Análise assintótica e projeto de distribuições de graus para uma classe de códigos fonte

Humberto V. Beltrão Neto e Valdemar C. da Rocha Jr.

***Resumo*— Este artigo apresenta resultados da aplicação da técnica de análise de processos aleatórios baseados em árvores E-OU para avaliar o desempenho de códigos fonte do tipo Luby Transform (LT) em presença de erros. Apresenta-se ainda, uma técnica de projeto de distribuições de graus para códigos LT baseada em programação linear, em conjunto com a análise de desempenho das distribuições projetadas utilizando tal técnica.**

***Palavras-Chave*— códigos LT, análise assintótica, árvores E-OU, projeto de distribuições de graus.**

***Abstract*— This article presents results obtained via the analysis of random processes via AND-OR tree evaluation applied to evaluate the error performance of LT codes. It is also presented a technique for developing degree distributions for LT codes based on linear programming, together with an analysis of the performance of the distributions developed using such technique.**

***Keywords*— LT codes, asymptotic analysis, AND-OR trees, development of degree distributions.**

I. INTRODUÇÃO

Muitos problemas concernentes a sistemas de comunicação atuais envolvem usuários trocando informação a partir de um canal que pode corromper ou perder parte dessa informação ao longo da transmissão. Um exemplo importante deste cenário é a comunicação realizada por meio de redes de computadores, dentre as quais o maior e talvez mais notável exemplo é a Internet. A troca de informação em tais redes é feita através de pacotes, em que cada pacote pode conter toda ou parte da mensagem original [1]. Ao longo da transmissão, pacotes podem ser corrompidos devido ao ruído ou até mesmo perdidos devido a mecanismos de controle de tráfego, por exemplo.

A abordagem mais utilizada para lidar com a perda de pacotes nestes sistemas é requerer a retransmissão dos pacotes corrompidos ou perdidos, através de um procedimento conhecido como *automatic repeat request* (ARQ) [2]. Sistemas que fazem uso dessa abordagem empregam protocolos, como por exemplo o *transport control protocol* (TCP) [1], nos quais cada um dos receptores envolvidos confirmam a recepção dos pacotes, e, avisam no pacote de resposta qual o último pacote recebido corretamente. Caso não tenham recebido algum deles ou caso algum dos pacotes recebidos esteja corrompido, o transmissor retransmite tais pacotes. Existe, no entanto, uma vasta gama de situações em que tal estratégia revela-se altamente dispendiosa, requerendo um grande número de retransmissões para que os receptores sejam capazes de receber a informação original por completo. Um exemplo de situação em que protocolos de retransmissão não são adequados é o caso de transmissões do tipo *multicast*, na qual um transmissor envia mensagens a um determinado grupo de receptores. Uma alternativa para lidar com o problema da retransmissão é empregar códigos corretores de erros [3], mais precisamente códigos para canais com apagamentos [4], [5].

Humberto V. Beltrão Neto e Valdemar C. da Rocha Jr., Departamento de Eletrônica e Sistemas, Universidade Federal de Pernambuco, Recife, Brasil, E-mails: hvbneto@hotmail.com, vcr@ufpe.br.

Os códigos Luby Transform (LT) [6] podem ser utilizados em transmissões nas quais pacotes são perdidos ou corrompidos, visto que estes podem ser interpretados pelo decodificador como apagamentos [2], [7].

Tendo em vista que para códigos LT o número de símbolos codificados que podem ser gerados a partir dos dados originais é potencialmente ilimitado podendo ser ajustado em tempo real de acordo com a necessidade da aplicação, os mesmos são classificados como códigos sem-taxa (*rateless codes*) [4], [5], [6]. Dentre os códigos sem-taxa, podemos citar ainda os códigos Raptor [8], os códigos em tempo real [9] e os códigos Slepian-Wolf sem-taxa [10].

O objetivo deste artigo consiste em apresentar uma técnica de análise assintótica da probabilidade de falha na decodificação para códigos LT além de um meio para construir distribuições de graus as quais gerem códigos LT com baixa probabilidade de falha ao longo do processo de decodificação. São apresentados ainda, resultados da análise assintótica aplicada para avaliação do desempenho dos códigos LT e algumas distribuições de graus projetadas.

Este artigo encontra-se organizado em seis seções. Na *Seção II* encontra-se uma revisão sobre códigos LT. Na *Seção III* apresenta-se a técnica utilizada para a análise assintótica da probabilidade de falha na decodificação, aplicando a mesma ao códigos LT na *Seção IV*. Na *Seção V* é descrito um método para construir distribuições de graus para códigos LT usando programação linear. Por fim, na *Seção VI* são apresentadas algumas conclusões.

II. CÓDIGOS LT

Os códigos LT formam uma classe de códigos criada por Michael Luby [6], sendo estes a primeira implementação do conceito de fontes digitais (*digital fountains*) [11] com tempo de codificação e decodificação viável para sistemas que requeiram a formação de um grande número de símbolos codificados n a partir de um arquivo com k símbolos de entrada. Os símbolos de entrada são símbolos binários os quais podem ser formados *bytes* de tamanho arbitrário.

Os códigos LT são capazes de recuperar, com probabilidade no mínimo $(1-\delta)$, um conjunto de k símbolos de entrada a

partir de quaisquer $k + O(\sqrt{k}.\ln^2(k/\delta))$[1] símbolos codificados, necessitando para isso de uma média de $O(k.\ln(k/\delta))$ operações [6]. Os códigos LT são classificados como códigos sem-taxa (*rateless codes*), ou seja, o número de símbolos codificados que podem ser gerados a partir dos dados originais é potencialmente ilimitado[2]. Assim, não importa qual o modelo de perdas do canal em uso, o codificador simplesmente envia dados até um número suficiente de pacotes chegarem ao receptor. A estatística do canal apenas influencia no tempo necessário para que símbolos de saída, em número suficiente para a decodificação bem sucedida, sejam recebidos pelo decodificador. Como o decodificador pode, com grande probabilidade, recuperar uma mensagem formada por k símbolos de entrada a partir de quaisquer $(1+\epsilon)k$ símbolos codificados, em que $0 < \epsilon < 1$, os códigos LT são quase ótimos com respeito a qualquer canal com apagamentos, considerando que um código ótimo é aquele que 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 [12].

A. Codificação

Ao longo da codificação, cada símbolo codificado t_n é obtido a partir de um arquivo original formado pelos símbolos $s_1, s_2, s_3, ..., s_k$ como segue:

1) Aleatoriamente escolher o número de vizinhos (grau) d_n do símbolo codificado a partir de uma distribuição de probabilidades $\Omega(x)$.
2) Escolher, uniforme e aleatoriamente, d_n símbolos de entrada distintos, e atribuir a t_n o valor da adição *bit* a *bit*, módulo 2 desses d_n símbolos.

Fica claro a partir da descrição do processo de codificação que existem duas distribuições de probabilidade envolvidas. Uma para determinar o número de vizinhos de cada símbolo (grau) e outra, uniforme, para reger a escolha de quais símbolos de entrada irão compor o conjunto de vizinhos do símbolo codificado. A distribuição de probabilidade que rege o número de vizinhos de um símbolo codificado é comumente referida como distribuição de graus.

Freqüentemente, denotar-se-á uma distribuição de probabilidades pelo seu polinômio gerador. Assim, uma possível representação para a distribuição de graus é dada pelo polinômio $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$, em que Ω_i denota a probabilidade de que o grau i seja escolhido. Usando essa notação, o valor esperado da distribuição de graus é dado simplesmente por $\Omega'(1)$, em que $\Omega'(x)$ denota a derivada primeira de $\Omega(x)$ com respeito a x.

B. Decodificação

O processo de codificação define um grafo conectando os símbolos de saída aos símbolos de entrada. A seguinte notação será adotada: os símbolos codificados (t_n) formarão os nós de verificação e os símbolos de entrada (s_n) formarão os nós de mensagem. O algoritmo de decodificação de códigos LT resume-se ao seguinte:

1) Encontrar um nó de verificação t_n que esteja conectado a apenas um símbolo de entrada s_k. Caso não haja tal nó de verificação, a decodificação é abortada e o processo de decodificação não pode ser concluído, sendo necessário receber mais símbolos codificados antes de se fazer nova tentativa de decodificação.
   a) Fazer $s_k = t_n$,
   b) Somar s_k a todos os nós t'_n que estejam conectados a s_k,
   c) Remover todas as conexões que chegam ao símbolo de entrada s_k.
2) Repetir (1) até que todos os s_k estejam determinados.

Os processos de análise da probabilidade de falha de códigos LT baseiam-se na análise do grafo gerado pelo processo de codificação. A seção seguinte apresenta a técnica de análise de grafos a ser empregada no cálculo do desempenho de códigos LT.

III. A TÉCNICA DE ANÁLISE VIA AVALIAÇÃO DE ÁRVORES E-OU

Esta seção explica de forma concisa a técnica de análise via árvores E-OU introduzida em [13], tendo em vista que esta é a ferramenta de análise assintótica a ser utilizada neste artigo. A versão simplificada do problema de avaliação por meio de árvores E-OU consiste no seguinte. Seja T_l uma árvore binária com profundidade $2l$. Atribui-se a cada uma de suas folhas os valores 0 ou 1 de forma aleatória e independente. À raiz da árvore atribui-se profundidade 0, estando as folhas a uma profundidade $2l$. Cada nó com profundidade $0, 2, 4, ..., 2l-2$ é rotulado como um nó porta lógica "OU" (e ele calcula o "OU" booleano do valor de seus filhos), e cada nó com profundidade $1, 3, 5, ..., 2l-1$ é rotulado como um nó porta lógica "E" (e ele calcula o "E" booleano de seus filhos).

Cada nó irá escolher independentemente quantos filhos terá. Para os nós "OU", a escolha é feita seguindo uma distribuição de probabilidades $(\alpha_0, \alpha_1, ..., \alpha_A)$ em que α_i é a probabilidade do nó "OU" possuir i filhos. Já para os nós "E" a escolha é feita seguindo uma distribuição de probabilidades $(\nu_0, \nu_1, ..., \nu_B)$ em que ν_j é a probabilidade do nó "E" possuir j filhos (aos nós "OU" sem filhos será atribuído o valor 0 e aos nós "E" sem filhos será atribuído o valor lógico 1). Por fim, a cada folha serão associados os valores 0 ou 1 independentemente, com y_0 sendo a probabilidade de uma folha assumir o valor 0. Como os nós com profundidade $2l$ serão folhas, não terão filhos.

Tratando a árvore como um circuito booleano, o interesse da presente análise reside em calcular a probabilidade de que a raiz da árvore assuma o valor 0 (y_l). O seguinte lema exposto em [13] nos municia com um método para calcular y_l. A prova deste lema é bem direta, basta considerar que os nós "OU" com profundidade igual a 2 em T_l são as raízes de árvores E-OU independentes T_{l-1}. Desta forma, a probabilidade y_l pode

[1] Sejam $f(x)$ e $g(x)$ duas funções definidas em algum subconjunto dos números reais. Diz-se que $f(x)$ é $O(g(x))$ quando $x \to a$ se e somente se: $\exists\ \delta > 0, \exists\ M > 0$ tal que $|f(x)| \leq M|g(x)|$ para $|x - a| < \delta$.

[2] Na verdade, apesar do termo códigos sem-taxa ter sido largamente adotado na literatura, os códigos LT são códigos de taxa variável, sendo o número de símbolos codificados (n) variável de acordo com o número de pacotes necessários à decodificação bem sucedida.

ser calculada como função de y_{l-1}, a probabilidade de que a raiz de uma árvore E-OU T_{l-1} assuma o valor 0.

Lema 1 (Lema da árvore E-OU): A probabilidade y_l de que a raiz de uma árvore E-OU T_l assuma o valor 0 é $y_l = f(y_{l-1})$, em que y_{l-1} é a probabilidade de que a raiz de uma árvore E-OU T_{l-1} assuma o valor 0, e

$$f(x) = \alpha(1 - \nu(1-x)), \text{ em que}$$

$$\alpha(x) = \sum_{i=0}^{A} \alpha_i x^i \text{ e } \nu(x) = \sum_{i=0}^{B} \nu_i x^i.$$

IV. Análise assintótica de códigos LT

A ferramenta introduzida na seção anterior, tem sido amplamente utilizada no cálculo da probabilidade de falha na decodificação de alguns códigos sem-taxa quando o número de símbolos de entrada tende a infinito (análise assintótica). Abordagens que utilizam tal ferramenta podem ser encontradas em [8], [9] e [14]; aqui utilizar-se-á tal ferramenta na análise de códigos LT. Para iniciar o estudo, alguns conceitos devem ser relembrados. Seja $\mathcal{G}$ um grafo bipartite com k nós do lado esquerdo, n nós no lado direito e $\mathcal{E}$ ramos no total entre os lados esquerdo e direito. Será associado um símbolo de entrada a cada nó do lado esquerdo (nós de símbolo) e um símbolo de saída a cada nó do lado direito (nós de verificação). Seguindo a abordagem exposta em [8] e [9] pode-se modelar o processo de decodificação da seguinte maneira: a cada iteração do algoritmo de decodificação, mensagens (0 ou 1) são enviadas através dos ramos dos nós de verificação para os nós de símbolo e vice-versa. Um nó de símbolo envia 0 para para um nó de verificação adjacente se e somente se seu valor não foi recuperado ainda; da mesma forma, um nó de verificação envia 0 para um nó de símbolo adjacente, se e somente se, ele não puder recuperar tal nó de símbolo. Não é difícil verificar que os nós de símbolo podem ser vistos como nós "OU" e de forma análoga os nós de verificação podem ser vistos como nós "E".

A. Distribuição de graus dos nós de símbolo

O primeiro passo para aplicação da técnica de análise baseada em árvores E-OU consiste na demonstração de que a distribuição de graus de um nó de símbolo segue uma distribuição de Poisson com média constante. A análise aqui segue a abordagem mostrada em [9]. A demonstração inicia-se com a descrição do processo de formação dos nós de verificação. Considere que serão formados βk nós de verificação, com $\beta > 1$. Em primeiro lugar, cada um dos βk nós de verificação escolhe seu grau. Feito isto, os nós de símbolo são conectados aleatória e uniformemente através de ramos aos nós de verificação. A primeira etapa determina o número total de ramos $\mathcal{E}$, o qual é calculado (tratando-se aqui de valores esperados) como sendo $\mathcal{E} = \beta\gamma k$, em que $\gamma = \sum_{i=1}^{k} i\Omega_i$ é o grau médio dos nós de verificação. Uma vez determinado o número de ramos, a segunda etapa pode ser encarada como o lançamento de $\beta\gamma k$ bolas (os ramos) em n urnas (os nós de verificação). Assim, a probabilidade λ_d de um nó de símbolo ter grau igual a d é dada por:

$$\lambda_d = \binom{\beta\gamma k}{d} p^d (1-p)^{\beta\gamma k - d}, \tag{1}$$

em que $p = p(k) = \frac{1}{k}$. Assintoticamente ($k \to \infty$) pode-se aproximar a distribuição de probabilidade (1) por uma distribuição de Poisson se as seguintes condições forem satisfeitas [15]:

1) $p(k) = o(1)$[3].
2) $\beta\gamma kp$ é uma constante.

Satisfazendo essas condições, λ_d pode ser aproximada por:

$$\lambda_d = \frac{e^{-\gamma\beta}(\gamma\beta)^d}{d!}.$$

No que segue será admitido que as condições 1 e 2 são satisfeitas. A primeira delas é claramente satisfeita visto que $p(k) = \frac{1}{k}$. Já a condição 2 será atingida se $\beta\gamma$ for uma constante, o que é facilmente conseguido fazendo-se β (o *overhead*) e γ (o grau médio dos nós de símbolo) constantes. Assim, assintoticamente, ambos os nós da esquerda e da direita possuem graus limitados por constantes (para os nós da direita tal limite advém das regras de codificação).

B. Cálculo da probabilidade de falha na decodificação

Feitas as considerações preliminares, a probabilidade de um nó de símbolo ser recuperado pode ser calculada utilizando a análise via árvores E-OU em um subgrafo $\mathcal{G}_l$ construído da seguinte forma:

1) Escolha um ramo (v, w) de $\mathcal{G}$ aleatória e uniformemente. O nó v será a raiz de $\mathcal{G}_l$.
2) Remova (v, w) de $\mathcal{G}$.
3) $\mathcal{G}_l$ consistirá de todos os vértices de $\mathcal{G}$ que são alcançáveis percorrendo no máximo $2l$ ramos a partir de v e todos os ramos de $\mathcal{G}$ que conectam quaisquer pares dos referidos vértices.

Pode-se mostrar [13] que o subgrafo $\mathcal{G}_l$ é uma árvore quando k tende a infinito. Isso porque, conforme dito anteriormente, ambos os nós da esquerda e da direita possuem graus limitados por constantes. Assim, $\mathcal{G}_l$ não possui mais do que um número constante de vértices, logo, a probabilidade que $\mathcal{G}_l$ não seja uma árvore, ou seja, possua um ciclo, tende a 0 quando k tende a ∞. Desta forma, podemos usar o lema da árvore E-OU para calcular a probabilidade de falha na decodificação de um símbolo ao fim de l iterações, que nada mais é que a probabilidade y_l que a raiz de $\mathcal{G}_l$ assuma o valor 0; tal probabilidade será denotada por SER do inglês *symbol error rate*. Usando a abordagem baseada em árvores E-OU, tem-se que ν_i é a probabilidade de que o ramo selecionado aleatoriamente na construção de $\mathcal{G}_l$ esteja ligado a um nó de verificação com $i+1$ filhos. Esta é a probabilidade de que o ramo esteja ligado a um nó de verificação de grau $(i+1)$. Logo:

$$\nu_i = \frac{(i+1)\Omega_{i+1}}{\Omega'(1)},$$

[3]Diz-se que uma função $f(x)$ é $o(g(x))$ se $\lim_{x\to\infty} \left| \frac{f(x)}{g(x)} \right| = 0$.

o que implica em $\nu(x) = \frac{\Omega'(x)}{\Omega'(1)}$. Analogamente, α_i é a probabilidade de que o nó de símbolo conectado ao ramo selecionado aleatoriamente na construção de $\mathcal{G}_l$ tenha grau $i+1$, o que pode ser facilmente calculado como sendo:

$$\alpha_i = \frac{(i+1)\lambda_{i+1}}{\beta\gamma},$$

o que, após substituições adequadas, resulta em $\alpha(x) = e^{\beta\gamma(x-1)}$. Resumindo tais resultados tem-se:

Teorema 1: Considere um código LT com parâmetros $\Omega(x)$, k, e $\beta = (1+\epsilon)$. Seja y_l a probabilidade de que um nó de variável não seja recuperado após l iterações. Tem-se então:

$$y_0 = 1,$$

e

$$y_l = \alpha(1 - \nu(1 - y_{l-1})), \quad l \geq 1,$$

em que

$$\nu(x) = \frac{\Omega'(x)}{\Omega'(1)} \quad e \quad \alpha(x) = e^{\beta\gamma(x-1)}.$$

Desta forma, pode-se calcular assintoticamente a probabilidade de falha na decodificação de um código LT, com distribuição de graus $\Omega(x)$. A probabilidade de falha na decodificação de códigos LT pode ser facilmente obtida a partir de y_l, basta perceber que o processo de decodificação falha se ao menos um símbolo de entrada não for recuperado ao fim da decodificação. A Figura 1 ilustra o uso da técnica de análise assintótica para códigos LT para 20, 40, 80 e 100 iterações, utilizando a seguinte distribuição disponível em [8].

$$\Omega_1(x) = 0.007969x + 0.493570x^2 + 0.166220x^3 + 0.072646x^4$$
$$+0.082558x^5 + 0.056058x^8 + 0.037229x^9 + 0.055590x^{19}$$
$$+0.025023x^{65} + 0.0003135x^{66}.$$

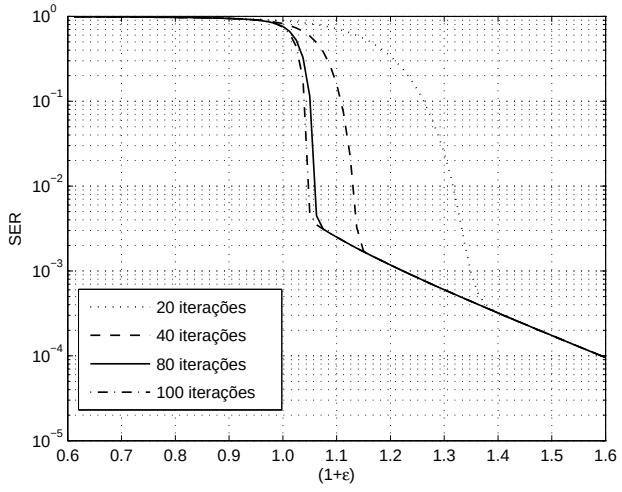

Fig. 1. Curva da taxa de erro de símbolo em função do *overhead* para a distribuição $\Omega_1(x)$ obtida através da análise assintótica baseada em árvores E-OU. Cada uma das curvas foi obtida variando o número de iterações da análise.

V. Projeto de distribuições de graus usando programação linear

A distribuição comumente utilizada com os códigos LT é a distribuição Sóliton Robusta [6]. A análise assintótica da distribuição Sóliton Robusta encontra-se ilustrada na Figura 2. Estritamente, a análise baseada em árvores E-OU não poderia ser utilizada para a distribuição Sóliton Robusta visto que, para esta distribuição, o grau máximo dos símbolos de saída é k, ou seja, depende do número de símbolos de entrada. No entanto, como a probabilidade de ocorrência de graus altos tende a zero quando $k \to \infty$, a análise baseada em árvores E-OU gera boas aproximações.

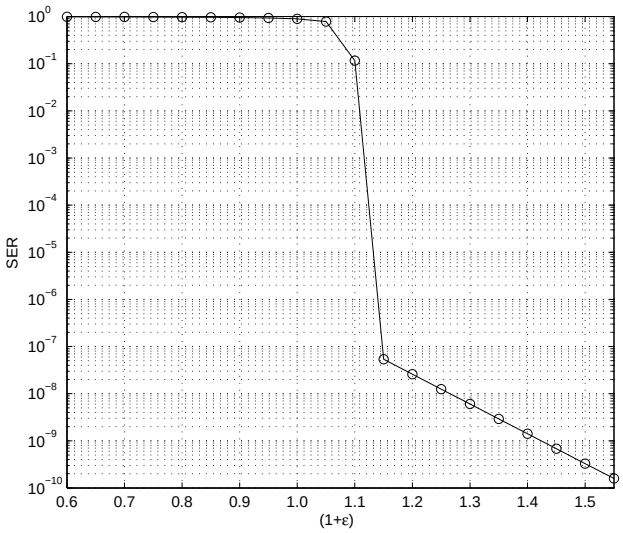

Fig. 2. Taxa de erro de símbolo em função do *overhead* para a distribuição Sóliton Robusta. Curva obtida utilizando a análise assintótica baseada em árvores E-OU.

Embora a distribuição Sóliton Robusta apresente um bom desempenho, não existe qualquer controle sobre o grau médio dos símbolos codificados e sobre o grau máximo permitido para um dado símbolo de saída. Visto que, à medida que o grau médio dos símbolos codificados cresce, aumenta também a complexidade dos processos de codificação e decodificação, é interessante desenvolver um método de construção de distribuições que possibilite o controle sobre o grau médio dos símbolos de saída. O método descrito nessa seção visa satisfazer tal requisito, ao mesmo tempo, mantendo constante o grau máximo da distribuição de graus.

Em [16] os autores sugerem um processo baseado em programação linear para desenvolver distribuições de graus que proporcionem uma alta probabilidade de sucesso na decodificação dos códigos Tornado. Posteriormente, Shokrollahi em [8] fez uma abordagem do referido processo para o desenvolvimento de códigos Raptor [8]; explanar tal abordagem e aplicá-la aos códigos LT é o objetivo dessa seção.

O projeto usando programação linear baseia-se na análise de processos aleatórios através de árvores E-OU. Seja p_l a probabilidade de um ramo no grafo de decodificação enviar o valor 1 de um nó de saída para um nó de entrada na l-ésima iteração. Usando o lema da árvore E-OU obtém-se

$$p_{l+1} = \nu(1 - \alpha(1 - p_l)), \quad l \geq 1,$$

em que

$$\nu(x) = \frac{\Omega'(x)}{\Omega'(1)} \quad e \quad \alpha(x) = e^{\beta\gamma(x-1)},$$

sendo $\nu(x)$ e $\alpha(x)$ as distribuições de graus dos nós de saída e entrada, respectivamente. Seja u_l a probabilidade de um símbolo de entrada ser recuperado na iteração l. Convém lembrar que um símbolo de entrada é recuperado, se e somente se, estiver ligado a um ramo que carrega o valor 1. A probabilidade de um símbolo de entrada de grau d ser recuperado pode ser facilmente calculada como sendo $1-(1-p_l)^d$. Como $(1 - p_l)^d = \alpha(1 - p_l)$, a probabilidade de um símbolo de entrada ser recuperado na l-ésima iteração do algoritmo é $u_l = 1 - \alpha(1 - p_l) = 1 - e^{-\beta\gamma p_l}$. Logo, escrevendo em forma de recursão em u tem-se

$$u_{l+1} = 1 - e^{-\beta\gamma\nu(u_l)}.$$

A recursão acima mostra que, se uma fração esperada de x símbolos de entrada foi recuperada em alguma iteração, na próxima iteração essa fração aumenta para $1 - e^{-\beta\gamma\nu(x)}$. Logo, a fração esperada de símbolos de grau 1 ainda não processados[4] será $1 - x - e^{-\beta\gamma\nu(x)}$. Sendo o número de símbolos de saída igual a $\beta k = (1 + \epsilon)k$, então $\beta\gamma\nu(x) = (1 + \epsilon)\Omega'(x)$ sendo a fração esperada de símbolos no *ripple* dada por

$$1 - x - e^{-\Omega'(x)(1+\epsilon)}.$$

Dito isto, é preciso projetar uma distribuição de graus para os símbolos de saída de forma a garantir que uma grande fração dos k símbolos de entrada sejam recuperados. Assume-se então, para fins de projeto, que a inserção e retirada de elementos do *ripple* é feita de forma independente com probabilidade $1/2$, então o *ripple* deve ser maior por um fator c da raiz quadrada do número de símbolos de entrada ainda não recuperados, ou seja, maior que $c\sqrt{(1-x)k}$. Esse valor surge a partir da aplicação de uma cota de Chernoff [17] para a soma de variáveis aleatórias i.i.d. [5] X, ou seja, $X = \sum_{i=1}^{N} X_i$ sendo $Pr[X_i = 1] = 1/2$ e $Pr[X_i = -1] = 1/2$. Seja R o valor esperado do *ripple*, o processo de entrada e saída de símbolos no mesmo pode ser visto como um passeio aleatório [15] de tamanho N, em que N é igual ao número de símbolos ainda não recuperados. Utilizando a seguinte cota de Chernoff [17],

$$Pr[X \leq a] \leq e^{\frac{-a^2}{2N}}$$

$$= Pr[X \leq -R] \leq e^{\frac{-R^2}{2(1-x)k}} \leq c',$$

em que c' é uma constante que determina uma cota superior para a máxima probabilidade de falha admitida e após algumas manipulações, chega-se à condição a seguir para o tamanho do *ripple*

[4]O conjunto de símbolos de grau 1 ainda não processados será referido como *ripple*. Diz-se que um símbolo é processado quando o mesmo é retirado da vizinhança dos símbolos codificados que o tenham como vizinho.

[5]independentes e identicamente distribuídas.

TABELA I

DISTRIBUIÇÕES DE GRAUS PARA VÁRIOS VALORES DE K.

k	1000	5000	10000	50000
Ω_1	0,069168	0,060131	0,056290	0,049966
Ω_2	0,385100	0,395150	0,398850	0,407210
Ω_3	0,209760	0,210130	0,209710	0,210000
Ω_4	0,095456	0,093559	0,092477	0,090704
Ω_5	0,057914	0,056956	0,056532	0,055473
Ω_8	0,042095	0,042853	0,043572	0,044244
Ω_9	0,041593	0,042165	0,042853	0,043530
Ω_{18}	0,028249	0,024327	0,022371	0,019771
Ω_{19}	0,027125	0,023004	0,020882	0,018070
Ω_{20}	0,026175	0,021916	0,019662	0,016678
Ω_{65}	0,003281	0,006324	0,008120	0,010137
Ω_{66}	0,005645	0,009709	0,012002	0,014481
Ω_{67}	0,008442	0,013771	0,016678	0,019735
ϵ	0.05	0.05	0.05	0.05
γ	5.55	6.15	6.50	6.85
γ_{SR}	10.35	13.28	14.58	17.57

$R \geq c\sqrt{(1-x)k}$ para um dado $c = \sqrt{2\ln(1/c')}$.

Usando essa condição o problema do projeto da distribuição dos símbolos de saída torna-se o seguinte: dados ϵ, δ e o número de símbolos de entrada k, encontre uma distribuição de graus tal que, para $x \in [0, 1 - \delta]$,

$$k(1 - x - e^{-\Omega'(x)(1+\epsilon)}) \geq c\sqrt{(1-x)k},$$

em que $k(1 - x - e^{-\Omega'(x)(1+\epsilon)})$ é o tamanho do *ripple*. Reescrevendo a equação em termos de $\Omega'(x)$ tem-se,

$$\Omega'(x) \geq \frac{-\ln\left(1 - x - c\sqrt{\frac{1-x}{k}}\right)}{1+\epsilon}, \qquad (2)$$

para $x \in [0, 1-\delta]$. Note que para a desigualdade ter solução, δ deve ser maior que $c/\sqrt{k}$. Esse problema pode ser facilmente resolvido através de uma abordagem utilizando programação linear [18]. O primeiro passo consiste em escolher um conjunto de inteiros M o qual deseja-se que contenha os possíveis graus dos símbolos de saída. Em seguida, deve-se discretizar o intervalo $[0, 1 - \delta]$ e fazer com que a desigualdade (2) seja obedecida em tais pontos, obtém-se assim um conjunto de desigualdades lineares em que as incógnitas são os coeficientes de $\Omega(x)$[6]. Por fim, escolhe-se minimizar a função objetivo $\Omega'(1)$ a fim de obter uma distribuição de graus com o menor valor médio possível.

Na Tabela I encontram-se algumas distribuições obtidas a partir de simulações implementadas fazendo uso da abordagem acima exposta. Nela $(1 + \epsilon)$ é o *overhead*, γ é o grau médio dos símbolos de saída e γ_{SR} é o grau médio dos símbolos de saída da distribuição Sóliton Robusta para os mesmos valores de k. Em todas as distribuições projetadas usando programação linear fez-se $\delta = 0.01$. O conjunto M escolhido nas presentes simulações foi o mesmo escolhido em [8].

A Figura 3 ilustra o desempenho das distribuições projetadas usando programação linear. É importante citar que as otimizações foram feitas incluindo as restrições de que para todo $i > 0$, $\Omega_i \geq 0$. Uma outra restrição foi o grau médio das distribuições, essa restrição é importante pois, apesar de

[6]$\Omega'(1)$ é função dos coeficientes Ω_i onde $i \in M$.

estar-se minimizando a função objetivo $\Omega'(1)$, graus médios muito baixos não resultam em boas distribuições no que diz respeito à probabilidade de falha na decodificação.

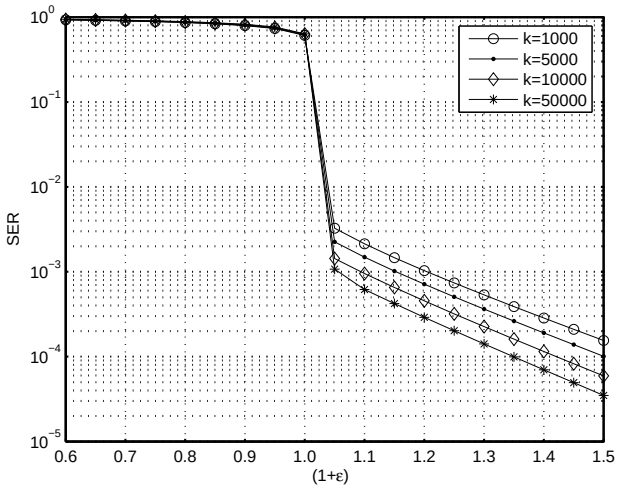

Fig. 3. Desempenho das distribuições projetadas por meio da abordagem baseada em programação linear expostas na Tabela I

É possível observar na Figura 3 que, quanto maior o grau médio da distribuição, menor a probabilidade de erro para $(1+\epsilon)k > 1$. Essa melhoria no desempenho, no entanto, implica uma maior complexidade computacional, visto que mais operações terão quer ser realizadas, em média, para codificar e decodificar um símbolo. Cada uma das probabilidades de erro foi calculada com 100 iterações. Faz-se mister lembrar que probabilidade de erro aqui denota a probabilidade de um símbolo de entrada não ser recuperado após $(1+\epsilon)k$ símbolos de saída terem sido recebidos, acarretando assim, falha no processo de decodificação.

Analisando os gráficos de desempenho pode-se questionar porque usar tal abordagem se a distribuição Sóliton Robusta, apresenta menores probabilidades de falha quando $(1+\epsilon) > 1.0$. A resposta é simples; usando programação linear pode-se controlar o grau médio dos símbolos de saída, algo impossível se a distribuição Sóliton Robusta for utilizada. Tal controle, é extremamente importante quando se quer trocar uma baixa probabilidade de erro, por uma baixa complexidade computacional nos processos de codificação e decodificação, isso porque quanto maior for o grau médio de uma distribuição, maior será o número de operações a serem realizadas a fim de recuperar a mensagem original. Observamos na Tabela I que o melhor desempenho da distribuição Sóliton Robusta ocorre às custas de um alto grau médio, isso acarreta uma maior complexidade computacional aos processos de codificação e decodificação.

VI. Conclusões

Neste artigo foi analisado o desempenho assintótico dos códigos LT, mediante a utilização da análise de processos aleatórios via avaliação de árvores E-OU. Foi mostrado que tais códigos são capazes de atingir uma baixa probabilidade de falha no processo de decodificação à medida que o número de símbolos de saída recebidos supera o número de símbolos de entrada. Foi apresentado ainda, um método para a construção de distribuições de graus as quais permitem o controle dos graus médio e máximo dos símbolos codificados. A aplicação da análise assintótica às distribuições geradas pelo método exposto, mostra que as mesmas geram códigos LT com bom desempenho.

Agradecimentos

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro. Este trabalho foi parcialmente financiado pelo CNPq, Projeto 305226/2003-7.

Referências

[1] A. S. Tanenbaum, *Computer Networks, 4a ed*. Prentice Hall, Agosto 2002.

[2] J. C. Moreira and P. G. Farrell, *Essentials of Error-Control Coding*. John Wiley & Sons, Setembro 2006.

[3] S. Lin and D. J. Costello, *Error-Control Coding*. 2a ed. Prentice Hall, Abril 2004.

[4] M. Luby, M. Mitzenmacher and J. W. Byers, "A digital fountain approach to asynchronous reliable multicast," *IEEE J. on Selected Areas in Communications, Special Issue on Network Support for Multicast Communications*, 2002.

[5] J. W. Byers, M. Luby, M. Mitzenmacher and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *In: Proc. ACM SIGCOMM*, p. 56-67, Agosto 1998.

[6] M. Luby, "LT codes", *In: Procedings of the ACM Symposium on Foundations of Computer Science*, 2002.

[7] D. J. C. Mackay, "Fountain codes", *IEE Proc.-Communication*, v. 152, n. 6, p. 1062-1068, Dezembro 2005.

[8] A. Shokrollahi, "Raptor codes", *IEEE International Symposium on Information Theory (ISIT) 2004*, Junho 2004.

[9] P. Maymounkov, "Online codes", *NYU, Relatório Técnico TR2003-883*, Novembro 2002.

[10] A. W. Eckford and W. Yu, "Rateless Slepian Wolf codes", *In: Proc. Asilomar Conference on Signals, Systems and Computers*,p. 1757-1761, Outubro - Novembro 2005.

[11] M. Mitzenmacher,"Digital fountains: A survey and look forward", *IEEE Information Theory Workshop*, p. 24-29, Outubro 2004.

[12] P. Maymounkov and D. Mazieres, "Rateless codes and big downloads", *In: Proc. of the 2nd Int. Workshop Peer-to-Peer Systems (IPTPS)*, Fevereiro 2003.

[13] M. Luby, M. Mitzenmacher and A. Shokrollahi, "Analysis of random processes via and-or tree evaluation", *In: Symposium on Discrete Algorithms*, 1998.

[14] N. Rahnavard, N. Badri, R. Vellambi and F. Fekri, "Rateless codes with unequal error protection", não publicado.

[15] S. M. Ross, *Introduction to Probability Models*, 8a ed. Academic Press, 2003.

[16] M. Luby, M. Mitzenmacher and A. Shokrollahi, "Efficient erasure correcting codes", *IEEE Transactions on Information Theory*, v. 47, p. 569-584, Fevereiro 2001.

[17] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2003.

[18] A. G. Glicksman, *An introduction to linear programming and the theory of games*. John Wiley & Sons, 1963.