3
Projetando a distribuição de graus para códigos LT

Neste capítulo é analisado o processo de construção de uma boa distribuição de graus para os códigos LT. Quando estudamos o processo de codificação e decodificação destes códigos, fica clara a necessidade de obter-se uma distribuição de probabilidades que atinja os dois objetivos primordiais em relação ao *ripple*: conjunto de símbolos de entrada cobertos mas ainda não processados.

1. Ao longo de processo de decodificação o tamanho do *ripple* deve ser o menor possível para evitar um desperdício de esforço computacional, visto que símbolos codificados que são liberados não irão cobrir nenhum símbolo de entrada ainda não-processado que se encontra no *ripple*.

2. O tamanho do *ripple* deve ser suficientemente grande tal que não fique vazio antes da conclusão bem sucedida da decodificação, ou seja, não deve ficar vazio antes que todos os símbolos de entrada estejam devidamente cobertos. Caso contrário, este provocaria uma falha na decodificação.

Uma característica básica requerida de uma boa distribuição de graus é permitir que os símbolos do *ripple* sejam processados na mesma taxa em que outros são adicionados ao mesmo. Esta característica originou o nome Sóliton, pois um Sóliton é uma onda que equilibra perfeitamente a dispersão e a refração. Além desses objetivos, uma boa distribuição de graus deve:

- Requerer, em média, a recepção da menor quantidade de símbolos de saída possível para garantir o sucesso da decodificação.
- 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.

Uma distribuição de graus que consiga atender os objetivos anteriormente descritos seria aquela que liberasse 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,

conseguindo assim, manter o *ripple* com apenas um símbolo ao longo de todo o processo de decodificação até que a decodificação seja completada com sucesso. A seguir é descrita a distribuição Sóliton Ideal, a qual foi projetada para atingir os objetivos mencionados.

3.1 Distribuição Sóliton Ideal

Esta seção apresenta a dedução da distribuição Sóliton Ideal seguindo a notação proposta em [8]. Na primeira iteração ($t = 0$), que corresponde ao início do processo de decodificação, seja $h_0(d)$ o número de símbolos codificados de grau d. Assim, o valor esperado do número de símbolos codificados de grau d que tem seu grau reduzido para $d-1$ na iteração seguinte é

$$h_0(d)\frac{d}{k}, \tag{3-1}$$

onde k é o número de símbolos de entrada. A expressão (3-1) é o valor esperado de uma variável binomial, em que d/k é a probabilidade de um dos d vizinhos ser o símbolo presente no *ripple* a ser processado na iteração seguinte, reduzindo a $d-1$ o grau de todos os símbolos codificados que o tenham como vizinho. Na t-ésima iteração, quando t dos k símbolos de entrada tiverem sido recuperados e o número de símbolos codificados de grau d for $h_t(d)$, o valor esperado do número de símbolos codificados de grau d que terão seu grau reduzido para $d-1$ é

$$h_t(d)\frac{d}{k-t}. \tag{3-2}$$

Lembrar que a cada iteração um símbolo de entrada presente no *ripple* é processado. Assim, a fim de que o valor esperado do número de símbolos codificados de grau 1 satisfaça a condição

$$h_t(1) = 1 \quad \forall\ t \in \{0, ..., k-1\}, \tag{3-3}$$

devemos ter inicialmente $h_0(1) = 1$ e $h_0(2) = k/2$ e, mais genericamente

$$h_t(2) = \frac{k-t}{2}. \tag{3-4}$$

Para deduzir a distribuição Sóliton Ideal, basta observar que na segunda iteração ($t = 1$), o número de símbolos de grau 2 é o número de símbolos de grau 3 que tiveram seu grau reduzido para 2, mais o número de símbolos de grau 2 que não foram reduzidos a grau 1, ou seja,

$$h_1(2) = \frac{3}{k}\, h_0(3) + \left(h_0(2) - \frac{2}{k}\, h_0(2)\right).$$

Da mesma forma, na terceira iteração ($t = 2$), o número de símbolos de grau 2 é o número de símbolos de grau 3 que tiveram seu grau reduzido para 2, mais o número de símbolos de grau 2 que não foram reduzidos a grau 1 na iteração anterior, ou seja,

$$h_2(2) = \frac{3}{k-1}\, h_1(3) + \left(h_1(2) - \frac{2}{k-1}\, h_1(2) \right).$$

Assim, mais geralmente tem-se para $t > 0$,

$$h_t(d) = \frac{d+1}{k-(t-1)}\, h_{t-1}(d+1) + \left(h_{t-1}(d) - \frac{d}{k-(t-1)}\, h_{t-1}(d) \right). \quad (3\text{-}5)$$

Logo, após algumas manipulações na equação (3-5), chega-se à seguinte equação

$$h_t(d+1) = \frac{k-t}{d+1}\Big(h_{t+1}(d) - h_t(d) \Big) + \frac{d}{d+1} h_t(d). \quad (3\text{-}6)$$

Sendo o interesse da presente dedução encontrar uma fórmula geral para $h_0(d)$, é necessário eliminar na equação (3-6) a dependência entre termos de iterações distintas, ou seja, eliminar a recursividade. Para este fim, o teorema apresentado a continuação oferece uma forma mais simples.

Teorema 3.1 *O número de símbolos codificados de grau d na t-ésima iteração que leva à distribuição de grau ótima, ou seja, que o tamanho do ripple seja 1 é:*

$$h_t(d) = \frac{k-t}{d(d-1)}. \quad (3\text{-}7)$$

■

A prova desse teorema é feita por indução usando a equação (3-6). Detalhes da prova por indução são apresentados no Apêndice A.

Agora, fazendo uso do Teorema 3.1, os valores reais que estamos procurando para $h_0(d)$ são

$$h_0(d) = \frac{k}{d(d-1)}, \quad \forall\ d \in \{2, ..., k\}. \quad (3\text{-}8)$$

A fim de obter a distribuição de probabilidade necessária, nós dividimos $h_0(d)$ o número de símbolos codificados de grau d na primeira iteração, pelo número total de símbolos de entrada k que formam a mensagem original [25], ou seja, normalizamos os valores $h_0(d)$ e obtemos

$$\rho(1) = \frac{1}{k} \qquad \text{visto que} \;\; h_0(1) = 1,$$

$$\rho(d) = \frac{1}{d(d-1)} \quad \text{para} \;\; d = 2, \ldots, k$$

que nada mais é que a distribuição Sóliton Ideal desenvolvida por Luby em [6]. Nas Figuras 3.1 e 3.2 tem-se uma ilustração da evolução dos valores de $h_t(d)$ ao longo do processo de decodificação. Os blocos sombreados representam os símbolos de grau d que são reduzidos a grau $d-1$ a cada iteração. Perceba que em ambas iterações, apenas um símbolo de grau 2 é reduzido a grau 1, conforme requerido pela distribuição Sóliton Ideal.

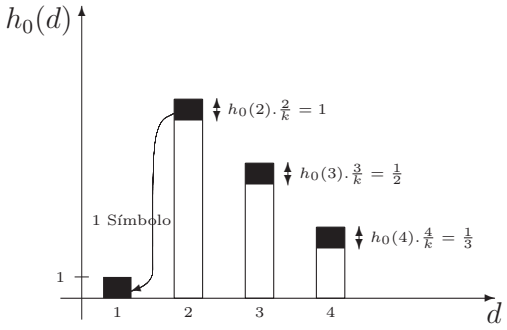

Figura 3.1: Distribuição Sóliton Ideal: primeira iteração.

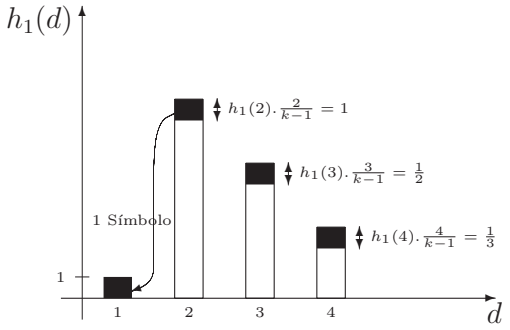

Figura 3.2: Distribuição Sóliton Ideal: segunda iteração.

A distribuição Sóliton Ideal, apesar de apresentar um comportamento ideal, mostra-se pouco útil na prática, já que uma pequena mudança no valor esperado do número de símbolos no *ripple* (um símbolo) ao longo do processo de decodificação, causa um esvaziamento do mesmo, o que leva a uma falha na decodificação. Portanto, a distribuição Sóliton Ideal pode ser modificada com a finalidade de conseguir uma alta probabilidade de que o *ripple* não se esvazie antes que o processo de decodificação haja sido completado. Essa modificação leva a outra distribuição de probabilidade chamada, distribuição Sóliton Robusta. A Figura 3.3 ilustra o desempenho com a distribuição Sóliton Ideal, ou seja, a probabilidade de falha (P_F) na decodificação após $(1+\epsilon)k$ símbolos de saída terem sido recebidos. Neste caso, as simulações foram realizadas com 1000 blocos de tamanho $k = 1000$ símbolos de entrada.

3.2 Distribuição Sóliton Robusta

O grande problema com a distribuição Sóliton Ideal é que o tamanho do *ripple* é de apenas um símbolo. Qualquer variação no tamanho deste

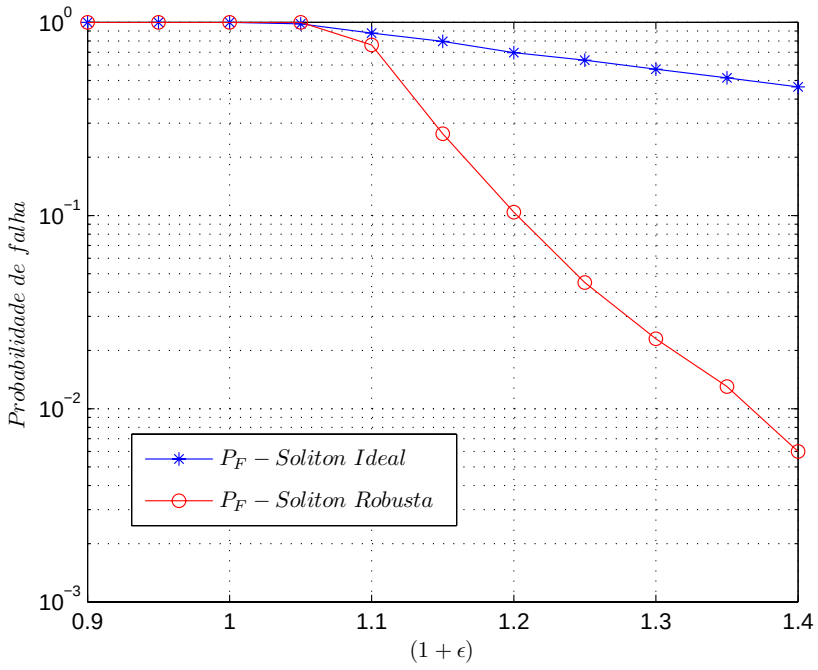

Figura 3.3: Desempenho das distribuições Sóliton Robusta com parâmetros $c = 0.03$ e $\delta = 0.1$ e Sóliton Ideal.

pode fazer que ele seja esvaziado e o processo de decodificação falhe. A distribuição Sóliton Robusta procura corrigir essa falha, fazendo com que o tamanho do *ripple* seja grande o suficiente tal que, a cada passo do processo de decodificação, a probabilidade de que o *ripple* esvazie seja bem pequena. Ao mesmo tempo, a distribuição Sóliton Robusta procura obter o menor tamanho esperado possível para o *ripple*, a fim de minimizar a redundância surgida de símbolos codificados liberados cobrindo símbolos de entrada que já se encontrem no referido *ripple*.

A idéia aqui é projetar uma distribuição que mantenha o valor esperado para o tamanho do *ripple* em torno de $\ln(k/\delta)\sqrt{k}$ ao longo do processo de decodificação, sendo δ a probabilidade de falha no processo. A escolha desse valor se dá devido à observação de que o processo de decodificação (crescimento e diminuição do *ripple*) se assemelha a um *"random walk"*; e em um *"random walk"* de comprimento k, a probabilidade de que o mesmo se desvie de sua média por um valor maior que $\ln(k/\delta)\sqrt{k}$ é no máximo δ [6]. Assim, δ é o limite superior da probabilidade de falha na decodificação.

Repetindo a análise utilizada na dedução da distribuição Sóliton Ideal,

porém com o valor esperado do número de símbolos codificados de grau 1 sendo agora $h_t(1) = 1 + \bar{R}$ para todo t, em vez de 1. Percebe-se entretanto, que ao contrário da distribuição Sóliton Ideal, o valor esperado do número de símbolos codificados de grau 2 que devem ter o grau reduzido para a unidade não mais vale 1, isso porque dentre os $\bar{R}$ símbolos adicionais pode haver repetição. Assim, existe uma probabilidade igual a $\bar{R}/k$ de que o símbolo processado na iteração encontre-se entre os $\bar{R}$ símbolos já presentes no *ripple*. Então, o valor esperado do número de símbolos codificados de grau 2 que devem ter seu grau reduzido para 1 vale $1 + \bar{R}/k$, ou seja,

$$\frac{2}{k}\, h_0(2) = 1 + \frac{\bar{R}}{k},$$

o que implica,

$$h_0(2) = \frac{k}{2} + \frac{\bar{R}}{2}.$$

Na t-ésima iteração, quando t símbolos tiverem sido recuperados, têm-se

$$\frac{2}{k-t}\, h_t(2) = 1 + \frac{\bar{R}}{k-t}.$$

Usando o mesmo procedimento que culminou na obtenção da Equação (3-8) para a dedução da distribuição Sóliton Ideal, obtém-se

$$h_0(d) = \frac{k}{d(d-1)} + \frac{\bar{R}}{d}, \quad \text{para } d > 1. \tag{3-9}$$

Logo, dividimos o número de símbolos codificados de grau d na primeira iteração, $h_0(d)$, pelo número total de símbolos de entrada k, e obtém-se

$$\mu'_d = \frac{1}{d(d-1)} + \frac{\bar{R}}{kd} = \rho(d) + \tau(d), \quad \text{para } d > 1, \tag{3-10}$$

onde μ'_d é a probabilidade de um símbolo ter grau d.

A seguir é explicado o porquê da escolha do incremento $\tau(d)$. A razão para truncar o segundo termo da Equação (3-10), $\tau(d)$, para $d > k/\bar{R}$, e substituí-lo pelo incremento $\tau(k/\bar{R})$, é para assegurar que a complexidade da decodificação não cresça mais que $O(k \ln(k))$ [8].

Perceba que no início da decodificação $\tau(1)$ assegura que o tamanho do *ripple* seja $1 + \bar{R}$ inicialmente. O incremento final $\tau(k/\bar{R})$ assegura que todos os símbolos de entrada não cobertos, sejam cobertos quando o número destes for igual ao tamanho do *ripple*. Em outras palavras, o incremento final $\tau(k/\bar{R})$ assegura que todos os símbolos de entrada sejam selecionados, ao menos uma

vez, como vizinhos de um símbolo codificado. Isto é similar ao processo de liberar simultaneamente $\bar{R}\ln(\bar{R}/\delta)$ bolas (símbolos codificados) para cobrir $\bar{R}$ urnas (símbolos de entradas)[1]. Assim, o desgaste causado pela liberação de vários símbolos codificados para cobrir cada um destes símbolos de entrada ao menos uma vez é somente uma fração pequena do número total de símbolos de entrada k. Então, o incremento $\tau(d)$ é definido como

$$\tau(d) = \begin{cases} \frac{\bar{R}}{dk}, & \text{para } 1 \leq d \leq \frac{k}{\bar{R}} - 1, \\ \frac{\bar{R}}{k}\ln\left(\frac{\bar{R}}{\delta}\right), & \text{para } d = \frac{k}{\bar{R}}, \\ 0, & \text{para } d = \frac{k}{\bar{R}} + 1, ..., k. \end{cases}$$

Finalmente, normalizando μ'_d obtemos

$$\mu(d) = \frac{\rho(d) + \tau(d)}{\beta},$$

a distribuição Sóliton Robusta definida anteriormente.

Nas Figuras 3.4 e 3.5 tem-se uma ilustração do processo de decodificação usando a distribuição Sóliton Robusta. A interpretação é a mesma das figuras Sóliton Ideal. Os blocos escuros representam o valor esperado dos símbolos de grau d que são reduzidos a grau $d-1$, portanto, para $d = 2$, 3 e 4 os blocos escuros valem 1, $\frac{1}{2}$ e $\frac{1}{3}$ de símbolo respectivamente. Os blocos mais claros representam os símbolos liberados os quais já se encontram no *ripple* e não acrescentam nenhuma informação ao processo de decodificação. Eles sempre têm o mesmo valor para uma dada iteração, sendo o valor esperado desses símbolos igual a $\frac{\bar{R}}{(k-t)}$, onde t representa a iteração.

3.2.1
Análise da distribuição Sóliton Robusta

Nesta subseção são apresentados alguns resultados básicos envolvendo a distribuição Sóliton Robusta. Calcula-se aqui o número de símbolos codificados e o grau médio de um símbolo codificado.

Teorema 3.2 *O número de símbolos codificados é* $n = k + O\left(\sqrt{k}\ln^2(k/\delta)\right)$.

[1] A análise do processo clássico de lançar bolas aleatoriamente em um conjunto de urnas mostra que são necessários $k\ln(k/\delta)$ bolas para assegurar-se que com probabilidade $(1-\delta)$, todas as k urnas sejam cobertas, ou seja, possuam no mínimo uma bola.

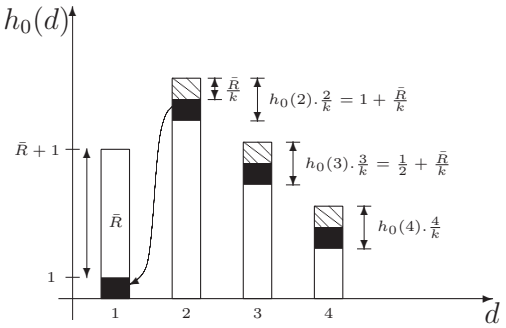

Figura 3.4: Distribuição Sóliton Robusta: primeira iteração.

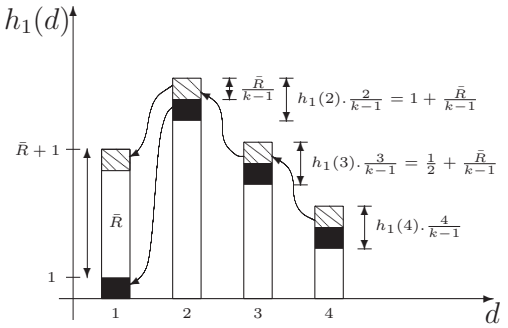

Figura 3.5: Distribuição Sóliton Robusta: segunda iteração.

Prova:

$$\begin{aligned} n &= k\beta \\ &= k\left(\sum_i \rho(i) + \tau(i)\right) \\ &= k + \sum_{i=1}^{\frac{k}{\bar{R}}-1} \frac{\bar{R}}{i} + \bar{R}\ln\frac{\bar{R}}{\delta} \\ &\leq k + \bar{R}\,\mathrm{H}(k/\bar{R}) + \bar{R}\ln\frac{\bar{R}}{\delta}. \end{aligned}$$

Para continuar com a prova do teorema, definimos a função $\mathrm{H}(k)$ como a série harmônica truncada $\mathrm{H}(k) = \sum_{j=1}^{k} \frac{1}{j}$. Usando a seguinte aproximação $\mathrm{H}(k) \approx \ln(k)$, a última expressão é equivalente a:

$$\begin{aligned} &\approx k + \bar{R}\ln\frac{k}{\bar{R}} + \bar{R}\ln\frac{\bar{R}}{\delta} \\ &= k + \sqrt{k}\ln^2\frac{k}{\delta} \end{aligned}$$

e, finalmente,

$$n = k + O\left(\sqrt{k}\ln^2(k/\delta)\right). \ \blacksquare$$

A função $O(\cdot)$ é somente uma notação utilizada para medir a complexidade computacional de um determinado algoritmo.

Teorema 3.3 *O grau médio de um símbolo codificado é* $\bar{D} = O(\ln(k/\delta))$.

Prova:

$$\begin{aligned}
\bar{D} &= \frac{\sum_i i\,(\rho(i)+\tau(i))}{\beta} \\
&\leq \sum_i i\,(\rho(i)+\tau(i)) \\
&= \sum_{i=2}^{k+1} \frac{1}{i-1} + \sum_{i=1}^{\frac{k}{\bar{R}}-1} \frac{\bar{R}}{k} + \ln\frac{\bar{R}}{\delta} \\
&\leq \mathrm{H}(k) + 1 + \ln\frac{\bar{R}}{\delta}
\end{aligned}$$

Como $\mathrm{H}(k) \approx \ln(k)$ vem,

$$\bar{D} = O\left(\ln(k/\delta)\right). \blacksquare$$

Na Figura 3.3 encontra-se ilustrado o desempenho da distribuição Sóliton Robusta com parâmetros $c = 0.03$ e $\delta = 0.1$. Encontra-se ilustrada a probabilidade de falha da decodificação, P_F, quando $(1+\epsilon)k$ símbolos de saída são recebidos. As simulações mais uma vez foram implementadas para $k = 1000$.