ANÁLISE DA PARTIÇÃO ÓTIMA DO LZ78

Marcelo S. Pinho

UNESP - Guaratinguetá
mpinho@feg.unesp.br

Weiler A. Finamore

CETUC - PUC-Rio
weiler@cetuc.puc-rio.br

Resumo

O Lempel-Ziv'78 ($LZ78$) é um codificador universal que tem inspirado vários esquemas práticos de compressão de dados. Embora ele seja assintoticamente ótimo, i.e., sua taxa de compressão converge para a entropia da fonte, o procedimento de partição da seqüência de entrada não é ótimo. Encontrar a partição ótima de uma seqüência de entrada para o $LZ78$ é um problema NP-completo e por esta razão pouco se sabe sobre o ganho que uma partição mais eficiente produziria no $LZ78$. Neste trabalho, a partição ótima do $LZ78$ é analisada. É apresentado um resultado que mostra que o ganho da partição ótima não é significativo, no sentido de que o número de segmentos produzidos por esta partição tem o mesmo comportamento assintótico do número de segmentos produzidos pela partição original do $LZ78$. Este trabalho ainda apresenta um algoritmo que calcula um limitante inferior para o número de segmentos gerados pela partição ótima, para uma seqüência de entrada arbitrária. Este algoritmo é utilizado para medir o desempenho de dois métodos de partição diferentes.

Palavras-Chave: codificação de fonte, codificação universal, compressão de dados, codificadores via recorrência de padrões.

1. INTRODUÇÃO

O algoritmo $LZ78$ é um compressor de dados baseado na técnica de casamento de padrões recorrentes. Ele foi proposto em [1] para resolver o problema da codificação universal. No artigo original, foi mostrado que dada uma fonte ergódica arbitrária, a taxa de compressão obtida pelo $LZ78$, quando aplicado em uma seqüência produzida pela fonte, converge com probabilidade 1 para a entropia da fonte. Devido a sua simplicidade e a sua eficiência na prática, o $LZ78$ se tornou bastante popular, dando origem a diversas variantes do algoritmo original. Um dos algoritmos mais utilizados na prática é conhecido como LZW e foi proposto em [2]. O LZW possui um desempenho melhor que o $LZ78$, com um ganho de aproximadamente 10% na prática.

Um codificador é dito *universal* para uma determinada classe de fontes se a taxa de compressão obtida por este codificador, quando aplicado em uma seqüência emitida por qualquer fonte desta classe, converge para a entropia desta fonte. Esta convergência ocorre quando o comprimento da seqüência de entrada cresce indefinidamente. No entanto, as seqüências de dados que existem na prática são limitadas. Por esta razão, é fundamental analisar o desempenho dos codificadores universais quando eles são aplicados em seqüências de comprimento finito. Esta análise pode ser realizada através da redundância de um codificador universal. Seja Ψ uma classe de fontes e $\mathcal{S} \in \Psi$ uma fonte cuja saída é uma seqüência aleatótria $X_1^n = X_1 \dots X_n$, onde X_i pode assumir valores em um conjunto $\mathcal{A}$, denominado alfabeto da fonte. Seja C um codificador universal para a classe Ψ. A redundância de C relativa à fonte $\mathcal{S}$ é definida por

$$R_C^n(\mathcal{S}) = \frac{E[|C(X_1^n)|] - H(X_1^n)}{n}, \qquad (1)$$

onde $|C(X_1^n)|$ representa o comprimento da palavra código $C(X_1^n)$ e $E[|C(X_1^n)|]$ representa o seu valor esperado. A entropia de X_1^n é representada por $H(X_1^n)$. A redundância de C relativa à classe Ψ é definida por

$$R_C^n(\Psi) = \max_{\mathcal{S}\in\Psi}\{R_C^n(\mathcal{S})\} \qquad (2)$$

Um importante resultado da teoria da codificação de fonte diz que se Ψ é a classe de fontes sem memória, então a menor redundância que um codificador universal pode atingir é $R_C^n(\Psi) = O(\frac{logn}{n})$ [3]. Este mesmo resultado vale se Ψ é a classe de fontes markovianas ou a classe de fontes com número de estados finito (FSM) [3, 4, 5, 6]. Recentemente, foi mostrado que a redundância do $LZ78$ é igual a $O(\frac{1}{logn})$, quando Ψ é a classe de fontes sem memória, a classe de fontes markovianas, ou a classe de fontes FSM [7, 8, 9].

Embora o $LZ78$ não seja ótimo para seqüências finitas, emitidas por fontes com memória finita, ele é amplamente utilizado na prática. Na verdade, os codificadores

Este trabalho foi desenvolvido com o apoio do CNPq

que atingem a redundância ótima possuem alta complexidade computacional e sua utilização em diversas aplicações é inviável.

Seja x_1^n uma realização de X_1^n. Para codificar a seqüência x_1^n, o $LZ78$ particiona esta seqüência em segmentos s_1, $\ldots$, s_m ($x_1^n = s_1 \ldots s_m$) e depois codifica cada segmento s_i separadamente. A partição e a codificação são realizadas com base em um dicionário adaptativo que depende dos segmentos que estão sendo criados. Desta forma, o desempenho do codificador depende diretamente da partição. No entanto, é possível mostrar que a partição realizada pelo $LZ78$ não é ótima [10, 11, 12]. Além disso, é possível mostrar que o problema de encontrar a partição ótima de uma seqüência arbitrária x_1^n é NP-completo. Este mesmo resultado também vale para a versão LZW [11].

Em [10] é apresentado um procedimento para encontrar a partição ótima de uma seqüência, quando o dicionário é fixo. Embora este não seja o caso do $LZ78$ e nem do LZW, este procedimento pode ser utilizado para melhorar o desempenho destes codificadores. Em [12], é apresentado uma versão do LZW que utiliza esta técnica e produz um ganho em torno de 5% sobre o LZW. A partição realizada pelo codificador proposto em [12], conhecido como mm-$LZ78$, está entre as melhores da literatura para o LZW.

Como o problema da partição ótima do $LZ78$ é NP-completo, o ganho que pode ser obtido nesta etapa do codificador é desconhecido. Neste trabalho, este problema é analisado. É apresentado um teorema que mostra que a partição ótima não altera o comportamento assintótico do número de segmentos da partição. Embora isso não seja suficiente para garantir que o comportamento assintótico da redundância do codificador não se altera, este resultado mostra que o ganho que pode ser obtido é pequeno. Este trabalho ainda apresenta um algoritmo que calcula um limitante inferior para o número de segmentos produzidos pela partição ótima do $LZ78$ (que com uma simples modificação pode ser aplicado para o LZW). Este algoritmo é utilizado para medir o desempenho da partição original do LZW e o desempenho da partição do mm-$LZ78$.

A seção 2 apresenta o conceito de partição genérica do $LZ78$, indicando a partição original do $LZ78$ e a partição ótima. O algoritmo prático que calcula um limitante inferior para o número de frases da partição ótima do $LZ78$ é proposto na seção 3. A seção 4 é dedicada aos resultados obtidos. A conclusão é apresentada na seção 5, fechando o trabalho.

2. PARTIÇÃO ÓTIMA DO LZ78

Para definir o que é a partição ótima do $LZ78$, é preciso introduzir o conceito de uma partição genérica para o $LZ78$. A idéia é estabelecer as propriedades que uma partição deve satisfazer para que possa ser utilizada em conjunto com o codificador $LZ78$. Com base nestas propriedades, é possível definir uma partição genérica.

Conforme descrito anteriormente, o $LZ78$ particiona a seqüência de entrada x_1^n em segmentos $s_1, \ldots, s_m$ e codifica cada segmento com base em um dicionário adaptativo. Na verdade, cada segmento s_i é tal que $s_i = s_j v$, onde $j < i$ e $v \in \mathcal{A}$, i.e., o segmento s_i é uma cópia de um segmento anterior s_j concatenado a um símbolo v do alfabeto da fonte $\mathcal{A}$. Sendo assim, o segmento s_i pode ser codificado através de j e v.

2.1. Partição Genérica do LZ78

Seja $s_1 = x_1^{n_1}$, $s_2 = x_{1+n_1}^{n_2}$, $\ldots$, $s_m = x_{1+n_{m-1}}^{n}$ uma partição da seqüência x_1^n. Seja $\pi(x_j^n|D)$ o maior prefixo de x_j^n que pertence ao conjunto D. Seja $\{D_i\}$ a seqüência de dicionários obtidos pelas seguintes regras.

1. $D_1 = \mathcal{A}$,
2. $\forall\ i = 1, \ldots, m$, $D_{1+i} = D_i \cup \{\pi(x_{1+n_{i-1}}^n|D_i)v : v \in \mathcal{A}\}$.

A partição $s_1, \ldots, s_m$ de x_1^n é dita ser uma partição genérica do $LZ78$ se $s_i \in D_i$, $i = 1, \ldots, m$.

2.2. Partição Original do LZ78

O codificador $LZ78$ comprime a seqüência de entrada x_1^n, particionando a seqüência em segmentos $s_1, \ldots, s_{m(LZ78)}$, $s_1 = x_1^{n_1}$, $s_2 = x_{1+n_1}^{n_2}$, $\ldots$, $s_{m_{LZ78}} = x_{1+n_{m_{LZ78}-1}}^{n}$, tal que s_i é o maior prefixo de $x_{1+n_{i-1}}^n$ que pertence ao dicionário $D_i = \{s_j v : j < i, v \in \mathcal{A}\}$. Após a partição, o $LZ78$ codifica o segmento s_i utilizando um código que mapeia D_i em seqüências de bits de comprimento fixo. Desta forma, o comprimento da palavra código de um segmento é dado por $\lceil \log_2 |D_i| \rceil$ e o comprimento da palvara código da seqüência de entrada é dada por

$$|LZ78(x_1^n)| = \sum_{i=1}^{m_{LZ78}} \lceil \log_2 |D_i| \rceil. \qquad (3)$$

O $LZ78$ é um algoritmo que busca minimizar a razão $\frac{|C_i(s_i)|}{|s_i|}$ a cada passo, escolhendo s_i como o maior prefixo em D_i (como no passo i, $|D_i|$ é constante, então $|C_i(s_i)|$ é constante e portanto minimizar a razão $\frac{|C_i(s_i)|}{|s_i|}$ é equivalente a maximizar $|s_i|$). No entanto, é fácil ver que este é um procedimento que busca uma minimização local. Na verdade, o objetivo do codificador deveria ser minimizar a razão $\frac{|LZ78(x_1^n)|}{n}$. Conforme ressaltado em [10, 11, 12], esta minimização local não atinge o mínimo global.

2.3. Partição Ótima do LZ78

A melhor partição genérica do $LZ78$ é aquela que minimiza a razão $\frac{|LZ(x_1^n)|}{n}$. Através da equação (3) é possível observar que este problema é equivalente ao problema de encontrar a partição genérica que produz o menor número de segmentos. De fato, seguindo as regras estabelecidas na seção 2.1 para o dicionário $|D_i|$, é possível observar que $|D_i| = i|\mathcal{A}|$. Portanto, a equação (3) pode ser reescrita da forma

$$|LZ78(x_1^n)| = \sum_{i=1}^{m} \lceil \log_2 i|\mathcal{A}| \rceil,$$

e isso mostra que minimizar $\frac{|LZ(x_1^n)|}{n}$ é equivalente a minimizar o número de segmentos.

Sendo assim, a partição ótima do $LZ78$ pode ser definida facilmente. Uma partição genérica do $LZ78$ é ótima se não existe outra partição genérica com um número menor de segmentos. É importante ressaltar que a partição ótima pode não ser única.

Embora seja fácil definir a partição ótima do $LZ78$, é importante ressaltar que a tarefa de encontrar uma partição ótima não é fácil. De fato, conforme mencionado na seção 1, este é um problema NP-completo, vide [11].

3. ALGORITMO PRÁTICO PARA O CÁLCULO DE UM LIMITANTE INFERIOR PARA A PARTIÇÃO ÓTIMA

A partir da definição da partição genérica do $LZ78$, é possível observar que existem dois obstáculos intrínsicos ao algoritmo $LZ78$ que dificultam a procura do ponto ótimo. O primeiro obstáculo está relacionado à estrutura do dicionário. O segundo é devido ao procedimento de atualização do dicionário, que para uma dada seqüência de entrada x_1^n fixa, pode produzir dicionários diferentes (pois eles dependem da partição). Para encontrar um limitante inferior para o número de segmentos da partição ótima do $LZ78$, este trabalho apresenta um algoritmo de partição que retira estes obstáculos. Retirando os obstáculos, este algoritmo faz com que a minimização global possa ser atingida através de minimizações locais. No entanto, esta modificação faz com que a partição não seja mais uma partição genérica do $LZ78$.

Embora esta partição não seja uma partição genérica do $LZ78$, ela pode ser utilizada para medir o desempenho de partições do $LZ78$, pois produz um número de segmentos menor que o número de segmentos da partição ótima. Esta partição será chamada de partição-$LZ78^*$.

Na partição do codificador $LZ78$, a estrutura do dicionário permite que um par de frases (s_i, s_{1+i}), obtido por minimizações locais (i.e., o par é composto pelo prefixo mais longo de $x_{1+n_{i-1}}^n$ e pelo prefixo mais longo de $x_{1+n_i}^n$ em $D_i \times D_{1+i}$) não seja o melhor par (i.e. o par em $D_i \times D_{1+i}$ que concatenado forma o maior prefixo de $x_{1+n_{i-1}}^n$), para maiores detalhes ver [12]. Em alguns casos, é interessante tomar um segmento s_i reduzido, pois isso pode tornar o próximo segmento s_{1+i} mais longo e fazer com que o par com s_i reduzido seja melhor que o par original. No entanto, se o dicionário fosse tal que se $d \in D_i$ então todos os sufixos de d também pertencessem a D_i, este problema nunca ocorreria. Isso evitaria o primeiro obstáculo.

O segundo obstáculo, que faz com que diferentes partições produzam diferentes dicionários, tambéé pode ser evitado. Em cada passo do $LZ78$, apenas as extensões $s_i v$, $v \in \mathcal{A}$, de $s_i = x_{1+n_i}^{n_1+i}$ são adicionadas no dicionário. O problema é que o segmento s_i começa em um ponto bem específico que depende do segmento s_{i-1} e consequentemente depende da partição. Note que as extensões de $x_{2+n_i}^{n_1+i}$ não serão incluídas no dicionário. A princípio, não existe razão alguma para que isso ocorra. Isso pode ser evitado expandindo o dicionário. A cada passo, ao invéz de incluir apenas as extensões de s_i, a partição-$LZ78^*$ também introduz no dicionário todas as extensões dos sufixos de s_i. É interessante observar que este procedimento é suficiente para transpor os dois obstáculos.

$LZ78^*$-parsing

Seja x_1^n uma seqüência de entrada. Então a partição-$LZ78^*$ é definida por $s_1 = x_1^{n_1}, \ldots, s_{m_{LZ78^*}} = x_{1+n_{m_{LZ78^*}-1}}^n$, tal que para todo $i = 1, \ldots, m_{LZ78^*}$, s_i é o prefixo mais longo de $x_{1+n_{i-1}}^n$ que pertence ao dicionário D_i, onde D_i é tal que

1. $D_1 = \mathcal{A}$,
2. $D_{1+i} = D_i^\ell$, onde ℓ é o comprimento de s_i e $D_i^k = D_i \cup \{\pi(x_{k+n_{i-1}}^n | D_i^{k-1})v : v \in \mathcal{A}\}$, $k = 1, \ldots, \ell$.

A partir da discussão acima, é possível concluir que a partição-$LZ78^*$ produz um número de segmentos menor que o número de segmentos produzido pela partição ótima do $LZ78$. É importante observar no entanto, que esta partição não é uma partição válida para o $LZ78$.

É fácil observar que o mesmo procedimento realizado para o $LZ78$ poderia ter sido realizado para o LZW. Sendo assim, também é possível criar um algoritmo que calcula um limitante inferior para o número de segmentos da partição ótima do LZW. Este algoritmo será chamado de partição-LZW^*.

4. RESULTADOS

O objetivo deste trabalho é medir qual o ganho que um algoritmo de partição eficiente pode fornecer ao codificador $LZ78$. Para isso, a partição ótima do LZ78 é analisada. O primeiro resultado do trabalho mostra que o número de segmentos de uma partição ótima do $LZ78$ possui o mesmo

Tabela 1: Número de Segmentos produzidos pelo LZW e pelo LZW^*

Arquivo - x_1^n	$\lvert x_1^n \rvert$	LZW	LZW^*	Ganho
alice29.txt	152089	35074	28385	23.5%
asyoulik.txt	125179	31374	25856	21.3%
cp.html	24603	7474	6125	22.0%
fields.c	11150	3542	2848	24.4%
grammar.lsp	3721	1409	1185	19.0%
kennedy.xls	1029744	156979	153713	2.1%
lcet10.txt	426754	84345	67320	25.3%
plrabn12.txt	481861	100954	83485	20.9%
ptt5	513216	35058	28555	22.8%
sum	38240	12527	10536	18.9%
xargs.1	4227	1792	1542	14.0%

Tabela 2: Número de Segmentos produzidos pelo mm-$LZ78$

Arquivo - x_1^n	$\lvert x_1^n \rvert$	mm-LZW	Ganho
alice29.txt	152089	33329	17.4%
asyoulik.txt	125179	30140	16.6%
cp.html	24603	6943	13.4%
fields.c	11150	3246	14.0%
grammar.lsp	3721	1321	11.5%
kennedy.xls	1029744	155990	1.5%
lcet10.txt	426754	78883	17.2%
plrabn12.txt	481861	97196	16.4%
ptt5	513216	33663	17.9%
sum	38240	11773	11.7%
xargs.1	4227	1712	11.0%

comportamento assintótico que o número de segmentos produzidos pela partição original do $LZ78$. Este resultado é apresentado a seguir.

Teorema 1 *Seja x_1^n uma seqüência emitida por uma fonte ergódica. Seja m_o o número de segmentos de uma partição ótima do $LZ78$ e m_{LZ78} o número de segmentos produzidos pela partição original do $LZ78$. Então a seguinte relação é satisfeita.*

$$\lim_{n\to\infty} \frac{m_o}{m_{LZ78}} = 1, \textit{com prob. 1}$$

A demonstração do teorema é apresentada no apêndice. É importante ressaltar que este resultado não garante que a otimização da partição não produza uma melhora no comportamento assintótico da redundância do codificador. No entanto, ele garante que se existir algum ganho, ele será pequeno. De fato, é possível mostrar que este ganho, se houver, não faz com que o codificador atinja a redundância mínima para fontes sem memória, para fontes markovianas, ou para fontes FSM.

O segundo resultado deste trabalho é relativo ao algoritmo prático que calcula um limitante inferior para o número de segmentos da partição ótima. Como na prática o codificador LZW é a versão mais utilizada do $LZ78$, os testes foram realizados para esta versão. Para medir o desempenho dos algoritmos de partição, foi utilizado o conjunto de teste para codificadores sem perda, conhecido como *Canterbury Corpus* [13]. Os números de segmentos obtidos para os arquivos do *Canterbury Corpus* através do LZW e através da partição-LZW^* são apresentados na Tabela 1.

Os resultados da Tabela 1 mostram um ganho em torno de 20%. No entanto, é importante ressaltar que o algoritmo LZW^* utiliza um dicionário com muito mais elementos que o dicionário do LZW. Isso também contribui para o ganho do LZW^*. No caso da partição ótima do LZW, o dicionário teria menos elementos que o dicionário do LZW, pois o número de elementos do dicionário é diretamente proporcional ao número de segmentos, quando a partição é uma partição genérica do LZW. Sendo assim, este ganho não seria tão elevado. O resultado do LZW^* também foi comparado com o resultado do mm-$LZ78$, proposto em [12]. Os resultados são apresentados na Tabela 2.

Da Tabela 2, é possível observar que o LZW^* produz um ganho em torno de 15% sobre o mm-$LZ78$. Este trabalho conjectura que grande parte deste ganho seja devido ao crescimento acelerado do dicionário do LZW^*.

5. CONCLUSÃO

Este trabalho apresentou uma análise da partição ótima do codificador $LZ78$. Foi provado que a partição ótima não produz ganho algum no comportamento assintótico do número de segmentos gerados pela partição. Embora este fato não seja suficiente para garantir que o comportamento assintótico da redundância não se altera, ele é suficiente para garantir que a partição ótima não torna o $LZ78$ um codificador ótimo, quando a figura de mérito é o comportamento assintótico da redundância. Utilizando um algoritmo prático, o $LZ78^*$, este trabalho apresentou uma forma de calcular um limitante inferior para o número de segmentos gerados pela partição ótima do $LZ78$. Este algoritmo foi implementado para a versão LZW e os resultados foram comparados com os resultados de algoritmos de partição utilizados em versões do LZW. Os resultados mostram um ganho em torno de 20% sobre o LZW e em torno de 15% sobre o mm-$LZ78$.

Apêndice

Este apêndice apresenta a demontração do teorema 1, introduzido na seção 4.

Demonstração: Seja x_1^n uma seqüência emitida por uma fonte ergódica. Seja m_o o número de segmentos de uma partição ótima da seqüência x_1^n para o codificador $LZ78$ e seja m_{LZ78} o número de segmentos produzido pelo algoritmo original do $LZ78$. Pelo Teorema 12.10.1 de [14], é possível afirmar que

$$\frac{m_{LZ78}\lceil \log_2 (m_{LZ78}|\mathcal{A}|)\rceil}{n} \rightarrow H, \text{ com Prob. 1}, \qquad (4)$$

onde H é a entropia da fonte. Além disso, como a taxa de compressão do $LZ78$ converge para H,

$$\frac{\sum_{i=1}^{m_o}\lceil \log_2 (i|\mathcal{A}|)\rceil}{n} \rightarrow H, \text{ com Prob. 1}$$

Portanto,

$$\frac{\sum_{i=1}^{m_{LZ78}}\lceil \log_2 i|\mathcal{A}|\rceil}{n} - \frac{\sum_{i=1}^{m_o}\lceil \log_2 i|\mathcal{A}|\rceil}{n} \rightarrow 0, \text{ com Prob. 1} \qquad (5)$$

Os somatórios da equação acima podem ser calculados utilizando a seguinte identidade.

$$\sum_{i=K_1}^{K_2} a_i = (K_2-K_1+1)a_{K_2} - \sum_{i=K_1}^{K_2-1}(i+1-K_1)(a_{i+1}-a_i), \qquad (6)$$

que é uma simples generalização do exemplo 1.2.4-42 de [15]. Sendo assim, de (5) e (6),

$$0 \leq \frac{1}{n}\left((m_{LZ78}-m_o)\lceil log_2 (|\mathcal{A}|m_{LZ78})\rceil - K\right) \rightarrow 0,$$

com Prob. 1. Mas $\frac{K}{n} \rightarrow 0$, com Prob. 1, logo,

$$\frac{1}{n}(m_{LZ78}-m_o)\lceil log_2 (|\mathcal{A}|m_{LZ78})\rceil \rightarrow 0, \text{ com Prob. 1},$$

ou

$$\frac{m_{LZ78}\lceil log_2 (|\mathcal{A}|m_{LZ78})\rceil}{n}\left(1-\frac{m_o}{m_{LZ78}}\right) \rightarrow 0, \qquad (7)$$

com Prob. 1. De (4) e (7),

$$H\left(1-\frac{m_o}{m_{LZ78}}\right) \rightarrow 0, \text{ com Prob. 1}$$

Portanto $\frac{m_{opt}}{m_{LZ78}} \rightarrow 1$, com Prob. 1, o que conclui a demonstração do teorema.

Referências

[1] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, pp. 530–536, 1978.

[2] T. A. Welch, "A technique for high-performance data compression," *IEEE Computer*, vol. 17, pp. 8–19, 1984.

[3] L. D. Davisson and A. Leon-Garcia, "A source matching approach to finding minimax codes," *IEEE Transactions on Information Theory*, vol. 26, pp. 166–174, 1980.

[4] L. D. Davisson, "Minimax noiseless universal coding for markov sources," *IEEE Transactions on Information Theory*, vol. 29, pp. 211–215, 1983.

[5] J. Rissanen, "A universal data compression system," *IEEE Transactions on Information Theory*, vol. 29, pp. 656–664, 1983.

[6] J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Transactions on Information Theory*, vol. 30, pp. 629–636, 1984.

[7] G. Louchard and W. Szpankowski, "On the average redundancy rate of the lempel-ziv code," *IEEE Transactions on Information Theory*, vol. 43, pp. 1–8, 1997.

[8] S. A. Savari, "Redundancy of the lempel-ziv incremental parsing rule," *IEEE Transactions on Information Theory*, vol. 43, pp. 9–21, 1997.

[9] J. C. Kieffer and E. Yang, "A simple technique for bounding the pointwise redundancy of the 1978 lempel-ziv algorithm," *Proc. of IEEE Data Compress Conference*, pp. 434–442, 1999.

[10] A. Hartman and M. Rodeh, "Optimal parsing of strings," *Combinatorial Algorithms on Words, Springer-Verlag, A. Apostolico and Z. Galil, editors,*, pp. 155–167, 1985.

[11] S. D. Agostino and J. Storer, "On-line versus off-line computation in dynamic text compression," *Information Processing Letters*, vol. 59, pp. 169–174, 1996.

[12] M. S. Pinho, W. A. Finamore, and W. A. Pearlman, "Fast multi-match lempel-ziv," *Proc. IEEE Data Compress Conference*, p. 545, 1999.

[13] R. Arnold and T. Bell, "A corpus for the evaluation of lossless compression algorithms," *Proc. of IEEE Data Compress Conference*, pp. 201–210, 1997.

[14] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[15] D. Knuth, *The Art of Computer Programming: Fundamental Algorithms*, vol. 1. Massachusetts: Addison-Wesley, 1973.