

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA  
INDUSTRIAL - CPGEI

GUILHERME LUIZ MORITZ

**ANÁLISE DE COMPLEXIDADE DE CÓDIGOS TURBO UTILIZANDO  
A TRELIÇA MÍNIMA E SECCIONADA**

DISSERTAÇÃO

CURITIBA

2012

GUILHERME LUIZ MORITZ

**ANÁLISE DE COMPLEXIDADE DE CÓDIGOS TURBO UTILIZANDO  
A TRELIÇA MÍNIMA E SECCIONADA**

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciências, do Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná. Área de Concentração: Informática Industrial

Orientador: Prof. Dr. Richard Demo Souza

Co-orientador: Prof. Dr. Cecílio José Lins Pimentel

Curitiba  
2012

Dados Internacionais de Catalogação na Publicação

---

M862 Moritz, Guilherme Luiz  
Análise de complexidade de códigos turbo utilizando a treliça mínima e seccionada /  
Guilherme Luiz Moritz. — 2012.  
56 f. : il. ; 30 cm

Orientador: Richard Demo Souza.

Coorientador: Cecílio José Lins Pimentel.

Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa  
de Pós-graduação em Engenharia Elétrica e Informática Industrial. Curitiba, 2012.

Bibliografia: f. 55-56.

1. Teoria da codificação. 2. Códigos corretores de erros (Teoria da informação). 3.  
Algoritmos. 4. Teoria de sinais (Telecomunicações). 5. Convoluções (Matemática). 6.  
Engenharia elétrica – Dissertações. I. Souza, Richard Demo, orient. II. Pimentel,  
Cecílio José Lins, coorient. III. Universidade Tecnológica Federal do Paraná. Programa  
de Pós-graduação em Engenharia Elétrica e Informática Industrial. IV. Título.

---

CDD (22. ed.) 621.3

**Título da Dissertação Nº: 587**

**“Análise de Complexidade de Códigos Turbo Utilizando  
as Trelças Mínima e Seccionada ”**

por

**Guilherme Luiz Moritz**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM CIÊNCIAS – Área de Concentração: Telemática, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial – CPGEI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Curitiba, às 09h do dia 17 de fevereiro de 2012. O trabalho foi aprovado pela Banca Examinadora, composta pelos professores:

*Richard Demo Souza*

Prof. Richard Demo Souza, Dr.  
(Orientador – UTFPR - CT)

*Marcelo Eduardo Pellenz*

Prof. Marcelo Eduardo Pellenz, Dr.  
(PUC - PR)

*João Luiz Rebelatto*

Prof. João Luiz Rebelatto, Dr.  
(UTFPR - CT)

Visto da coordenação:

*Fábio Kurt Schneider*

Prof. Fábio Kurt Schneider, Dr.  
(Coordenador do CPGEI)

## RESUMO

MORITZ, Guilherme Luiz. Análise de complexidade de códigos turbo utilizando as treliças mínima e seccionada. 56 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2012.

A dissertação apresentada utiliza a representação de treliça mínima e seccionada para a decodificação de códigos turbo, analisando os impactos da aplicação desta técnica no desempenho (taxa de erro em função da relação sinal ruído) em um canal AWGN e avaliando a redução de complexidade de processamento. O processo de decodificação proposto utilizará a teoria de minimização de treliça proposta em (MCELIECE, 1996) e a teoria de seccionamento de treliça proposto em (VARDY, 1998). Decodifica-se utilizando o algoritmo max-log-MAP (VUCETIC; YUAN, 2000). Desenvolve-se uma métrica de complexidade baseada no número de operações e mostra-se técnicas para escolher-se seccionamentos que são simples e apresentam pouca ou nenhuma perda de desempenho em função da decodificação convencional.

**Palavras-chave:** Códigos Turbo, Seccionamento de Treliça, Treliça Mínima, Complexidade de treliça, Códigos convolucionais

## *ABSTRACT*

MORITZ, Guilherme Luiz. Complexity reduction of turbo codes using minimal trellis. 56 p. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2012.

We present a technique for reducing the turbo decoding complexity based on minimal and sectionalized trellises. A novel complexity metric is proposed and the complexity of all possible sectionalizations for some selected codes is evaluated. We use the minimal trellis representation proposed in (MCELIECE, 1996) for BCJR decoding and conclude that one can find less complex sectionalizations (when compared to the conventional trellis) which show small or none performance degradation.

Keywords: Turbo Codes, Trellis Sectionalization, Minimal Trellis, Trellis Complexity, Convolutional Codes

## LISTA DE FIGURAS

FIGURA 01	– MODELO DE SISTEMA DE COMUNICAÇÃO.....	15
FIGURA 02	– CODIFICADOR TURBO SISTEMÁTICO DE TAXA 1/3.....	17
FIGURA 03	– DECODIFICADOR TURBO.....	18
FIGURA 04	– ESTRUTURA GERAL DE UM CODIFICADOR CSR DE TAXA ELEVADA.....	21
FIGURA 05	– EXEMPLO DE MÁQUINA DE ESTADOS DE UM CODIFICADOR CONVOLUCIONAL $C_0(2,1,2)$ AO LADO DE SEU CIRCUITO CODIFICADOR.....	23
FIGURA 06	– EXEMPLO DE TRELIÇA CONVENCIONAL PARA UM CÓDIGO CONVOLUCIONAL $C_0(2,1,2)$ .....	24
FIGURA 07	– TRELIÇA CONVENCIONAL DO CÓDIGO TURBO DUOBINÁRIO DO WIMAX (IEEE STD 802.16E, 2009).....	25
FIGURA 08	– TRELIÇA MÍNIMA DO CÓDIGO TURBO DUOBINÁRIO DO WIMAX.....	26
FIGURA 09	– TRELIÇA DO WIMAX SECCIONADA COM $VETSEC = \{0,1,0\}$ .....	27
FIGURA 10	– DESEMPENHO DO CÓDIGO $C_1(4,2,3)$ , $D_2 = 7$ DO WIMAX.....	41
FIGURA 11	– TRELIÇA DE $C_2(4,2,3)$ , $D_2 = 6$ .....	42
FIGURA 12	– DESEMPENHO DO CÓDIGO $C_2(4,2,3)$ , $D_2 = 6$ , BLOCO DE 480 BITS, SEM CORREÇÃO.....	43
FIGURA 13	– TRELIÇA EXPERIMENTAL MÍNIMA 2, $D_2 = 13$ .....	44
FIGURA 14	– DESEMPENHO DO CÓDIGO $C_3(4,2,4)$ , $D_2 = 13$ , BLOCO DE 480BITS, SEM CORREÇÃO.....	45
FIGURA 15	– DESEMPENHO DO CÓDIGO $C_1(4,2,3)$ , $D_2 = 7$ DO WIMAX, BLOCO DE 96 BITS, SEM CORREÇÃO.....	46
FIGURA 16	– DESEMPENHO DO CÓDIGO $C_1(4,2,3)$ , $D_2 = 7$ DO WIMAX, BLOCO DE 480 BITS, COM CORREÇÃO.....	47
FIGURA 17	– TRELIÇA MÍNIMA DE $C_3(4,2,4)$ .....	48
FIGURA 18	– DESEMPENHO DO CÓDIGO $C_4(4,3,4)$ , $D_2 = 6$ , BLOCO DE 720 BITS, SEM CORREÇÃO.....	49
FIGURA 19	– TRELIÇA MÍNIMA DE $C_5(5,3,5)$ .....	50
FIGURA 20	– DESEMPENHO DO CÓDIGO $C_5(5,3,5)$ , $D_2 = 10$ , BLOCO DE 720 BITS SEM CORREÇÃO.....	50

## LISTA DE TABELAS

TABELA 01	- COMPLEXIDADE DO CÓDIGO $C_1(4,2,3)$ DO WIMAX (IEEE STD 802.16E, 2009).....	37
TABELA 02	- COMPLEXIDADE DO CÓDIGO $C_2(4,2,3)$ COM $D_2 = 6$ (BENCHIMOL, 2011).....	42
TABELA 03	- COMPLEXIDADE DO CÓDIGO $C_3(4,2,4)$ COM $D_2 = 13$ (BENCHIMOL, 2011).....	46
TABELA 04	- COMPLEXIDADE DO CÓDIGO $C_4(4,3,5)$ DE (BENCHIMOL, 2011).....	48
TABELA 05	- COMPLEXIDADE DO CÓDIGO $C_5(5,3,5)$ DE (BENCHIMOL, 2011).....	51

## LISTA DE SIGLAS

dB	Decibel
CSR	Convolucional sistemático recursivo
BPSK	<i>Binary Phase Shift Keying</i>
AWGN	<i>Additive white gaussian noise</i>
LLR	<i>Log-likelihood ratio</i>
CRC	<i>Cyclic Redundancy Check</i>
MAP	<i>Maximum a posteriori</i>

## LISTA DE SÍMBOLOS

$\mathbf{c}$	Mensagem binária
$S_t$	Estado da máquina de codificação no instante $t$
$\mathbf{v}$	Vetor de saída de decodificação
$\mathbf{x}$	Palavra codificada modulada
$\mathbf{r}$	Palavra recebida distorcida por ruído AWGN
$N$	Comprimento do entrelaçador
$k$	Número de bits de entrada
$\mathbf{c}'$	Vetor de entrada entrelaçado
$\Lambda_{2e}$	Informação extrínseca do segundo decodificador
$\Lambda_1$	LLR do primeiro decodificador
$\Lambda_{1e}$	Informação extrínseca do primeiro decodificador
$\Lambda_2$	LLR do segundo decodificador
$\nu$	Número de memórias do codificador
$M_c$	Matriz de conexão de código de taxa elevada
$\mathbf{T}_t$	Vetor de conexão no instante $t$
$m$	Número de bits de entrada em cada instante de codificação de um codificador de taxa elevada
$d_2$	Peso mínimo da palavra codificada gerada por uma sequência de informação de peso 2
$C_j(n, k, \nu)$	Código convolucional constituinte
$n$	Número de saídas binárias
$M$	Módulo da treliça
$M_{min}$	Módulo da treliça mínima
$n'$	Número de seções da treliça seccionada
$vetsec$	Vetor de seccionamento de treliça

$v_i^{sec}$	Complexidade de estado da seção de índice $i$
$b_i^{sec}$	Complexidade de ramo da seção $i$
$l_i^{sec}$	Número de bits de rótulo da seção $i$
$R$	Taxa do código
$\rho_q(c_i)$	Métrica de rótulo de treliça
$\Lambda_q(c_i)$	Razão logarítmica de probabilidade da palavra $q$ em relação à métrica de referência para o símbolo transmitido $c_i$
$\bar{\alpha}_i(l)$	Logaritmo da probabilidade do caminho de codificação na treliça ter passado pelo estado $l$ , no instante $i$ , quando se calcula do início da treliça para seu fim
$\bar{\beta}_i(l)$	Logaritmo da probabilidade do caminho de codificação na treliça ter passado pelo estado $l$ , no instante $i$ , quando se calcula do final da treliça para seu início
$\bar{\gamma}_i(l', l)$	Logaritmo da probabilidade de ocorrer uma transição entre o estado $l'$ e $l$ , no instante $i$
$r_{i,u}$	$u$ -ésimo símbolo da $n$ -tupla recebida no instante $i$
$x_{i,u}$	Símbolo que deveria ter sido transmitido no instante $i$ caso ocorresse a transição entre os estados $l'$ e $l$
$TC(M)$	Métrica de complexidade de módulo de treliça segundo McEliece
$\mathcal{S}$	Número de somas
$\mathcal{M}$	Número de multiplicações
$\mathcal{C}$	Número de comparações
$T_z$	Complexidade algorítmica para o cálculo de $z$ , onde $z$ é uma variável do algoritmo de max-log-MAP

## SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	<b>12</b>
1.1 MOTIVAÇÃO .....	12
1.2 OBJETIVOS .....	13
1.2.1 Resultados obtidos .....	14
<b>2 FUNDAMENTAÇÃO TEÓRICA</b> .....	<b>15</b>
2.1 CÓDIGOS TURBO .....	15
2.1.1 Introdução .....	15
2.1.2 Modelamento do sistema .....	15
2.1.3 Codificação Turbo .....	17
2.1.4 A Importância do Entrelaçador .....	17
2.1.5 Decodificação Turbo .....	18
2.1.6 Códigos Turbo de Taxa Elevada .....	20
2.1.7 Parâmetros de desempenho de códigos turbo .....	22
2.2 REPRESENTAÇÕES DE TRELIÇAS PARA CÓDIGOS CONVOLUCIONAIS ....	22
2.2.1 Preliminares: Máquinas de Estado .....	22
2.2.2 Treliza Convencional .....	23
2.2.3 Treliza Mínima .....	24
2.2.4 Treliza Seccionada .....	26
2.2.5 Obtenção da treliza mínima para códigos convolucionais sistemáticos recursivos ...	28
2.3 O ALGORITMO MAP .....	28
2.3.1 Versão logarítmica do BCJR .....	28
2.3.2 Correção do algoritmo de max-log-MAP .....	30
<b>3 MÉTRICAS DE COMPLEXIDADE</b> .....	<b>32</b>
3.1 MÉTRICA DE COMPLEXIDADE PROPOSTA POR MCELIECE E LIN .....	32
3.2 MÉTRICA PROPOSTA PARA COMPLEXIDADE DO MAX-LOG-MAP .....	32
3.2.1 Considerações iniciais .....	32
3.2.2 Treliza seccionada .....	33
3.2.3 Treliza convencional .....	35
3.2.4 Treliza mínima .....	35
3.3 EXEMPLO - $C_1(4, 2, 3)$ DO WIMAX (IEEE STD 802.16E, 2009) .....	36
3.3.1 Definição do código .....	36
3.3.2 Complexidade do código .....	37
3.3.3 Comentários sobre o código .....	37
3.4 CONSIDERAÇÕES SOBRE A MÉTRICA DE COMPLEXIDADE .....	37
<b>4 AVALIAÇÃO DE DESEMPENHO DE CÓDIGOS TURBO EM FUNÇÃO DO SEC-</b>	
<b>CIONAMENTO</b> .....	<b>39</b>
4.1 INTRODUÇÃO .....	39
4.2 PARÂMETROS GERAIS DOS TESTES .....	39
4.3 CARACTERÍSTICAS DAS TRELIÇAS UTILIZADAS PARA TESTES .....	39
4.4 TESTE 1 - TRELIÇA DO WIMAX .....	40
4.4.1 Análise de complexidade e desempenho .....	40

4.5	TESTE 2 - REDUÇÃO DA COMPLEXIDADE DA TRELIÇA .....	41
4.5.1	Análise de complexidade e desempenho .....	42
4.6	TESTE 3 - AUMENTO DE COMPLEXIDADE DA TRELIÇA .....	43
4.6.1	Análise de complexidade e desempenho .....	44
4.7	TESTE 4 - REDUÇÃO DO TAMANHO DO ENTRELAÇADOR .....	45
4.8	TESTE 5 - UTILIZAÇÃO DA CORREÇÃO .....	45
4.9	TESTE 6 - AUMENTO DA TAXA DO CÓDIGO .....	46
4.9.1	Análise de complexidade e desempenho .....	48
4.10	TESTE 7 - SEGUNDO EXEMPLO DE AUMENTO DA TAXA DO CÓDIGO .....	49
4.10.1	Análise de complexidade e desempenho .....	51
4.11	CONCLUSÕES .....	51
<b>5</b>	<b>CONCLUSÃO .....</b>	<b>53</b>
	<b>REFERÊNCIAS .....</b>	<b>55</b>

# 1 INTRODUÇÃO

## 1.1 MOTIVAÇÃO

A evolução da computação mudou radicalmente o modo de vida da população mundial. Vivenciamos a expansão do mercado de computadores pessoais, que a cada geração aumentam seu poder de processamento, diminuem em custo, e se tornam cada vez menores. Observamos a evolução do uso da internet que há menos de uma década era um simples método alternativo de pesquisa para hoje atuar como a principal, ou no mínimo uma das maiores fontes de informação e entretenimento e até socialização da população. Assim, podemos afirmar que vivemos numa sociedade que deseja estar sempre conectada, o que enfatiza novos requerimentos para os sistemas computacionais.

O mercado demanda um computador que não só tenha grande poder de processamento, mas que seja portátil (e portanto alimentado a bateria) e tenha conexão permanente com a internet, para ser capaz de trocar informação a qualquer momento, sem ser a vida útil da bateria um empecilho para a mobilidade. Estes requisitos tornam de suma importância o aperfeiçoamento dos sistemas de comunicações, a fim de equilibrar os desempenhos contrastantes de taxas de comunicações cada vez mais elevadas com autonomia de bateria cada vez maior.

Aos desejos de desempenho e conexão, alia-se a consciência de que a sociedade deve ser sustentável. O sistemas não podem buscar desempenho a qualquer custo. Energia é um recurso que deve ser economizado em prol de nosso ambiente. Equipamentos sem fio com maior autonomia significam menos ciclos de recarga, que por sua vez significam baterias com maior vida útil, que desta maneira demoram mais para precisarem ser descartadas ou recicladas. Além disso, a minimização de potência requerida não é importante apenas no mercado de dispositivos móveis de consumo: há outras áreas onde a energia é limitada, como em redes de sensores, implantes médicos e transmissões via satélite.

## 1.2 OBJETIVOS

Objetiva-se atuação na redução dos requerimentos de potência de enlaces de comunicação modernos. Entre estes sistemas, destacam-se redes sem fio metropolitanas WiMax (IEEE STD 802.16E, 2009), comunicações celulares de terceira e quarta geração e transmissão de TV digital.

Segundo (MASSELOS; BLIONAS; RAUTIO, 2002), o processamento da correção de erros convolucional corresponde a até 76% do total de processamento em um decodificador HYPERLAN/2. Em (BOUGARD et al., 2004) é analisada a complexidade de decodificadores para o padrão 802.11, indicando que o algoritmo de Viterbi (VITERBI, 1967) contribui com até 35% da complexidade do decodificador. Ressalta-se que maior complexidade computacional se traduz em maiores exigências de potência.

A proporção de processamento necessária para a correção de erros tende a aumentar quando o código convolucional é substituído por outros de maior desempenho, como os códigos Turbo (BERROU; GLAVIEUX; THITIMAJSHIMA, 1993) e LDPC (GALLAGER, 1962) já que estes códigos utilizam processos iterativos que podem ser comparados em complexidade a várias iterações do algoritmo de Viterbi. Apesar de apresentarem uma maior latência e complexidade de codificação, códigos Turbo e LDPC possuem desempenho que pode chegar, num canal AWGN, a até 0.3dB do limite teórico definido por Shannon (SHANNON, 1948). Por este motivo, são previstos em padrões modernos de comunicação pois são úteis para permitir comunicação confiável em baixas relações sinal-ruído.

Há várias maneiras de reduzir-se a complexidade de um decodificador turbo, entre elas podemos citar otimizações específicas de implementação (LEE; PARK, 2005) (LEUNG et al., 1999), utilização de algoritmos mais simples de decodificação ou utilizar representações mais simples de treliça (GARELLO et al., 2000) (KIM et al., 2000) (MINOWA; IMAI, 2000) (PAHARALABOS et al., 2009).

Podem existir diversas treliças para um único código, como a representação convencional, a puncionada (CAIN; CLARK; GEIST, 1979), a mínima (MCELIECE; LIN, 1996) e a seccionada (VARDY, 1998), entre outras. Cada uma destas representações possui uma complexidade distinta para o mesmo código. A treliça convencional é a mais complexa e existe para todos os códigos. A treliça puncionada é uma representação simplificada para apenas alguns códigos convolucionais, a treliça mínima é uma representação para todos os códigos convolucionais, possuindo complexidade de treliça menor ou igual à complexidade da treliça convencional. Já a treliça seccionada é uma representação intermediária entre a treliça mínima e convencional.

O objetivo da dissertação apresentada é utilizar representações menos complexas, como a treliça mínima e seccionada para a decodificação de códigos turbo a fim de tornar menor sua complexidade de decodificação. A técnica de obtenção de treliças mínimas para códigos turbo utilizada é a desenvolvida em (PIMENTEL et al., 2011). Analisa-se os impactos da aplicação da técnica no desempenho (taxa de erro em função da relação sinal-ruído) e avalia-se a redução de complexidade de processamento, que reflete diretamente ou na potência consumida pelo processo de decodificação e/ou na área de silício utilizada para realização do hardware de decodificação. O processo de decodificação proposto utilizará a teoria de minimização de treliça proposta em (MCELIECE, 1996) para decodificação de códigos turbo baseados no algoritmo max-Log-MAP (VUCETIC; YUAN, 2000).

### 1.2.1 Resultados obtidos

Os estudos sobre a complexidade do algoritmo mostram que nem sempre a minimização de treliça resulta em redução de complexidade do processo de decodificação. Por este motivo foi proposta a utilização de seccionamento de treliça (VARDY, 1998), que é uma técnica que representa o código de uma maneira intermediária entre a representação convencional e a mínima.

Conclui-se que a técnica de representação seccionada é capaz de reduzir a complexidade de decodificação turbo sem alterar seu desempenho, mas não em todos os casos. As restrições que devem ser seguidas para obtenção deste objetivo mencionado são enumeradas. Em adição, estudos sobre o compromisso entre complexidade e desempenho são desenvolvidos. Mostra-se que nem todo código apresenta redução de complexidade significativa sem afetar o desempenho e descreve-se meios de identificar se o código é bom candidato para a redução de complexidade, demonstrando-se o processo de cálculo de complexidade de todos os possíveis seccionamentos e enumerando-se os impactos de desempenho em função da redução de complexidade.

O restante desta dissertação está organizado da seguinte maneira. O Capítulo 2 apresenta conceitos necessários para entendimento do método utilizado tanto para a decodificação turbo, quanto para a representação das treliças. O Capítulo 3 apresenta o desenvolvimento das métricas utilizadas para avaliação da complexidade dos códigos. O Capítulo 4 apresenta o estudo da aplicação da técnica proposta a vários códigos disponíveis na literatura. Finalizando, no Capítulo 5 apresenta-se os comentários finais sobre a técnica apresentada, assim como eventuais trabalhos futuros.

## 2 FUNDAMENTAÇÃO TEÓRICA

### 2.1 CÓDIGOS TURBO

#### 2.1.1 Introdução

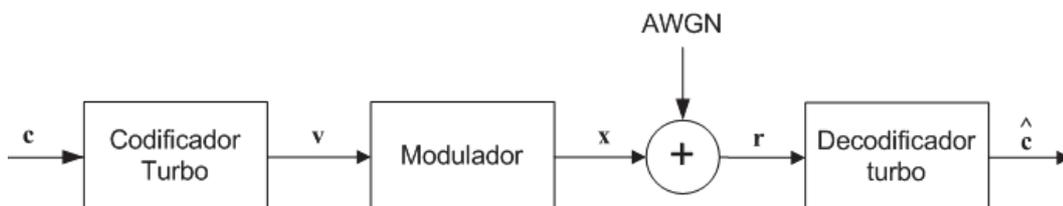
Códigos turbo foram apresentados primeiramente em 1993, em (BERROU; GLAVIEUX; THITIMAJSHIMA, 1993) e representaram uma revolução na área de códigos corretores de erros. A principal característica alcançada, que logo despertou interesse da comunidade científica foi o excelente desempenho desta classe de código, afastado apenas 0.3 dB da capacidade teórica quando o canal de transmissão é AWGN.

Um codificador turbo pode ser descrito como a concatenação paralela de dois codificadores convolucionais sistemáticos recursivos (Codificador CSR), baseados em máquinas de estados com resposta ao impulso infinita, conforme descrito nas seções subsequentes.

#### 2.1.2 Modelamento do sistema

Toda discussão sobre o processo de codificação e decodificação será baseado em um sistema como descrito na Figura 1, onde a mensagem binária  $\mathbf{c}$ , onde

$$\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_t, \dots, \mathbf{c}_N) \quad (1)$$



**Figura 1: Modelo de sistema de comunicação.**

**Fonte: Autoria Própria**

é composta por  $N$  símbolos  $c_t$ , compostos por  $k$  bits de informação:

$$\mathbf{c}_t = (c_{t,1}, c_{t,2}, \dots, c_{t,k}). \quad (2)$$

A operação de codificação será descrita por um modelo de Markov discreto de estados finitos, que pode ser representado por diagramas de estados de treliça. Uma entrada  $\mathbf{c}_t$  gera uma saída  $\mathbf{v}_t$  e faz o sistema transicionar do estado  $S_t$  para  $S_{t+1}$ , o que pode ser descrito pelas funções de saída

$$\mathbf{v}_t = f(S_t, \mathbf{c}_t, t) \quad (3)$$

e transição

$$S_{t+1} = g(S_t, \mathbf{c}_t, t). \quad (4)$$

O vetor de saída  $\mathbf{v}$  é dado por:

$$\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \dots, \mathbf{v}_t, \dots, \mathbf{v}_N) \quad (5)$$

onde

$$\mathbf{v}_t = (v_{t,1}, v_{t,2}, \dots, v_{t,n}). \quad (6)$$

A sequência codificada  $\mathbf{v}$  é modulada em BPSK, gerando o vetor  $\mathbf{x}$

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_t, \dots, \mathbf{x}_N) \quad (7)$$

onde

$$\mathbf{x}_t = (x_{t,1}, x_{t,2}, \dots, x_{t,n}). \quad (8)$$

À sequência  $\mathbf{x}$  é adicionado ruído branco aditivo Gaussiano (AWGN), o que resulta na sequência recebida  $\mathbf{r}$ :

$$\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \dots, \mathbf{r}_t, \dots, \mathbf{r}_N) \quad (9)$$

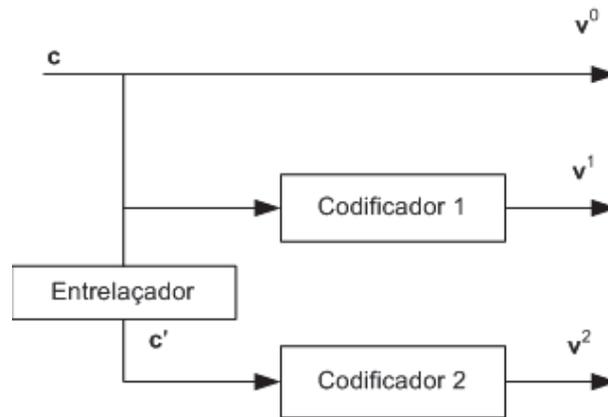
onde

$$\mathbf{r}_t = (r_{t,1}, r_{t,2}, \dots, r_{t,n}) \quad (10)$$

e

$$r_{t,i} = (x_{t,i} + n_{t,i}), \quad (11)$$

sendo  $n_{t,i}$  uma variável aleatória Gaussiana de média zero e variância  $\sigma^2 = \frac{N_0}{2}$ , sendo  $N_0$  a densidade espectral de potência unilateral do ruído do canal. O decodificador gera uma estimativa da entrada do modelo de Markov operando sobre a sequência recebida  $\mathbf{r}$ .



**Figura 2: Codificador Turbo Sistemático de Taxa 1/3**

**Fonte: Autoria Própria**

### 2.1.3 Codificação Turbo

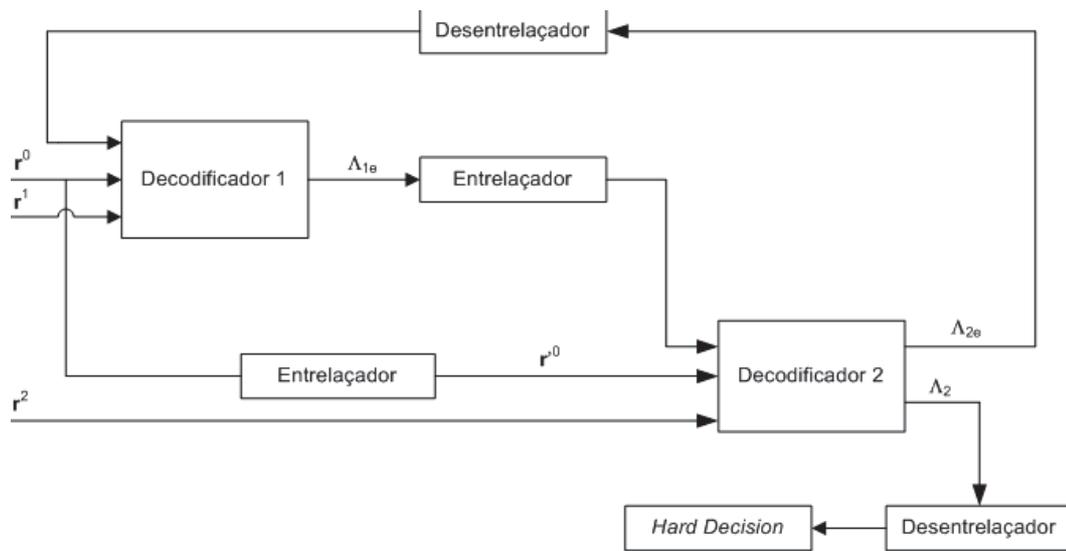
A Figura 2 descreve o sistema de codificação turbo. Dois codificadores são conectados em paralelo através de um entrelaçador. Desta maneira, cada bit será codificado duas vezes, porém em ordens diferentes. O primeiro codificador apresenta duas saídas, recebendo a entrada de maneira direta (representada como  $\mathbf{c}$  na figura, composta por  $N$  palavras de  $k$  bits). A primeira saída, denotada  $\mathbf{v}^0$  é idêntica à entrada  $\mathbf{c}$ , pois o codificador é sistemático. A outra saída é o vetor de paridade do Codificador 1, representado por  $\mathbf{v}^1$ . Para servir de entrada para o Codificador 2, o vetor de entrada  $\mathbf{c}$  é entrelaçado, gerando-se o vetor  $\mathbf{c}'$ . O Codificador 2 gera, então, a informação de paridade  $\mathbf{v}^2$ . Repara-se que não é necessária a transmissão da entrada  $\mathbf{c}'$ , pois a mesma pode ser construída no decodificador a partir da sequência  $\mathbf{c}$ , já que no decodificador também há conhecimento da operação de entrelaçamento. Os vetores  $\mathbf{v}^0$ ,  $\mathbf{v}^1$  e  $\mathbf{v}^2$  são multiplexados gerando a palavra codificada  $\mathbf{v}$  onde  $\mathbf{v}_t = \mathbf{v}^0 \mathbf{v}^1 \mathbf{v}^2$ . A taxa do esquema apresentado é 1/3.

### 2.1.4 A Importância do Entrelaçador

A função principal do entrelaçador é descorrelacionar as entradas dos dois codificadores convolucionais. Desta maneira, um algoritmo sub-ótimo baseado na troca de informação entre os dois decodificadores pode ser aplicado na decodificação. Trabalhando com entradas não correlacionadas surge uma alta probabilidade de que os erros que não possam ser corrigidos pelo primeiro decodificador possam ser corrigidos pelo segundo. A troca iterativa de informação faz com que o processo de decodificação convirja para resultados muito melhores do que os resultados da decodificação individual de seus codificadores constituintes.

Um entrelaçador pseudo-aleatório de permutação de  $N$  elementos, é suficiente para descorrelacionar as entradas. Todavia, estudos comprovam que versões mais complexas de entrelaçamento podem acarretar em ganhos de desempenho, como, entre outros, o entrelaçador uniforme proposto (ou de permutação) em (DOUILLARD; BERROU, 2005) e o entrelaçador *s-random* proposto por Divsalar (DIVSALAR; POLLARA, 1995).

### 2.1.5 Decodificação Turbo



**Figura 3: Decodificador turbo**

**Fonte: Autoria Própria**

A Figura 3 apresenta a estrutura de um decodificador turbo. Observa-se um arranjo similar ao codificador: dois decodificadores constituintes separados por um entrelaçador, além de outros entrelaçadores e desentrelaçadores responsáveis por adequar a saída de um codificador ao formato esperado na entrada do outro. O nome do código se deve ao fato da saída do decodificador realimentar sua entrada, assim como um motor turbo realimenta sua entrada com a pressão dos gases de saída.

O algoritmo de decodificação será detalhado na Seção 2.3.1, por ora, analisa-se apenas as entradas e saídas do algoritmo. A saída do decodificador é denominada *log-likelihood ratio* (LLR), que é definida para o caso binário (onde  $k = 1$ ) como (VUCETIC; YUAN, 2000):

$$\Lambda(\mathbf{c}_t) = \ln \frac{P_r(\mathbf{c}_t = 1 | \mathbf{r})}{P_r(\mathbf{c}_t = 0 | \mathbf{r})}, \quad (12)$$

e representa o logaritmo da probabilidade de um bit 1 em relação a um bit 0. O módulo da LLR

$(|\Lambda(\mathbf{c}_t)|)$  indica quão confiável é uma decisão de bit tomada pelo algoritmo. Já o sinal da LLR é utilizado para estimar-se o valor do bit transmitido (denominado  $\mathbf{c}_t$ ), como segue:

$$\hat{\mathbf{c}}_t = \begin{cases} 1 & \text{se } \Lambda(\hat{\mathbf{c}}_t) \geq 0 \\ 0 & \text{caso contrário.} \end{cases} \quad (13)$$

Este comportamento é característico de algoritmos de decisão suave.

Para geração da saída, três entradas são requeridas:

- a informação suave dos bits de informação recebidos;
- a informação suave sobre os bits de paridade recebidos;
- a informação extrínseca gerada pelo outro decodificador.

Dos itens de entrada, tanto a informação suave dos bits de informação quanto dos bits de paridade representam os bits que foram gerados no codificador, distorcidos pelo ruído do canal. Somente a informação gerada pelos bits de paridade é utilizada como entrada do outro codificador (denominada informação extrínseca). A informação dos bits recebidos é passada ao decodificador diretamente. Uma segunda utilização afetaria o processo de convergência da decodificação. Desta maneira, o processo iterativo funciona da seguinte maneira:

- ao Decodificador 1 são aplicados os vetores representando a palavra de informação recebida ( $\mathbf{r}^0$ ) e de paridade correspondente ( $\mathbf{r}^1$ ), a informação extrínseca do segundo codificador ( $\Lambda_{2e}$ ) naturalmente é nula, pois ainda não há estimativa deste codificador (considerando-se as palavras de entrada equiprováveis);
- como saída o Decodificador 1 gera um vetor de LLRs ( $\Lambda_1$ );
- a informação extrínseca gerada pelo primeiro decodificador ( $\Lambda_{1e}$ ) é calculada subtraindo-se a informação dos bits de informação obtidas diretamente do canal ( $\mathbf{r}^0$ ) da LLR  $\Lambda_1$ ;
- o segundo decodificador tem dados suficientes para operar. Repara-se que tanto a informação extrínseca gerada ( $\Lambda_{1e}$ ) quanto os bits de informação ( $\mathbf{r}^0$ ) precisam ser entrelaçados para se adequarem ao formato utilizado pelo Decodificador 2;
- o Decodificador 2 gera sua própria LLR ( $\Lambda_2$ ), que geralmente é uma estimativa melhor do que a previamente calculada pelo Decodificador 1, pois além das informações obtidas da recepção, utiliza a informação extrínseca calculada pelo Decodificador 1;

- a nova informação extrínseca ( $\Lambda_{2e}$ ) é calculada a partir da segunda LLR ( $\Lambda_2$ ), além de ser devidamente desentrelaçada para se adequar à ordem esperada pelo Decodificador 1;
- o processo se reinicia, porém agora a informação extrínseca para o Decodificador 1 ( $\Lambda_{2e}$ ) não é mais nula.

A cada nova iteração do processo a contribuição extrínseca de cada codificador vai diminuindo, até que o processo convirja. Então, a última LLR gerada pelo Decodificador 2 é utilizada na determinação da palavra decodificada ( $\mathbf{c}_t$ ) utilizando-se (13). Geralmente utiliza-se um número fixo de iterações. Segundo (VUCETIC; YUAN, 2000), para um entrelaçador de 4096 bits, não há ganho de desempenho significativo após a décima segunda iteração. Para um entrelaçador de 16384, este valor sobe para 15. Na prática, pode-se também utilizar um código detetor de erros (por exemplo, CRC) ao fim de cada iteração para determinar se a palavra já foi decodificada corretamente.

#### 2.1.6 Códigos Turbo de Taxa Elevada

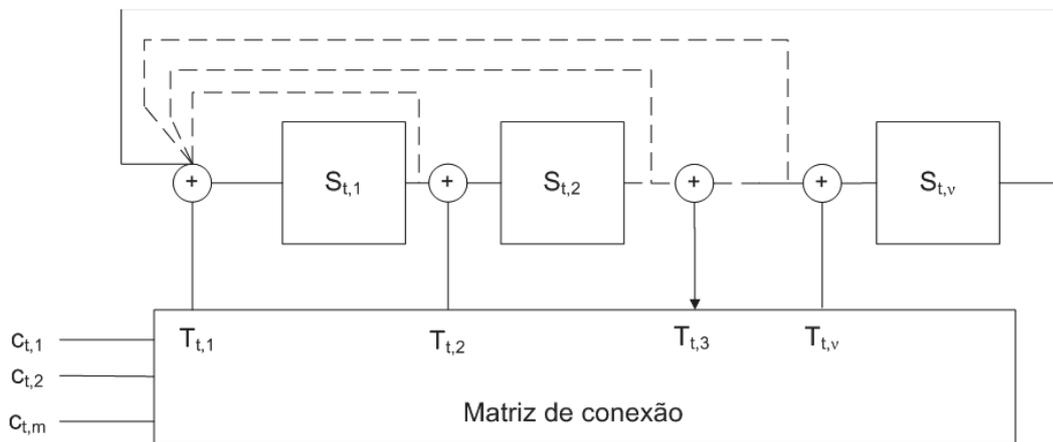
Todo o desenvolvimento de códigos turbo apresentado até o momento baseou-se em codificadores de taxa 1/3, relativamente baixa. Por sua vez, há dois métodos principais de gerar códigos de taxa elevada. No primeiro, denominado puncionamento, alguns bits de paridade são ignorados de forma regular no momento da transmissão. No processo de decodificação, as posições onde estariam os bits deletados são inicializadas com um valor equiprovável para 0 e 1, não influenciando nas decisões do processo de decodificação, porém diminuindo o desempenho do código quando comparado aos códigos de menor taxa, não puncionados. O segundo método, desenvolvido em (DOUILLARD; BERROU, 2005), alia taxas mais elevadas com melhor desempenho. Em vez de puncionar bits de paridade no momento da transmissão, a técnica utiliza códigos convolucionais constituintes de taxa naturalmente mais elevada.

A Figura 4 descreve a estrutura de um codificador CSR de taxa elevada com  $v$  memórias binárias, onde no instante  $t$  o vetor  $m$ -ário de entrada  $\mathbf{c}_t = (c_{t,1} \dots c_{t,l} \dots c_{t,m})^T$  está conectado às  $v$  memórias através de uma matriz de conexão  $M_c$ , de dimensões  $v \times m$ . O vetor de conexão no instante  $t$ , denotado  $\mathbf{T}_t$  é definido como

$$\mathbf{T}_t = M_c \mathbf{c}_t. \quad (14)$$

A saída do codificador, não representada na figura, é calculada como

$$\mathbf{v}_t = \sum_{j=1, \dots, m} c_{t,j} + \mathbf{Q} \mathbf{S}_t, \quad (15)$$



**Figura 4: Estrutura geral de um codificador CSR de taxa elevada**

**Fonte: (DOUILLARD; BERROU, 2005).**

onde  $\mathbf{S}_t$  é um vetor que representa o estado do codificador no instante  $t$  e  $\mathbf{Q}$  é um vetor com  $v$  componentes, que vale 1 na posição  $p$  se a  $p$ -ésima componente de  $\mathbf{S}_t$  está presente no cálculo de  $\mathbf{v}_t$  e zero caso contrário.

A complexidade de decodificação de códigos de alta taxa é fortemente influenciada pelo valor escolhido para  $m$ . Seguindo (DOUILLARD; BERROU, 2005), na presente dissertação estudou-se códigos com  $m = 2$  ou  $m = 3$ , pois códigos com  $m > 3$  apresentam complexidade muito elevada para serem utilizados em aplicações reais levando em consideração o atual estágio de poder de processamento dos sistemas.

Como exemplo, considera-se o código turbo do WiMax. Neste código tem-se  $m = 2$ , que faz com que o código constituinte tenha taxa  $\frac{m}{m+1} = 2/3$ . Assim, temos um bit de paridade para cada dois bits aplicados ao codificador. Como o código turbo é constituído por dois codificadores desta taxa, gera-se um código com taxa  $2/4$ . A taxa do código pode ser elevada aplicando-se punçãoamento.

Em relação aos códigos turbo clássicos, destaca-se as seguintes vantagens dos códigos turbo de taxa naturalmente elevada (DOUILLARD; BERROU, 2005):

- melhor convergência do processo iterativo;
- distâncias mínimas maiores, que se traduzem em códigos de maior desempenho;
- necessidade de punçãoamento menos agressivo para alcance de taxas maiores;
- observou-se que as simplificações do algoritmo de decodificação (utilização do max-log-MAP no lugar do log-MAP) surtem menor efeito negativo no desempenho do código

duobinário.

Em contrapartida, a utilização de codificadores de taxa  $\frac{m}{m+1}$  faz com que o processo de decodificação seja mais complexo.

As vantagens apresentadas motivam o uso da técnica em padrões modernos de comunicação, como o 802.16a (IEEE STD 802.16E, 2009).

### 2.1.7 Parâmetros de desempenho de códigos turbo

Em (VUCETIC; YUAN, 2000), uma análise dos limites de desempenho dos códigos turbo é realizada. Demonstra-se que a taxa de erro de bit do código é inversamente proporcional ao comprimento do entrelaçador (mas o valor possui comportamento assintótico, não diminuindo indefinidamente). Além disso, conclui-se que a taxa de erro de bit é dominada pelos caminhos na treliça que possuem peso de informação  $w_{min} = 2$ . Assim, define-se  $d_2$  como o peso mínimo da sequência de paridade nos caminhos da treliça do código CSR constituinte gerados por uma sequência de informação de peso  $w_{min} = 2$ . Por dominar a probabilidade de erro de bit, este parâmetro define o poder de correção do código.

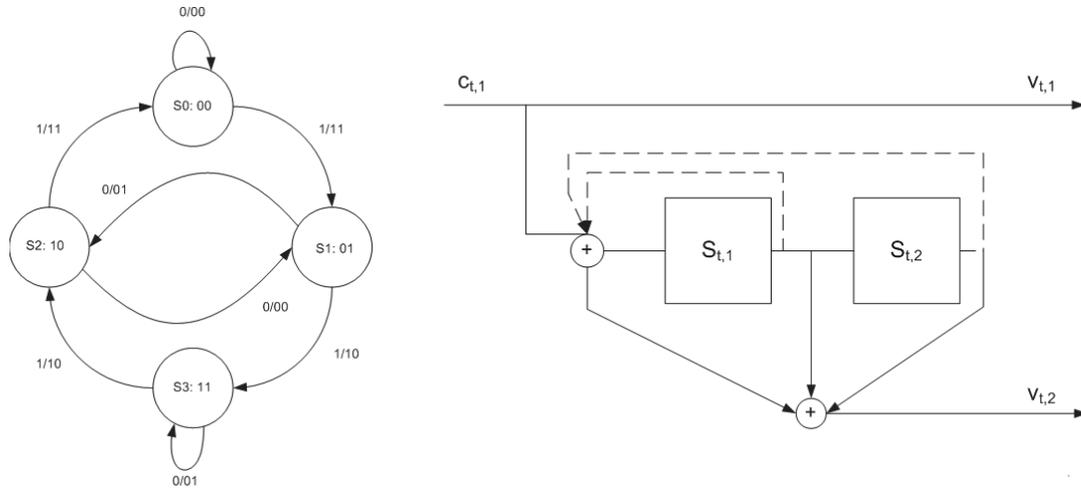
## 2.2 REPRESENTAÇÕES DE TRELIÇAS PARA CÓDIGOS CONVOLUCIONAIS

Estuda-se, nas seções subsequentes, a representação de treliça para códigos convolucionais. A principal importância deste estudo se deve ao fato de que os principais algoritmos de decodificação convolucional (que conseqüentemente constituem os decodificadores turbo) se baseiam na representação de treliça para operarem. Desta maneira, o método proposto para minimização de complexidade de códigos turbo se baseará em desenvolvimento de representações menos complexas de treliças. O impacto da minimização de treliça na complexidade dos algoritmos de decodificação turbo será investigado no Capítulo 3.

### 2.2.1 Preliminares: Máquinas de Estado

Uma máquina de estado finita pode representar a cadeia de Markov utilizada no modelamento do codificador. No evento de codificação, quando uma nova palavra binária  $\mathbf{c}_t$  de comprimento  $k$  é aplicada ao codificador, apenas o conteúdo desta palavra e o estado atual de codificação são necessários para a determinação do novo estado do codificador. Assim, o número de estados da cadeia de Markov, e conseqüentemente da máquina de estados resultante

será  $2^v$ , onde  $v$  é o número de memórias binárias do circuito de codificação. O número de transições de cada estado é  $2^k$ , cada uma rotulando um par entrada/saída. A Figura 5 exemplifica



**Figura 5: Exemplo de máquina de estados de um codificador convolucional  $C_0(2, 1, 2)$  ao lado de seu circuito codificador**

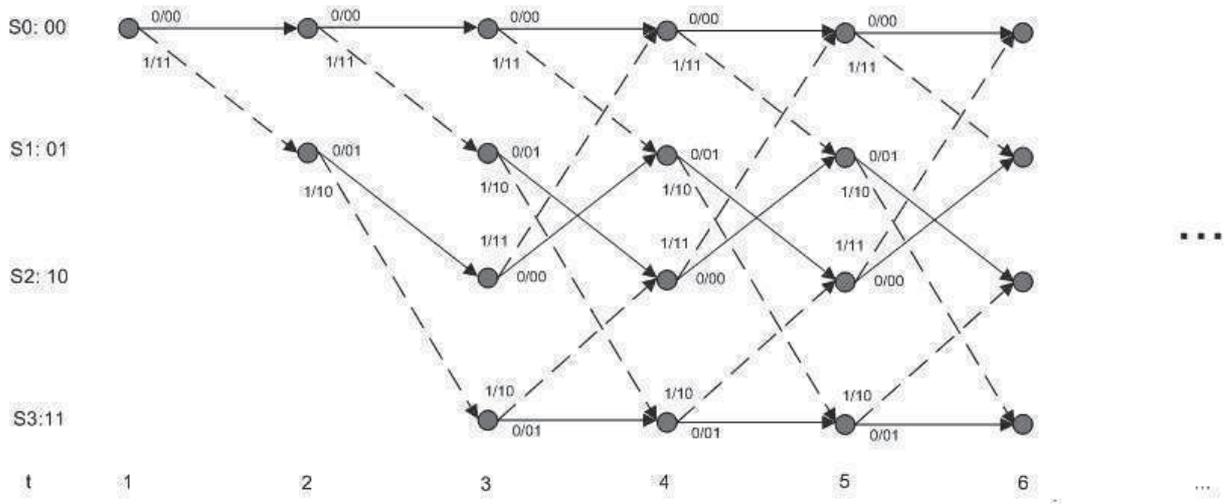
**Fonte: Autoria própria**

uma máquina de estados como descrito.

### 2.2.2 Treliça Convencional

Considere um código convolucional  $C_j(n, k, v)$ , onde  $v$ ,  $k$  e  $n$  são o número de memórias binárias do codificador, o número de entradas binárias e o número de saídas binárias, respectivamente enquanto  $j$  é um índice dos códigos mencionados nesta dissertação. A treliça do código é uma representação temporal do processo de codificação. Há uma linha para cada um dos  $2^v$  estados e uma coluna de pontos (onde cada ponto é denominado nó), para cada instante de codificação (cada instante de codificação representa a aplicação de uma palavra ao codificador). De cada nó partem  $2^k$  ramos, equivalentes às transições da máquina de estados na qual a treliça é baseada. Assim, uma treliça pode ser vista como várias cópias alinhadas da máquina de estados que a gera, sendo o número de cópias igual ao número de bits codificados. A Figura 6 é um exemplo de treliça gerada a partir da máquina de estados da Figura 5. Convencionou-se que bits de entrada '1' geram transições pontilhadas, enquanto bits de entrada '0' geram transições de linhas contínuas. Esta convenção será adotada no restante do documento.

Num processo de codificação, o estado inicial do codificador é conhecido. Assim, nos primeiros instantes de codificação nem todos os estados são possíveis e, por isso, não estão representados na figura. A cada instante de codificação o número de estados possíveis é multi-



**Figura 6: Exemplo de treliça convencional para um código convolucional  $C_0(2, 1, 2)$**

**Fonte: Autoria própria**

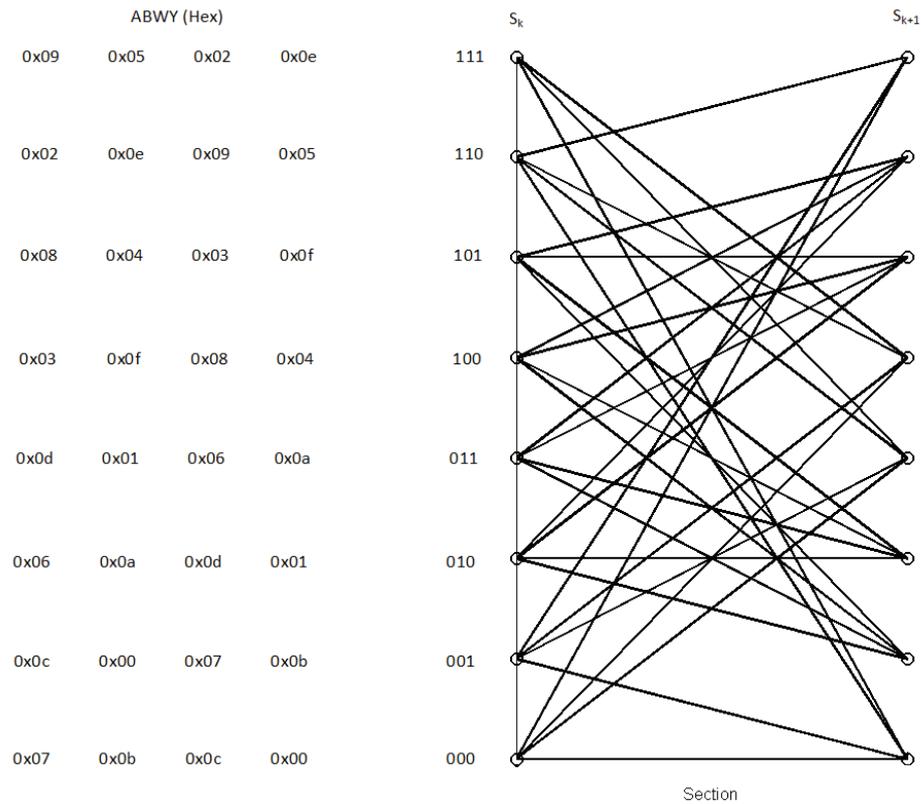
plicado por  $2^k$ , até que todos os  $2^v$  estados se tornem acessíveis. Passada esta etapa transitória, observa-se que a treliça passa a apresentar uma estrutura regular que se repete indefinidamente, com  $2^v$  estados iniciais e finais. Esta estrutura regular é denominada módulo da treliça convencional ( $M$ ). No exemplo da Figura 6 observa-se que o regime transitório termina no instante de codificação  $t = 2$ , observando-se a repetição do módulo da treliça a partir do instante  $t = 3$ . A sequência de saída pode ser obtida através da treliça traçando-se o caminho especificado pelas palavras de entrada.

### 2.2.3 Treliça Mínima

McEliece e Lin em (MCELIECE, 1996) afirmam que o formato da treliça está diretamente relacionado à complexidade de decodificação do algoritmo de Viterbi. Além disso, no artigo prova-se que a menor representação de treliça para um código de bloco é a proposta em (BAHL et al., 1974). Por ser a menor representação, esta classe de treliça foi batizada de treliça mínima.

A teoria proposta foi posteriormente expandida para códigos convolucionais em (MCELIECE; LIN, 1996), onde apresentou-se um novo método de construção de treliças para código convolucionais. Em (PIMENTEL et al., 2011), a técnica de criação de treliça foi expandida para códigos CSR. O método de obtenção desta treliça está fora do escopo desta dissertação.

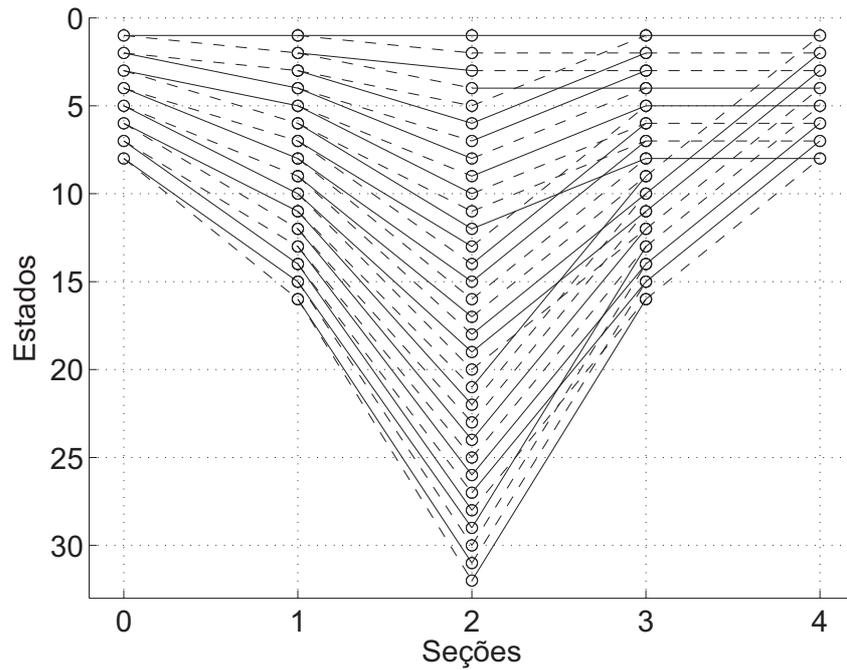
A Figura 7 representa o módulo de treliça convencional do código turbo do Wimax (IEEE STD 802.16E, 2009) com apenas uma seção. Este código é  $C_1(4, 2, 3)$  e conseqüentemente sua única seção apresenta  $2^v = 8$  estados, cada um sendo origem de  $2^k = 4$  ramos rotulados



**Figura 7: Treliça convencional do código turbo duobinário do Wimax (IEEE STD 802.16E, 2009)**

**Fonte: Adaptado de (VALENTI; SESHADRI, 2005)**

com  $n = 4$  bits. Os rótulos das transições foram posicionados à esquerda da treliça, sendo AB o par de bits sistemáticos, W a paridade gerada pelo primeiro codificador e Y a paridade gerada pelo segundo. Nas linhas, o rótulo mais à esquerda está associado à transição superior, o segundo rótulo à transição logo abaixo e os outros dois rótulos seguindo a lógica. Já a treliça mínima apresenta mais de uma seção para a composição do módulo de treliça  $M_{min}$ , conforme observado na Figura 8. Cada seção representa um bit codificado, totalizando  $n$  seções para a codificação de uma  $k$ -upla. Além disso, observa-se que algumas seções apresentam dois ramos de saída e outras apenas um. As seções com dois ramos de saída representam os bits sistemáticos, consequentemente, no exemplo existem  $k = 2$  seções (seções 0 e 1) com esta propriedade. As seções restantes (2 e 3) não carregam informação pois há apenas um caminho a ser seguido na treliça, desta maneira não possuem LLR associada e representam os bits de paridade. No exemplo, observa-se também que os 8 estados iniciais se expandem em até 32 estados.



**Figura 8: Treliça mínima do código turbo duobinário do Wimax.**

**Fonte: Autoria própria**

#### 2.2.4 Treliça Seccionada

As representações de treliça apresentadas nas Seções 2.2.2 e 2.2.3 podem ser considerados extremos, já que na primeira é utilizada apenas uma seção na representação e na última utiliza-se uma seção para cada bit codificado, o que resulta no número máximo de seções para uma treliça. Estudam-se agora os casos intermediários, onde qualquer número de seções (entre 1 e  $n$ ) pode ser utilizado para construção do módulo, fazendo com que os ramos de seção possam ter de 1 a  $n$  bits de rótulo. A técnica da seccionamento é interessante pois tem a capacidade de unir as vantagens das treliças mínima e convencional, desde que o vetor de seccionamento seja corretamente escolhido. Um estudo sobre a escolha do vetor de seccionamento será desenvolvido no Capítulo 3. Faz-se agora a definição formal de uma treliça seccionada (que também inclui os casos mínimo e convencional), conforme (BENCHIMOL, 2011).

Considera-se um módulo de treliça mínima  $M_{min}$  definido na Seção 2.2.3. Um módulo no instante  $t$  possui  $n$  seções compreendidas entre os índices  $i = 0, \dots, n$ . Define-se o seccionamento do tempo  $t$  para  $i = 1, \dots, n - 1$  como a remoção dos estados no índice  $i$  e a conexão dos estados dos índices  $i - 1$  diretamente com os estados do índice  $i + 1$ , desde que exista, em  $M_{min}$  um caminho entre os estados de  $i - 1$  e  $i + 1$ . O rótulo dos ramos no módulo seccionado é formado pela concatenação dos rótulos dos ramos dos caminhos entre  $i - 1$  e  $i + 1$  em  $M_{min}$ .

O módulo seccionado resultante possui  $n' = n - 1$  seções. O processo de seccionamento pode então ser repetido, fazendo que sejam possíveis  $2^{n-1}$  formas de seccionamento. O número de seções do módulo de treliça seccionada varia de 1 (treliça convencional) a  $n$  (treliça mínima).

Define-se um vetor binário de seccionamento, denotado  $vetsec$ , com  $n - 1$  elementos. Caso o  $i$ -ésimo elemento do vetor seja 1, indica que o índice  $i$  de  $M_{min}$  está seccionado. A treliça mínima é definida por  $vetsec = 0, i = 1, \dots, n$  e a treliça convencional é definida por  $vetsec = 1, i = 1, \dots, n$ . Além de  $vetsec$  define-se:

- $v_i^{sec}$ , complexidade de estado da seção de índice  $i$ , sendo que a seção  $i$  possui  $2^{v_i^{sec}}$  estados;
- $b_i^{sec}$ , complexidade de ramo da seção  $i$  (número de bits de informação que rotulam a seção  $i$ );
- $l_i^{sec}$ , número de bits de rótulo da seção  $i$ .

Como exemplo, secciona-se a treliça do Wimax apresentada na Figura 8 utilizando  $vetsec = \{0, 1, 0\}$ , gerando a Figura 9. Observa-se que o seccionamento fez com que o número máximo de estados da treliça caísse de 32 para 16 e o número de seções caísse de 4 para 3.

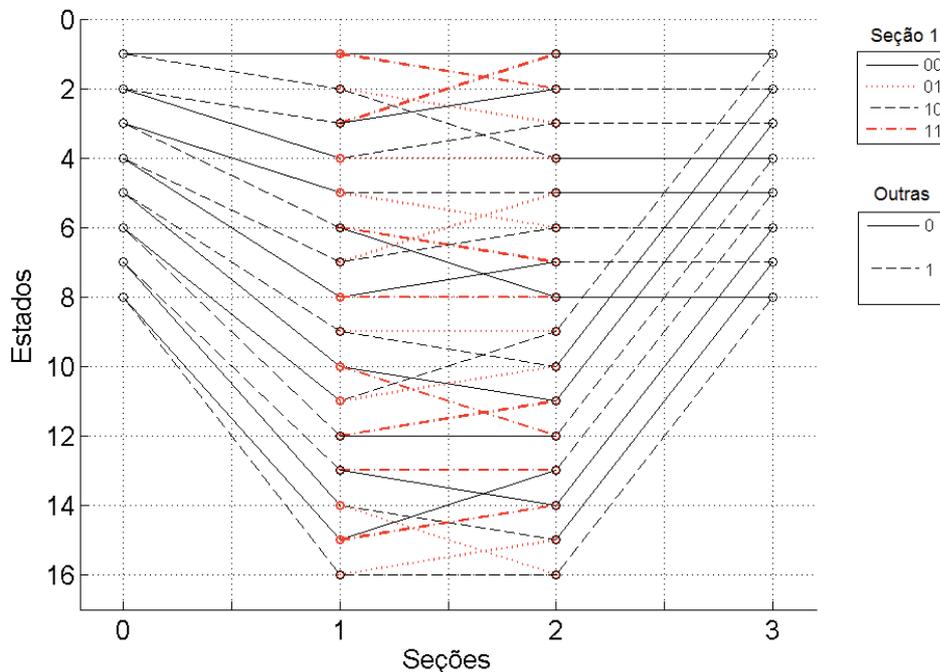


Figura 9: Treliça do Wimax seccionada com  $vetsec = \{0, 1, 0\}$

Fonte: Autoria própria

No Capítulo 4 todos os possíveis seccionamentos de alguns códigos disponíveis na literatura são analisados, a fim de observar-se a variação de complexidade e desempenho obtida.

### 2.2.5 Obtenção da treliça mínima para códigos convolucionais sistemáticos recursivos

Técnicas para representação de códigos de bloco por treliça foram apresentadas pela primeira vez em (BAHL et al., 1974). Posteriormente, McEliece e Lin expandiram a técnica para códigos convolucionais (MCELIECE; LIN, 1996). Porém, esta técnica não pode ser diretamente aplicada no caso de matrizes geradoras racionais, como em um código turbo (BERROU; GLAVIEUX; THITIMAJSHIMA, 1993). Desta maneira, esta dissertação utiliza a técnica desenvolvida em (PIMENTEL et al., 2011) para geração da treliça mínima de um código recursivo.

O procedimento inicia-se com o cálculo de uma matriz não racional (ou não recursiva) que tenha uma treliça mínima equivalente à matriz racional geradora do código. O processo gera uma treliça equivalente, porém o mapeamento entre as palavras de entrada e as palavras código é alterado, tornando a treliça resultante não sistemática. A segunda etapa do processo é forçar que o mapeamento volte a ser sistemático, simplesmente forçando que as transições de bits voltem a ser sistemáticas, adotando a convenção que a transição é gerada pelo bit idêntico ao seu rótulo.

Ressalta-se que as matrizes apresentadas nesta dissertação já se encontram na forma não racional, partindo-se da matriz racional e aplicando-se o procedimento descrito em (PIMENTEL et al., 2011).

## 2.3 O ALGORITMO MAP

Na Seção 2.1.5 descreveu-se o processo de decodificação turbo. Notou-se que o algoritmo de decodificação precisa gerar uma informação sobre a confiabilidade de decisão de valor de um bit em processo de decodificação. Desta maneira, apresenta-se o algoritmo BCJR, também conhecido como MAP e proposto por (BAHL et al., 1974). Calcula-se a LLR em (12) utilizando-se cálculos recursivos ao longo da treliça do código CSR.

### 2.3.1 Versão logarítmica do BCJR

O algoritmo de BCJR, em sua forma original, necessita a computação de um grande número de multiplicações e exponenciações, tendo, desta maneira, complexidade elevada que o torna impraticável em muitos casos. Assim, apresenta-se sua versão logarítmica que substitui as operações de exponenciação e multiplicação do algoritmo original em somas e comparações. Nesta versão, utiliza-se uma aproximação no cálculo das exponenciações e por isso o algoritmo tem desempenho sub-ótimo porém muito próximo da versão sem aproximação. Este algoritmo

é denominado max-log-MAP (VUCETIC; YUAN, 2000).

Para uma seção do código CSR de taxa  $R = k/n$ , que possui  $2^{b_{sec}}$  bits de informação são calculadas  $2^{b_{sec}}$  métricas de rótulo  $\rho_q(c_i)$ , com  $q = 0, \dots, 2^{b_{sec}} - 1$ . Uma métrica é utilizada como referência (geralmente  $\rho_0(c_i)$ ) para o cálculo de  $2^{b_{sec}} - 1$  LLRs ( $\Lambda_q(c_i)$ ), definidas como:

$$\Lambda_q(c_i) \approx \rho_0(c_i) - \rho_q(c_i), \quad (16)$$

$$\rho_q(c_i) \approx \max_{q \rightarrow l', l} \left[ \bar{\alpha}_{i-1}(l') + \bar{\gamma}_i(l', l) + \bar{\beta}_i(l) \right], \quad (17)$$

sendo  $\bar{\alpha}_i(l)$  e  $\bar{\beta}_i(l)$  definidos como:

$$\bar{\alpha}_i(l) = \max_{l, l'} \left[ \bar{\alpha}_{i-1}(l') + \bar{\gamma}_i(l', l) \right] \quad (18)$$

e

$$\bar{\beta}_i(l) = \max_{l, l'} \left[ \bar{\beta}_{i+1}(l') + \bar{\gamma}_{i+1}(l, l') \right], \quad (19)$$

para  $0 \leq l' \leq 2^{v_i^{sec}}, 0 \leq l \leq 2^{v_i^{sec}}, q = 0, \dots, 2^{b_{sec}} - 1$  e  $\bar{\gamma}_i(l', l)$  definido como

$$\bar{\gamma}_i(l', l) = \left\{ \left[ \sum_{u=1}^n r_{i,u} \cdot x_{i,u}(l, l') \right] + \Lambda_e(l, l') \right\}, \quad (20)$$

onde  $r_{i,u}$  é o  $u$ -ésimo símbolo da  $n$ -tupla recebida no instante  $i$ ,  $x_{i,u}$  é o símbolo que deveria ter sido transmitido no instante  $i$  caso ocorresse a transição entre os estados  $l'$  e  $l$ ,  $\Lambda_e(l, l')$  é a informação extrínseca (*a priori*) daquela transição. Em (20) considera-se o fato dos algoritmos que utilizam *max* não necessitarem da informação de canal para operação confiável (WANG; WANG; CHAO, 2004).

Observa-se que o cálculo de (18) e (19) é recursivo, sendo as condições de contorno derivadas do fato da decodificação sempre começar e terminar num estado conhecido, que se assume zero neste desenvolvimento:

$$\bar{\alpha}_0(l) = \begin{cases} 0 & \text{com } l = 0 \\ -\infty & \text{caso contrário;} \end{cases} \quad (21)$$

e

$$\bar{\beta}_N(l) = \begin{cases} 0 & \text{com } l = 0 \\ -\infty & \text{caso contrário.} \end{cases} \quad (22)$$

A aproximação utilizada para se chegar em (20) a partir da equação original do algoritmo BCJR

é:

$$\ln \left( \sum_u e^{\delta_u} \right) = \max_u \delta_u \quad (23)$$

a qual simplifica o algoritmo, porém faz com que seu desempenho seja ligeiramente inferior ao MAP.

Descreve-se a seguir uma interpretação dos resultados das equações do algoritmo:

- $\bar{\gamma}_i(l', l)$ : logaritmo da probabilidade de ocorrer uma transição entre o estado  $l'$  e  $l$ , no instante  $i$ ;
- $\bar{\alpha}_i(l)$ : logaritmo da probabilidade do caminho de codificação na treliça ter passado pelo estado  $l$ , no instante  $i$ , quando se calcula do início da treliça para seu fim;
- $\bar{\beta}_i(l)$ : logaritmo da probabilidade do caminho de codificação na treliça ter passado pelo estado  $l$ , no instante  $i$ , quando se calcula do final da treliça para seu início;
- $\Lambda_q(\mathbf{c}_i)$ : relação logarítmica entre as probabilidades da  $k$ -upla codificada ( $\mathbf{c}_i$ ) no instante  $i$  ser a  $t$ -upla  $q$  ou a  $t$ -upla de referência.

### 2.3.2 Correção do algoritmo de max-log-MAP

A aproximação apresentada em (23), torna o algoritmo max-log-MAP sub-ótimo. Apresenta-se uma correção que melhora o desempenho do algoritmo. Do algoritmo Jacobiano têm-se:

$$\ln \{ \exp \delta_1 + \exp \delta_2 \} = \max(\delta_1, \delta_2) + \ln(1 + e^{-|\delta_2 - \delta_1|}) = \max(\delta_1, \delta_2) + f_c(|\delta_2 - \delta_1|), \quad (24)$$

onde  $f_c(|\delta_2 - \delta_1|)$  é uma função de correção unidimensional que pode ser implementada na forma de uma tabela. Além disso, (24) pode ser aplicada recursivamente:

$$\ln \{ e^{\delta_1} + \dots + e^{\delta_n} \} = \ln(\Delta + e^{\delta_n}), \Delta = e^{\delta_1} + \dots + e^{\delta_{n-1}} = e^{\delta} \quad (25)$$

$$= \max(\ln \Delta, \delta_n) + f_c(|\ln \Delta - \delta_n|) \quad (26)$$

$$= \max(\delta, \delta_n) + f_c(|\delta - \delta_n|). \quad (27)$$

Substituindo-se as operações de *max* de (16), (18) e (19) por (27), obtém-se o algoritmo log-MAP, cujo desempenho é idêntico ao algoritmo MAP, sob a desvantagem de ter complexidade superior ao max-log-MAP devido à adição da função de correção. O aumento de complexidade é aceitável por poder ser implementado como uma simples pesquisa em tabela. A função de correção foi utilizada em um dos testes da dissertação. Sua utilização não alterou as conclusões obtidas na dissertação pois o comportamento do desempenho em função do seccionamento foi

o mesmo que nas versões sem correção. Desta maneira, optou-se por não utilizar a correção como a configuração padrão dos testes executados no Capítulo 4.

### 3 MÉTRICAS DE COMPLEXIDADE

#### 3.1 MÉTRICA DE COMPLEXIDADE PROPOSTA POR MCELIECE E LIN

Conforme estudado na Seção 2.2.2, o módulo de treliça convencional possui  $2^v$  estados iniciais e  $2^v$  estados finais. Cada estado é conectado por  $2^k$  ramos, fazendo com que o módulo apresente um total de  $2^{v+k}$  ramos. Como cada ramo é rotulado por  $n$  bits, temos que o número total de bits no módulo é  $n(2^{v+k})$  bits. Como cada módulo representa a codificação de  $k$  bits, define-se a complexidade de treliça convencional como (MCELIECE; LIN, 1996):

$$\frac{n}{k} \cdot 2^{v+k} \quad (29)$$

símbolos por bit codificado. Generalizando para as treliças seccionadas, utilizando as definições da Seção 2.2, obtêm-se a seguinte definição de complexidade de treliça  $TC(M)$ :

$$TC(M) = \frac{1}{k} \sum_{i=0}^{n'-1} l_i^{sec} \cdot 2^{v_i^{sec} + b_i^{sec}}. \quad (30)$$

Observa-se que para o algoritmo de Viterbi a métrica descrita em (30) é proporcional à complexidade de decodificação (MCELIECE; LIN, 1996). Para a decodificação turbo utiliza-se um algoritmo que maximiza a verossimilhança bit a bit, o que não ocorre no algoritmo de Viterbi, que maximiza a verossimilhança de caminho na treliça. Desta maneira, investiga-se na sequência se a métrica proposta por McEliece está relacionada com a complexidade computacional da decodificação turbo.

#### 3.2 MÉTRICA PROPOSTA PARA COMPLEXIDADE DO MAX-LOG-MAP

##### 3.2.1 Considerações iniciais

Para a análise proposta, serão consideradas apenas operações aritméticas ( somas (  $\mathcal{S}$  ), multiplicações (  $\mathcal{M}$  ), e comparações (  $\mathcal{C}$  ) ). Numa implementação real, vários outros fatores influenciam na complexidade final, como, por exemplo, acessos em memória e diferenças nos

pesos computacionais das operações envolvidas. Optou-se por não considerar outros fatores por estes serem dependentes de arquitetura e da implementação. Faz-se, assim, uma análise puramente matemática, que pode ser utilizada na comparação das diferentes abordagens de decodificação e auxiliar na análise a ser realizada após a definição de arquitetura. Conforme desenvolvido na Seção 2.2, as treliças mínima e convencional são casos particulares de treliças seccionadas. Assim, o estudo iniciará com o caso genérico e depois se demonstrará as equações para os casos particulares de treliça mínima e convencional. A complexidade das várias seções do algoritmo será denotada  $T_z$  significando a complexidade para cálculo de  $z$ , onde  $z$  deve ser substituído por uma variável do algoritmo max-log-MAP.

### 3.2.2 Treliça seccionada

Inicia-se o cálculo com a complexidade da computação de  $\bar{\gamma}_i(l', l)$  para cada um dos  $2^{v_i^{sec}}$  estados da seção  $i$ , para isso, decompõe-se (20), indicando-se as partes em (31):

$$\bar{\gamma}_i(l', l) = \left\{ \left[ \underbrace{\sum_{u=1}^n r_{i,u} \cdot x_{i,u}(l, l')}_B \right]_A + \underbrace{\Lambda_e(l, l')}_C \right\}. \quad (31)$$

O somatório indicado por  $A$  em (31) possui  $l_{sec}$  termos, cada um exigindo uma multiplicação. Desta maneira, são necessários  $l_i^{sec} \cdot \mathcal{M}$  para os operandos mais  $(l_i^{sec} - 1) \cdot \mathcal{S}$  para somá-los. Ao valor encontrado adiciona-se a informação extrínseca (representada por  $C$ ) que contribui com  $1 \cdot \mathcal{S}$ . Totaliza-se, assim,  $l_i^{sec} \cdot \mathcal{M} + l_i^{sec} \cdot \mathcal{S}$  operações por ramo. Como cada um dos  $2^{v_i^{sec}}$  estados possui  $2^{b_i^{sec}}$  ramos obtém-se:

$$T_{\gamma_i}^{sec} = \begin{cases} 2^{v_i^{sec} + b_i^{sec}} \cdot l_i^{sec} \cdot (\mathcal{M} + \mathcal{S}) & \text{se } b_i^{sec} > 0 \\ 2^{v_i^{sec} + b_i^{sec}} [l_i^{sec} \cdot \mathcal{M} + (l_i^{sec} - 1) \cdot \mathcal{S}] & \text{caso contrário.} \end{cases} \quad (32)$$

Observa-se a complexidade de algumas etapas da decodificação são dependentes da ausência/presença de bits de informação na seção, já que seções sem bits de informação (e consequentemente apresentam apenas um ramo de partida) não possuem informação extrínseca para ser somada, o que elimina a parte  $C$  de (31). Por esse motivo, faz-se (32) condicional, excluindo-se uma soma do resultado total caso não haja bit de informação na seção.

Comparando-se (32) com (30) repara-se que a métrica proposta por McEliece e Lin é proporcional ao número de multiplicações da métrica proposta. Por definição a métrica de McEliece é dividida pelo parâmetro  $k$  do código.

Procede-se para o cálculo de  $\bar{\alpha}_i(l)$  e  $\bar{\beta}_i(l)$ . A complexidade desta etapa é dependente do número de ramos que chegam ao estado em questão no caso de  $\bar{\alpha}_i(l)$ , e do número de ramos que partem do estado, no caso de  $\bar{\beta}_i(l)$ . Pode-se calcular o número de ramos que chegam num estado a partir do número total de ramos da seção anterior dividido pelo número de estados da seção corrente. Para calcular-se o número de ramos que partem inverte-se a razão.

Assim, para cada um dos  $2^{v_{(i+1)}^{sec}}$  estados de destino da seção  $i$  temos  $\Psi = \frac{2^{(v_i^{sec}+b_i^{sec})}}{2^{v_{(i+1)}^{sec}}}$  ramos de chegada. Cada cálculo de  $\bar{\alpha}_i(l)$  exige, assim,  $\Psi \cdot \mathcal{S}$  e  $(\Psi - 1) \cdot \mathcal{C}$  para ser efetuado:

$$T_{\alpha_i}^{sec} = 2^{v_{(i+1)}^{sec}} \cdot \left\{ \left( \frac{2^{(v_i^{sec}+b_i^{sec})}}{2^{v_{(i+1)}^{sec}}} \right) \cdot \mathcal{S} + \left( \frac{2^{(v_i^{sec}+b_i^{sec})}}{2^{v_{(i+1)}^{sec}}} - 1 \right) \cdot \mathcal{C} \right\} \quad (33)$$

$$= \left( 2^{(v_i^{sec}+b_i^{sec})} \right) \cdot \mathcal{S} + \left( 2^{(v_i^{sec}+b_i^{sec})} - 2^{v_{(i+1)}^{sec}} \right) \cdot \mathcal{C}. \quad (34)$$

Para o cálculo de  $\bar{\beta}_i(l)$  leva-se em consideração os estados de partida, que são  $2^{v_i^{sec}}$  para o estágio  $i$ . Cada estado possui  $2^{b_i^{sec}}$  ramos de partida, que exigem  $2^{b_i^{sec}} \cdot \mathcal{S}$  e  $(2^{b_i^{sec}} - 1) \cdot \mathcal{C}$ , para cada um dos  $2^{b_i^{sec}}$  estados, resultando em:

$$T_{\beta_i}^{sec} = 2^{v_i^{sec}} \cdot \left[ 2^{b_i^{sec}} \cdot \mathcal{S} + (2^{b_i^{sec}} - 1) \cdot \mathcal{C} \right] \quad (36)$$

$$= 2^{v_i^{sec}+b_i^{sec}} \cdot \mathcal{S} + (2^{v_i^{sec}+b_i^{sec}} - 2^{v_i^{sec}}) \cdot \mathcal{C}. \quad (37)$$

A terceira etapa do processo, o cálculo da LLR, é descrita por (16), que pode ser dividida em duas etapas, sendo a primeira, o cálculo das métricas de rótulo (17). Observa-se que o número de métricas de rótulo por seção depende do número de bits de informação codificados pela seção  $i$ . É possível demonstrar que, numa treliça seccionada, cada estado do estágio  $i$  possui  $2^{b_i^{sec}} = z$  ramos de saída, gerados por uma  $z$ -upla sistemática diferente (dentre as  $2^{b_i^{sec}}$  possíveis). Cada  $z$ -upla possuirá uma métrica de rótulo, e cada um dos  $2^{v_i^{sec}}$  estados contribuem com um operador para o  $\max_{l',l}$  em (17). Cada operador é constituído por duas somas e são definidas  $2^{b_i^{sec}}$  métricas. Desta maneira define-se a complexidade de cálculo de cada métrica de ramo como:

$$T_{\rho_q} = \left( 2^{v_i^{sec}+1} \right) \cdot \mathcal{S} + \left( 2^{v_i^{sec}} - 1 \right) \cdot \mathcal{C}. \quad (39)$$

Como estão definidas  $2^{b_i^{sec}}$  métricas, temos a complexidade total:

$$T_{\rho_q}^{total} = 2^{b_i^{sec}} \left[ \left( 2^{v_i^{sec}+1} \right) \cdot \mathcal{S} + \left( 2^{v_i^{sec}} - 1 \right) \cdot \mathcal{C} \right]. \quad (40)$$

Calculadas as métricas de ramo, passa-se para a segunda etapa do processo, que resulta nas LLRs ( $\Lambda_q(\mathbf{c}_i)$ ). Escolhe-se uma métrica de rótulo de referência da qual se subtrai cada uma das métricas restantes. Cada subtração gera a  $\Lambda_q(\mathbf{c}_i)$  do rótulo  $q$  em questão. Assim, adicionam-se

mais  $(2^{b_i^{sec}} - 1) \cdot \mathcal{S}$  à  $T_{\rho_q}^{total}$ , resultando numa complexidade de:

$$T_{\Lambda_q(\mathbf{c}_i)}^{sec} = \begin{cases} 2^{b_i^{sec}} [(2^{v_i^{sec}+1}) \cdot \mathcal{S} + (2^{v_i^{sec}} - 1) \cdot \mathcal{C}] + (2^{b_i^{sec}} - 1) \cdot \mathcal{S} & \text{se } b_i^{sec} > 0 \\ 0 & \text{caso contrário.} \end{cases} \quad (41)$$

Novamente, o cálculo está condicionado à existência de bits de informação na seção calculada, sendo desnecessário calcular  $\Lambda^{sec}(\mathbf{c}_i)$  quando nenhum bit de informação rotula a seção. Finalmente, define-se a complexidade total do processo de decodificação da treliça seccionada como:

$$T_{MAP}^{sec} = \sum_{i=0}^{n'-1} T_{\gamma_i}^{sec} + T_{\alpha_i}^{sec} + T_{\beta_i}^{sec} + T_{\Lambda_i^q(\mathbf{c}_i)}^{sec}, \quad (42)$$

sendo  $n'$  o número de seções de cada módulo.

### 3.2.3 Treliça convencional

Define-se os parâmetros da seção única da treliça convencional da seguinte maneira:

- $v_i^{sec} = v$ ;
- $b_i^{sec} = k$ ;
- $l_i^{sec} = n$

Como resultado temos a seguinte complexidade:

$$T_{\gamma_i}^{conv} = 2^{k+v} \cdot (n \cdot \mathcal{M} + n \cdot \mathcal{S}), \quad (43)$$

$$T_{\alpha_i}^{conv} = T_{\beta_i}^{conv} = 2^v [2^k \cdot \mathcal{S} + (2^k - 1) \cdot \mathcal{C}], \quad (44)$$

$$T_{\Lambda_q(\mathbf{c}_i)}^{conv} = 2^k [(2^{v+1}) \cdot \mathcal{S} + (2^v - 1) \cdot \mathcal{C}] + (2^k - 1) \cdot \mathcal{S} \quad (45)$$

e

$$T_{MAP}^{conv} = T_{\gamma_i}^{conv} + T_{\alpha_i}^{conv} + T_{\beta_i}^{conv} + T_{\Lambda_q(\mathbf{c}_i)}^{conv}. \quad (46)$$

### 3.2.4 Treliça mínima

Para a treliça mínima, tem-se os seguintes parâmetros:

- $v_i^{sec} = \tilde{v}_i$ ;
- $b_i^{sec} = \tilde{b}_i, 0 \leq \tilde{b}_i \leq 1$ ;
- $l_i^{sec} = 1$ .

O resultado é a seguinte métrica de complexidade:

$$T_{\gamma_i}^{min} = 2^{\tilde{b}_i + \tilde{v}_i} \left( \mathcal{M} + \tilde{b}_i \cdot \mathcal{S} \right), \quad (47)$$

$$T_{\alpha_i}^{min} = \left( 2^{(\tilde{v}_i + \tilde{b}_i)} \right) \cdot \mathcal{S} + \left( 2^{(\tilde{v}_i + \tilde{b}_i)} - 2^{\tilde{v}_{(t+1)}} \right) \cdot \mathcal{C}, \quad (48)$$

$$T_{\beta_i}^{min} = 2^{\tilde{v}_i + \tilde{b}_i} \cdot \mathcal{S} + \left( 2^{\tilde{v}_i + \tilde{b}_i} - 2^{\tilde{v}_i} \right) \cdot \mathcal{C}, \quad (49)$$

$$T_{\Lambda_q(\mathbf{c}_i)}^{min} = \tilde{b}_i \cdot \left[ \left( 2^{\tilde{v}_i + 2} + 1 \right) \cdot \mathcal{S} + \left( 2^{\tilde{v}_i + 1} - 2 \right) \cdot \mathcal{C} \right], \quad (50)$$

e

$$T_{\text{MAP}}^{min} = \sum_{i=0}^{n-1} T_{\gamma_i}^{min} + T_{\alpha_i}^{min} + T_{\beta_i}^{min} + T_{\Lambda_q(\mathbf{c}_i)}^{min}. \quad (51)$$

### 3.3 EXEMPLO - $C_1(4, 2, 3)$ DO WIMAX (IEEE STD 802.16E, 2009)

Apresenta-se o código do Wimax, e estuda-se a complexidade dos seus possíveis seccionamentos.

#### 3.3.1 Definição do código

A matriz geradora do código do Wimax, na forma não recursiva e cujo  $d_2 = 7$ , é:

$$G_1 = \begin{bmatrix} 5 & 2 & 5 & 5 \\ 2 & 3 & 3 & 1 \end{bmatrix}. \quad (52)$$

e gera a treliça mínima está apresentada na Figura 8.

### 3.3.2 Complexidade do código

A Tabela 1 apresenta a complexidade de todos os possíveis seccionamentos da treliça, segundo a métrica desenvolvida. Grifa-se o código de menor complexidade, notação que será adotada em todos os estudos de complexidade subsequentes. Ressalta-se que o seccionamento com  $vetsec = \{0,0,0\}$  representa a treliça mínima enquanto  $vetsec = \{1,1,1\}$  representa a treliça convencional.

**Tabela 1: Complexidade do código  $C_1(4,2,3)$  do WiMax (IEEE STD 802.16E, 2009)**

Sec.	$\bar{\gamma}_i(l',l)$		$\bar{\alpha}_i(l)$		$\bar{\beta}_i(l)$		$\Lambda_q(\mathbf{c}_i)$		Total			$TC(M)$
	$\mathcal{S}$	$\mathcal{M}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{M}$	
0,0,0	48	96	96	24	96	24	98	44	338	92	96	48
0,0,1	80	112	80	24	80	24	98	44	338	92	112	56
0,1,0	80	96	64	24	64	24	98	44	306	92	96	48
0,1,1	112	112	48	24	48	24	98	44	306	92	112	56
1,0,0	64	112	80	24	80	24	67	28	291	76	112	56
1,0,1	96	128	64	24	64	24	67	28	291	76	128	64
1,1,0	<b>96</b>	<b>112</b>	<b>48</b>	<b>24</b>	<b>48</b>	<b>24</b>	<b>67</b>	<b>28</b>	<b>259</b>	<b>76</b>	<b>112</b>	<b>56</b>
1,1,1	128	128	32	24	32	24	67	28	259	76	128	64

### 3.3.3 Comentários sobre o código

Na Tabela 1, observa-se que a minimização da treliça do Wimax reduz o número de multiplicações necessárias para a aplicação do algoritmo max-log-MAP, porém, no processo, o número adicional de somas e comparações acaba por reverter o ganho obtido, tornando a complexidade da treliça mínima maior do que a treliça convencional se for considerado que o custo computacional de todas as operações são iguais. Desta maneira, torna-se vantajoso o uso de um seccionamento intermediário, que reduz o número de multiplicações necessárias para o cálculo sem afetar o número de somas e comparações. Este seccionamento é o que possui  $vetsec = (1,1,0)$ .

## 3.4 CONSIDERAÇÕES SOBRE A MÉTRICA DE COMPLEXIDADE

Analisando a métrica exposta na Seção 3.1, a treliça convencional do Wimax possui  $TC(M) = 64$ , que é reduzida para  $TC(M) = 48$  na treliça mínima. Porém, na treliça de menor complexidade computacional ( $vetsec = \{1,1,0\}$ ) temos  $TC(M) = 56$ . Conclui-se que a treliça mínima, sob a definição de (MCELIECE; LIN, 1996) não é a que apresenta menor complexidade computacional para decodificação do max-log-MAP, fazendo com que a métrica exposta na Seção 3.1

não seja indicada para avaliação de complexidade de códigos turbo. Este comportamento pode ser explicado pelo modo de operação do algoritmo, que é dependente do número de estados da treliça. No algoritmo de Viterbi, o número de estados afeta a complexidade de maneira diferente do max-log-MAP. Observa-se que a métrica de McEliece representa apenas uma pequena parcela da complexidade total de decodificação para o max-log-MAP, já que é proporcional ao número de multiplicações realizadas pelo algoritmo, não levando em consideração o número de somas e comparações. Existem treliças seccionadas que possuem complexidade superior à da treliça convencional, pois o processo de minimização e seccionamento adiciona seções que apresentam muitos estados. Com a ajuda de tabelas como a apresentada neste capítulo, estas treliças podem ser identificadas como auxílio do processo de escolha da melhor treliça. No Capítulo 4, adiciona-se mais uma variável na escolha do seccionamento a ser adotado: o desempenho do código.

## 4 AVALIAÇÃO DE DESEMPENHO DE CÓDIGOS TURBO EM FUNÇÃO DO SECCIONAMENTO

### 4.1 INTRODUÇÃO

Investigam-se as diferenças de desempenho do algoritmo max-Log-MAP quando se altera o tipo de treliça empregada. Os parâmetros dos códigos foram variados a fim de verificar se há alguma influência de acordo com o seccionamento.

### 4.2 PARÂMETROS GERAIS DOS TESTES

Caso não sejam mencionados diferentemente na seção específica, todos os testes utilizam os parâmetros abaixo:

- Simula-se até que  $1.0 \times 10^7$  bits sejam transmitidos ou 800 erros ocorram;
- A mesma semente aleatória é utilizada para todos os testes, e reinicializada para cada ponto de SNR;
- O decodificador utiliza 10 iterações;
- O ponto inicial de SNR é  $0dB$ , simula-se até se obtenha BER de no mínimo de  $1.0 \times 10^{-5}$ ;
- Utiliza-se modulação BPSK;
- O canal utilizado é AWGN;
- Utiliza-se o entrelaçador de permutação descrito em (DOUILLARD; BERROU, 2005);
- Decodifica-se utilizando o algoritmo max-log-MAP.

### 4.3 CARACTERÍSTICAS DAS TRELIÇAS UTILIZADAS PARA TESTES

Com o objetivo de verificar a influência do seccionamento no desempenho dos códigos, simulou-se todas os possíveis seccionamentos para os códigos selecionados. Sete testes foram

executados, variando-se:

- A complexidade da treliça (avaliada pelo  $d_2$  do código);
- Tamanho do entrelaçador;
- Utilização do fator de correção demonstrado na Seção 2.3.2;
- Taxa do código utilizado.

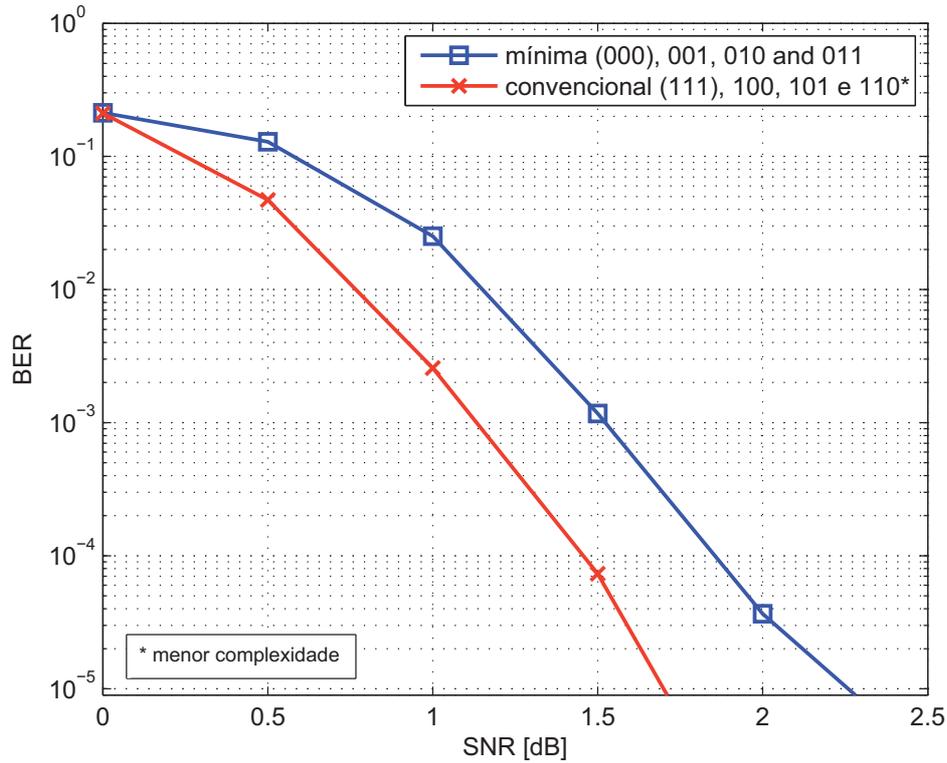
#### 4.4 TESTE 1 - TRELIÇA DO WIMAX

Simulou-se todos os possíveis seccionamentos da treliça do Wimax, que é uma treliça gerada por um código  $C_1(4,2,3)$ , seu módulo convencional apresenta oito estados e seu  $d_2 = 7$ . O código está descrito em detalhes na Seção 3.3. Obtiveram-se os resultados apresentados na Figura 10. Utilizou-se um entrelaçador de 480 bits.

Na legenda da Figura 10 e em todos os gráficos de desempenho subsequentes descrevem-se os seccionamentos a partir de seu vetor de seccionamento (*vetsec*). Como exemplo, a curva apresentada na figura como '010' é a curva de desempenho da treliça cujo  $vetsec = \{0, 1, 0\}$ . Observa-se que as oito curvas de desempenho formaram dois grupos. O desempenho das treliças onde uma única LLR é calculada do agrupamento dos bits sistemáticos é superior aos das treliças que calculam a LLR individualmente para os bits. Esta diferença é de aproximadamente 1dB para  $BER = 1 \times 10^{-5}$ . A curva indica que a diferença tende a crescer com a SNR. Como exemplo, observamos os seccionamentos  $vetsec = \{1, 1, 1\}$  (convencional) e  $vetsec = \{0, 1, 1\}$ . Os dois diferem apenas no agrupamento ou não das duas primeiras seções (que carregam os bits de informação), mas este fato os coloca em grupos de desempenho diferentes. Já quando o seccionamento  $vetsec = \{0, 1, 1\}$  é comparado com o seccionamento da treliça mínima  $vetsec = \{0, 0, 0\}$  o desempenho é igual, mesmo se alterando completamente a maneira de se agrupar os bits de paridade. Os motivos deste comportamento serão discutidos na Seção 4.11.

##### 4.4.1 Análise de complexidade e desempenho

Na Tabela 1 (Seção 3.3) observa-se que o código de menor complexidade para o Wimax é o que possui  $vetsec = \{1, 1, 0\}$ . A Figura 10 indica que o desempenho deste seccionamento é o melhor dentre os possíveis. Assim, conclui-se que para os parâmetros indicados no Teste da Seção 4.4, o seccionamento  $vetsec = \{1, 1, 0\}$  resulta em menor complexidade de decodificação sem alteração de desempenho.



**Figura 10: Desempenho do código  $C_1(4,2,3)$ ,  $d_2 = 7$  do WiMax. Bloco de 480 bits, sem correção.**

**Fonte: Autoria própria**

#### 4.5 TESTE 2 - REDUÇÃO DA COMPLEXIDADE DA TRELIÇA

Escolheu-se um código  $C_2(4,2,3)$  com  $d_2 = 6$ , inferior ao do Wimax ( $d_2 = 7$ ) para a realização do teste subsequente. Sua matriz geradora na forma não recursiva é:

$$G_2 = \begin{bmatrix} 2 & 2 & 2 & 3 \\ 6 & 5 & 3 & 2 \end{bmatrix}, \quad (53)$$

o que gera a treliça mínima da Figura 11.

O estudo de complexidade mostra que a redução de  $d_2$  também reduz a complexidade do código. A única variação de parâmetro frente ao teste da Seção 4.4 é a treliça utilizada. Obtém-se a curva de desempenho da Figura 12. Observa-se que a tendência do teste da Seção 4.4 se manteve. O gráfico apresenta dois grupos de curvas, diferenciados pelo agrupamento ou não de bits sistemáticos. Na região de *error floor*, o desempenho dos dois grupos se equiparam.

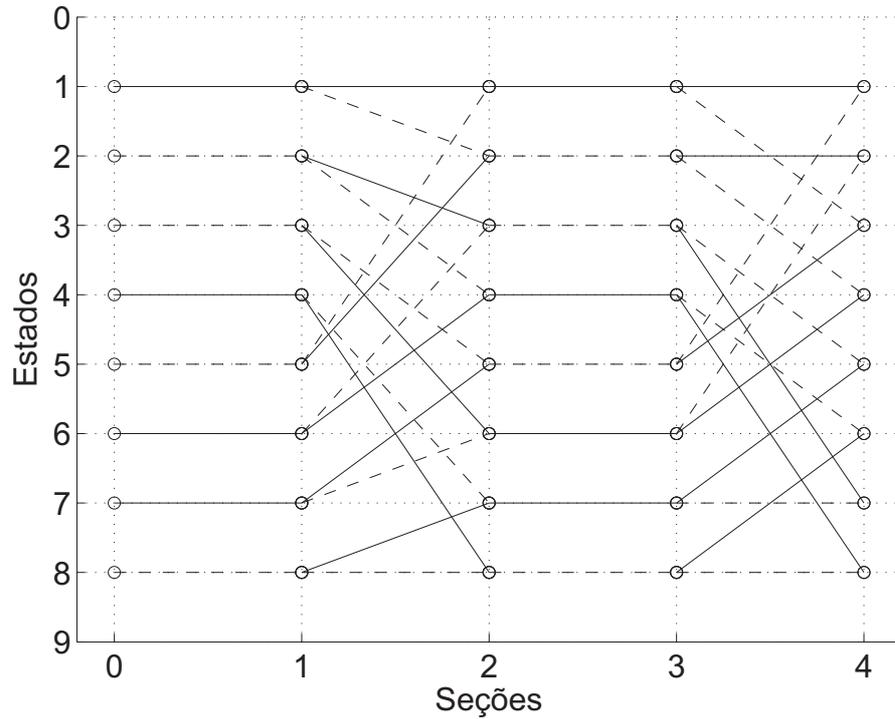


Figura 11: Treliça de  $C_2(4,2,3)$ ,  $d_2 = 6$

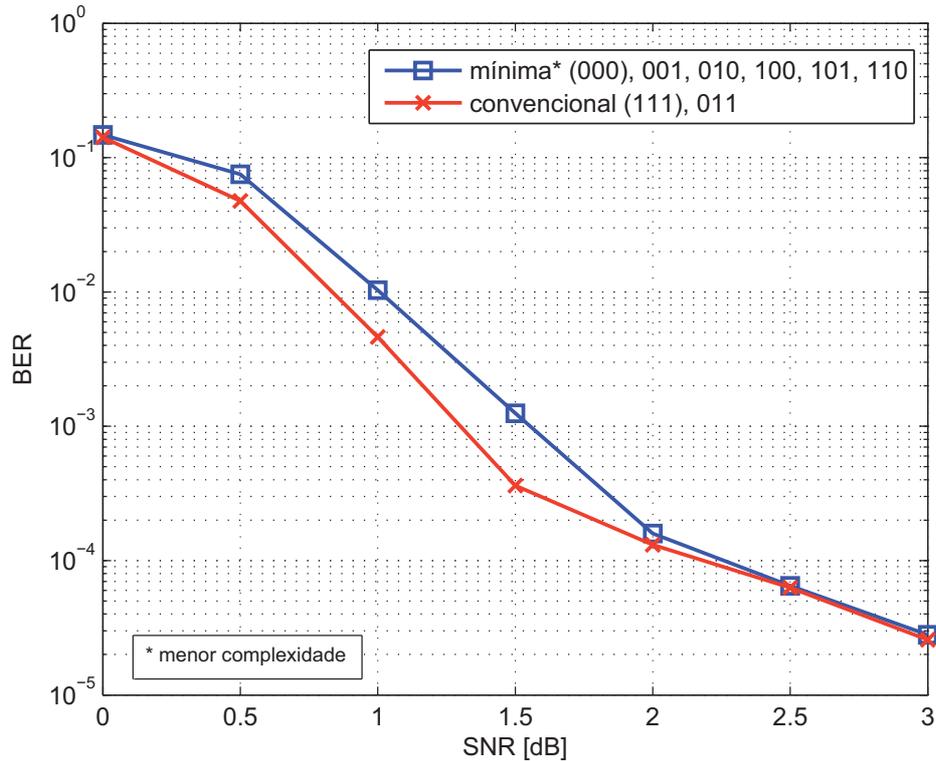
#### 4.5.1 Análise de complexidade e desempenho

Estudou-se a complexidade do código, obtendo-se a Tabela 2.

Tabela 2: Complexidade do código  $C_2(4,2,3)$  com  $d_2 = 6$  (BENCHIMOL, 2011)

Sec.	$\bar{\gamma}_i(l', l)$		$\bar{\alpha}_i(l)$		$\bar{\beta}_i(l)$		$\Lambda_q(\mathbf{c}_i)$		Total			$TC(M)$
	$\mathcal{S}$	$\mathcal{M}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{M}$	
<b>0,0,0</b>	<b>32</b>	<b>48</b>	<b>48</b>	<b>16</b>	<b>48</b>	<b>16</b>	<b>66</b>	<b>28</b>	<b>194</b>	<b>60</b>	<b>48</b>	<b>24</b>
0,0,1	48	56	40	16	40	16	66	28	194	60	56	28
0,1,0	48	56	40	16	40	16	66	28	194	60	56	28
0,1,1	96	104	40	24	40	24	67	28	243	76	104	52
1,0,0	48	56	40	16	40	16	66	28	194	60	56	28
1,0,1	64	64	32	16	32	16	66	28	194	60	64	32
1,1,0	64	64	32	16	32	16	66	28	194	60	64	32
1,1,1	128	128	32	24	32	24	67	28	259	76	128	64

Conforme Tabela 2, o código do teste da Seção 4.5 tem como seccionamento de menor complexidade a treliça mínima, o que pode ser explicado pelo fato de todas as seções da treliça do código terem 8 estados. Este comportamento faz com que o número de somas e comparações não suba significativamente. Além disso, comprova-se que o código apresenta decodificação menos complexa que a do Wimax. Na Figura 12, observa-se que o desempenho da treliça mínima não é o melhor entre os seccionamentos. Desta maneira, a escolha do código deverá ser



**Figura 12: Desempenho do código  $C_2(4,2,3)$ ,  $d_2 = 6$ , bloco de 480 bits, sem correção.**

**Fonte: Autoria própria**

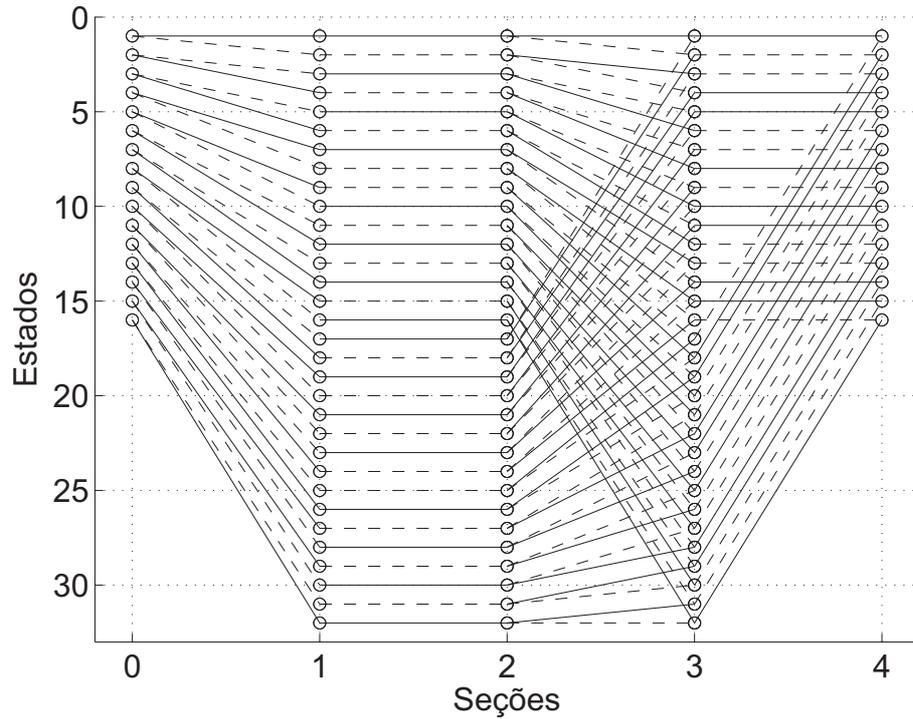
realizada de acordo com critérios de uso. Caso seja melhor sacrificar desempenho em favor de complexidade, utiliza-se  $vetsec = \{0,0,0\}$ . Caso deseje-se desempenho máximo, utiliza-se  $vetsec = \{0,1,1\}$ , sacrificando-se a redução de complexidade. Novamente o seccionamento de treliça mostra-se útil na redução de complexidade, mas a redução máxima de complexidade vem ao custo de perda de desempenho. Assim, pode ser interessante o uso de um seccionamento intermediário, que apresenta redução menor de complexidade, mas não apresenta perda de desempenho.

#### 4.6 TESTE 3 - AUMENTO DE COMPLEXIDADE DA TRELIÇA

Utilizando a mesma metodologia do teste da Seção 4.5, procedeu-se na simulação de desempenho de um código  $C_3(4,2,4)$  com  $d_2 = 13$ , cuja matriz geradora na forma não recursiva é

$$G_3 = \begin{bmatrix} 7 & 1 & 6 & 3 \\ 2 & 4 & 7 & 5 \end{bmatrix} \quad (54)$$

e que gera a treliça mínima da Figura 13. O desempenho do código pode ser observado na



**Figura 13: Treliça experimental mínima 2,  $d_2 = 13$**

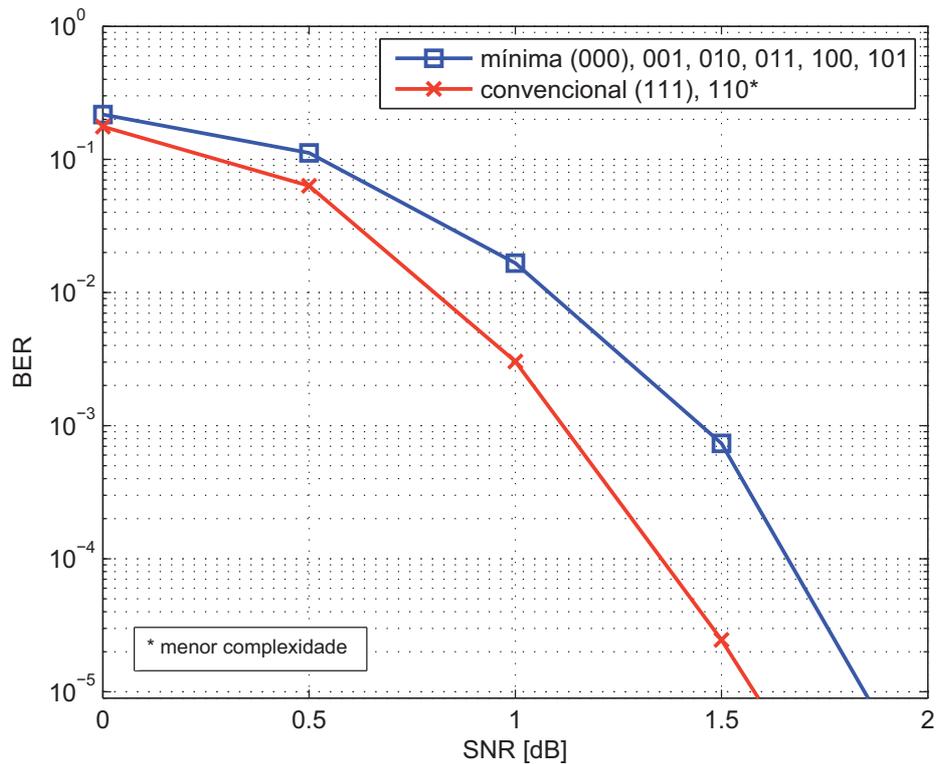
Figura 14.

Novamente a tendência dos testes anteriores se manteve, concluindo-se que a variação de complexidade não altera o fato das treliças que agrupam bits sistemáticos possuírem desempenho superior às que não agrupam. Observa-se, também, que nesta treliça os bits sistemáticos estão em seções diferentes que na treliça do teste da Seção 4.5. Assim, os seccionamentos que compõem cada grupo de desempenho foram alterados.

#### 4.6.1 Análise de complexidade e desempenho

Estudou-se a complexidade do algoritmo max-log-MAP quando aplicado à treliça, obtendo-se os resultados apresentados na Tabela 3.

Para o código utilizado no teste da Seção 4.6, o código de menor complexidade é também o de melhor desempenho ( $vetsec = \{1, 1, 0\}$ ), mostrando-se o seccionamento eficaz na redução de complexidade sem perda de desempenho.



**Figura 14: Desempenho do código  $C_3(4,2,4)$ ,  $d_2 = 13$ , bloco de 480 bits, sem correção.**

**Fonte: Autoria própria**

#### 4.7 TESTE 4 - REDUÇÃO DO TAMANHO DO ENTRELAÇADOR

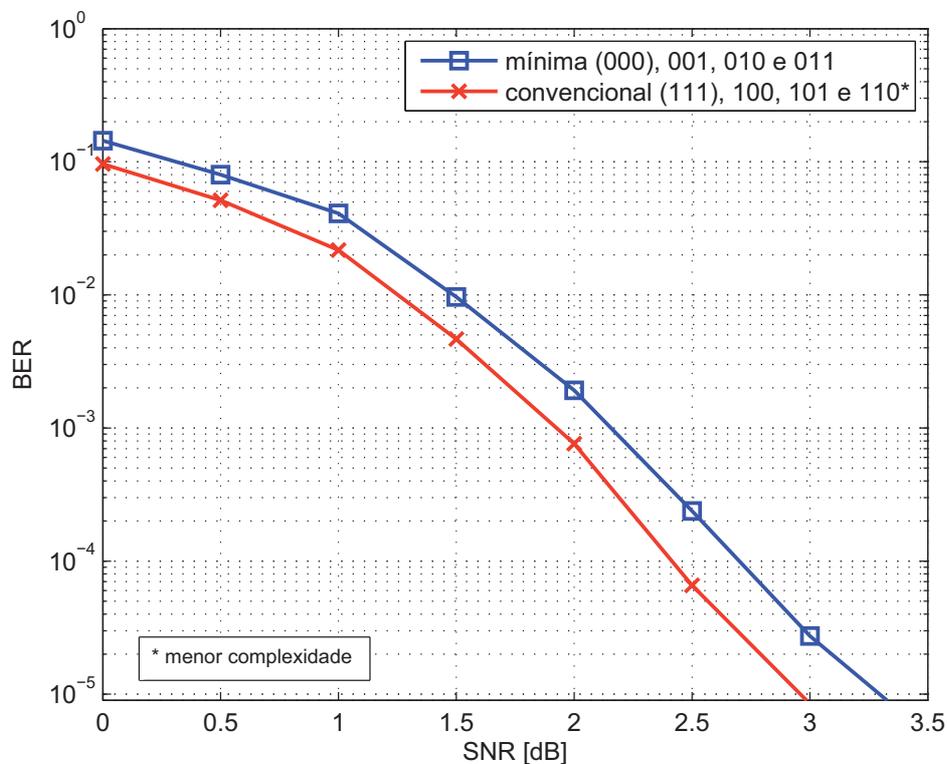
Novamente utilizando-se o código  $C_1(4,2,3)$  do Wimax, reduziu-se o tamanho do entrelaçador para 96, mantendo-se os outros parâmetros iguais aos do teste da Seção 4.4, para verificar a tendência observada nos testes anteriores. O desempenho obtido está apresentado na Figura 15. Apesar de se observar uma diferença menor de desempenho entre os grupos, a tendência é idêntica, concluindo-se que o tamanho do entrelaçador também não afeta o comportamento observado.

#### 4.8 TESTE 5 - UTILIZAÇÃO DA CORREÇÃO

Nesta rodada, repetiu-se o teste da Seção 4.4, apenas adicionando-se a correção proposta na Seção 2.3.2 à simulação. Obteve-se a Figura 16. O aumento de desempenho do código não alterou a tendência observada.

**Tabela 3: Complexidade do código  $C_3(4, 2, 4)$  com  $d_2 = 13$  (BENCHIMOL, 2011)**

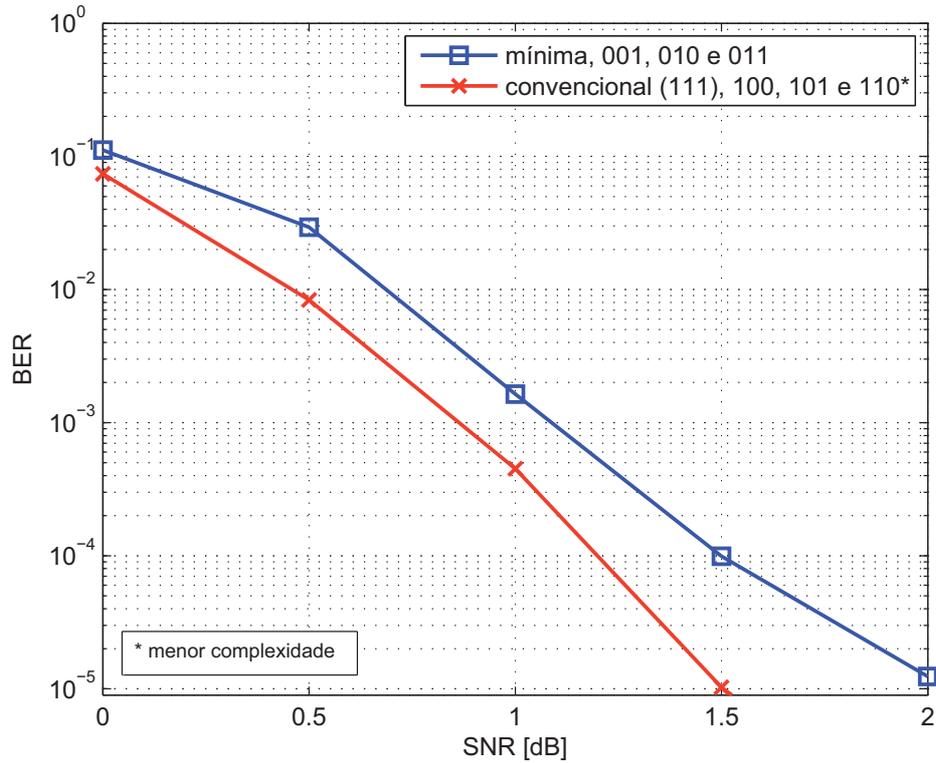
Sec.	$\bar{\gamma}_i(l', l)$		$\bar{\alpha}_i(l)$		$\bar{\beta}_i(l)$		$\Lambda_q(\mathbf{c}_i)$		Total			$TC(M)$
	$\mathcal{S}$	$\mathcal{M}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{M}$	
0,0,0	96	160	160	48	160	48	194	92	610	188	160	80
0,0,1	160	192	128	48	128	48	194	92	610	188	192	96
0,1,0	160	192	128	48	128	48	194	92	610	188	192	96
0,1,1	224	224	96	48	96	48	194	92	610	188	224	112
1,0,0	128	160	128	48	128	48	194	92	578	188	160	80
1,0,1	192	192	96	48	96	48	194	92	578	188	192	96
<b>1,1,0</b>	<b>192</b>	<b>224</b>	<b>96</b>	<b>48</b>	<b>96</b>	<b>48</b>	<b>131</b>	<b>60</b>	<b>515</b>	<b>156</b>	<b>224</b>	<b>112</b>
1,1,1	256	256	64	48	64	48	131	60	515	156	256	128

**Figura 15: Desempenho do código  $C_1(4, 2, 3)$ ,  $d_2 = 7$  do WiMax, bloco de 96 bits, sem correção.**

Fonte: Autoria própria

#### 4.9 TESTE 6 - AUMENTO DA TAXA DO CÓDIGO

Os códigos utilizados nos testes de 1 a 5 possuem  $k = 2$ , e, portanto, há apenas duas alternativas de agrupamento dos bits. Por este motivo, repete-se o teste de agrupamento de bits utilizando um código de taxa  $R = 3/4$ , que possui quatro maneiras diferentes de se agrupar os



**Figura 16: Desempenho do código  $C_1(4,2,3)$ ,  $d_2 = 7$  do WiMax, bloco de 480 bits, com correção.**

**Fonte: Autoria própria**

bits sistemáticos. O código escolhido possui a seguinte matriz geradora:

$$G_4 = \begin{bmatrix} 2 & 2 & 2 & 3 \\ 4 & 1 & 1 & 2 \\ 6 & 6 & 1 & 0 \end{bmatrix}, \quad (55)$$

que gera a treliça mínima apresentada na Figura 17.

O comprimento do entrelaçador utilizado é de 720 bits. O resultado obtido está descrito na Figura 18. Na figura, observa-se que as oito curvas formam quatro grupos de desempenho, sendo estes agrupamentos função do vetor de seccionamento (*vetsec*). Quanto mais agrupados estiverem os bits sistemáticos melhor é o desempenho do código. Além disso, o agrupamento de bits não sistemáticos não influencia no desempenho do código. Desta maneira, observou-se coerência entre o comportamento do código duobinário testado na Seção 4.7 e sugerindo então que o comportamento é uma característica do seccionamento de códigos.

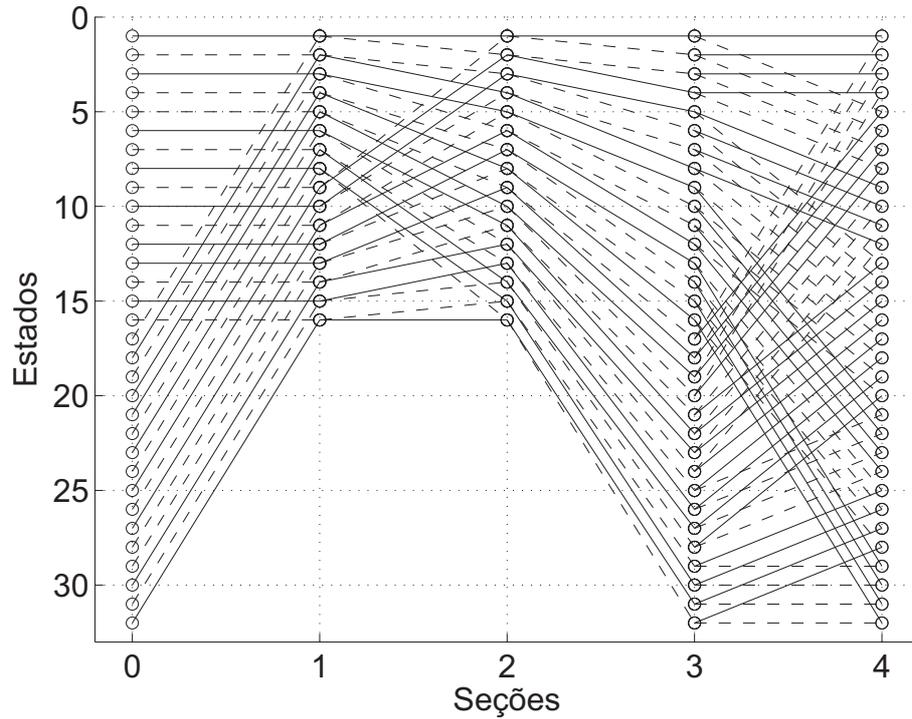


Figura 17: Treliça mínima de  $C_3(4,2,4)$

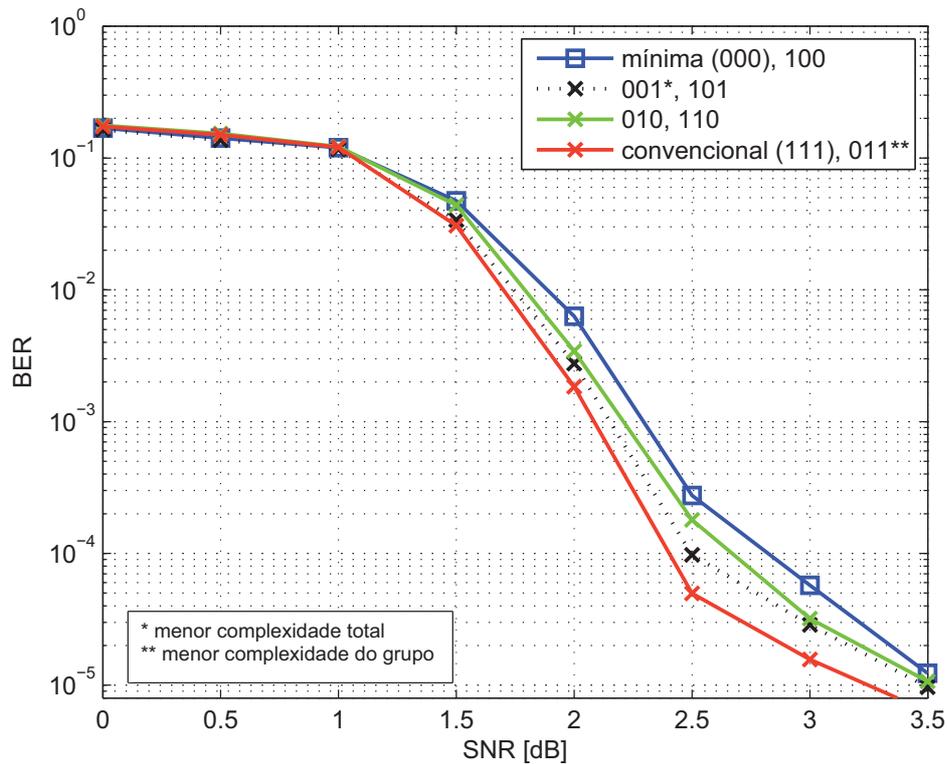
#### 4.9.1 Análise de complexidade e desempenho

Estudou-se a complexidade do código, obtendo-se a Tabela 4. Para o código utilizado na

Tabela 4: Complexidade do código  $C_4(4,3,5)$  de (BENCHIMOL, 2011).

Sec.	$\bar{\gamma}_i(l', l)$		$\bar{\alpha}_i(l)$		$\bar{\beta}_i(l)$		$\Lambda_q(\mathbf{c}_i)$		Total			$TC(M)$
	$\mathcal{S}$	$\mathcal{M}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{M}$	
0,0,0	128	160	160	64	160	64	259	122	707	250	160	53.3
<b>0,0,1</b>	<b>160</b>	<b>192</b>	<b>128</b>	<b>64</b>	<b>128</b>	<b>64</b>	<b>196</b>	<b>90</b>	<b>612</b>	<b>218</b>	<b>192</b>	<b>64</b>
0,1,0	192	224	160	80	160	80	260	122	772	282	224	74.7
0,1,1	384	416	160	112	160	112	263	120	967	344	416	138.7
1,0,0	224	224	160	80	160	80	323	154	867	314	224	74.7
1,0,1	256	256	128	80	128	80	260	122	772	282	256	85.3
1,1,0	448	448	192	128	192	128	388	186	1220	442	448	149.3
1,1,1	1024	1024	256	224	256	224	519	248	2055	696	1024	341.3

Seção 4.9, o seccionamento de menor complexidade ( $vetsec = \{0,0,1\}$ ) é o que apresenta segundo melhor desempenho. Novamente uma escolha entre maior desempenho ou maior redução de complexidade deve ser efetuada. Mesmo assim, o estudo aponta que os dois seccionamentos que constituem o grupo de melhor desempenho ( $vetsec = \{0,1,1\}$  e  $vetsec = \{1,1,1\}$ ) possuem métricas de complexidade que diferem em mais de 50%, mostrando que o seccionamento é muito vantajoso mesmo no caso em que a redução de complexidade não é máxima.



**Figura 18: Desempenho do código  $C_4(4,3,4)$ ,  $d_2 = 6$ , bloco de 720 bits, sem correção.**

**Fonte: Autoria própria**

#### 4.10 TESTE 7 - SEGUNDO EXEMPLO DE AUMENTO DA TAXA DO CÓDIGO

A treliça mínima do teste de aumento de taxa apresentado na Seção 4.9 apresenta apenas uma seção que não tem bit de informação. Realiza-se nesta seção um teste com um código de taxa  $R = 3/5$  e  $d_2 = 10$ . Este código apresenta 16 possíveis seccionamentos, tornando a análise do comportamento mais abrangente. A matriz geradora do código, na forma não recursiva é:

$$G_5 = \begin{bmatrix} 6 & 6 & 4 & 0 & 1 \\ 2 & 7 & 1 & 2 & 0 \\ 2 & 2 & 0 & 3 & 2 \end{bmatrix}, \quad (56)$$

e gera a treliça mínima da Figura 19.

O desempenho obtido pode ser observado na Figura 20 onde conclui-se que o comportamento é semelhante ao teste da Seção 4.9: 4 grupos de desempenho foram encontrados.

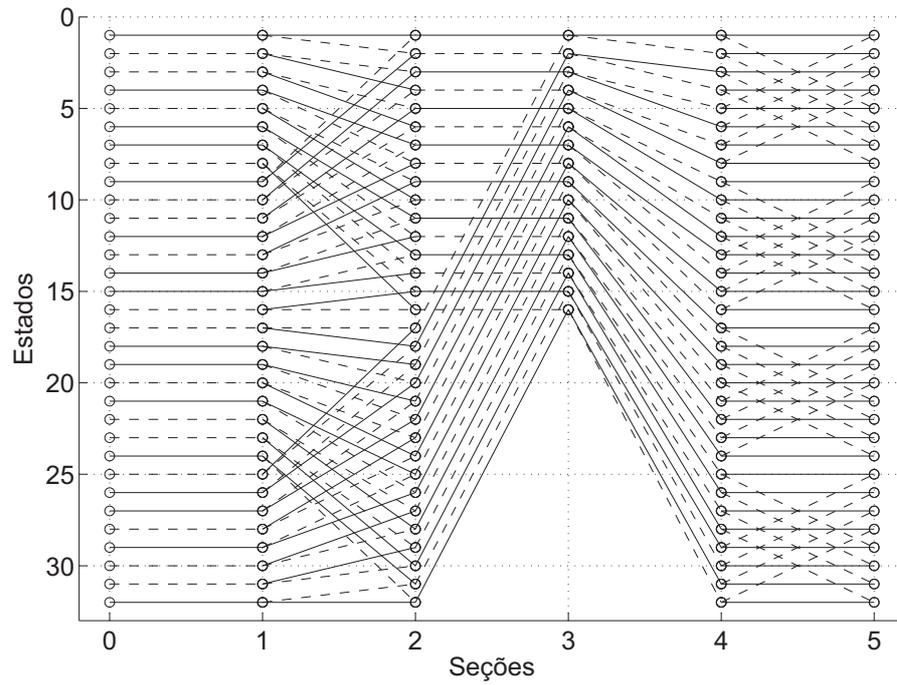


Figura 19: Treliça mínima de  $C_5(5,3,5)$

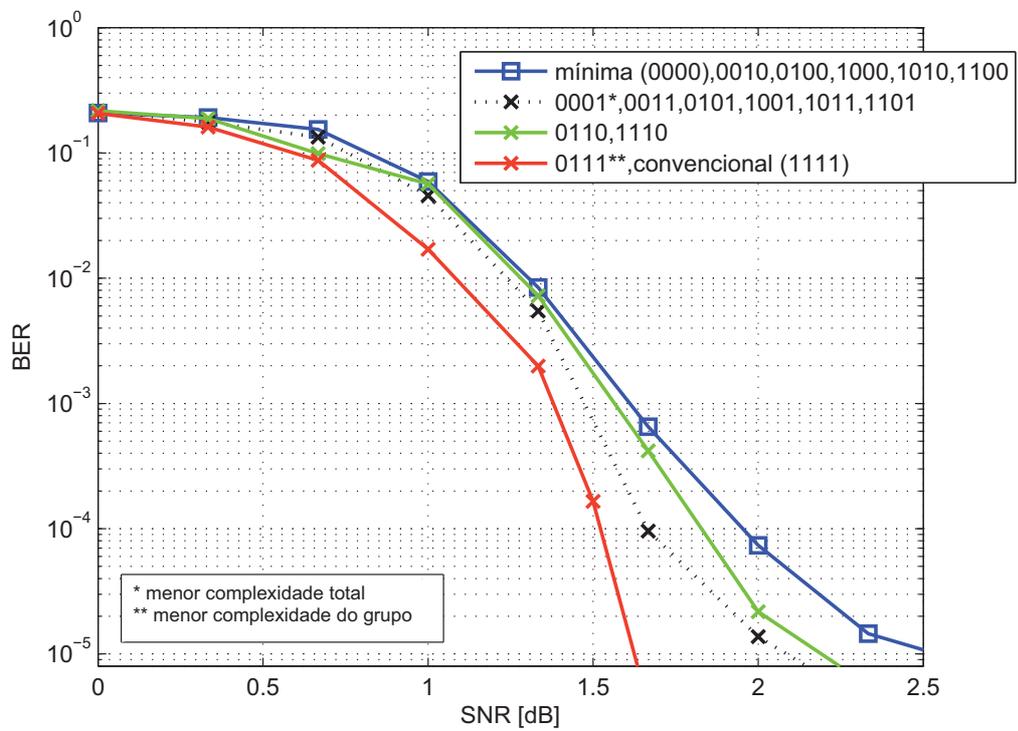


Figura 20: Desempenho do código  $C_5(5,3,5)$ ,  $d_2 = 10$ , bloco de 720 bits sem correção.

Fonte: Autoria própria

## 4.10.1 Análise de complexidade e desempenho

Estudou-se a complexidade do código, obtendo-se a Tabela 5. Observa-se que existe um

**Tabela 5: Complexidade do código  $C_5(5, 3, 5)$  de (BENCHIMOL, 2011).**

Sec.	$\bar{\gamma}_i(l', l)$		$\bar{\alpha}_i(l)$		$\bar{\beta}_i(l)$		$\Lambda_q(\mathbf{c}_i)$		Total			TC(M)
	$\mathcal{S}$	$\mathcal{M}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}$	$\mathcal{M}$	
0,0,0,0	160	224	224	111	224	80	323	154	931	345	224	74.7
<b>0,0,0,1</b>	<b>192</b>	<b>256</b>	<b>192</b>	<b>80</b>	<b>192</b>	<b>80</b>	<b>260</b>	<b>122</b>	<b>836</b>	<b>282</b>	<b>256</b>	<b>85.3</b>
0,0,1,0	256	288	224	96	224	96	387	186	1091	378	288	96
0,0,1,1	448	480	224	128	224	128	388	186	1284	442	480	160
0,1,0,0	224	256	192	80	192	80	323	154	931	314	256	85.3
0,1,0,1	256	288	160	80	160	80	260	122	836	282	288	96
0,1,1,0	448	480	224	128	224	128	388	186	1284	442	480	160
0,1,1,1	1024	1056	288	224	288	224	519	248	2119	696	1056	352
1,0,0,0	224	256	192	80	192	80	323	154	931	314	256	85.3
1,0,0,1	256	288	160	80	160	80	260	122	836	282	288	96
1,0,1,0	320	320	192	96	192	96	387	186	1091	378	320	106.7
1,0,1,1	512	512	192	128	192	128	388	186	1284	442	512	170.7
1,1,0,0	288	288	160	80	160	80	323	154	931	314	288	96
1,1,0,1	320	320	128	80	128	80	260	122	836	282	320	106.7
1,1,1,0	576	576	192	128	192	128	388	186	1348	442	576	192
1,1,1,1	1280	1280	256	224	256	224	519	248	2311	696	1280	426.7

seccionamento cuja complexidade de decodificação é 68% inferior à complexidade de decodificação da treliça convencional ( $vetsec = \{0, 1, 1\}$ ), porém a redução de complexidade resulta num sacrifício de desempenho. Mesmo que não seja desejável trocar desempenho por redução de complexidade, há seccionamentos no grupo de desempenho máximo que apresentam redução de complexidade na ordem de 10%. O fato indica que o seccionamento de treliça é vantajoso e reforça as conclusões obtidas nos testes anteriores.

## 4.11 CONCLUSÕES

Através de simulação observa-se que os desempenhos dos seccionamentos podem ser divididos em grupos que são função das diferentes maneiras de se agrupar os bits sistemáticos nas seções da treliça. Nas treliças com mais de um bit sistemático por seção, uma LLR da combinação dos bits sistemáticos é calculada, diferentemente das treliças simples, onde a decisão de bits é individual, uma vez que o algoritmo de decodificação utilizado otimiza a verossimilhança de símbolo da treliça e não de caminho, como faria o algoritmo Viterbi. A separação das decisões de bits da treliça faz com que o código duobinário se apresente como um código binário simples, fazendo com que ele perca as vantagens descritas na Seção 2.1.6.

O agrupamento ou não de ramos que não carregam bits de informação não altera os caminhos possíveis na treliça. Assim, o agrupamento de bits de paridade altera apenas a complexidade da treliça, sem influenciar no desempenho da decodificação. Desta maneira, determinam-se as diferenças de desempenho do código, em função do vetor de seccionamento empregado. Do Capítulo 3, tem-se um estudo da variação de complexidade em função da mesma variável. Observou-se que em alguns códigos o seccionamento de menor complexidade é também o de melhor desempenho. Mesmo quando o efeito não ocorre, observa-se que ainda existem seccionamentos que trazem redução de complexidade sem sacrifício de desempenho, então, é função do projetista optar entre redução de complexidade máxima, com perda de desempenho, ou menor redução de complexidade sem perda de desempenho. Estabelece-se, assim, uma maneira de escolher o vetor de seccionamento mais indicado para um determinado código, levando em consideração o compromisso entre desempenho e complexidade.

## 5 CONCLUSÃO

A dissertação apresentada desenvolve uma análise de complexidade de códigos turbo, assim como uma técnica de redução de complexidade baseada na métrica proposta. Utilizou-se treliças seccionadas. Iniciou-se o estudo avaliando as métricas já existentes de complexidade, chegando-se a conclusão que as métricas existentes não são indicadas para a avaliação de códigos turbo. Criou-se, então, uma métrica baseada no número de operações a serem executadas pelo algoritmo. Do estudo da métrica de complexidade, conclui-se que não há um padrão para escolher o seccionamento de menor complexidade: cada treliça tem seu formato particular, e por isso, um estudo de todos os casos possíveis de seccionamento deve ser realizado.

Do ponto de vista de desempenho, observou-se um comportamento que é independente da treliça utilizada ou dos parâmetros de *codec* utilizados (utilização de correção ou tamanho do entrelaçador, utilização de correção do max-log-MAP): quanto mais agrupados estiverem os bits sistemáticos na treliça, melhor é o desempenho do código.

Assim, a escolha de um código para minimização de complexidade deve começar com uma análise da estrutura da treliça mínima do código, seguido de um estudo de complexidade dos possíveis seccionamentos. Nesta análise, leva-se em consideração a complexidade em função do agrupamento de bits sistemáticos, escolhendo-se um ponto ótimo entre a complexidade e o desempenho do código. Repara-se que nem todos os códigos têm o mesmo potencial de redução de complexidade sem perda significativa de desempenho, assim, este fato ser considerado no projeto de um *codec* turbo.

Como trabalhos futuros, sugere-se:

- um estudo mais aprofundado da métrica de complexidade, comparando-se a complexidade de uma arquitetura específica com a complexidade genérica desenvolvida, a fim de verificar-se a correspondência entre as duas;
- buscar a explicação matemática para a redução de desempenho do código quando se separam os bits sistemáticos. Observou-se, durante os testes de desempenho, que a diferença

de desempenho dos diferentes seccionamentos começa a surgir assim que as trocas de informação entre os dois decodificadores inicia-se. Pode-se investigar o comportamento da LLR em função das iterações do algoritmo de decodificação e, se possível, sugerir uma correção para extinguir o efeito da perda de desempenho.

## REFERÊNCIAS

- BAHL, L. et al. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). **Information Theory, IEEE Transactions on**, v. 20, n. 2, p. 284 – 287, mar 1974. ISSN 0018-9448.
- BENCHIMOL, I. B. **Módulo de Treliça Mínimo Para Códigos Convolucionais**. Julho 2011. Exame de Qualificação, Universidade Federal de Pernambuco.
- BERROU, C.; GLAVIEUX, A.; THITIMAJSHIMA, P. Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In: **Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on**. [S.l.: s.n.], 1993. v. 2, p. 1064 –1070 vol.2.
- BOUGARD, B. et al. Energy-scalability enhancement of wireless local area network transceivers. In: **Signal Processing Advances in Wireless Communications, 2004 IEEE 5th Workshop on**. [S.l.: s.n.], 2004. p. 449 – 453.
- CAIN, J.; CLARK, G.; GEIST, J. Punctured convolutional codes of rate  $(n-1)/n$  and simplified maximum likelihood decoding (corresp.). **Information Theory, IEEE Transactions on**, v. 25, n. 1, p. 97 – 100, jan 1979. ISSN 0018-9448.
- DIVSALAR, D.; POLLARA, F. Multiple turbo codes for deep-space communications. **TDA Progress Report. Jet Propulsion Laboratory, California Institute of Technology**, v. 1, p. 42–121, 1995.
- DOUILLARD, C.; BERROU, C. Turbo codes with rate- $m/(m+1)$  constituent convolutional codes. **Communications, IEEE Transactions on**, v. 53, n. 10, p. 1630 – 1638, oct. 2005. ISSN 0090-6778.
- GALLAGER, R. Low-density parity-check codes. **Information Theory, IEEE Transactions on**, v. 8, n. 1, p. 21 –28, january 1962. ISSN 0096-1000.
- GARELLO, R. et al. Analysis of the trellis complexity of interleavers and turbo codes. In: **Information Theory, 2000. Proceedings. IEEE International Symposium on**. [S.l.: s.n.], 2000. p. 65.
- IEEE STD 802.16E. **IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems**. May 2009.
- KIM, W. T. et al. Reduction of computational complexity in two-step sova decoder for turbo code. In: **Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE**. [S.l.: s.n.], 2000. v. 3, p. 1887 –1891 vol.3.
- LEE, D.-S.; PARK, I.-C. Low-power log-map turbo decoding based on reduced metric memory access. In: **Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on**. [S.l.: s.n.], 2005. p. 3167 – 3170 Vol. 4.

- LEUNG, O.-H. et al. Reducing power consumption of turbo code decoder using adaptive iteration with variable supply voltage. In: **Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on**. [S.l.: s.n.], 1999. p. 36 – 41.
- MASSELOS, K.; BLIONAS, S.; RAUTIO, T. Reconfigurability requirements of wireless communication systems. In: **IEEE Workshop on Heterogeneous Reconfigurable Systems on Chip**. [S.l.: s.n.], 2002.
- MCELIECE, R. On the bcjr trellis for linear block codes. **Information Theory, IEEE Transactions on**, v. 42, n. 4, p. 1072 –1092, jul. 1996. ISSN 0018-9448.
- MCELIECE, R.; LIN, W. The trellis complexity of convolutional codes. **Information Theory, IEEE Transactions on**, v. 42, n. 6, p. 1855 –1864, nov. 1996. ISSN 0018-9448.
- MINOWA, T.; IMAI, H. Reduced-complexity iterative decoding of high-rate turbo codes. In: **Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE**. [S.l.: s.n.], 2000. v. 1, p. 193 –197 vol.1.
- PAPAHARALABOS, S. et al. Modified log-map algorithm for simplified decoding of turbo and turbo tcm codes. In: **Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th**. [S.l.: s.n.], 2009. p. 1 –5. ISSN 1550-2252.
- PIMENTEL, C. et al. Minimal trellis for systematic recursive convolutional encoders. **ISIT**, 2011.
- SHANNON, C. A mathematical theory of communications. **Bell Syst. Tech. J.**, v. 27, p. 379–423 e 623–656, 1948.
- VALENTI, S. C. M.; SESHADRI, R. I. **Turbo Code Applications: A Journey from a Paper to Realization**. [S.l.]: Springer, 2005.
- VARDY, A. Handbook of coding theory. In: \_\_\_\_\_. [S.l.]: Elsevier Science Inc., 1998. cap. Trellis Structure of Codes, p. 1106.
- VITERBI, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. **Information Theory, IEEE Transactions on**, v. 13, n. 2, p. 260 – 269, apr 1967. ISSN 0018-9448.
- VUCETIC, B.; YUAN, J. **Turbo Codes, Principles and Applications**. [S.l.]: Kluwer Academic Publishers, 2000.
- WANG, C.-H.; WANG, W.-T.; CHAO, C. chao. A unified structure of trellis-based soft-output decoding algorithms for turbo codes. **Communications, IEEE Transactions on**, v. 52, n. 8, p. 1355 – 1366, 2004. ISSN 0090-6778.