2
Códigos LT

Os códigos LT são abordados neste capítulo. Antes, porém, é necessário que se faça uma breve abordagem a respeito do conceito de Fontana Digital (*Digital Fountain*). Introduzido em [Byres02], este conceito surgiu da intenção de se desenvolver um esquema de codificação no qual, a partir de um subconjunto qualquer de pacotes codificados com cardinalidade igual à do conjunto de pacotes que constituem os dados originais, o receptor seja capaz de recuperar a mensagem originalmente transmitida [Beltrão07]. Uma definição de Fontana Digital Ideal é apresentada a seguir.

Definição 2.1 (Fontana Digital Ideal) *Um esquema capaz de produzir, a partir de k símbolos de entrada* $\mathbf{u}$*, uma sequência de símbolos codificados* $\mathbf{c}$ *que obedece às propriedades [Mitzenmacher04]:*

1. *O comprimento da sequência de símbolos codificados,* $|\mathbf{c}|$*, é um número arbitrariamente grande (possivelmente infinito) de símbolos codificados. (Idealmente, dada a mensagem original* $\mathbf{u}$*, cada símbolo codificado* c_i *pode ser formado em um tempo constante).*

2. *Existe um receptor capaz de reconstruir a mensagem original* $\mathbf{u}$ *de tamanho k, a partir de qualquer conjunto* $\{c_{i_1}, c_{i_2}, \ldots, c_{i_j}, \ldots, c_{i_k}\}$ *de k símbolos codificados. (Essa reconstrução deve ser rápida, de preferência linear em k).*

é denominado uma Fontana Digital Ideal. □

As fontanas digitais são sistemas que, a partir de um bloco, ou arquivo de entrada — na realidade, um vetor de comprimento k, ou ainda uma matriz $1 \times k$,

$$\mathbf{u} = (u_1, u_2, \ldots, u_k), \tag{2-1}$$

com os k símbolos de entrada gerados por uma fonte de informação, produzem na sua saída, um bloco

$$\mathbf{c} = (c_1, \ldots c_j, \ldots, c_n), \tag{2-2}$$

potencialmente muito grande, com n símbolos codificados.

Os códigos LT (Luby *Transform*), propostos por Michael Luby em 2002 [Luby02], foram a primeira implementação do conceito de fontanas digitais com tempos de codificação e decodificação viáveis em sistemas que requeiram a formação de um grande número de símbolos codificados n a partir de k símbolos de entrada [Beltrão07]. Por serem baseados em operações XOR (OU-exclusivo), estes códigos apresentam altas velocidades de codificação e decodificação, além de oferecerem uma boa capacidade na correção de outros erros [Sasaki05]. Os códigos LT foram originalmente concebidos para canais com apagamento (*Erasure Channel*), livres de erros, isto é, canais em que o alfabeto de entrada é o conjunto de símbolos $\mathcal{A} = \{a_0, \ldots, a_{q-1}\}$ e o alfabeto de saída é o conjunto de símbolos $\mathcal{B} = \{a_0, \ldots, a_{q-1}, a_q\}$.

A saída do codificador LT (entrada do canal, portanto) é uma sequência de variáveis aleatórias (v.a.'s) independentes $X = \left(X_1, \ldots, X_k, \ldots, X_n\right)$ e tem-se que a probabilidade condicional, que relaciona a v.a. X_j, à entrada do canal, com a v.a. Y_j, à saída do canal, também designada por probabilidade de transição do canal, é dada por

$$P(Y_j = a_\ell | X_j = a_i) = \begin{cases} 1 - P_a & \text{para } \ell = i \in \{0, q-1\}, \\ P_a & \text{para } \ell = q, \end{cases} \tag{2-3}$$

em que P_a denota a probabilidade de apagamento do canal.

Em geral, tem-se $q = 2^L$ mas, nesta tese, por simplicidade e sem perda de generalidade, é considerado o caso $L = 1$ e $q = 2$. Portanto, a transmissão é feita através de um BEC (*Binary Erasure Channel*). Ainda por simplicidade, para representar o alfabeto de entrada e saída do canal, é usada a notação

$$\mathcal{A} = \{a_0, a_1\} = \{0, 1\} \quad \text{e} \quad \mathcal{B} = \{a_0, a_1, a_2\} = \{0, 1, \text{E}\}, \tag{2-4}$$

onde E representa um apagamento ou indefinição do símbolo.

Nos códigos LT, cada símbolo codificado pode ser gerado de maneira independente e o conjunto de k símbolos originais de entrada pode ser recuperado, com probabilidade $(1 - \delta)$, a partir de quaisquer $k + O\left(\sqrt{k} \ln^2\left(\frac{k}{\delta}\right)\right)$ símbolos codificados, com uma média de $O\left(k \ln\left(\frac{k}{\delta}\right)\right)$ operações-símbolo (operação XOR

ou cópia de um símbolo de um registrador para outro) [Luby02]. O número de símbolos codificados que podem ser gerados a partir dos dados de entrada é potencialmente ilimitado. Por isso, como já dito no Capítulo 1, os códigos LT são ditos de taxa versátil (*rateless*). Assim, independentemente do modelo de perdas do canal em uso, o codificador simplesmente gera e envia dados até o decodificador e este envio só cessará quando o número de dados recebido for suficiente para a recuperação da mensagem original. A característica do canal somente influencia no tempo necessário para que símbolos de saída sejam recebidos em quantidade suficiente pelo decodificador para que a decodificação seja realizada com sucesso [Beltrão07].

Em qualquer canal com apagamento, os códigos LT são considerados quase ótimos, ou seja, considera-se que o decodificador pode recuperar a mensagem original formada por k símbolos de entrada a partir de quaisquer $(1+\epsilon)k$ símbolos codificados, onde $0 < \epsilon < 1$. Considera-se que um código é ótimo, se ele consegue recuperar a mensagem original formada por k símbolos de entrada, a partir de qualquer subconjunto de k símbolos de saída dentre os n gerados [Maymounkov03]. Por serem considerados quase ótimos e por sua eficiência aumentar com o crescimento do tamanho dos blocos de entrada, os códigos LT são ditos universais [Luby02].

2.1 Codificação

No processo de codificação dos códigos LT, a mensagem original a ser transmitida consiste de k símbolos de informação e é denotada por $\mathbf{u} = (u_1, u_2, \ldots, u_k)$. Em geral, $u_i \in GF(2^L)$, mas por simplicidade considera-se $L = 1$. À saída do codificador, um vetor $\mathbf{c}$ de dimensão n é obtido. Mesmo que $\mathbf{u}$ seja determinado, cada símbolo codificado c_j é produzido de forma aleatória de acordo com:

$$c_j = \mathbf{u} \cdot \mathbf{g}_j = \sum_{\ell=1}^{k} u_\ell g_{j,\ell} \,, \tag{2-5}$$

onde o símbolo $\sum$ representa a adição módulo 2 (XOR) dos símbolos e $\mathbf{g}_j = (g_{j,1}, g_{j,2}, \ldots, g_{j,k})$ são os vetores-coluna da matriz geradora $\mathbf{G}$ $(k \times n)$. Vale observar que $\mathbf{c} = (c_1, \ldots, c_j, \ldots, c_n) \in \{0,1\}^*$, onde $\{0,1\}^*$ é o conjunto de todos os blocos binários. Percebe-se que os códigos LT pertencem ao grupo dos códigos de bloco lineares. Porém, ao contrário dos códigos de bloco lineares tradicionais, que têm seus projetos baseados na maximização de sua distância mínima, os códigos LT são projetados através da construção de uma

distribuição de pesos ou graus que tem papel fundamental no desempenho do sistema, como será visto adiante.

A construção da matriz geradora **G** envolve uma teoria desenvolvida primeiramente por Luby [Luby02] e posteriormente por outros trabalhos [Nguyen07]. Esta teoria leva à determinação dos pesos de Hamming $\omega(\mathbf{g}_j)$ dos vetores-coluna e também à decisão de quais componentes desses vetores (elementos da matriz em cada coluna j) terão valor igual a 1. Os pesos dos vetores-coluna $\mathbf{d} = \big(d_1, d_2, \ldots, d_n\big)$ são escolhidos aleatoriamente de acordo com uma distribuição de probabilidade (DP). Sendo assim, têm-se que $\big(d_1, d_2, \ldots, d_n\big)$ é uma realização da sequência $\big(D_1, D_2, \ldots, D_n\big)$ de v.a.'s discretas i.i.d.'s (independentes e identicamente distribuídas), que assumem valores no conjunto de inteiros $\mathbb{Z}_k = \{1, \ldots, k\}$ com distribuição de probabilidade caracterizada pela probabilidade

$$P(D_j = i) \;\; = \;\; p_i, \;\; (i = 1, \ldots, k; \; j = 1, \ldots, n). \tag{2-6}$$

Uma vez estabelecido o peso d_j do vetor-coluna $\mathbf{g}_j$, deve-se decidir quais dos seus componentes devem ter o valor igual a 1. Em outras palavras, quais dos símbolos de entrada u_ℓ, $\ell \in \mathbb{Z}_k$, devem ser somados (módulo 2) para formar a componente de palavra-código c_j, conforme a Equação 2-5. Os valores dos índices ℓ das d_j parcelas em (2-5) são realizações da sequência de v.a.'s $\big(L_1, \ldots, L_{d_j}\big)$ definidas a seguir.

$L_1 \in \mathbb{Z}_k$ é uma v.a. discreta uniforme com DP,

$$\mu_1(i_1) = P(L_1 = i_1) \;\; = \;\; \frac{1}{k}, \; i_1 \in \mathbb{Z}_k. \tag{2-7}$$

L_2 é também uma v.a. discreta com DP condicional uniforme, definida da seguinte maneira para $(i_1, i_2) \in \mathbb{Z}_k^2$:

$$\mu_2(i_2 \big| i_1) = P(L_2 = i_2 | L_1 = i_1) \;\; = \;\; \begin{cases} \frac{1}{k-1} & \text{para } i_2 \neq i_1, \\ 0 & \text{para } i_2 = i_1. \end{cases} \tag{2-8}$$

Analogamente, a v.a. L_3 também tem DP condicional uniforme, esta definida para $(i_1, i_2, i_3) \in \mathbb{Z}_k^3$, por:

$$\begin{aligned} \mu_3(i_3 \big| i_1, i_2) \;\; &= \;\; P\big(L_3 = i_3 | (L_1, L_2) = (i_1, i_2)\big) \\ &= \;\; \begin{cases} \frac{1}{k-2} & \text{para } i_3 \notin \{i_1, i_2\}, \\ 0 & \text{para } i_3 \in \{i_1, i_2\}. \end{cases} \end{aligned} \tag{2-9}$$

Generalizando, têm-se para $(i_1, i_2, \ldots, i_\ell) \in \mathbb{Z}_k^\ell$:

$$\begin{aligned} \mu_\ell(i_\ell | i_1, \ldots, i_{\ell-1}) &= P\big(L_\ell = i_\ell \big| \, (L_1, \ldots, L_{\ell-1}) = (i_1, \ldots, i_{\ell-1})\big) \quad (2\text{-}10) \\ &= \begin{cases} \frac{1}{k-(\ell-1)} & \text{para } i_\ell \notin \{i_1, \ldots, i_{\ell-1}\}, \\ 0 & \text{para } i_\ell \in \{i_1, \ldots, i_{\ell-1}\}. \end{cases} \end{aligned}$$

O processo de codificação apresentado pode ser descrito usando um grafo $\mathcal{G}$ bipartido que associa os símbolos de entrada $\mathbf{u}$ a nós denominados *nós-de-mensagem* e associa os símbolos de saída $\mathbf{c}$ a nós denominados *nós-de-verificação*. Cada valor u_i é associado a um nó-de-mensagem α_i e cada c_j é associado a um nó-de-verificação β_j. Este grafo terá uma aresta e_{ij} conectando o nó α_i ao nó β_j, se o elemento da matriz geradora é $g_{ij} = 1$. A Tabela 2.1 [Beltrão07] ilustra um exemplo do processo de codificação correspondente à seguinte matriz geradora, para $\mathbf{u} = (u_1, u_2, u_3, u_4, u_5) = (1, 0, 0, 1, 1)$:

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}. \quad (2\text{-}11)$$

Tabela 2.1: Processo de codificação ($\oplus$ denota a operação XOR).

Símbolo	**Grau** (d_j)	**Nós vizinhos**	**Valor**
c_1	2	$u_1 u_2$	$u_1 \oplus u_2 = 1$
c_2	3	$u_1 u_2 u_4$	$u_1 \oplus u_2 \oplus c_4 = 0$
c_3	4	$u_1 u_2 u_3 u_5$	$u_1 \oplus u_2 \oplus c_3 \oplus s_5 = 0$
c_4	4	$u_1 u_3 u_4 u_5$	$u_1 \oplus u_3 \oplus c_4 \oplus s_5 = 1$
c_5	1	u_5	$u_5 = 1$
c_6	2	$u_2 u_5$	$u_2 \oplus u_5 = 1$

Note que o vetor de pesos é $\mathbf{d} = (d_1, d_2, d_3, d_4, d_5, d_6) = (2, 3, 4, 4, 1, 2)$.

A Figura 2.1 mostra o grafo $\mathcal{G}$ correspondente ao processo de codificação exemplificado na Tabela 2.1. O grafo é particionado em dois conjuntos: o conjunto $\mathcal{A} = \{\alpha_1, \ldots, \alpha_k\}$ de nós-de-mensagem e o conjunto $\mathcal{B} = \{\beta_1, \ldots, \beta_n\}$ de nós-de-verificação. A cada nó-de-mensagem α_i associa-se um símbolo de entrada u_i (às vezes identifica-se o nó α_1, indistintamente, como nó u_i) e a

cada nó-de-verificação β_j corresponderá um símbolo codificado c_j (este nó será também identificado como nó c_j). Por simplicidade iremos nos referir aos nós pertencentes ao conjunto $\mathcal{A}$ como α-nós e aos nós pertencentes ao conjunto $\mathcal{B}$ como β-nós.

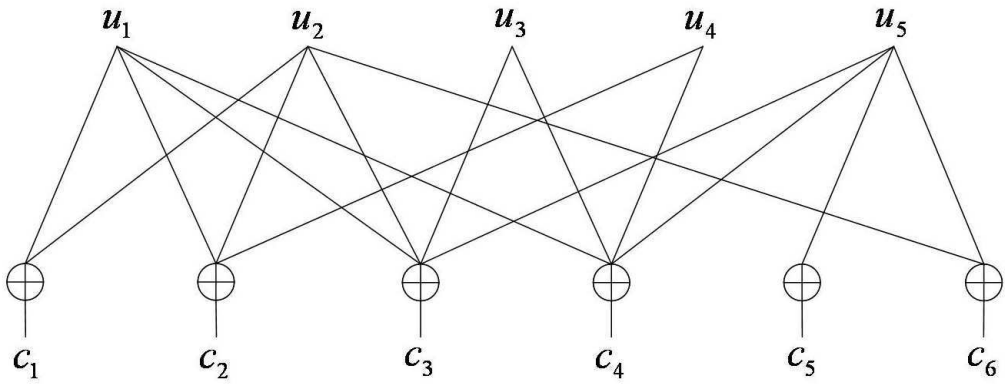

Figura 2.1: Grafo $\mathcal{G}$ resultante da codificação da Tabela 2.1.

O grafo $\mathcal{G}$ é construído de maneira aleatória. Cada nó-de-verificação c_j está conectado a d_j nós-de-mensagem $\{u_{i_1}, \ldots, u_{i_{d_j}}\}$, onde o índice ℓ do nó α_ℓ (ou valor u_ℓ) é uma v.a. $L_\ell \in \mathbb{Z}_k$ escolhido aleatoriamente de acordo a uma DP μ_ℓ. Tem-se, assim, que $\mathcal{V}_j = \{u_{i_1}, \ldots, u_{i_{d_j}}\}$ é o conjunto de nós-de-mensagem vizinhos do nó c_j. O número de vizinhos, d_j, de um nó-de-verificação c_j será referido como o grau do nó c_j e a DP μ_j será designada por *distribuição de graus* do código LT. Então, o algoritmo de codificação LT é descrito a seguir:

Algoritmo de codificação LT:

1. *Inicializar o codificador*:

- **(a)** Para cada símbolo codificado c_j, $j \in \mathbb{Z}^n$, escolher aleatoriamente e segundo a DP ρ, um número d_j. Ou seja, determinar o vetor de graus $\mathbf{d} = (d_1, d_2, \ldots, d_n)$;
- **(b)** Especificar o conjunto $\mathcal{V}_j = \{\alpha_{i_1}, \ldots, \alpha_{i_{d_j}}\}$ de d_j nós-de mensagem (distintos), indexados por $\{i_1, \ldots, i_{d_j}\}$, valores estes escolhidos aleatoriamente, de acordo com a distribuição de probabilidade μ_j. Estes serão os d_j nós-de-mensagem vizinhos do nó-de-verificação β_j associado ao símbolo codificado c_j. Fazendo-se $g_{j,\ell} = 1$ se $\alpha_\ell \in \mathcal{V}_j$ e $g_{j,\ell} = 0$ caso contrário, constrói-se os vetores-coluna $\mathbf{g}_j = (g_{j,1}, g_{j,2}, \ldots, g_{j,k})$, de acordo com a sequência de graus $\mathbf{d}$ e, no final, a matriz-$\mathbf{d}$, $\mathbf{G}^{[\mathbf{d}]}$;

2. *Gerar a palavra-código a ser transmitida*: encontrar $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ de acordo com a Equação 2-5. □

Conforme visto, várias DP's são usadas no processo de codificação dos códigos LT: uma, ρ, dada pela Equação 2-6, para determinar o grau d_j de cada símbolo codificado c_j e as demais, uniformes, conforme dado pelas Equações (2-7) a (2-10), para escolher os símbolos de entrada *vizinhos* (que entram na formação) do símbolo codificado. As principais DP's usadas para a determinação dos graus dos símbolos codificados estão descritas na Seção 2.3.

2.2 Decodificação

Pressupõe-se que também o decodificador conhece o vetor de graus $\mathbf{d}$ e, $\mathbf{G}^{[\mathbf{d}]}$ a correspondente matriz-$\mathbf{d}$. Em outras palavras, o decodificador deve conhecer o grafo $\mathcal{G}$. Admitindo-se que a palavra-código $\mathbf{c}$ é transmitida em um canal BEC, na saída deste canal, o vetor $\mathbf{v}_j = (v_{j,1}, v_{j,2}, \ldots, v_{j,n})$ é entregue ao receptor. Levando em consideração que os (possíveis) símbolos apagados devem ser descartados antes de iniciar o primeiro passo da decodificação, o decodificador, então, identifica e expurga tais símbolos. Considera-se agora um novo e atualizado vetor de graus $\mathbf{d}^{[1]} = (d_1^{[1]}, d_2^{[1]}, \ldots, d_{n'}^{[1]})$ que representa os graus dos $n' < n$ símbolos recebidos não apagados $\mathbf{c}^{[1]} = \left(c_1^{[1]}, c_2^{[1]}, \ldots, c_{n'}^{[1]}\right)$.

O objetivo do decodificador é estimar o vetor de informação transmitido $\mathbf{u}$. No Passo 1, na entrada do decodificador têm-se o vetor $\mathbf{c}^{[1]}$, de dimensão n', associado ao seu vetor de graus $\mathbf{d}^{[1]}$. Inicialmente, a informação transmitida estimada, $\mathbf{u}^{[1]} = \left(u_1^{[1]}, u_2^{[1]}, \ldots, u_k^{[1]}\right)$ é considerada uma sequência de apagamentos, ou seja, $u_\ell^{[1]} = \mathrm{E}$ para todos $\ell = 1, \ldots, k$.

O processo de decodificação inicia com os vetores $\left(\mathbf{c}^{[1]}, \mathbf{d}^{[1]}, \mathbf{u}^{[1]}\right)$ e produz uma sequência de vetores $\{\left(\mathbf{c}^{[1]}, \mathbf{d}^{[1]}, \mathbf{u}^{[1]}\right), \left(\mathbf{c}^{[2]}, \mathbf{d}^{[2]}, \mathbf{u}^{[2]}\right), \ldots, \left(\mathbf{c}^{[m]}, \mathbf{d}^{[m]}, \mathbf{u}^{[m]}\right)\}$ através de sucessivas transformações do trio de vetores $\left(\mathbf{c}^{[\ell]}, \mathbf{d}^{[\ell]}, \mathbf{u}^{[\ell]}\right)$, associado à matriz-$\mathbf{d}_\ell$, $\mathbf{G}^{[\mathbf{d}_\ell]}$, no trio de vetores $\left(\mathbf{c}^{[\ell+1]}, \mathbf{d}^{[\ell+1]}, \mathbf{u}^{[\ell+1]}\right)$, associado à matriz-$\mathbf{d}_{\ell+1}$, $\mathbf{G}^{[\mathbf{d}_{\ell+1}]}$. Estas transformações serão designadas pelo nome *redução de grafo* (mais detalhes nas definições que se seguirão). Vale ressaltar que, a cada trio de vetores $\left(\mathbf{c}^{[\ell]}, \mathbf{d}^{[\ell]}, \mathbf{u}^{[\ell]}\right)$, há um grafo, $\mathcal{G}^{[\ell]}$, único, associado.

Considere agora o trio de vetores $\left(\mathbf{c}^{[m]}, \mathbf{d}^{[m]}, \mathbf{u}^{[m]}\right)$ e o seu grafo associado $\mathcal{G}^{[m]} = (\mathcal{A}^{[m]}, \mathcal{B}^{[m]}, \mathcal{E}^{[m]})$. O conjunto $\mathcal{B}^{[m]} = \{\beta_{j_1}^{[m]}, \beta_{j_2}^{[m]}, \ldots \beta_{J_m}^{[m]}\}$, de cardinalidade J_m, é o conjunto de nós associados ao vetor $\mathbf{c}^{[m]}$ — por simplicidade estes serão chamados de nós-β. O conjunto de nós-α, $\mathcal{A}^{[m]} = \{\alpha_{i_1}^{[m]}, \alpha_{i_2}^{[m]}, \ldots \alpha_{I_m}^{[m]}\}$,

de cardinalidade I_m, é o conjunto de nós associados ao vetor $\mathbf{u}^{[m]}$. E, $\mathcal{E}^{[m]}$, com cardinalidade $\sum_{x=1}^{J_m} \widetilde{d}_{m,x}$, é o conjunto de arestas (ou conexões) $(\beta_j^{[m]}, \alpha_i^{[m]})$ onde (j,i) são as posições da matriz $\mathbf{G}^{[\mathbf{d}_m]}$ correspondentes a $g_{j,i}^{[m[} = 1$.

A seguir apresenta-se algumas definições que irão auxiliar a definir o conceito de *ripple* introduzido em [Luby02] e a transformação de um grafo em outro, denominada por *redução*.

Definição 2.2 (Nó Mono-conectado) *Um nó é dito mono-conectado se possui apenas um nó vizinho.* □

Definição 2.3 (Grafo β-Redutível) *Um grafo $\mathcal{G}^{[m]}$ é dito β-redutível se possui pelo menos um nó-β (nó-de-verificação) $\beta_{j'}^{[m]}$ mono-conectado e, além disto, o valor $u_{i'}^{[m]}$ do símbolo associado ao vizinho único $\alpha_{i'}^{[m]}$ deste nó (nó-de-mensagem) satisfaz a uma das duas condições seguintes:*

1. *$u_{i'}^{[m]} = E$ (é um apagamento), ou;*
2. *$u_{i'}^{[m]} = c_{i'}^{[m]}$ (possui valor idêntico ao valor do símbolo associado ao nó $\beta_{j'}^{[m]}$).* □

É fácil observar que o vetor de graus $\mathbf{d}^{[m]}$ correspondente a um grafo $\mathcal{G}^{[m]}$, β-redutível, possui pelos menos uma componente de grau unitário.

Definição 2.4 (β-Redução de Grafo) *A β-redução consiste em transformar um grafo $\mathcal{G}^{[m]}$, β-redutível, em outro grafo $\mathcal{G}^{[m+1]}$, identificando-se, inicialmente, o nó-β de grau* 1, *$\beta_{j'}^{[m]}$, de menor índice j'. Seja $\left(\beta_{j'}^{[m]}, \alpha_{i'}^{[m]}\right)$ a aresta conectada a este nó. O processo de β-redução é realizado construindo-se o novo grafo $\mathcal{G}^{[m+1]}$ com nós e respectivos valores associados idênticos aos do grafo $\mathcal{G}^{[m]}$ exceto pela exclusão da aresta $\left(\beta_{j'}^{[m+1]}, \alpha_{i'}^{[m+1]}\right)$ e respectivo nó $\beta_{j'}^{[m+1]}$ e, atribuição do valor $c_{i'}^{[m]}$ ao nó $\alpha_{i'}^{[m+1]}$ — ou seja fazendo-se $u_{i'}^{[m+1]} = c_{j'}^{[m]}$.* □

Definição 2.5 (Grafo α-Redutível) *Um grafo $\mathcal{G}^{[m]}$ é dito α-redutível se possui pelo menos um nó-α $\alpha_{i'}^{[m]}$ com símbolo associado de valor $u_{i'}^{[m]} \neq E$ conectado a seus vizinhos (tais nós receberão denominação* nó-marcado*).* □

Definição 2.6 (α-Redução de Grafo) *A α-redução consiste em transformar um grafo $\mathcal{G}^{[m]}$, α-redutível, em outro grafo $\mathcal{G}^{[m+1]}$, identificando-se, inicialmente, o nó-marcado , $\alpha_{i'}^{[m]}$, de menor índice i'. Seja $\left(\alpha_{i'}^{[m]}, \beta_{j'}^{[m]}\right)$ a aresta conectada a este nó. O processo de α-redução é realizado construindo-se o novo grafo $\mathcal{G}^{[m+1]}$ com nós e respectivos valores associados idênticos aos do grafo $\mathcal{G}^{[m]}$ exceto pela exclusão da aresta $\left(\alpha_{i'}^{[m+1]}, \beta_{j'}^{[m+1]}\right)$ e, atribuição do valor $c_{j'}^{[m+1]} = c_{j'}^{[m]} \oplus u_{i'}^{[m]}$ ao nó $\beta_{j'}^{[m+1]}$.* □

O algoritmo de decodificação LT é apresentado a seguir.

Algoritmo de decodificação LT:

1. *Inicializar o decodificador*:

(a) Inicializar o valor do contador de passos: $m = 0$;

(b) Definir a sequência inicial de graus: $\mathbf{d}^{[1]} = \left(d_1^{[1]}, d_2^{[1]}, \ldots, d_{n'}^{[1]}\right)$;

(c) Especificar a matriz $\widetilde{\mathbf{G}}^{\mathbf{d}^{[1]}}$: a partir da construção dos vetores-coluna $\mathbf{g}_j^{[1]} = \left(g_{j,1}^{[1]}, \ldots, g_{j,k}^{[1]}\right)$, de acordo com a sequência de graus $\mathbf{d}^{[1]}$;

(d) Definir a entrada inicial do decodificador: $\mathbf{c}^{[1]} = \left(c_1^{[1]}, \ldots, c_{n'}^{[1]}\right)$;

(e) Definir a saída inicial do decodificador: $\mathbf{u}^{[1]} = \left(u_1^{[1]}, u_2^{[1]}, \ldots, u_k^{[1]}\right)$ com $u_\ell^{[1]} = \mathrm{E}$ para todo $\ell = 1, 2, \ldots, k$;

(f) Fazer $\mathcal{G}^{[1]} = \left(\mathcal{A}^{[1]}, \mathcal{B}^{[1]}, \mathcal{E}^{[1]}\right)$, onde

(i) $\mathcal{G}^{[1]} = \mathcal{G}$ é o grafo de decodificação inicial,

(ii) $\mathcal{B}^{[1]} = \left\{\beta_{j_1}^{[1]}, \ldots \beta_{J_1}^{[1]}\right\}$ é o conjunto de nós-β, de cardinalidade $J_1 = n'$, associados ao vetor $\mathbf{c}^{[1]}$,

(iii) $\mathcal{A}^{[1]} = \left\{\alpha_{i_1}^{[1]}, \ldots \alpha_{I_1}^{[1]}\right\}$, de cardinalidade $I_1 = k$, é o conjunto de nós-α associados ao vetor $\mathbf{u}^{[1]}$,

(iv) $\mathcal{E}^{[1]}$, com cardinalidade $\sum_{x=1}^{J_1} d_x^{[1]}$, é o conjunto de arestas (ou conexões) $\left(\beta_j^{[1]}, \alpha_i^{[1]}\right)$, com (j, i) sendo as posições da matriz $\mathbf{G}^{\mathbf{d}^{[1]}}$ correspondentes a $g_{j,i}^{[1]} = 1$;

2. Faça $m = m + 1$. Se o grafo $\mathcal{G}^{[m]}$ for β-irredutível, siga para o último passo (Passo 6) – ocorreu uma falha de decodificação;

3. Realizar uma sequência de β-reduções de grafo, iniciando com o grafo $\mathcal{G}^{[m]}$ até que se produza o grafo β-irredutível $\mathcal{G}^{[\mathrm{temp}]} = \left(\mathcal{A}^{[\mathrm{temp}]}, \mathcal{B}^{[\mathrm{temp}]}, \mathcal{E}^{[\mathrm{temp}]}\right)$;

4. Realizar uma operação de redução, que elimina um nó-marcado e suas conexões a nós-de-verificação (α-redução), do grafo $\mathcal{G}^{[\mathrm{temp}]}$ e produza o grafo $\mathcal{G}^{[m+1]} = \left(\mathcal{A}^{[m+1]}, \mathcal{B}^{[m+1]}, \mathcal{E}^{[m+1]}\right)$, associado ao trio de vetores $\left(\mathbf{d}^{[m+1]}, \mathbf{c}^{[m+1]}, \mathbf{u}^{[m+1]}\right)$;

5. Seja $\mathcal{M}^{[m+1]} = \left\{\beta^{[m+1]}, \ldots, \beta_j^{[m+1]}, \ldots, \beta_{M_{m+1}}^{[m+1]}\right\}$ o conjunto de β-nós mono-conectados. Se $\left|\mathcal{M}^{[m+1]}\right| > 0$ voltar para o Passo 2;

6. Parar. A sequência transmitida estimada é dada por $\widehat{\mathbf{u}} = \mathbf{u}^{[m+1]}$. □

2.3 Distribuições de Graus para Projeto de um Código LT

Conforme já mencionado, para cada símbolo codificado c_j é escolhido um grau d_j (número de vizinhos), seguindo-se uma distribuição de probabilidade. Algumas distribuições vêm sendo aplicadas com este intuito. Para um melhor entendimento da sequência desta seção, faz-se necessária a definição do termo *ripple*.

Ao final da realização das α-reduções, Passo 5 do algoritmo de decodificação, cada nó $\beta_j^{[m+1]}$, pertencente ao conjunto de β-nós mono-conectados, $\mathcal{M}^{[m+1]} = \left\{\beta_1^{[m+1]}, \ldots, \beta_j^{[m+1]}, \ldots, \beta_{M_{m+1}}^{[m+1]}\right\}$, está conectado ao seu nó vizinho, único, $\alpha_i^{[m+1]}$. O conjunto $\mathcal{R}^{[m+1]}$, formado por α-nós $\alpha_i^{[m+1]}$ associados a símbolos $u_j^{[m+1]}$ e conectados a nós $\beta_j^{[m+1]} \in \mathcal{M}^{[m+1]}$, é denominado conjunto de *nós-marcados* (em [Luby02] estes nós são denominados *covered input symbols*). O conjunto $\mathcal{R}^{[m+1]}$ de cardinalidade ondulante, é denominado, na literatura inglesa, *ripple set* — nesta tese iremos usar a mesma denominação da literatura em inglês (a denominação conjunto ondulante parece ser uma tradução apropriada). Estas observações motivam a definição seguinte.

Definição 2.7 (*Ripple*) *O conjunto $\mathcal{R}^{[m+1]}$ de nós-marcados que resta após a redução (α-redução) que elimina do grafo $\mathcal{G}^{[m]}$, os vizinhos destes nós-marcados e suas respectivas arestas é denominado* Ripple.

Ao longo do processo de decodificação LT, caso haja algum símbolo codificado com apenas um único vizinho, ele é liberado para transferir valor a este seu vizinho único. Em cada passo do algoritmo, o conjunto $\mathcal{R}$, de símbolos de entrada aguardando trasferência de valor (que ainda não foram processados), é chamado *ripple*. Em cada passo subseqüente do processo LT, um símbolo de entrada pertencente ao *ripple* é processado, ou seja, ele é removido como vizinho de todos os símbolos codificados que o tenham como tal, e posteriormente, após a remoção, todos os símbolos de entrada vizinhos dos símbolos codificados mono-conectados são integrados ao *ripple*. Alguns destes vizinhos podem ser símbolos de entrada para os quais nenhum valor foi transferido (iguais a E), fazendo que o *ripple* cresça, enquanto outros podem ser símbolos de entrada para os quais já houve valor transferido (valor diferente de E), e não causam alteração no tamanho dp *ripple*. O processo termina quando o *ripple* encontrar-se vazio em algum passo. O processo falha quando existe ao menos um símbolo de entrada igual a E no final do processo e tem sucesso se, no final, todos os símbolos de entrada são diferentes de E.

Um projeto de distribuição de graus apropriado assegura que o processo de decodificação LT libera símbolos codificados para marcar (*cover*) de forma incremental todos os símbolos de entrada. Os objetivos de uma apropriada distribuição de graus são:

- Liberar lentamente símbolos codificados enquanto o processo evolui para manter o conjunto de símbolos mono-conectados (e, conseqüentemente, o *ripple*) pequeno, de modo a evitar a marcação (*covering*) redundante dos símbolos de entrada no *ripple* por vários símbolos codificados. Ou seja, o ideal seria que $\left|\mathcal{R}^{[m]}\right| = \left|\mathcal{M}^{[m]}\right|$
- Não permitir que o *ripple* fique vazio antes que todos os símbolos de entrada estejam devidamente marcados (*covered*). O ideal seria ter sempre $\left|\mathcal{R}^{[m]}\right| = 1$.

A seguir estão apresentadas as três distribuições de graus abordadas nesta tese, ou seja,

1. a Distribuição Sóliton Robusta (DSR)[Luby02],
2. a Distribuição Sóliton Robusta Melhorada (DSRM)[Tee06] e
3. a Distribuição Sóliton Robusta Melhorada Truncada para códigos LT sistemáticos (DSRMT)[Nguyen07].

2.3.1 Distribuição Sóliton Robusta (DSR)

Uma boa distribuição de graus requer uma propriedade básica de adicionar símbolos de entrada ao *ripple* na mesma taxa em que eles são processados. Além disso, a distribuição de graus deve requerer, em média, a recepção da menor quantidade de símbolos de saída possível para garantir o sucesso da decodificação e, também, utilizar o menor número de operações XOR para gerar um símbolo de saída. Isso é obtido mantendo-se baixo o grau médio dos símbolos de saída.

Para uma distribuição de graus atingir os objetivos anteriormente descritos, ela precisa liberar somente um símbolo a cada iteração, pois assim o *ripple* seria mantido no menor tamanho possível (um símbolo), já que o símbolo processado a cada iteração seria imediatamente substituído por outro. Desta forma, o *ripple* seria mantido com apenas um símbolo ao longo de todo o processo de decodificação, até que esta fosse completada com sucesso. A distribuição Sóliton Ideal foi projetada para atingir tais objetivos [Luby02].

Porém, apesar de apresentar um comportamento ideal, a distribuição Sóliton Ideal, dada pela Equação 2-12, mostra-se pouco útil na prática. Isto deve-se ao fato de que, qualquer variação no tamanho do *ripple* (que é de um símbolo) ao longo do processo de decodificação, causa um esvaziamento do mesmo, o que leva a uma falha na decodificação.

$$\rho(d) = \begin{cases} \frac{1}{k} & \text{para } d = 1, \\ \frac{1}{d \cdot (d-1)} & \text{para } d = 2, 3, ..., k. \end{cases} \tag{2-12}$$

Buscando corrigir essa falha, a distribuição Sóliton Robusta (DSR) — originada de uma modificação da Sóliton Ideal — procura fazer com que o tamanho do *ripple* seja grande o suficiente para que, a cada passo do processo de decodificação, a probabilidade dele desaparecer seja bem pequena. Ao mesmo tempo, a distribuição Sóliton Robusta procura obter o menor tamanho possível para o *ripple*, a fim de minimizar a redundância surgida de símbolos codificados liberados para marcar (*cover*) símbolos de entrada que já se encontrem no *ripple*.

A idéia aqui é projetar a distribuição que mantenha o valor esperado da v.a. que representa o o tamanho do *ripple* em torno do valor $\ln\left(\frac{k}{\delta}\right)\sqrt{k}$, ao longo do processo de decodificação, onde δ corresponde à probabilidade de falha do processo de decodificação (na verdade, corresponde ao limite superior desta probabilidade [Luby02]).

De acordo com Luby [Luby02], a distribuição Sóliton Robusta μ é definida do seguinte modo. Considere a seguinte equação:

$$\overline{R} = c \ln\left(\frac{k}{\delta}\right)\sqrt{k}\ , \tag{2-13}$$

que representa o número médio de símbolos codificados de grau um. Para uma constante apropriada $c > 0$, têm-se:

$$\tau(d) = \begin{cases} \frac{\overline{R}}{kd} & \text{para } 1 \leq d \leq \frac{k}{\overline{R}} - 1, \\ \frac{\overline{R} \ln \frac{\overline{R}}{\delta}}{k} & \text{para } d = \frac{k}{\overline{R}}, \\ 0 & \text{para } d = \frac{k}{\overline{R}} + 1, ..., k. \end{cases} \tag{2-14}$$

Adicionando $\tau(d)$ a $\rho(d)$, dado pela Equação 2-12, e normalizando, obtém-se a distribuição Sóliton Robusta:

$$\mu(d) = \frac{\rho(d) + \tau(d)}{\beta}\ , \tag{2-15}$$

onde o fator de normalização $\beta = \sum_d \big(\rho(d) + \tau(d)\big)$, garante que as somas das probabilidades seja unitária. A distribuição Sóliton Robusta é mostrada na Figura 2.2.

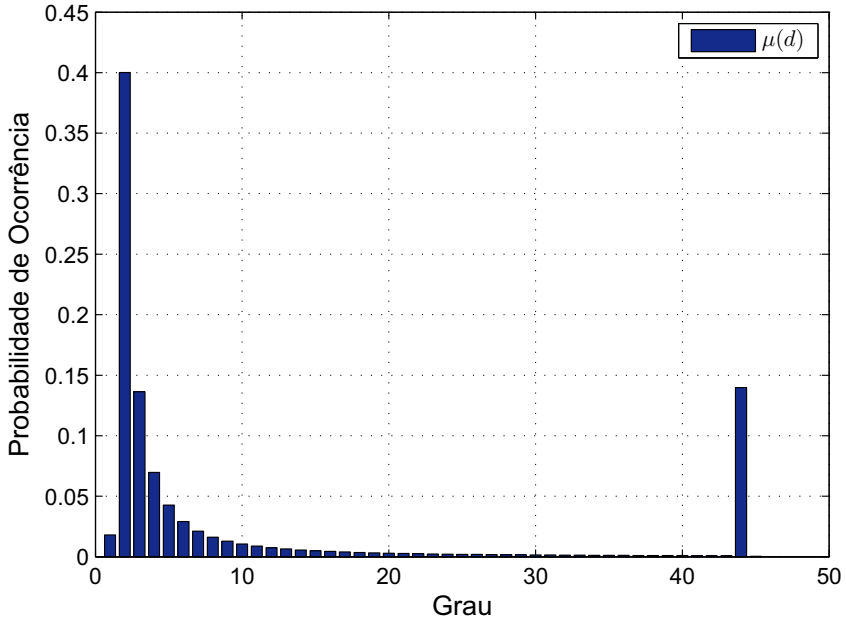

Figura 2.2: A distribuição Sóliton Robusta (DSR) para o caso $k = 10000$, $c = 0.2$ e $\delta = 0.1$.

A Figura 2.2, bem como a Figura 2.3, mostrada a seguir, foram obtidas para $c = 0.2$ e $\delta = 0.1$. Este valor de c foi escolhido para uma melhor visualização dos gráficos. No restante da tese, utiliza-se $c = 0.03$ e $\delta = 0.1$, valores considerados adequados através de simulações em [Paiba08].

Luby, em [Luby02], mostrou que, para um valor de $c > 0$ apropriado (independente de k e δ), o decodificador pode recuperar a mensagem original a partir de $n = k\beta = k + c\sqrt{k}\ln^2(k/\delta)$ símbolos codificados, com taxa de falha menor que δ.

O número de símbolos codificados é ajustado em $n = k\beta$. Isto implica que $k(\rho(i) + \tau(i))$ é o valor esperado do número de símbolos codificados de grau i.

Na prática, um código LT é capaz de recuperar a mensagem transmitida, de comprimento k, caso o decodificador receba ao menos 5% de símbolos a mais, em relação ao número original k. Isto, considerando-se as condições do canal adequadas. Os parâmetros c e δ da Equação 2-13 determinam o número

de símbolos requeridos para a recuperação de todos os k símbolos transmitidos.

2.3.2 Distribuição Sóliton Robusta Melhorada (DSRM)

Tee, Nguyen *et al.* observaram, em [Tee06], a existência de alguns valores de graus d_i, com probabilidades $\mu(d_i)$ tão baixas, que o número de símbolos dado pelo produto de $\mu(d_i)$ e k pode ser menor que um, indicando a ausência de símbolos com estes graus. Em alguns casos, a distribuição de Luby, definida na Equação 2-15, pode levar a um prematuro fracasso na decodificação e uma consequente perda de símbolos durante o processo de decodificação, a menos que o número de símbolos redundantes seja elevado. Quando o processo de decodificação é interrompido devido à ausência de pacotes de grau um, precisa-se receber mais símbolos para a recuperação de todos os símbolos originais.

A proposta de Tee, Nguyen *et al.*, portanto, é melhorar o comportamento da distribuição Sóliton Robusta no caso em que $\mu(d_i)k < 1$, através da introdução de um fator extra,

$$\nu = \sum \mu(d_i)k \ , \tag{2-16}$$

cuja a finalidade é criar uma distribuição de grau mais benéfica, onde d_i representa o termo grau-i da distribuição, e satisfazendo as seguintes condições:

$$\begin{cases} \left(\frac{1}{d(d-1)} + \frac{\overline{R}}{kd}\right)\frac{k}{\beta} < 1 & \text{para } 2 \leq d \leq (\frac{k}{\overline{R}} - 1), \\ \left(\frac{1}{d(d-1)}\right)\frac{k}{\beta} < 1 & \text{para } (\frac{k}{\overline{R}} + 1) \leq d \leq k. \end{cases} \tag{2-17}$$

Em referência à Sóliton Robusta de Luby, essa distribuição é denominada Sóliton Robusta Melhorada (DSRM). Nela, determina-se o parâmetro ν obedecendo ao conjunto de desigualdades dado pela Equação 2-17, no intuito de maximizar a frequência relativa dos símbolos de grau um. Ou seja, todos aqueles graus d_i que satisfazem a Equação 2-17 têm redefinidas as suas probabilidades de distribuição a zero e a soma de todas essas probabilidades de distribuição são adicionadas à distribuição de probabilidade de grau um. As probabilidades $\mu(d_i)$ da DSRM são mostradas na Figura 2.3.

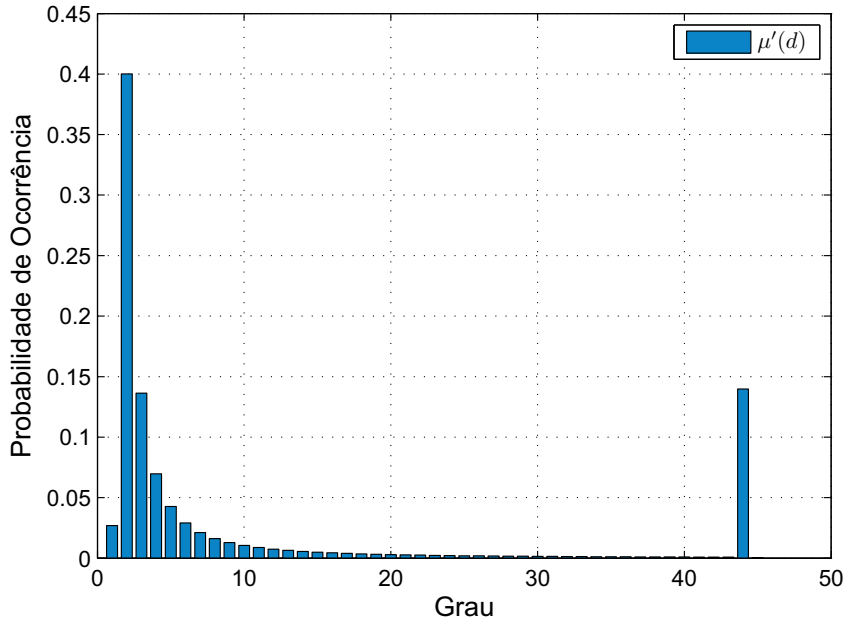

Figura 2.3: A distribuição Sóliton Robusta Melhorada (DSRM) para o caso $k = 10000$, $c = 0.2$ e $\delta = 0.1$.

2.3.3 Distribuição Sóliton Robusta Melhorada Truncada (DSRMT)

Na distribuição Sóliton Robusta, há duas condições a serem satisfeitas por todos os símbolos de entrada, a fim de serem recuperados no decodificador. Primeiramente, o número de símbolos codificados, n, deve satisfazer à desigualdade

$$n \geq k + 2R \ln \frac{\overline{R}}{\delta}.$$

Também, o número de símbolos que possuem o mais alto grau deve obedecer a $d \geq \frac{k}{\overline{R}}$ [Luby02, Mackay03]. Nguyen, Yang *et al.* ao desenvolverem um código LT sistemático em [Nguyen07], viram que estas condições continuavam válidas para a decodibilidade de tais códigos com taxas maiores que $R = \frac{1}{2+\epsilon}$, onde $\epsilon = 2\ln(\frac{\overline{R}}{\delta})$ é o *overhead* do código LT original. Entretanto, para taxas menores que $R = \frac{1}{2+\epsilon}$, essas condições não são suficientes, já que as densidades de ambas as matrizes, geradora e de paridade, dos códigos LT sistemáticos são baixas.

No intuito de aumentar essas densidades, a distribuição de graus usada para gerar a matriz de paridade dos códigos LT sistemáticos, denominada distribuição Sóliton Robusta Melhorada Truncada (DSRMT) [Nguyen08], foi

$$\Omega(d) = \mu(d,\gamma,\nu) = \begin{cases} \frac{1}{\beta'}\left(1 + \frac{\overline{R}}{k} + \nu\right) & \text{para } d = \gamma, \\ \frac{\gamma}{\beta'}\left(\frac{1}{d\left(\frac{d}{\gamma}-1\right)} + \frac{\overline{R}}{k}\cdot\frac{1}{d}\right) & \text{para } d = 2\gamma, 3\gamma, \\ & \quad ..., \frac{\gamma k}{\overline{R}} - 1, \\ \frac{\overline{R}}{\beta' k}\left(\log\left(\frac{\overline{R}}{\delta}\right) + \frac{1}{\left(\frac{k}{\overline{R}}-1\right)}\right) & \text{para } d = \frac{\gamma k}{\overline{R}}, \\ 0 & \text{para } d > \frac{\gamma k}{\overline{R}} \text{ e } d = 1, \end{cases} \tag{2-18}$$

onde

$$\beta' = \sum_{d}[(\rho(d) + \tau(d)) + \nu(\gamma)], \tag{2-19}$$

com γ um número inteiro maior que um, k é o número total de símbolos de informação originais e, $\overline{R}$ é o número de símbolos com um grau específico γ, que satisfaz a condição [Luby02, Nguyen08]

$$\overline{R} \equiv c \ln\left(\frac{k}{\delta}\right)\sqrt{k}.$$

Além do mais, ν representa o fator extra que garante a decodibilidade da distribuição Sóliton Robusta Melhorada, já visto na Equação 2-16. Mantendo-se o grau máximo em $d_{max} = \frac{\gamma k}{\overline{R}}$, garante-se que os símbolos originais de entrada serão representados no grupo de símbolos sistemáticos codificados, pelo menos γ vezes [Luby02, Shokrollahi06].

A Figura 2.4 mostra a DSRMT. Note que, assim como nas Figura 2.2 e Figura 2.3, utilizou-se $c = 0.2$ para uma melhor visualização do gráfico. Também por este motivo, o eixo das abscissas possui um passo de tamanho 6.

2.3.4 Comparação dos Desempenhos dos Códigos LT para Diferentes Distribuições

Com as três distribuições de graus utilizadas nesta tese, mostra-se uma comparação dos desempenhos no gráfico da Figura 2.5. Antes, porém, faz-se necessário definir Taxa de Apagamento de Símbolo (TAS) como a porcentagem dos símbolos que não puderam ser reconhecidos nem como 0, nem como 1. Ou

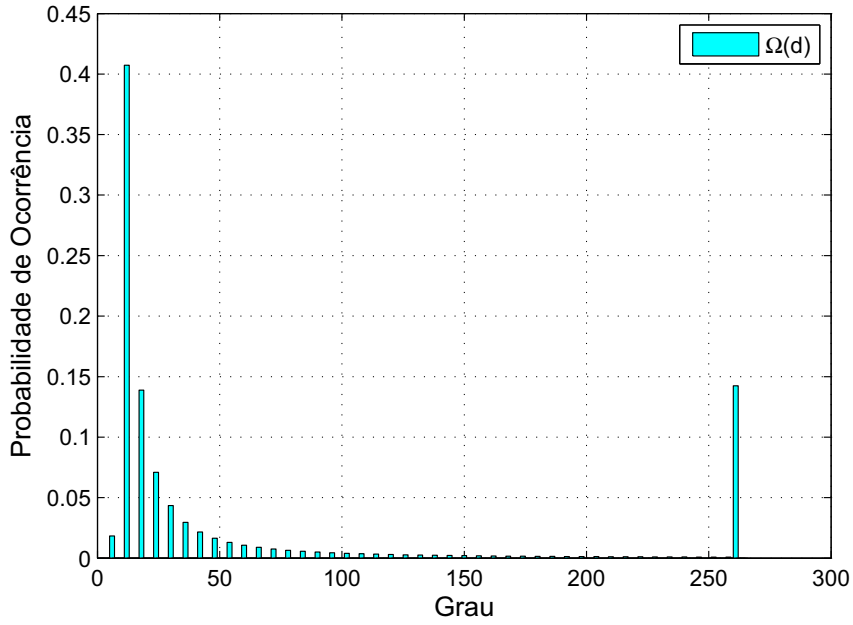

Figura 2.4: A distribuição Sóliton Robusta Melhorada Truncada (DSRMT) para o caso $k = 10000$, $c = 0.2$, $\delta = 0.1$ e $\gamma = 6$.

seja, é a porcentagem dos símbolos considerados apagados após a decodificação ($\hat{u}_i = \mathrm{E}$). A análise da TAS em função da qualidade do canal BEC $(1 - P_a)$, onde P_a é a probabilidade de apagamento do mesmo, na Figura 2.5, mostra o melhor desempenho da distribuição Sóliton Robusta Melhorada em relação à Sóliton Robusta e também da Sóliton Robusta Melhorada Truncada em relação à Sóliton Robusta Melhorada. Lembrando que a curva da distribuição Sóliton Robusta Melhorada Truncada foi gerada com um código LT sistemático, enquanto as outras duas curvas foram geradas com um código LT convencional [Luby02]. Estas simulações foram realizadas para 1000 blocos de entrada, com comprimento $k = 1000$ e *overhead* $\epsilon = 30\%$.

2.4 Códigos LT Sistemáticos

Os códigos sistemáticos são uma classe de códigos em que todos os símbolos de informação aparecem explicitamente no bloco a ser transmitido. São preferíveis em muitas situações práticas. A codificação e a decodificação quando se usa um código sistemático são simplificadas quando não ocorrem apagamentos num dado bloco transmitido, o que leva a uma redução de custos. Originalmente, os códigos LT foram propostos para uma cosntrução não-sistemática. Nguyen, Yang *et al.* desenvolveram um código LT sistemático

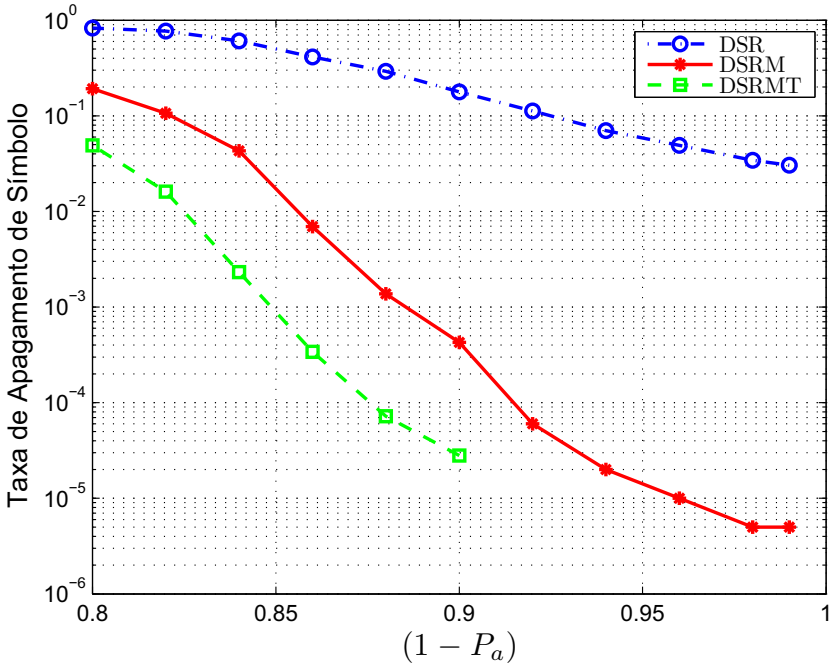

Figura 2.5: Desempenho das distribuições Sóliton Robusta (DSR), Sóliton Robusta Melhorada (DSRM) e Sóliton Robusta Melhorada Truncada (DSRMT), com $k = 1000$, $c = 0.03$, $\rho = 0.1$, $\gamma = 6$ e $\epsilon = 30\%$.

[Nguyen07] e encotraram uma distribuição de graus apropriada [Nguyen08].

Conforme ilustra a Figura 2.5, os códigos LT sistemáticos, usados com a distribuição Sóliton Robusta Melhorada Truncada, apresentam melhor desempenho em relação aos códigos LT não-sistemáticos com outras distribuições. Devido a esses melhores resultados e à menor complexidade que apresentam [Nguyen07, Yuan07, Yuan08, Yuan09], esta tese é baseada nos códigos LT sistemáticos usados com a distribuição de graus Sóliton Robusta Melhorada Truncada [Nguyen07, Nguyen08]. Os códigos LT sistemáticos são identificados ao longo desta tese por SLT. Analogamente, os códigos LT não-sistemáticos são referenciados por nSLT.

De forma análoga ao que foi apresentado para os códigos não-sistemáticos na Seção 2.1, a Tabela 2.2 apresenta um exemplo do processo de codificação LT sistemática, correspondente à matriz geradora abaixo e com $\mathbf{u} = (u_1, u_2, u_3, u_4, u_5) = (1, 0, 0, 1, 1)$:

Tabela 2.2: Processo de codificação LT sistemática.

Símbolo	**Grau** (d_j)	**Nós vizinhos**	**Valor**
c_1	1	u_1	$u_1 = 1$
c_2	1	u_2	$u_2 = 0$
c_3	1	u_3	$u_3 = 0$
c_4	1	u_4	$u_4 = 1$
c_5	1	u_5	$u_5 = 1$
c_6	3	$u_1 u_2 u_5$	$u_1 \oplus u_2 \oplus u_5 = 0$
c_7	2	$u_3 u_5$	$u_3 \oplus u_5 = 1$
c_8	2	$u_3 u_4$	$u_3 \oplus u_4 = 1$

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}, \tag{2-20}$$

onde o vetor de graus é $\mathbf{d} = (d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8) = (1, 1, 1, 1, 1, 3, 2, 2)$.

A média dos graus dos símbolos sistemáticos codificados é dada por [Nguyen08]:

$$\overline{D} = \sum_d \frac{d \cdot (\rho(d) + \tau(d) + \nu(d))}{\beta'} + 1 \ . \tag{2-21}$$

A Figura 2.6 mostra o grafo $\mathcal{G}_{sist}$ resultante da codificação da Tabela 2.2. Percebe-se facilmente através do grafo $\mathcal{G}_{sist}$ que os nós de verificação sistemáticos $(c_1, c_2, c_3, c_4, c_5)$ estão conectados através de apenas uma aresta aos respectivos nós de informação $(u_1, u_2, u_3, u_4, u_5)$, caracterizando que todos eles possuem graus $d_i = 1$.

A codificação e a decodificação dos códigos LT sistemáticos obedecem exatamente aos mesmos algoritmos apresentados, respectivamente, na Seção 2.1 e na Seção 2.2, para os códigos LT originalmente propostos (não-sistemáticos). O Capítulo 4, mais precisamente na Figura 4.4 e na Figura 4.5, apresenta resultados referentes a comparações entre códigos LT não-sistemáticos e sistemáticos, ficando claro o melhor desempenho destes últimos.

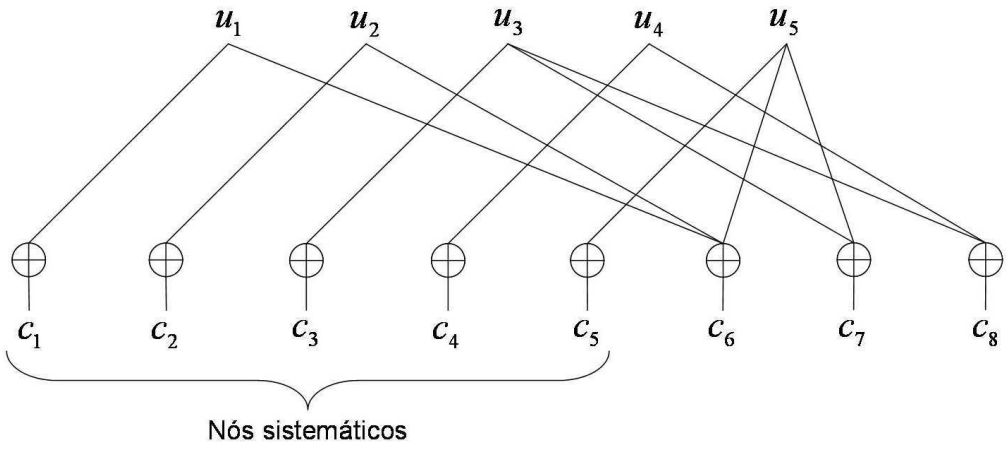

Figura 2.6: Grafo $\mathcal{G}_{sist}$ resultante da codificação da Tabela 2.2.

2.5 Códigos LT Bidimensionais

Alguns esquemas de codificação utilizam dois códigos atuando de forma combinada. Há várias formas de se realizar esta combinação. Os códigos Raptor [Shokrollahi06] utilizam, de forma serial, uma pré-codificação com um código LDPC antes de uma codificação LT. Baseados nessa idéia e com o objetivo de melhorar o desempenho dos códigos LT, foram propostos em [Paiba08], os códigos LT bidimensionais.

A Figura 2.7 mostra um diagrama em blocos de um sistema de comunicações utilizando um código Raptor. Tais como os códigos Raptor, os códigos LT bidimensionais também fazem uso de uma pré-codificação. Porém, neste caso, foram utilizadas em [Paiba08], duas codificações LT. Além desta diferença, também há o fato de que estas codificações são aplicadas sobre os dados organizados sob forma matricial — primeiro são codificadas as linhas e depois as colunas da matriz — esta matriz será chamada de matriz LT. Para isto, obviamente, os símbolos de informação provenientes da fonte são rearranjados nessa matriz, como descrito a seguir.

2.5.1 Codificação LT Bidimensional

O esquema de codificação dos códigos LT bidimensionais é feito de forma que os símbolos $\mathbf{u} = (u_1, ..., u_k)$ de entrada são rearranjados em uma matriz LT, como mostra a Figura 2.8.

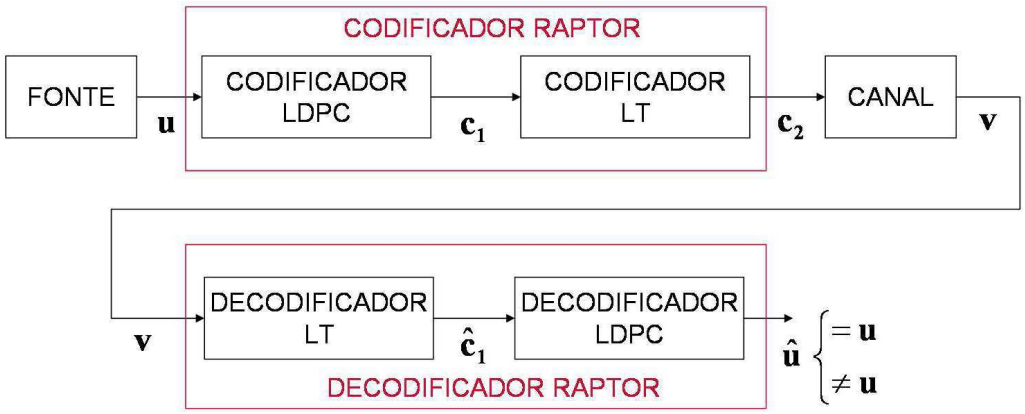

Figura 2.7: Diagrama em blocos de um sistema de comunicações fazendo uso de um código Raptor.

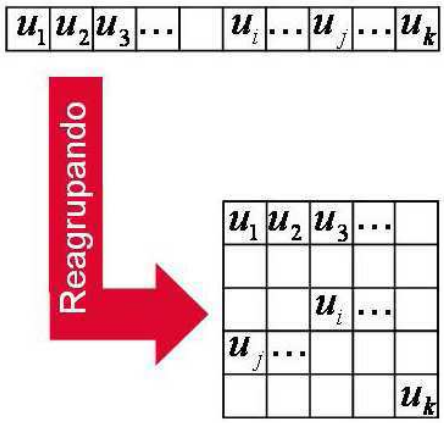

Figura 2.8: Reagrupamento bidimensional dos símbolos de entrada **u**.

Quando os símbolos de entrada estão rearranjados em uma matriz, aplica-se um código LT linha a linha. Por trabalhar horizontalmente, denominou-se este código de LT horizontal (LT_H). O código LT_H introduz um *overhead* em cada linha e então, um outro código LT (LT_V) é aplicado, agora verticalmente, coluna a coluna. O código LT_V, por sua vez, introduz um *overhead* em cada coluna. O *overhead* total do sistema é dado por:

$$(1+\epsilon) = (1+\epsilon_H)\cdot(1+\epsilon_V). \tag{2-22}$$

A Figura 2.9 ilustra esse processo. Então, os símbolos são novamente colocados na forma de vetor (ou *string*) para poderem ser transmitidos através do canal.

Apesar da Figura 2.9 mostrar que o *overhead* vertical é variável, neste

ponto — na codificação — o modo de operação é, inicialmente, modo taxa-fixa. A operação será em modo taxa-variável (*rateless*) caso o decodificador não consiga recuperar todos os símbolos de entrada. Uma explicação mais detalhada deste aspecto é dada na próxima seção, que aborda a decodificação dos códigos LT bidimensionais.

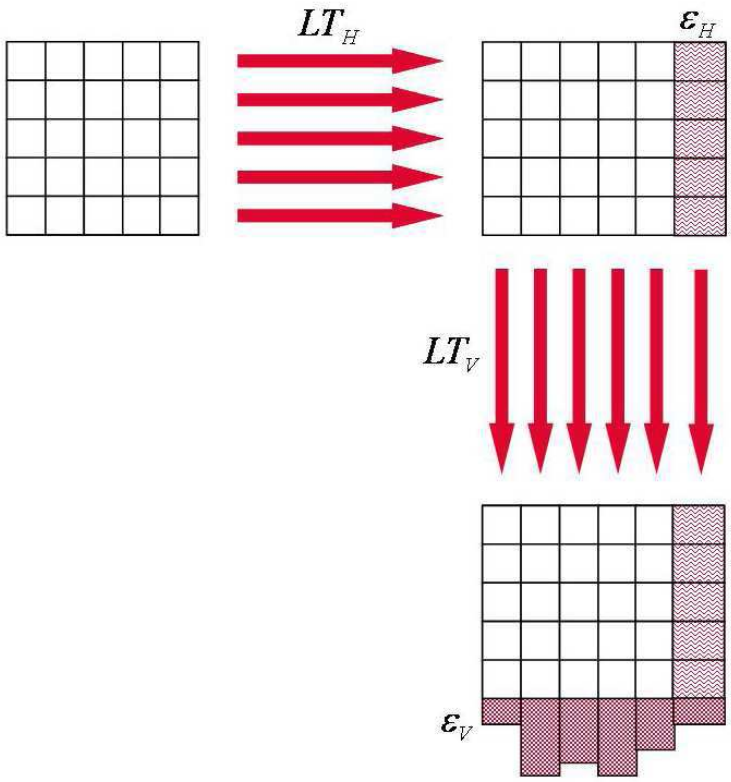

Figura 2.9: Processo de codificação LT bidimensional.

2.5.2 Decodificação LT Bidimensional

Após receber um bloco à saída do canal, o decodificador rearranja seus símbolos aos moldes da Figura 2.8, em uma matriz intermediária. Então, a decodificação bidimensional, que em cada dimensão é feita de acordo com o algoritmo de decodificação LT descrito na Seção 2.2, é realizada da seguinte forma:

1. **Decodificação vertical**: opera em cada coluna da matriz intermediária, com o intuito de recuperar os símbolos codificados verticalmente pelo código LT_V. Alguns destes símbolos podem ser considerados apagados devido ao fato de que não foi possível a sua recuperação.

2. **Decodificação horizontal**: atua linha a linha na matriz intermediária, tentando a recuperação dos símbolos de informação que formam a matriz original.

3. **Transmissão de símbolos adicionais**: enquanto não se consegue recuperar todos os símbolos u_i de entrada (a matriz original), ou seja, quando ocorre uma falha no processo de decodificação LT bidimensional, o transmissor continua enviando símbolos codificados pelo canal. Esses símbolos extras são adicionados, um a um, na(s) coluna(s) onde forem necessários. Por isso, diz-se que o *overhead* vertical ϵ_V é variável (vide Figura 2.9), sendo necessário determinar sua média para calcular-se o *overhead* ϵ total do sistema.

Ao receber esses símbolos adicionais, o decodificador realiza novamente as decodificações vertical e horizontal e o processo se repete até que todos os símbolos u_i de informação sejam recuperados, ou então até que algum limite máximo de *overhead* requerido (ϵ_{REQ}) pré-estabelecido seja alcançado. Por ϵ_{REQ} entende-se o *overhead* necessário para uma decodificação sem erros.