \int_x^\infty e^{-y^2/2} dy$. O limitante da união é apertado para relações sinal-ruído moderadas e elevadas, geralmente quando P_b é menor que 10^{-4}.

Da equação 3.2 pode-se notar que o valor de P_b varia diretamente com o conjunto de coeficientes D_m. Uma redução nos valores dos coeficientes causa uma redução em P_b. Da equação 3.1, nota-se que a redução nos coeficientes D_m pode ser obtida com o aumento do comprimento do entrelaçador N. Logo, o uso de entrelaçadores longos contribui para um melhor desempenho do sistema. A maioria dos codificadores turbo utiliza longos entrelaçadores aleatórios de forma a atingir um bom ganho de codificação na região de $P_b = 10^{-5}$. Portanto, o entrelaçador implementado foi do tipo aleatório, por apresentar o melhor desempenho para seqüências de comprimento longo, visto que as seqüências simuladas possuíam comprimento de 1024 bits.

3.4 Decodificação Turbo

O sistema decodificador turbo é apresentado na figura 3.9. $DEC1$ representa o decodificador 1, que recebe os dados referentes ao codificador 1 do sistema paralelo, enquanto $DEC2$ representa o decodificador 2, correspondente ao codificador 2. Caso os codificadores não sejam idênticos, os decodificadores não o serão. Cada um deles é preparado para lidar com os dados do codificador correspondente. O algoritmo BCJR, descrito na seção 2.6, pode ser utilizado na implementação destes decodificadores para o caso de canais sem memória.

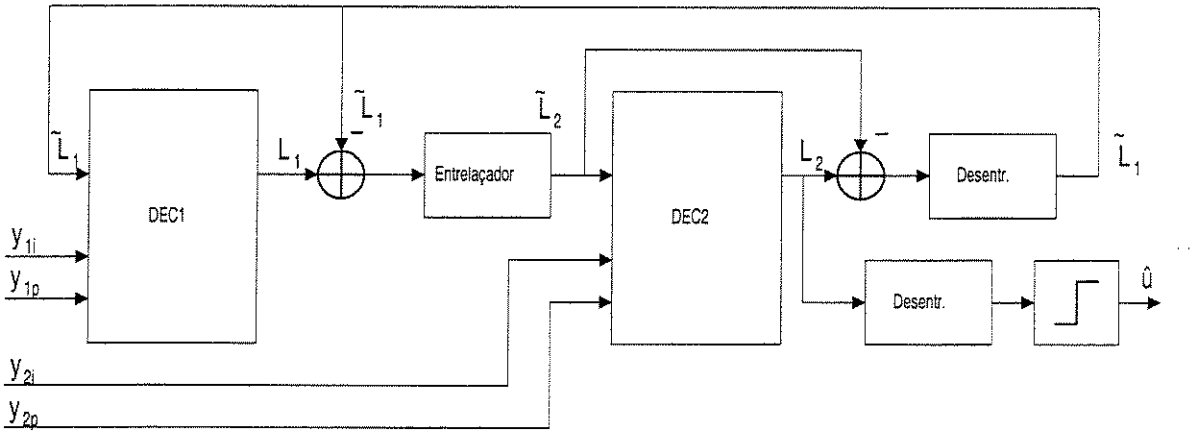

Figura 3.9: Estrutura de um decodificador turbo.

3.4.1 Probabilidades *a posteriori*

3.4.1.1 Razão de Verossimilhança

Considere uma seqüência de informação $\mathbf{u}$ cujas amostras u_k são variáveis aleatórias selecionadas do alfabeto $\{-1, +1\}$. Considere $\mathbf{x}$ a seqüência transmitida pelo canal, composta pelas subseqüências $\mathbf{x}_1$ e $\mathbf{x}_2$, referentes aos codificadores 1 e 2. A seqüência recebida é $\mathbf{y}$, composta pelas subseqüências $\mathbf{y}_1$ e $\mathbf{y}_2$.

Considere um algoritmo máximo *a posteriori* (MAP) que produza como saída uma razão de verossimilhança $L(k)$ definida por:

$$L(k) \triangleq \log \frac{P(u_k = +1|\mathbf{y})}{P(u_k = -1|\mathbf{y})}. \tag{3.3}$$

O sinal de $L(k)$ representa uma estimativa, $\widehat{u}_k$, de u_k, enquanto sua magnitude $\mid L(k) \mid$ é a confiabilidade dessa estimativa, conforme sugerido em [Hagen94].

3.4.1.2 Cálculo das probabilidades *a posteriori*

Para o sistema codificador paralelo apresentado na figura 3.6, a regra de decodificação ótima minimiza a probabilidade de erro de bit maximizando $P(u_k|\mathbf{y_1}, \mathbf{y_2})$. Ressalte-se que esta regra é distinta da regra de máxima verossimilhança, que minimiza a probabilidade de erro de seqüência através de $P(\mathbf{u}|\mathbf{y_1}, \mathbf{y_2})$. Uma vez que a regra é demasiado complexa para se calcular, recorre-se a uma regra de decodificação quase-ótima [Berr93] [Hagen94], a qual utiliza separadamente as observações das seqüências $\mathbf{y_1}$ e $\mathbf{y_2}$.

Na decodificação turbo, cada decodificador componente é um decodificador MAP cuja saída são as probabilidades *a posteriori* $P(u_k|\mathbf{y_i}, \widetilde{\mathbf{u}}_\mathbf{i})$, $i = 1, 2$ ou, de forma equivalente, a razão de verossimilhança $L_i(k) = \log \frac{P(u_k=+1|\mathbf{y_i}, \widetilde{\mathbf{u}}_\mathbf{i})}{P(u_k=-1|\mathbf{y_i}, \widetilde{\mathbf{u}}_\mathbf{i})}$. Os valores $\widetilde{\mathbf{u}}_\mathbf{i}$ são chamados "informação extrínseca", sendo fornecidos pelo estágio anterior de decodificação e são usados para se calcular novas probabilidades *a priori* para a seqüência de informação $\mathbf{u}$. Nesta descrição, $\widetilde{\mathbf{u}}_1$ é gerado

pelo decodificador 2 para servir de entrada para o decodificador 1, enquanto $\widetilde{\mathbf{u}}_2$ é gerado pelo decodificador 1.

Para que se calcule as probabilidades de transição na decodificação MAP é necessário gerar as probabilidades $P(u_k|\widetilde{u}_{i,k})$ ou, de forma equivalente, $\log \frac{P(u_k=+1|\widetilde{u}_{i,k})}{P(u_k=-1|\widetilde{u}_{i,k})}$, $i = 1, 2$. Um outro ponto a ser destacado é o fato de que o valor passado do decodificador 1 para o 2, $P(u_k|\widetilde{u}_{2,k})$ ou $\log \frac{P(u_k=+1|\widetilde{u}_{2,k})}{P(u_k=-1|\widetilde{u}_{2,k})}$, não deve incluir a contribuição devido a $\widetilde{u}_{1,k}$, visto que essa informação já foi gerada pelo decodificador 2. O que este decodificador espera é a "informação extrínseca", isto é, uma nova informação a respeito dos dados.

Pode-se então escrever a informação extrínseca como:

$$\log \frac{P(u_k = +1|\widetilde{u}_{2,k})}{P(u_k = -1|\widetilde{u}_{2,k})} = \log \frac{P(u_k = +1|\mathbf{y_i}, \widetilde{u}_{1,1}, ..., \widetilde{u}_{1,k-1}, \widetilde{u}_{1,k+1}, ..., \widetilde{u}_{1,N})}{P(u_k = -1|\mathbf{y_i}, \widetilde{u}_{1,1}, ..., \widetilde{u}_{1,k-1}, \widetilde{u}_{1,k+1}, ..., \widetilde{u}_{1,N})}. \tag{3.4}$$

Após manipulação, a equação 3.4 torna-se:

$$P(u_k|\mathbf{y_1}, \widetilde{\mathbf{u}}_1) = \frac{P(u_k|\mathbf{y_1}, \widetilde{u}_{1,1}, ..., \widetilde{u}_{1,k-1}, \widetilde{u}_{1,k+1}, ..., \widetilde{u}_{1,N}) P(\widetilde{u}_{1,k}|u_k, \mathbf{y_1}, \widetilde{u}_{1,1}, ..., \widetilde{u}_{1,k-1}, \widetilde{u}_{1,k+1}, ..., \widetilde{u}_{1,N})}{P(\widetilde{u}_{1,k}|\mathbf{y_1}, \widetilde{u}_{1,1}, ..., \widetilde{u}_{1,k-1}, \widetilde{u}_{1,k+1}, ..., \widetilde{u}_{1,N})}. \tag{3.5}$$

Como $\widetilde{u}_{1,k}$ é gerado pelo decodificador 2 e passa pelo desentrelaçador em seguida, seu valor possui pequena dependência com relação a $\mathbf{y_1}$ e $\widetilde{u}_{1,k}$, $j \neq k$. Então, pode-se fazer a seguinte aproximação:

$$P(\widetilde{u}_{1,k}|u_k, \mathbf{y_1}, \widetilde{u}_{1,1}, ..., \widetilde{u}_{1,k-1}, \widetilde{u}_{1,k+1}, ..., \widetilde{u}_{1,N}) \cong P(\widetilde{u}_{1,k}|u_k). \tag{3.6}$$

Como $P(\widetilde{u}_{1,k}|u_k) = \frac{P(u_k|\widetilde{u}_{1,k})P(\widetilde{u}_{1,k})}{P(u_k)}$ e supondo distribuição equiprovável dos bits na fonte de informação, $P(u_k) = 1/2$, temos

$$P(\widetilde{u}_{1,k}|u_k, \mathbf{y_1}, \widetilde{u}_{1,1}, ..., \widetilde{u}_{1,k-1}, \widetilde{u}_{1,k+1}, ..., \widetilde{u}_{1,N}) \cong 2P(u_k|\widetilde{u}_{1,k})P(\widetilde{u}_{1,k}). \tag{3.7}$$

Substituindo a equação 3.7 na equação 3.5, e isolando o termo referente à informação

extrínseca, $P(u_k|\mathbf{y_1}, \widetilde{u}_{1,1}, ..., \widetilde{u}_{1,k-1}, \widetilde{u}_{1,k+1}, ..., \widetilde{u}_{1,N})$, obtém-se:

$$P(u_k|\mathbf{y_1}, \widetilde{u}_{1,1}, ..., \widetilde{u}_{1,k-1}, \widetilde{u}_{1,k+1}, ..., \widetilde{u}_{1,N}) = \frac{P(u_k|\mathbf{y_1}, \widetilde{\mathbf{u}}_1)P(\widetilde{u}_{1,k}|\mathbf{y_1}, \widetilde{u}_{1,1}, ..., \widetilde{u}_{1,k-1}, \widetilde{u}_{1,k+1}, ..., \widetilde{u}_{1,N})}{2P(u_k|\widetilde{u}_{1,k})P(\widetilde{u}_{1,k})}. \tag{3.8}$$

Dessa forma, obtemos as probabilidades *a posteriori* desejadas.

3.4.1.3 Atualização iterativa da razão de verossimilhança

Mais uma vez, para facilidade de representação, define-se uma razão de verossimilhança $\tilde{L}_i$ através de:

$$\tilde{L}_i(k) = \log \frac{P(u_k = +1|\widetilde{u}_{i,k})}{P(u_k = -1|\widetilde{u}_{i,k})}, \ i = 1, 2. \tag{3.9}$$

Das equações 3.4 e 3.8 obtém-se, como saída do decodificador 1, antes do entrelaçamento,

$$\tilde{L}_2^{it}(k) = L_1^{it}(k) - \tilde{L}_1^{it-1}(k). \tag{3.10}$$

O índice it superscrito representa a iteração correspondente. Na primeira iteração, o conhecimento *a priori* utilizado para compor $\tilde{L}_1^{it-1}(k)$ vem da fonte de informação. Por exemplo, para uma fonte equiprovável, temos $P(u_k = +1 \mid \mathbf{y}) = P(u_k = -1 \mid \mathbf{y}) = 1/2$, portanto $\tilde{L}_1^{it-1}(k) = 0$.

Analogamente, obtém-se como saída do decodificador 2

$$\tilde{L}_1^{it}(k) = L_2^{it}(k) - \tilde{L}_2^{it-1}(k). \tag{3.11}$$

Conhecendo-se a razão de verossimilhança L_2, saída do decodificador 2, a obtenção das probabilidades *a posteriori* $P(u_k = \pm 1 \mid \mathbf{y})$ é direta.

Podemos ainda reescrever a equação para a it-ésima iteração através da substituição da eq. 3.10 na eq. 3.11 como

$$\tilde{L}_1^{it}(k) = \tilde{L}_1^{it-1}(k) + \alpha_{it}[L_2^{it}(k) - L_1^{it-1}(k)], \quad \alpha_{it} = 1. \tag{3.12}$$

A equação 3.12 tem a forma geral de uma equação de atualização de um método numérico de otimização pelo gradiente, onde o termo entre colchetes representa a taxa de variação de $\tilde{L}_1$ para um dado u_k e α_{it} é o valor do passo.

3.4.2 Algoritmo MAP

No processamento iterativo da decodificação turbo, é passada informação de um decodificador para outro. No entanto, como visto na seção anterior, não é necessário passar toda a informação referente a todas as decodificações, basta passar apenas as "novidades", a informação extrínseca.

A presente seção (3.4.2) apresenta uma outra maneira de se visualizar a informação extrínseca, na qual esta é representada pela razão de verossimilhança que utiliza a informação de paridade e os termos referente à última decodificação.

3.4.2.1 Expressão para a probabilidade de transição

Considere-se uma seqüência recebida de dados ruidosos Y_1^τ, produzida por um codificador convolucional sistemático recursivo que envia dados através de um canal sem memória. Deseja-se decodificar esta seqüência recebida e produzir estimativas *a posteriori* do valor de cada símbolo de entrada. Por fim, sejam os símbolos que compõem Y_1^τ definidos como y_k, $k = 1, 2, ...\tau$ onde os y_k podem ser números reais, números complexos ou vetores multidimensionais, dependendo-se do tipo de modulação utilizada.

A razão de verossimilhança L para o caso binário é dada por:

$$L_k = log \frac{\sum_m \alpha_k^1(m)\beta_k^1(m)}{\sum_m \alpha_k^0(m)\beta_k^0(m)} \tag{3.13}$$

onde, no instante k, o símbolo presente no codificador é u_k e m é o índice de estado.

Em seguida, define-se $\alpha_k^1(m)$ e $\beta_k^1(m)$ como:

$$\alpha_k^i(m) = \sum_{m'=0}^{2^\nu-1}\sum_{j}^{1}\alpha_{k-1}^j(m')\gamma_{i,j}(y_k,m,m') \tag{3.14}$$

$$\beta_k^i(m) = \sum_{m'=0}^{2^\nu-1}\sum_{j}^{1}\beta_{k+1}^j(m')\gamma_{i,j}(y_{k+1},m',m) \tag{3.15}$$

onde o termo $\gamma_{i,j}(y_k,m,m')$ é definido como:

$$\gamma_{i,j}(y_k,m,m') = P[y_k \mid u_k{=}\, i, S_k{=}\, m, u_{k-1}{=}\, j, S_{k-1}{=}\, m']P[u_k{=}\, i, S_k{=}\, m|u_{k-1}{=}\, j, S_{k-1}{=}\, m']. \tag{3.16}$$

onde S_k é o estado do codificador no instante k.

A expressão para $\gamma_{i,j}(y_k,m,m')$ pode ser simplificada pelas seguintes considerações. Uma vez que o símbolo enviado depende apenas do estado atual do codificador e da entrada atual e que o sinal recebido depende estatisticamente apenas do sinal enviado, o primeiro termo pode ser simplificado para

$$P[y_k \mid u_k = i, S_k = m, u_{k-1} = j, S_{k-1} = m'] = P[y_k \mid u_k = i, S_k = m].$$

A segunda parte da expressão de $\gamma_{i,j}(y_k,m,m')$ pode ser simplificada assumindo-se independência das entradas e considerando que a entrada u_k provoca mudança de estado no codificador do estado m' para o m temos:

$$\begin{aligned} P[u_k = i, S_k = m|u_{k-1} = j, S_{k-1} = m'] &= P[u_k = i] \\ &= 0, \text{ caso contrário.} \end{aligned}$$

Portanto, a expressão 3.16 fica:

$$\begin{aligned} \gamma_{i,j}(y_k,m,m') &= P[y_k \mid u_k = i, S_k = m]P[u_k = i], \text{ se houve transição de } m' \text{ para } m. \\ &= 0, \text{ caso contrário.} \end{aligned}$$

3.4.2.2 Extensão para sistemas binários

No caso de um sistema codificador turbo de taxa $1/2$, o modulador envia a saída sistemática x_{1i} e a paridade x_{1p}. Nesta situação, y_k pode ser expressa como

$$y_k = (y_k^i, y_k^p)$$

onde $y_k^i = x_{1i} + n_k^i$, $y_k^p = x_{1p} + n_k^p$ e $x_{1i}, x_{1p} \in \{\pm\sqrt{E_s}\}$. As variáveis n_k^i e n_k^p são ruído gaussiano branco adicionado à saída do modulador.

Para este caso, a seguinte simplificação pode ser feita

$$P[y_k \mid u_k = i, S_k = m] = P[y_k^i \mid u_k = i, S_k = m]P[y_k^p \mid u_k = i, S_k = m]. \tag{3.17}$$

Entretanto, visto que $y_k^i = x_{1i} + n_k^i$, a saída sistemática depende apenas da entrada atual, não do estado do codificador. Então

$$P[y_k^i \mid u_k = i, S_k = m] = P[y_k^i \mid u_k = i]. \tag{3.18}$$

3.4.2.3 Expressão compacta para a razão de verossimilhança

Substituindo-se as definições de α e γ na equação 3.3 temos a seguinte expressão para a razão de verossimilhança L das estimativas MAP dos bits (caso a transição de estados $m' \longrightarrow m$ exista)

$$L_k = \log \frac{\sum_m \sum_{m'} \sum_j \alpha_{k-1}^j(m') P[y_k \mid u_k = 1, S_k = m] P[u_k = 1] \beta_k^1(m)}{\sum_m \sum_{m'} \sum_j \alpha_{k-1}^j(m') P[y_k \mid u_k = 0, S_k = m] P[u_k = 0] \beta_k^0(m)} \tag{3.19}$$

Deve-se ressaltar que o termo $P[u_k = i]$, $i \in \{0, 1\}$ não depende do valor de m', m ou j . Então, o termo $\log \frac{P[u_k=1]}{P[u_k=0]}$ pode ser fatorado para fora do somatório. Este termo é a razão de verossimilhança (LLR) da informação *a priori* sobre o valor do bit de informação no instante k e será denotado por L_{Pk}.

Da seção anterior vimos que, para sistemas binários, expressões mais simples podem ser obtidas em termos das saídas sistemática e de paridade (equações 3.17 e 3.18). Além disso, conforme 3.18, o termo sistemático $\frac{P[y_k^i \mid u_k=1]}{P[y_k^i \mid u_k=0]}$ independe do estado do codificador e também pode ser retirado do somatório.

Após essas considerações, a equação 3.19 para a razão de verossimilhança torna-se

$$L_k = \log \frac{P[u_k=1]}{P[u_k=0]} + \log \frac{P[y_k^i \mid u_k=1]}{P[y_k^i \mid u_k=0]} + \log \frac{\sum_m \sum_{m'} \sum_j \alpha_{k-1}^j(m') P[y_k^p \mid u_k=1, S_k=m] \beta_k^1(m)}{\sum_m \sum_{m'} \sum_j \alpha_{k-1}^j(m') P[y_k^p \mid u_k=0, S_k=m] \beta_k^0(m)}.$$

Este valor, saída do decodificador MAP, é usado para a decisão final sobre o bit.

Por simplicidade de notação, define-se

$$\begin{aligned} L_{Pk} &= \log \frac{P[u_k=1]}{P[u_k=0]} \\ L_{sk} &= \log \frac{P[y_k^i \mid u_k=1]}{P[y_k^i \mid u_k=0]} \\ L_{ek} &= \log \frac{\sum_m \sum_{m'} \sum_j \alpha_{k-1}^j(m') P[y_k^p \mid u_k=1, S_k=m] \beta_k^1(m)}{\sum_m \sum_{m'} \sum_j \alpha_{k-1}^j(m') P[y_k^p \mid u_k=0, S_k=m] \beta_k^0(m)} \end{aligned}$$

Estas definições permitem descrever a saída do decodificador em um forma mais simples, assumindo-se que as variáveis aleatórias associadas às saídas sistemática e de paridade são independentes:

$$L_k = L_{Pk} + L_{sk} + L_{ek}. \tag{3.20}$$

Sobre os termos da equação 3.20 deve-se ressaltar que:

1. L_{Pk} é informação *a priori* inserida no decodificador.

2. L_{sk} é informação relacionada ao valor recebido correspondente à saída sistemática do codificador no instante k. Refere-se diretamente à variável aleatória que descreve o ruído no canal.

3. L_{ek} é informação que utiliza os dados recebidos, exceto a informação sistemática no instante k, para formar sua estimativa do símbolo no instante k. É neste termo que se expressa a influência da informação de paridade e dos termos α e β, os termos recursivos direto e reverso do algoritmo de decodificação. Este termo representa a informação extrínseca.

Resta definir qual informação deve ser passada de um decodificador para o outro de forma a aumentar a confiabilidade da seqüência decodificada. A razão de verossimilhança gerada pelo decodificador 2 para o símbolo no instante k é influenciada por três dados: a informação sistemática sobre o símbolo no instante k, um termo que reflete as restrições da treliça do codificador 2 e um termo que reflete as restrições da treliça do codificador 1.

Analisemos a saída do decodificador 2 referente ao símbolo inserido no codificador turbo no instante k. Os valores após o entrelaçamento são denotados por um sinal til ($\tilde{}$). Então, a saída do decodificador MAP pode ser expressa como

$$\widetilde{L}_k = \widetilde{L}_{Pk} + \widetilde{L}_{sk} + \widetilde{L}_{ek}. \tag{3.21}$$

A tarefa agora é definir que informação proveniente da equação 3.20 deverá ser introduzida no decodificador 2. Esta informação será utilizada como valor *a priori* pelo decodificador 2, será o valor de $\widetilde{L}_{Pk}$. Como foi exposto anteriormente, o valor de saída do decodificador 2 é influenciado por três termos. A influência da informação sistemática está presente no termo $\widetilde{L}_{sk}$ e a influência da estrutura da treliça do decodificador 2 (a própria decodificação por ele realizada) está presente no termo $\widetilde{L}_{ek}$.

Ainda falta o termo referente à treliça do decodificador 1. O termo que o decodificador 2 utiliza como informação *a priori* deve fornecer esse valor e essa informação pode ser simplesmente o termo L_{ek} da saída do decodificador 1, a informação extrínseca, pois contém a informação de paridade e a informação apenas sobre a última decodificação efetuada.

3.4.3 Passos Iterativos

Os passos iterativos da decodificação turbo seguem uma seqüência de eventos determinada. A descrição desta seção refere-se ao sistema decodificador turbo exibido na figura 3.9. $DEC1$ recebe como entradas 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, o valor de $\tilde{L}_1$ é igual a zero, pois na ausência de informação *a priori* assume-se $P(u_k = +1|\widetilde{u}_{i,k}) = P(u_k = -1|\widetilde{u}_{i,k}) = 1/2$ (eq. 3.9). L_1, a saída de $DEC1$, é o logaritmo da razão de verossimilhança das probabilidades *a posteriori*, $P[u_k \mid Y_1^\tau]$, $u_k = 0, 1$ (eq. 3.3).

Entretanto, não é o valor de L_1 que é diretamente passado para $DEC2$, e sim a diferença $\tilde{L}_2 = L_1 - \tilde{L}_1$, associada à informação extrínseca. Este novo valor é usado como informação *a priori* para $DEC2$. O fato de eliminar-se 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. L_1 carrega muita informação das iterações anteriores, enquanto $\tilde{L}_2$ enfatiza a contribuição da última decodificação de $DEC1$.

$DEC2$ recebe como entradas, além de $\tilde{L}_2 = L_1 - \tilde{L}_1$, as seqüências do canal y_{2i} e y_{2p}. $DEC2$ calcula novas informações *a posteriori*, obtendo como saída L_2. Para a próxima iteração, calcula-se o novo valor *a priori* para $DEC1$, $\tilde{L}_1 = L_2 - \tilde{L}_2$. De forma análoga, é necessário eliminar a informação referente à etapa anterior. Ao final de todas as I iterações, os bits decodificados são obtidos a partir de uma decisão sobre o valor de L_2.

Na decodificação turbo, cada iteração traz um ganho adicional em desempenho 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. O número de iterações necessárias para atingir-se a saturação é função direta do comprimento do entrelaçador, N. Quanto maior for N, maior será o número de iterações até o limite de saturação. A iteração limite para saturação também é função das

características dos codificadores envolvidos.

3.4.4 Algoritmos de Decisão Suave

Os algoritmos utilizados para a produção de decisões suaves pelos decodificadores do sistema turbo podem ser divididos em duas categorias: algoritmos baseados na estimação do símbolo de forma exata, como o BCJR, ou algoritmos de estimação de seqüência que evitam expressões exponenciais. Nesta última categoria temos simplificações do MAP e generalizações de saída suave do algoritmo de Viterbi (SOVA).

O algoritmo MAP ótimo considera a probabilidade de todos os possíveis caminhos na treliça quando calcula a estimativa suave do símbolo u_k. Isto implica uma considerável complexidade computacional e requer certa precisão, além de fazer uso de funções exponenciais. Uma simplificação do processamento, chamada max-log-MAP, pode ser feita. Esta pretende encontrar apenas o melhor caminho na treliça para decisões abruptas e para a decisão em cada u_k, avaliar a confiabilidade desta decisão. Para esta simplificação, utilizam-se aproximações como $\log(e^{L_1} + e^{L_2}) \approx max(L_1 + L_2)$.

Entretanto, o algoritmo implementado como referência de desempenho no sistema simulado foi o BCJR completo, para que se pudesse isolar apenas o efeito das modificações sugeridas para a redução do número de operações na decodificação turbo. As modificações sugeridas envolvem critérios para a redução do número médio e máximo de probabilidades de transição calculadas, conforme será visto no próximo capítulo.

3.4.5 Decodificador - Aspectos de Implementação

Os princípios da decodificação turbo e processamento iterativo que norteiam o sistema decodificador foram introduzidos no capítulo anterior. A figura 3.9 mostra a estrutura de um sistema decodificador turbo. O sistema possui dois decodificadores, cada decodificador

recebe duas entradas, correspondentes aos bits de informação e de paridade vindos do canal. Os decodificadores trocam informação a respeito dos bits estimados e através de processamento iterativo refinam os valores de saída.

Entrelaçadores e desentrelaçadores são necessários, pois a informação foi entrelaçada no codificador, a fim de aumentar a distância mínima e dispersar eventuais surtos de ruído. Uma vez que os decodificadores trabalham com decisão suave, a seqüência estimada $\hat{u}$ é obtida através de um decisor. A decisão é extraída ao final do processamento iterativo.

Inicialmente, o decodificador simulado cria os conjuntos de vetores $\boldsymbol{\alpha}_t$ e $\boldsymbol{\beta}_t$, os quais irão armazenar probabilidades referentes aos processamentos recursivos direto e reverso, respectivamente. A $\boldsymbol{\alpha}_0$ e $\boldsymbol{\beta}_\tau$, primeiros vetores de cada passo recursivo, são atribuídos os valores iniciais, correspondentes aos estados inicial e final do codificador. A seqüência recebida é indexada de 1 a τ, onde $\tau = N + m$. N é o comprimento da seqüência de informação e m é o comprimento da seqüência de bits de terminação da treliça, conforme exposto na seção 3.2.4.

Os vetores $\boldsymbol{\alpha}_t$, o passo direto, são indexados de 0 a τ, onde $\boldsymbol{\alpha}_0$ refere-se à situação do codificador antes de receber o primeiro bit de informação. Tipicamente, o codificador estará no estado 0. Já para o conjunto de vetores $\boldsymbol{\beta}_t$, como seu cálculo é feito de forma reversa, o índice de tempo começa em τ e decresce até 1, não havendo necessidade de um vetor $\boldsymbol{\beta}_0$. Uma vez que o codificador pode assumir qualquer valor para seu estado final, é possível definir qualquer estado inicial para o vetor $\boldsymbol{\beta}_\tau$, porém geralmente faz-se o codificador retornar para o estado 0. Assim, os valores iniciais dos vetores seriam:

$$\begin{aligned} \boldsymbol{\alpha}_0 &= [1, 0, ...0] \\ \boldsymbol{\beta}_\tau &= [1, 0, ...0] \end{aligned}$$

O processamento iterativo começa com a obtenção da entrada atual e cálculo da matriz Γ. Esta é calculada a partir da relação sinal-ruído $\rho = \sqrt{2E_s/N_0}$ e do conhecimento *a priori* η_k

sobre os bits a serem decodificados. A matriz Γ foi definida na seção 2.6.4 como

$$\Gamma_t(i,j) = P[S_t = j;\ Y_t \mid S_{t-1} = i] \tag{3.22}$$

Ou seja, representa a probabilidade de o estado atual ser j, com observação Y_t, dado o estado anterior i. As probabilidades de transição de canal e informações *a priori* sobre os dados são inseridas nesta matriz. Para obter-se a matriz Γ, partimos das probabilidades de transição de canal. Para um canal AWGN como o da figura 3.7, temos

$$\begin{aligned} y_1 &= \rho x_1 + n_1 \\ y_2 &= \rho x_2 + n_2 \end{aligned}$$

Como o ruído possui distribuição gaussiana,

$$p_y(y_1|x_1) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_1 - \rho x_1)^2}{2\sigma^2}).$$

E a probabilidade de símbolo, uma vez que as variáveis n_i são independentes e identicamente distribuídas (i.i.d.), fica

$$\begin{aligned} p_y(y|x) &= \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_1 - \rho x_1)^2}{2\sigma^2}) \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_2 - \rho x_2)^2}{2\sigma^2}) \\ &= \frac{1}{2\pi\sigma} \exp[-\frac{1}{2\sigma^2}(y_1^2 + y_2^2 - 2\rho(x_1 y_1 + x_2 y_2) + \rho^2(x_1^2 + x_2^2))] \end{aligned}$$

Para a decisão sobre o valor de y os termos de valor constante podem ser ignorados

$$p_y(y|x) = \exp[\rho(x_1 y_1 + x_2 y_2)]$$

O cálculo dos elementos da matriz Γ, caso exista a transição do estado i para o estado j fica

$$\Gamma_{ij} = \eta_k \cdot p_y(y|x) \cdot p(x) \tag{3.23}$$

onde $\eta_k = P(u_k = -1|\widetilde{u}_k)$ é o termo que carrega a informação extrínseca obtida pelo processamento iterativo. Na primeira iteração, $\eta_k = 1/2$, pois os bits são equiprováveis. São

também equiprováveis, em nosso caso, os valores de $p(x)$, a probabilidade de um dado símbolo x. Então:

$$\begin{aligned} \Gamma_{ij} &= \eta_k \exp[\rho(x_1 y_1 + x_2 y_2)], \text{ caso exista a transição } i \longrightarrow j \\ &= 0, \text{ caso contrário.} \end{aligned} \tag{3.24}$$

Obtido Γ, o próximo passo é calcular os valores de $\boldsymbol{\alpha}_t$, através do procedimento recursivo direto

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1}\Gamma_t.$$

Este processo é repetido para $1 \leqslant t \leqslant \tau$. Para fins de implementação é importante normalizar cada vetor $\boldsymbol{\alpha}_t$ a cada instante t. Isso ocorre porque os valores das probabilidades podem ser muito pequenos, menores que 10^{-9}, e o processamento iterativo requer que esses valores diminutos sejam multiplicados por termos menores que 1, o que leva a variável a tender a zero com o curso das iterações.

Normalizar $\boldsymbol{\alpha}_t$ a cada passo não interfere na precisão da decodificação, pois nos cálculos finais o que interessa são as relações entre as grandezas, que ficam preservadas. Embora não interfira na precisão, este procedimento aumenta o esforço computacional, mas é um requisito prático e se justifica pelo exposto anteriormente. Além disso, a normalização é importante no caso de empregar-se algoritmos de busca reduzida, pois alguns destes atuam de acordo com um limiar relativo aos valores dos elementos de α_t e através da normalização pode-se avaliar todos os vetores pela mesma medida.

Após o cálculo dos vetores $\boldsymbol{\alpha}_t$, inicia-se o procedimento recursivo reverso para cálculo dos vetores $\boldsymbol{\beta}_t$. O cálculo deste conjunto de vetores começa a partir de $\boldsymbol{\beta}_t$, então é necessário esperar a chegada de todo o bloco. Em virtude desse fato, um entrelaçador longo contribui significativamente no aumento do atraso de decodificação.

O cálculo dos vetores $\boldsymbol{\beta}_t$ é feito através do procedimento recursivo reverso

$$\boldsymbol{\beta}_t = \Gamma_t \boldsymbol{\beta}_{t+1}.$$

Os valores de Γ_t são os mesmos utilizados para o cálculo dos $\boldsymbol{\alpha}_t$ correspondentes. O comentário sobre normalização feito para os vetores $\boldsymbol{\alpha}_t$ também se aplica para $\boldsymbol{\beta}_t$.

Obtidos $\boldsymbol{\alpha}_t$ e $\boldsymbol{\beta}_t$, pode-se enfim calcular as probabilidades de transição na treliça. A probabilidade de haver ocorrido uma transição do estado i para o estado j é dada por

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

A probabilidade de um símbolo de entrada ter sido zero, $P[u_t = 0]$ é obtida pela soma de todos os elementos de $\boldsymbol{\sigma}_t$ que correspondam a uma transição (i,j) causada por entrada zero. O fator de normalização é a soma de todos os elementos de $\boldsymbol{\sigma}_t$. Portanto, o valor de saída do decodificador, $APP_0(t)$ é

$$APP_0(t) = \frac{\sum\limits_{i,j \leftrightarrow u_t = 0} \sigma_t}{\sum \sigma_t} = P[u_t = 0].$$

Para obter-se $P[u_t = 1]$ basta somar todos os elementos de $\boldsymbol{\sigma}_t$ que correspondam a uma transição (i,j) causada por entrada unitária e normalizar por $\sum \sigma_t$. No caso de dados binários, $P[u_t = 1] = 1 - P[u_t = 0]$.

Capítulo 4

Algoritmos de Busca Reduzida

Nos capítulos anteriores, analisamos o algoritmo BCJR, um algoritmo de decodificação que minimiza a probabilidade de erro de símbolo e analisamos a decodificação turbo, a qual pode fazer uso do algoritmo BCJR em seus decodificadores componentes. Entretanto, o algoritmo BCJR é bastante intensivo computacionalmente. Esforços têm sido feitos no sentido de simplificar o algoritmo, sem penalizar significativamente seu desempenho.

Neste capítulo investigaremos como é possível simplificar o número de operações na decodificação através de algoritmos que utilizam um critério para reduzir o número de operações efetuadas ao longo da treliça de decodificação. Estes algoritmos são denominados algoritmos de busca reduzida.

Nas próximas seções, analisaremos alguns algoritmos de busca reduzida já propostos e proporemos novos algoritmos para a redução no número de operações na decodificação. Um deles se baseia na aplicação simultânea de dois critérios de busca reduzida, outro apresenta um limiar variável para a busca e um terceiro utiliza dois critérios ao longo da treliça de forma não-simultânea.

4.1 Algoritmos de Busca Reduzida para BCJR

Houve um interesse natural em reduzir o número de operações do algoritmo BCJR visto que, para a decodificação turbo, este é o algoritmo de decodificação que apresenta melhor desempenho. Franz e Anderson propuseram mecanismos para simplificar o algoritmo BCJR que se baseiam na eliminação do cálculo de probabilidades associadas a transições entre estados na treliça [Franz98]. A cada seção da treliça, os valores de probabilidades de transição que não cumprirem um critério determinado não serão utilizados nos cálculos seguintes, ou seja, os ramos na treliça correspondentes a estas transições serão eliminados. Uma das técnicas propostas, o algoritmo M, baseia-se na manutenção dos M estados mais significativos da treliça, eliminando o cálculo dos demais estados. A outra técnica, o algoritmo T, elimina a computação para os estados da treliça cujas probabilidades caiam abaixo de um dado limiar.

Os algoritmos de busca reduzida exploram o fato de boa parte das componentes dos vetores $\boldsymbol{\alpha}_t$ e $\boldsymbol{\beta}_t$ possuírem valores pouco significativos quando comparados à 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 diagrama de fluxo da figura 4.1 corresponde a um decodificador genérico de busca reduzida. Após o cálculo do vetor $\boldsymbol{\alpha}_t$, referente ao passo recursivo direto, é introduzido um critério para limitar o número de estados calculados. A depender do critério utilizado, temos os diferentes algoritmos de busca reduzida.

No capítulo anterior, mencionamos que as iterações da decodificação turbo seguem a lei dos retornos diminuídos, ou seja, cada iteração apresenta um menor ganho em desempenho em relação à iteração anterior, até que este ganho torna-se desprezível. Destacamos ainda que o número de iterações necessárias para atingir a saturação tem relação direta com o comprimento do entrelaçador, N. Como cada iteração corresponde a duas decodificações MAP através do

algoritmo BCJR, a complexidade da decodificação turbo poderá tornar-se extremamente elevada com o aumento de N.

4.1.1 Por que não SOVA?

Na decodificação turbo, outros algoritmos podem ser utilizados no módulo decodificador além do BCJR, como o SOVA (*soft-output Viterbi algorithm* - algoritmo de Viterbi de saída suave), proposto por Hoeher e Hagenauer [Hagen89]. O SOVA associa uma estimativa simplificada da probabilidade de símbolo a cada estado da treliça e revisa essa estimativa a cada seção da treliça. Papke e Robertson propuseram implementações de decodificação turbo utilizando o SOVA [Papke96] e novas simplificações para o esquema.

O SOVA substitui a extensa seqüência de cálculos da seção 2.6.4 por uma série de estimativas da probabilidade $P[u_t = 0]$. A idéia é rodar o algoritmo de Viterbi com a diferença que, a cada estágio, a dispersão de métricas que entra em um dado nó é registrada. Desse fato e do estado estimado no estágio anterior $t-1$, uma estimativa de $P[u_t = 0]$ é formada.

Entretanto, embora o SOVA permita maior velocidade de decodificação que o BCJR, apresenta um desempenho inferior. Franz e Anderson (*op. cit.*) compararam o desempenho da decodificação turbo utilizando decodificadores SOVA e utilizando decodificadores BCJR. Como previsto, o sistema com algoritmo BCJR apresentou um desempenho melhor. Para uma probabilidade de erro de bit de 10^{-3}, decodificação turbo utilizando BCJR atingiu desempenho superior à decodificação turbo utilizando SOVA em pelo menos 1 *db*.

Uma provável explicação seria que as estimativas de símbolo geradas pelo SOVA são demasiado aproximadas e não tiram proveito das múltiplas iterações da decodificação turbo para melhorar significativamente essa estimativa. A convergência é muito lenta neste caso. Já o BCJR, com suas estimativas mais complexas, apresenta uma convergência rápida. Dependendo do comprimento do entrelaçador, 5 a 10 iterações são suficientes.

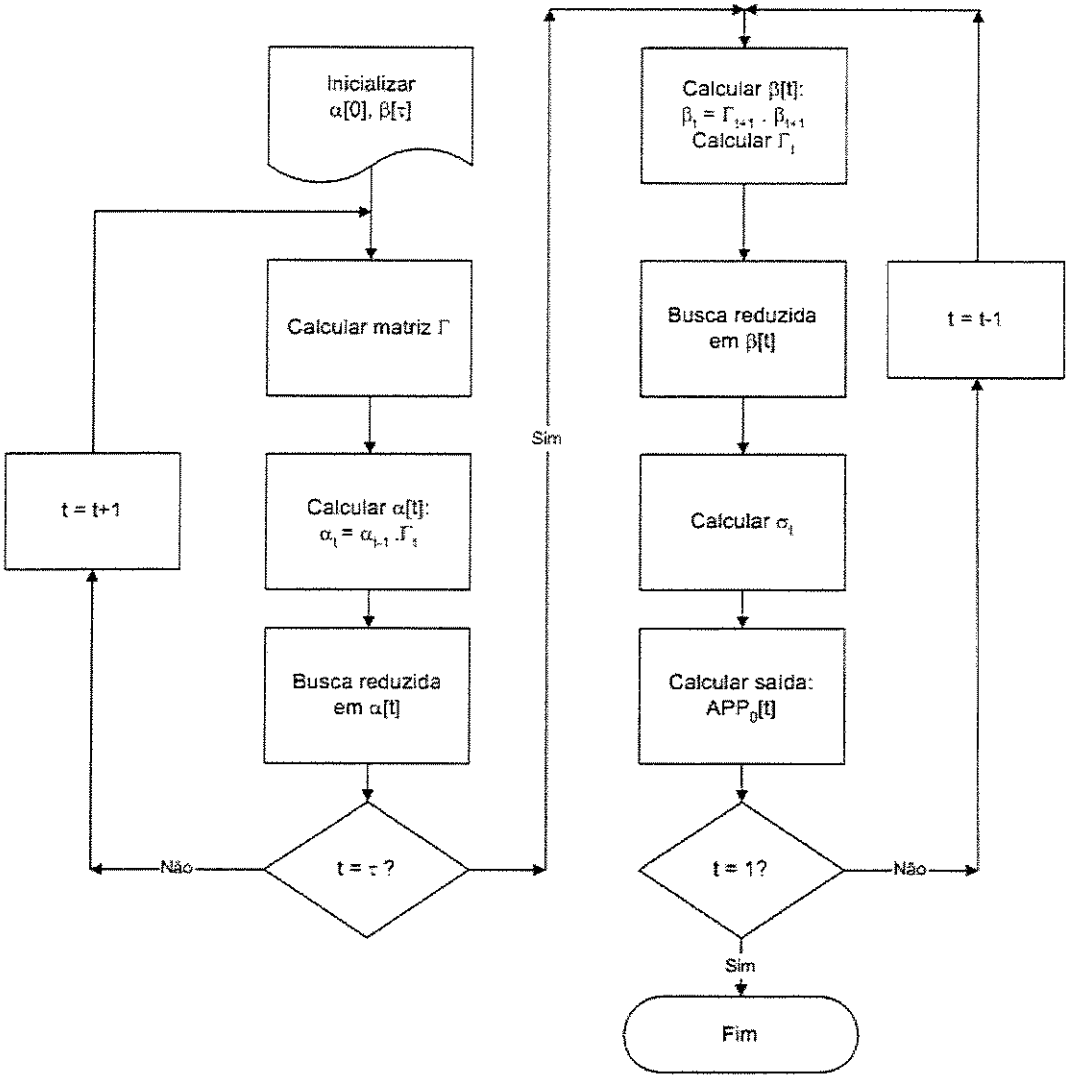

Figura 4.1: Diagrama de fluxo de um decodificador genérico de busca reduzida.

4.2 Algoritmo M

O algoritmo M preserva os M estados com maior probabilidade em cada seção da treliça. Aplicando-se o algoritmo M à decodificação BCJR temos um primeiro mecanismo de redução do número médio de estados, num esquema que pode ser denominado M-BCJR.

No M-BCJR, o procedimento recursivo que produz o vetor α_t (equação 2.30) utiliza apenas os M maiores elementos do vetor α_{t-1}, os demais são igualados a zero. O mesmo procedimento pode ser aplicado ao passo recursivo reverso, mas uma vez que os elementos de σ_t são produtos de α e β (equação 2.32), basta calcular os β_t apenas nas regiões da treliça onde existam os α_t correspondentes.

Os M estados com maior probabilidade são obtidos através de operações de comparação, não causando impacto no desempenho total da decodificação, que é limitado pelas operações de ponto flutuante. O diagrama de fluxo do algoritmo M está representado na figura 4.2.

O M-BCJR possibilita limitar o número máximo de operações, mas apresenta um desempenho pobre comparado ao BCJR sem busca reduzida, como também comparado a outros algoritmos de busca reduzida, que serão descritos nas próximas seções.

4.3 Algoritmo de Limiar (Algoritmo T)

O algoritmo de limiar estabelece um valor limite de probabilidade acima do qual os estados são preservados. Aqueles estados que ficarem abaixo desse limiar são igualados a zero e desconsiderados dos cálculos.

Aplicando o algoritmo T à decodificação BCJR temos o T-BCJR. Neste, os elementos de α e β são igualados a zero quando caírem abaixo de um certo limiar. Em cada seção da treliça, o vetor α_t correspondente é normalizado. Para calcular o vetor α_t na equação 2.30, utiliza-se apenas os elementos de α_{t-1} que estiverem acima do limiar. Assim como no algoritmo M-BCJR, a

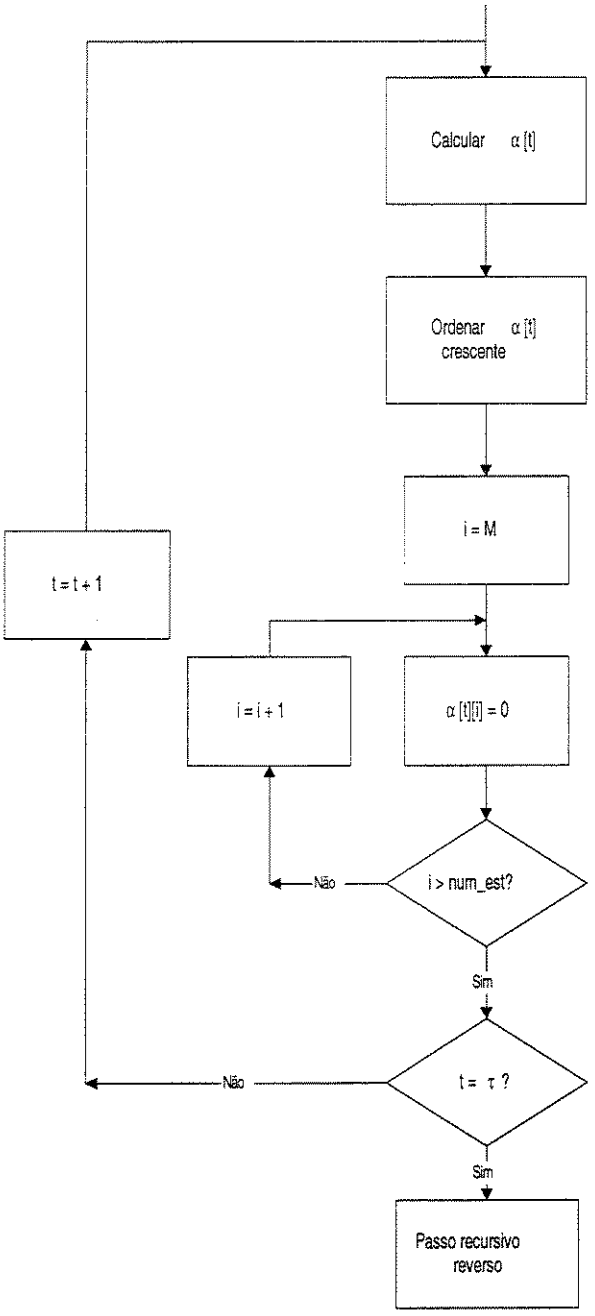

Figura 4.2: Diagrama de fluxo do algoritmo M.

computação dos β_t só precisa ser feita nas regiões da treliça onde existirem estados sobreviventes. O diagrama de fluxo do algoritmo T está representado na figura 4.3.

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 α caiam acima do limiar, logo o decodificador estará operando com a complexidade de um BCJR.

Nas próximas seções são propostos outros métodos de busca reduzida, bem como serão apresentados resultados de simulação do desempenho.

4.4 Estados sobreviventes

Antes de apresentar o primeiro algoritmo proposto, deve-se ressaltar uma distinção entre os algoritmos de busca reduzida. Em uma categoria, há aqueles que preservam um número fixo de estados a cada seção da treliça, como o algoritmo M. Em outra, há aqueles que permitem um número variável de estados sobreviventes a cada seção, a depender de algum critério limite, como o algoritmo T.

Para estes últimos, uma análise a respeito dos valores médio e instantâneo do número de estados sobreviventes é necessária. Para este tipo de algoritmo de redução do número de operações, uma informação relevante é o número médio de estados sobreviventes. Porém, para que se melhor avaliasse a questão, foi necessário obter a evolução do número de estados sobreviventes ao longo da treliça.

Visto que o algoritmo M preserva um número fixo de estados a cada seção da treliça, o algoritmo T foi avaliado. O resultado é mostrado na figura 4.4. O gráfico mostra o número médio de estados sobreviventes em cada seção da treliça, para cada iteração. Oito iterações foram efetuadas, sendo que 1000 palavras foram transmitidas. A relação sinal-ruído E_b/N_0 foi

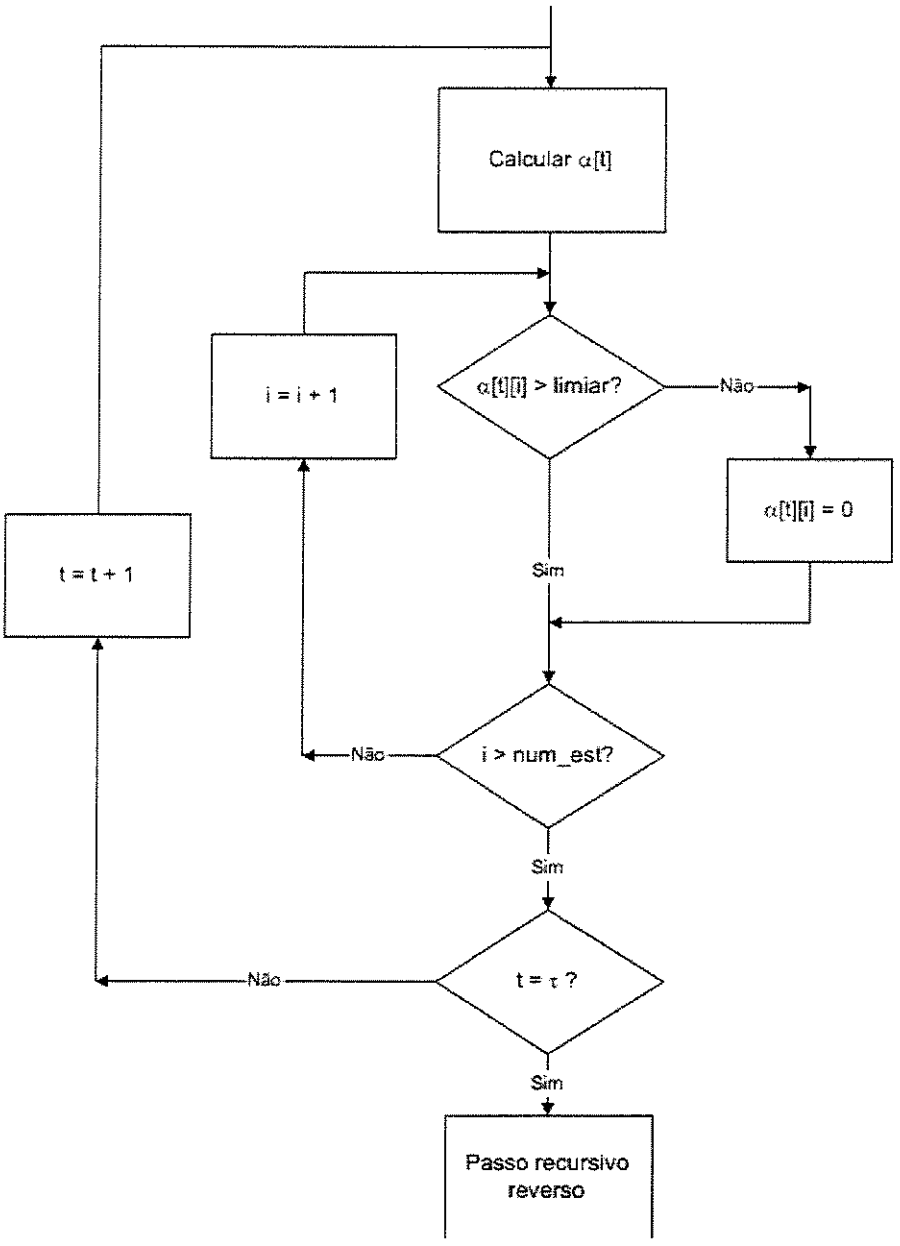

Figura 4.3: Diagrama de fluxo do algoritmo T.

fixada em $0,5\ dB$. Para outros valores de E_b/N_0, os resultados obtidos foram semelhantes. Cada uma das curvas do gráfico corresponte a uma dada iteração. Cada ponto representa a média do número de estados sobreviventes para cada seção da treliça.

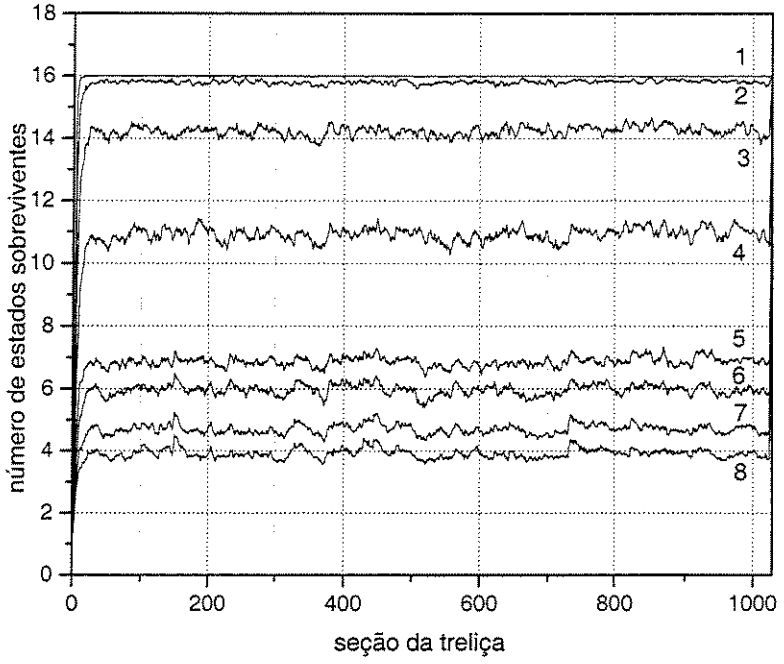

Figura 4.4: Estados sobreviventes para cada seção da treliça. $E_b/N_0 = 0,5\ dB$. 1000 palavras foram transmitidas e 8 iterações efetuadas. Cada curva é identificada pelo número da iteração correspondente.

Nota-se que ao longo da primeira iteração o número de estados sobreviventes, em média, é praticamente igual ao número máximo, 16, neste caso. Nota-se também que a separação vertical entre as linhas correspondentes às diversas iterações é não-uniforme e não-linear, há espaçamentos longos seguidos de espaçamentos curtos, sem uma taxa de variação definida. Destacam-se os espaçamentos verticais após a iteração 3 e após a iteração 4. Isso indica que a maior redução ocorre após essas iterações, tanto na média como, segundo observações mais detalhadas, nos valores instantâneos.

Um outro critério de redução no número de estados sobreviventes pode ser introduzido após as iterações 3 e 4 para fazer uso dessa queda na média dos valores e contribuir para uma redução

ainda maior no número de estados sobreviventes. Nas duas primeiras iterações, onde o número de estados sobreviventes é próximo do valor máximo, o desempenho é sensivelmente afetado caso haja restrições a esse valor máximo, como um número fixo de estados preservados. A partir de então, o uso de um critério de redução no número de estados sobreviventes tira proveito da queda do número de estados e mantém um bom desempenho na decodificação.

4.4.1 Estabelecimento de um teto

Outro possível uso dessa característica seria o estabelecimento de um limite máximo para o número de estados logo após a iteração 3 ou 4. Assim, as iterações posteriores estariam submetidas ao novo limite, mais apertado que o anterior, o que leva também a uma limitação no número máximo de operações. Este novo valor máximo é escolhido um pouco acima do número médio de estados sobreviventes. A adoção de um novo limite máximo para o número de estados é uma das bases do algoritmo proposto na próxima seção.

4.5 Algoritmo Composto

Os algoritmos de busca reduzida baseiam-se 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 (descrito na seção 2.6).

O primeiro algoritmo proposto baseia-se na introdução simultânea dos critérios de limiar e de número máximo de estados sobreviventes. As componentes dos vetores α e β 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

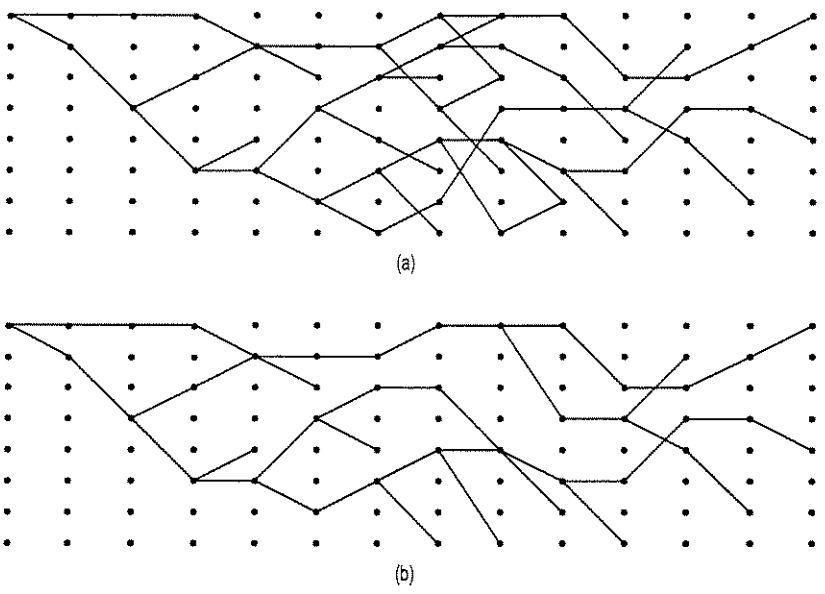

Figura 4.5: Evolução dos estados sobreviventes em algoritmos de busca reduzida para uma mesma palavra recebida. (a) Treliça para T-BCJR. (b) Treliça para C-BCJR com $M = 4$.

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. Por utilizar dois critérios distintos, foi denominado algoritmo composto e, para abreviar, é também referido como C-BCJR.

A figura 4.5 mostra a variação do número de estados sobreviventes para os algoritmos de busca reduzida T-BCJR e C-BCJR. Uma linha conectando dois nós indica que o nó à esquerda foi utilizado para calcular o nó à direita, e que este sobreviveu. Os dois fragmentos de treliça apresentados referem-se a uma mesma palavra recebida. Há 8 estados em cada seção da treliça. Da figura fica claro que, ao contrário do T-BCJR, o C-BCJR possui um limite para o número de estados sobreviventes. No caso, o limite para o C-BCJR foi definido como $M = 4$.

Fazendo-se uma análise do número médio de estados sobreviventes do algoritmo T-BCJR em função do número de iterações, de maneira análoga à que foi realizada na seção anterior,

observa-se que existe uma dada iteração I a partir da qual o número médio de estados não varia significativamente. Para um total de 8 iterações, as melhores escolhas para a iteração I foram a terceira e a quarta. Baseado nisso, a partir da iteração $I = 3$ ou $I = 4$ o algoritmo C-BCJR foi utilizado nos decodificadores MAP. Sendo assim, garantiu-se também uma redução do número máximo de operações da decodificação turbo.

O uso simultâneo de um limitante superior e de um limiar para os estados sobreviventes possibilita um controle sobre duas variáveis relevantes para a complexidade computacional, o número médio de operações e o atraso máximo de decodificação.

4.6 Algoritmo de Limiar Variável

Nesta seção propõe-se um algoritmo de limiar variável. O algoritmo considera como estados sobreviventes aqueles cuja probabilidade for superior a uma fator de redução f (< 1) multiplicado pela máxima probabilidade daquela seção. Ou seja, se $max = maximo(\boldsymbol{\alpha}_t)$, então o limiar será dado por $f.max$.

Dessa forma, temos um limiar que é mais sensível às variações introduzidas pelo canal na seqüência recebida, em vez de um limiar arbitrariamente escolhido. Assim, o limiar possui atualização dinâmica e pode acompanhar a cada seção as flutuações nas probabilidades.

No aspecto de implementação, o algoritmo baseia-se na manutenção dos elementos dos vetores $\boldsymbol{\alpha}_t$ e $\boldsymbol{\beta}_t$ maiores que uma determinada fração do máximo elemento do vetor. Os elementos que ficarem abaixo do valor limite serão igualados a zero. Assim, para cada seção da treliça de decodificação associada ao instante t, apenas os estados cujas probabilidades caiam acima deste limiar serão utilizados para o cálculo na próxima seção da treliça, ou seja, chamados estados sobreviventes.

O diagrama de fluxo da figura 4.6 representa o algoritmo proposto. O valor máximo de α_t é obtido através de operações de comparação, não causando impacto no desempenho total da

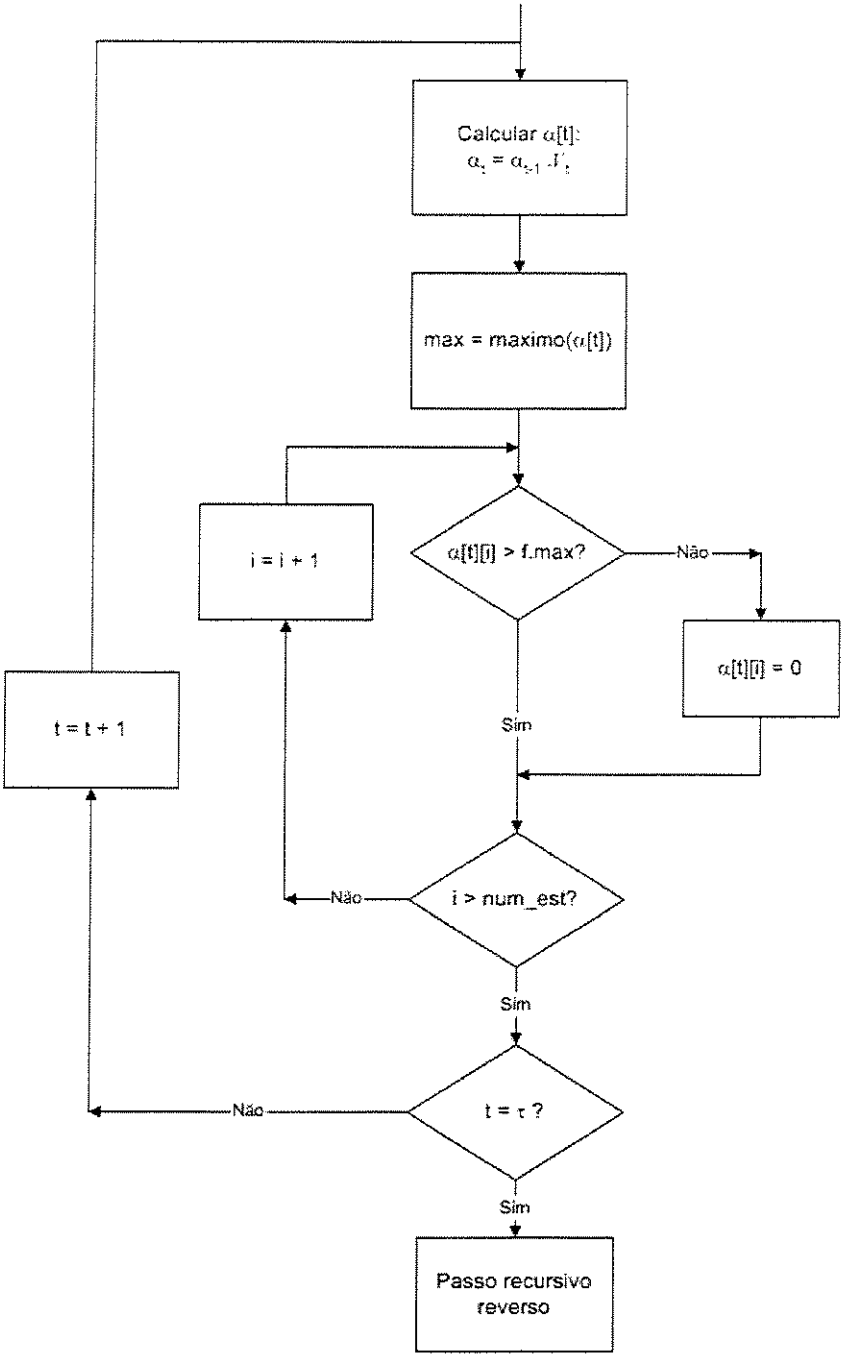

Figura 4.6: Diagrama de fluxo para Algoritmo de Limiar Variável.

decodificação, que é limitado pelas operações de ponto flutuante. O valor de f pode ser definido pelo programador, sendo geralmente entre 10^{-3} e 10^{-5}. Na simulação utilizamos $f = 10^{-4}$.

4.7 Algoritmo de Divisão de Treliça

Este algoritmo proposto divide a treliça de decodificação em 2 partes e utiliza critérios distintos para cada parte. A treliça é dividida ao longo do tempo, em um certo instante t_d. O ponto de divisão tem sido arbitrariamente escolhido. Considerou-se a divisão na metade da treliça ($t_d = 0,5.\tau$) como também em outros valores ($0,3.\tau$, $0,4.\tau$, $0,7.\tau$, $0,3.\tau$). τ é o comprimento da seqüência recebida.

Após a escolha do ponto de divisão, critérios de busca reduzida distintos são aplicados a cada parte da treliça. Na parte anterior, $0 < t < t_d$, é utilizado um critério de limiar, como o proposto na seção 4.6. Pode-se também permitir que a primeira parte não realize nenhum tipo de busca reduzida, operando com todos os estados da treliça. A idéia é que essas primeiras seções são decisivas para a consolidação das probabilidades na treliça e para a definição de um caminho ótimo, portanto se beneficiam de um limiar mais frouxo.

Na parte posterior, $t > t_d$, é aplicado um critério de limitação do número máximo de estados, como o algoritmo M. Dessa forma, nas primeiras seções da treliça há uma maior liberdade para a decodificação, permitindo-se até que o número de estados sobreviventes possa ser igual ao valor máximo. Nesta primeira parte, escolhe-se sacrificar o atraso máximo de decodificação em nome de um melhor desempenho, esperando-se que um caminho de maior probabilidade seja fortalecido na treliça.

Na segunda parte da treliça, a limitação do número máximo de estados sobreviventes contribui para a redução do atraso máximo de decodificação. O diagrama de fluxo da figura 4.7 ilustra o algoritmo proposto.

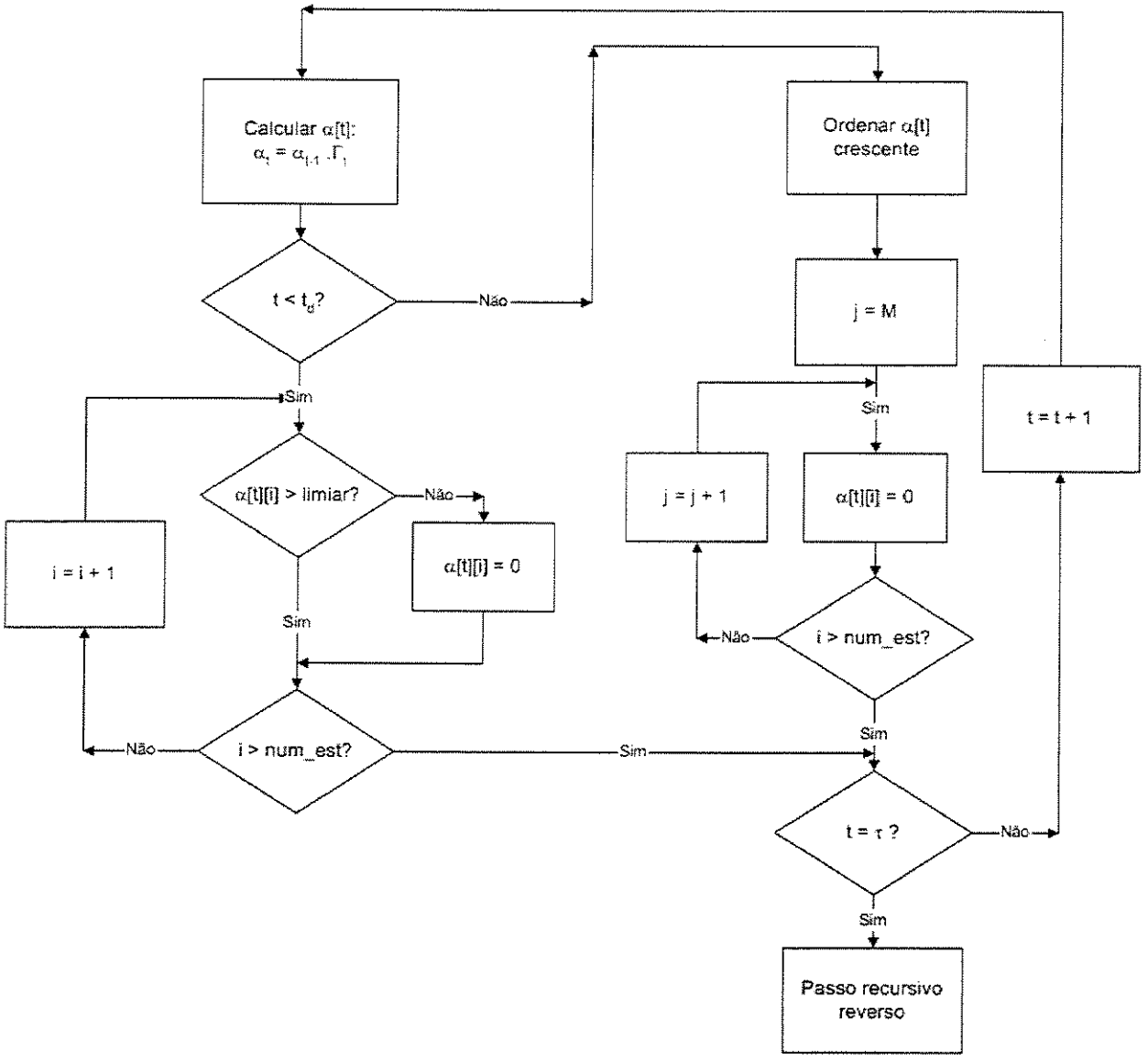

Figura 4.7: Diagrama de fluxo para Algoritmo de Divisão de Treliça.

Capítulo 5

Resultados de Simulação e Conclusões

Neste capítulo serão apresentados e discutidos os resultados de simulação para a decodificação com algoritmos de busca reduzida. Também serão apresentadas conclusões e perspectivas para trabalhos futuros.

Os resultados serão apresentados na forma de gráficos comparativos de desempenho, geralmente com respeito às variáveis independentes probabilidade de erro de bit e número médio de estados sobreviventes. A relação sinal-ruído foi obtida em termos de energia média por bit de informação, enquanto a probabilidade de erro de bit foi obtida através de freqüência relativa de bits incorretos. Em todas as simulações, o número de palavras transmitidas foi pelo menos dez vezes superior ao número de palavras necessárias para que um erro de bit seja observado. Os programas de simulação foram escritos em linguagem C e executados em um computador Pentium® III 450 MHz.

5.1 Resultados para o Algoritmo Composto

O Algoritmo Composto (C-BCJR) utiliza como critério de parada um limiar e também um número máximo de estados sobreviventes, a partir de uma dada iteração I. Esta iteração I era

escolhida em vista da evolução do número médio de estados. Para 8 ou 10 iterações, $I = 3$ ou $I = 4$ apresentaram o melhor compromisso entre redução da complexidade máxima sem penalizar significativamente o desempenho.

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 3.5. Os resultados foram obtidos para oito iterações do decodificador turbo.

A figura 5.1 apresenta o desempenho do algoritmo C-BCJR, para um limiar 10^{-5} e teto igual a 12. A figura mostra resultados para $I = 1$ a $I = 4$. 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 de simulação para valores de I menores que 3 apresentaram degradação de desempenho significativa,

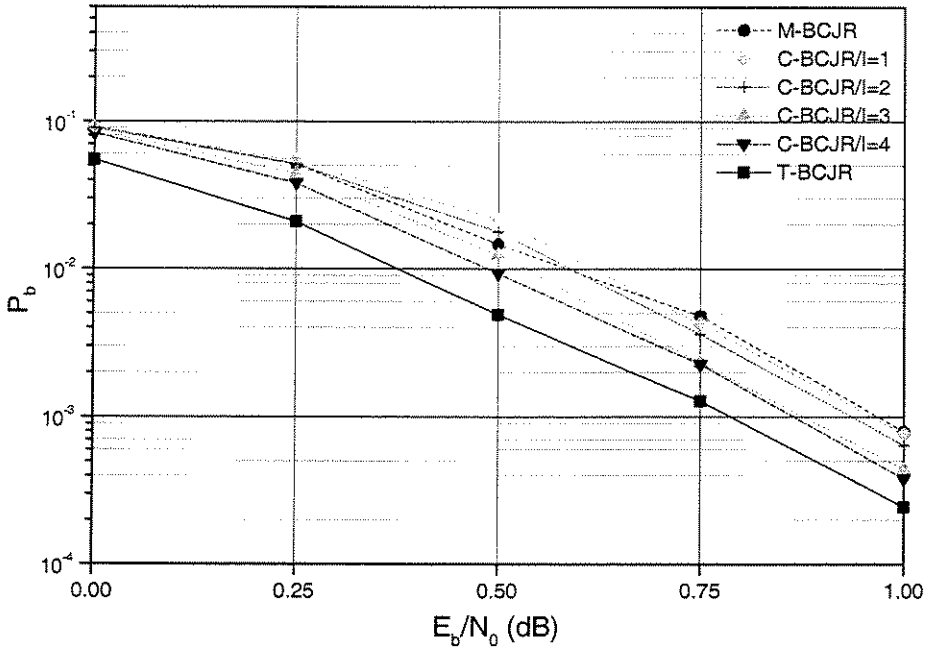

Figura 5.1: Probabilidade de erro de bit (P_b) para vários algoritmos de busca reduzida. Total de 8 iterações.

aproximando-se ou até ultrapassando a curva do 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 C-BCJR. A figura 5.2 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 C-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.

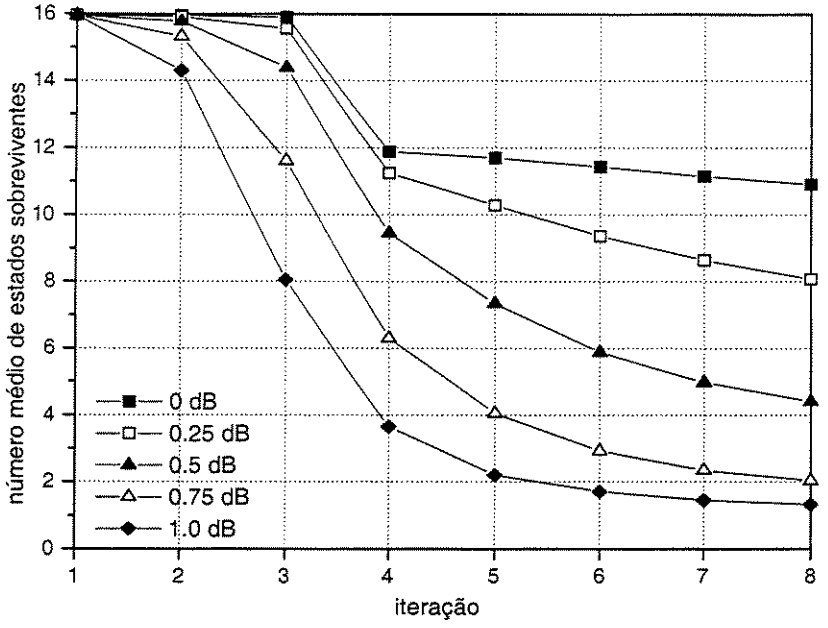

Figura 5.2: Decodificação Turbo C-BCJR/$I = 3$. Número médio de estados sobreviventes para cada iteração, para diversos valores de relação sinal-ruído. Limiar 10^{-5} e teto 12. Total de 8 iterações.

5.2 Resultados para Canais com Desvanecimento Rayleigh

A seção 2.2.4 apresentou o modelo para canais com desvanecimento. O desempenho da codificação utilizando o Algoritmo Composto (indicado por C-BCJR) num canal com desvanecimento Rayleigh é apresentado na figura 5.3. Os resultados foram obtidos para um canal completamente entrelaçado e com perfeita estimação da informação dos estados do canal do receptor. As métricas de transição (variáveis γ do algoritmo BCJR) são como dados em [Hall98]. Para fins de comparação, são também apresentados o desempenho do canal sem codificação e o desempenho da decodificação BCJR sem utilizar busca reduzida. Foram efetuadas simulações para o Algoritmo Composto utilizando diferentes valores do parâmetro I, sendo exibidos os resultados para $I = 3$ e $I = 4$, por apresentarem melhor compromisso entre desempenho e redução no esforço computacional.

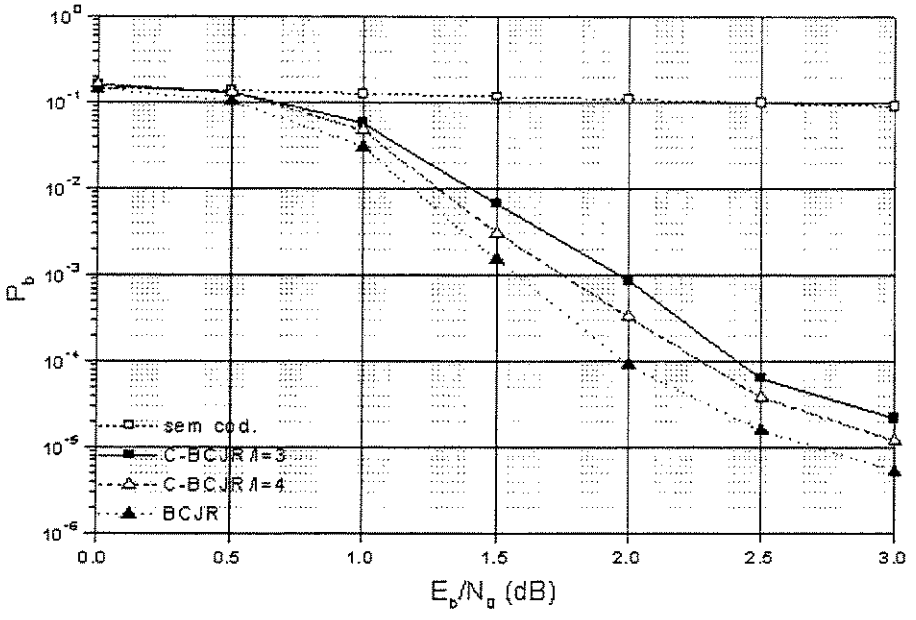

Figura 5.3: Desempenho do Algoritmo Composto em canal com desvanecimento Rayleigh.

Das curvas de desempenho observa-se que, em contraste com o desempenho obtido para o canal AWGN, para probabilidades de erro de bit pequenas, definir o parâmetro I como $I = 4$ resulta em desempenho melhor que escolher $I = 3$. O parâmetro I indica a iteração a partir da qual o número máximo de operações torna-se limitado. Portanto, $I = 4$ significa que o o decodificador passa uma iteração a mais sem se comprometer com um teto no número máximo de operações, o que explica o melhor desempenho desta configuração. Em compensação, com $I = 3$ temos uma maior limitação no volume de processamento total.

5.3 Resultados para o Algoritmo de Limiar Variável

O gráfico 5.4 apresenta resultados de simulação para o Algoritmo de Limiar Variável (LV). O fator de redução f utilizado foi de $f = 10^{-4}$. O desempenho do algoritmo foi muito similar ao desempenho do T-BCJR.

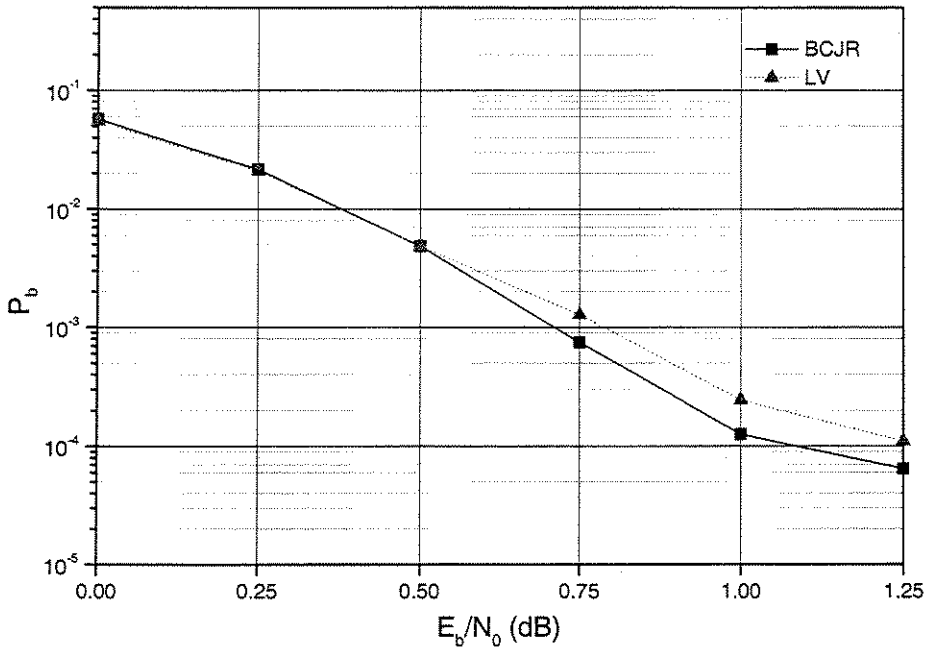

Figura 5.4: Desempenho do Algoritmo de Limiar Variável (LV) comparado ao BCJR.

5.4 Resultados para o Algoritmo de Divisão de Treliça

O gráfico 5.5 apresenta os resultados de simulação para o Algoritmo de Divisão de Treliça (DT). Foram usados como parâmetros para a divisão $t_d = 0,3.\tau$ e $t_d = 0,4.\tau$, onde τ é o comprimento da seqüência recebida.

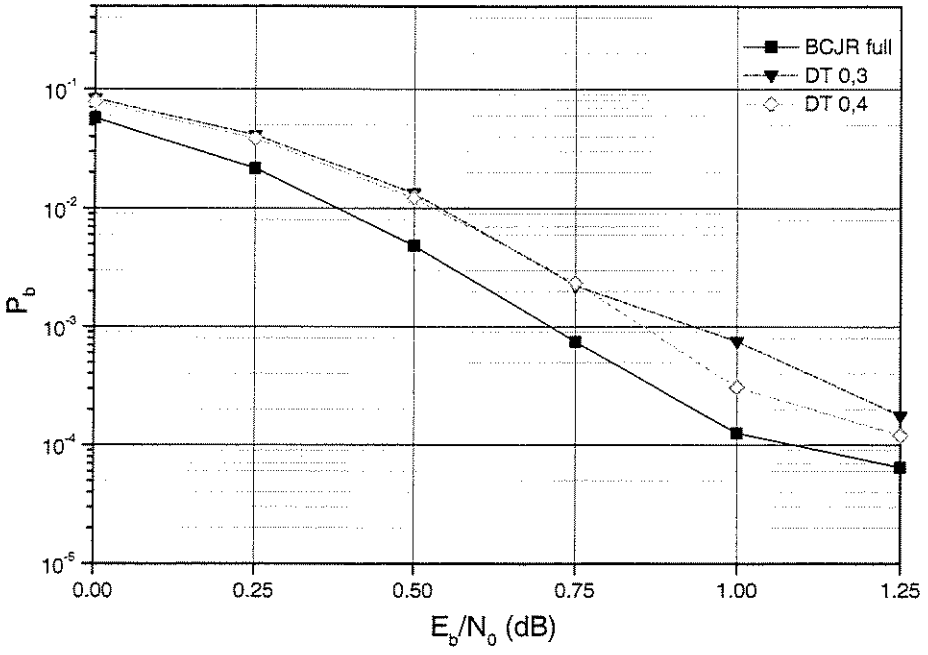

Figura 5.5: Desempenho comparativo do Algoritmo de Divisão de Treliça (DT) e BCJR.

5.5 Saturação do número de iterações

Já se comentou que no processamento iterativo ocorre uma saturação. A cada iteração o ganho de codificação é menor, até que este torna-se insignificante. A figura 5.6 mostra a evolução do desempenho do algoritmo DT ao longo de 8 iterações. Como parâmetro foi utilizado $t_d = 0,3.\tau$.

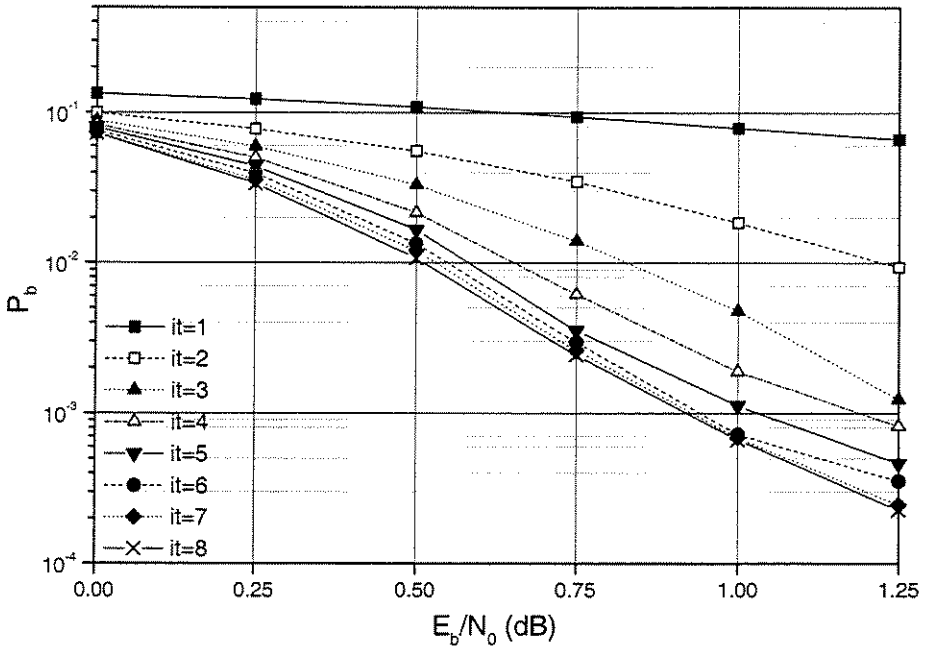

Figura 5.6: Evolução do desempenho do Algoritmo DT ao longo de 8 iterações.

5.6 Comparação de Resultados

Esta seção realiza uma comparação entre os algoritmos de busca reduzida propostos. A figura 5.7 apresenta o número médio de estados sobreviventes a cada iteração para algoritmos de busca reduzida. A figura 5.8 mostra a evolução da probabilidade de erro de bit a cada iteração. O algoritmo T-BCJR utilizou um limiar de 10^{-5}. O algoritmo composto teve como parâmetros $I = 4$ e $M = 12$. O algoritmo divisão teve como parâmetro $t_d = 0,4.\tau$. Todos os dados foram obtidos para $E_b/N_0 = 0,5\ dB$.

Pela figura 5.8, nas primeiras iterações o desempenho do T-BCJR e do algoritmo composto foram idênticos. Após a quarta iteração, quando o algoritmo composto passou a limitar o número máximo de estados buscados em $M = 12$, seu desempenho distanciou-se um pouco do T-BCJR. O T-BCJR apresenta uma aparente saturação na iteração 7, mas este comportamento é melhor explicado atribuindo-se a uma flutuação estatística da simulação do que a uma característica do algoritmo. O algoritmo de divisão apresentou um desempenho sempre inferior e a conclusão a que se chega é que isto ocorre por ele selecionar um menor número de estados sobreviventes desde as primeiras iterações.

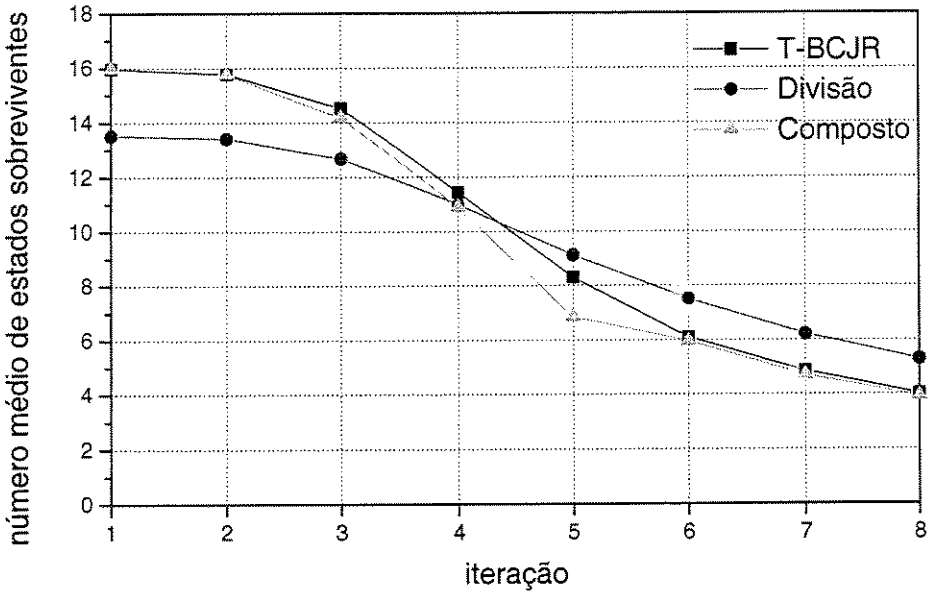

Figura 5.7: Número médio de estados sobreviventes para algoritmos de busca reduzida.

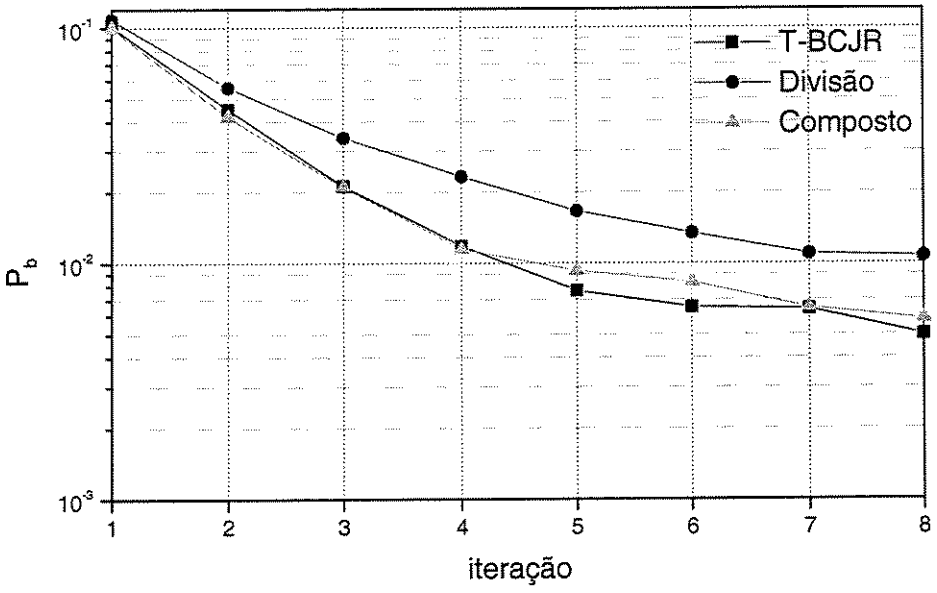

Figura 5.8: Evolução da probabilidade de erro de bit para algoritmos de busca reduzida.

5.7 Conclusões e Perspectivas Futuras

Observando o desempenho do algoritmo de divisão de treliça e do algoritmo composto, podemos concluir que as primeiras iterações possuem uma grande influência no desempenho final dos algoritmos de busca reduzida. Esta importância das primeiras iterações foi também utilizada na proposta de um algoritmo de complexidade reduzida feita por Frey e Kschischang em [Frey98]. Isto fica ainda mais ressaltado quando observamos o desempenho do algoritmo composto num canal com desvanecimento.

Todas as comparações entre algoritmos apresentadas neste trabalho consideraram um total de 8 iterações do decodificador turbo. Não foi considerada a possibilidade de se efetuar um número distinto de iterações para cada algoritmo a ser comparado. Para que uma tal comparação pudesse ser feita de maneira justa, seria preciso definir um pârametro de complexidade. Por exemplo, tal parâmetro deveria levar em consideração o fato de que os algoritmos propostos possuem uma redução significativa no número de computações por seção também na recursão reversa. Uma tal avaliação fica como sugestão para um trabalho futuro. Adicionalmente, baseado em tal avaliação, uma comparação dos algoritmos propostos com outros algoritmos da literatura fica também como sugestão.

Referências Bibliográficas

[Bahl74] 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.

[Berr93] C. Berrou, A. Glavieux e P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes", Proc. IEEE ICC'93, Geneva, Switzerland, May 1993, pp. 1740-1745.

[Dasg01] U. Dasgupta e K. R. Narayanan, "Parallel Decoding of turbo Codes Using soft Output T-Algorithms", IEEE Comm. Letters, vol. 5, n. 8, Aug. 2001, pp. 352-354.

[Divs95] D. Divsalar e F. Pollara, "Turbo codes for deep-space communications", TDA Progress Report 42-120, JPL, Feb. 1995, pp. 29-39.

[Franz98] 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.

[Frey98] Brendam J. Frey e Frank R. Kschischang, "Early Detection and Trellis Splicing: Reduced-Complexity Iterative Decoding", IEEE Journal Selec. Areas Commun., vol 16, Feb. 1998, pp. 153-159.

[Hagen89] J. Hagenauer e P. Hoeher, "A Viterbi algorithm with soft decision outputs and its applications", Proc. IEEE Globecom, Dallas, TX, Nov. 1989, pp. 47.1.1-47.1.6.

[Hagen94] J. Hagenauer e P. Robertson,"Iterative (Turbo) Decoding of Systematic Convolutional Codes With the MAP and SOVA Algorithms", Proc. of the ITG Conference on Source and Channel Coding, Frankfurt, Germany, October 1994, pp. 1-9.

[Hall98] E. K. Hall, S. G. Wilson, "Design and analysis of turbo codes on Rayleigh fading channels", IEEE Journal Selec. Areas Commun., vol 16, Feb. 1998, pp. 160-174.

[Herz98] Hanan Herzberg, "Multilevel Turbo Coding with Short Interleavers", IEEE Journal Selec. Areas Commun., vol 16, Feb. 1998, pp. 303-309.

[Shu83] Shu Lin e D. J. Costello,"Error-Control Coding", Prentice-Hall, 1983

[Papke96] L. Papke e P. Robertson, "Improved Decoding with the SOVA in a parallel concatenated (turbo-code) scheme", Conf. Rec., IEEE Int. Conf. Commun., June 1996, pp. 102-106.

[Proak83] John G. Proakis,"Digital Communications", McGraw-Hill, 1983.

[Vitb67] A. J. Viterbi,"Error bounds for convolutional codes in an asymptotically optimum decoding algorithm", IEEE Trans. Inform. Theory, vol IT-13, Apr 1967, pp. 260-269.

[Vitb79] Andrew J.Viterbi e Jim K. Omura,"Principles of digital communication and coding", McGraw-Hill, c1979.