招請論文

유니버샬 데이타 압축의 최근의 연구동향

正會員 朴 志 煥* 正會員 陳 庸 玉**

Recent Trends of Universal Data Compression

Ji Hwan PARK* Yong Ohk CHIN** *Regular Members*

要 約 텍스트 데이타 중심의 컴퓨터 통신의 활성화와 멀티 미디어의 등장에 따른 영상 데이타의 취급은 대용량의 기억공간과 전송시간을 요구하게 되어 데이타 압축의 중요성이 더욱 크게 되었다. 정보원의 사전 지식을 전제로 하지 않고도 효율적인 압축을 달성하는 유니버샬 부호는 동일 알고리즘으로 다양한 데이타에 적용할 수 있는 만능성을 갖는 적응성이 뛰어난 방식으로 최근 주목되고 있다.

이 논문에서는 유니버샬 부호의 기본원리와 분류를 통한 실현 알고리즘에 대하여 알아보고 그 응용을 중심으로 한 최근의 연구활동에 관하여 기술한다.

ABSTRACT Data compression has important application in the areas of file storage and distributed computer systems. The universal data compression achieves asymptotically optimum compression ratio for strings generated by any stationary ergodic source without a priori source probabilities.

This paper describes the principle and the recent research trends on universal data compression and its applications.

Ⅰ. 서 론

정보화 사회의 발달에 따라 컴퓨터에서 처리하는 데이타는 대량화 및 다양화 되어 가고 있다. 이와같은 각종 데이타를 효율 좋게 저장하고 전송하기 위한 방법으로 데이타 압축 기술이 여러가지 고안되어 실용화 되고 있다.

데이타 압축은 원래의 데이타 계열에 포함되어 있는 불필요한 부분(redundancy)을 제거하여 유용한 정보만을 추출함으로써 보다 짧은 데이타 계열로 변환하는 것이다. 압축 데이타 계열로부터 원래의 데이타 계열로 재생할때 복원 오류를 허용하지 않는 무잡음 압축(noiseless compression) 과 일정 허용 범위내에서 오류를 허용하는 유잡음 압축(compression with distortion)으로 분류된다. 유잡음 압축은 연속 정보인 TV, Radio 및 Sonar신호 등의 1차 압축에 응용되며 rate distortion 함수의 해석이론에 그 이론적 바탕을 두고 있다.[1] 무잡음 압축은 데이타 처리계나 디지탈 팩시밀리등의 이산 정보를 대상으로 하며 Shannon의 정보론에 그 기초를 두고 있다.[2] 이 논문에서는 이산 정보를 대상으로 하는 무잡음 압축에 한하여 다루기로 한다.

Huffman부호[3]를 비롯한 기존의 많은 압축법에서는 정보원 기호(source symbols)의 출현빈도를 알고 있다는 전제하에 최적 부호(optimum code)를 구성하는 통계적 기법을 사용하고 있다. 그러나 현실의 정보원으로부터 발생하는 많은 데이타는 그 확률 구조를 알지 못하며 심지어 마르코프 모델로 표현할 수 있는지 없는지조차 알지 못하는 경우가 대부분이다. 이와같은 배경아래 등장된 유니버샬 데이타 압축법은 동일한 알고리즘으로 보다 넓은 클래스의 정보원에 적용 가능한 만능성을 가진다.[4] 또한 데이타의 길이가 충분히 긴 경우, 평균 부호의 길이를

* 釜山水産大學校 電子計算學科
Natl. Fisheries Univ. of PUSAN
** 慶熙大學校 電子工學科
Kyunghee University
論文番號 : 91-84

압축의 이론적 한계인 엔트로피에 얼마든지 접근시킬 수 있는 점근적 최량성(asymptotically optimal) 이 보장되는 강력한 부호로 주목되고 있다.

유니버샬 부호의 명확한 정의는 확립되어 있지 않으나, 첫째 어떤 클래스에 속하는 모든 정보원 출력을 최적 또는 효율좋게 압축할 수 있는 부호 둘째 정보원의 확률분포에 의존하지 않는 부호화·복호화 알고리즘을 이용하고 있는 부호등의 의미를 포함하고 있으며 크게 다음의 두 가지로 분류된다.[5]

(1)데이타 계열로부터 정보원 모델링(source modeling)을 적응적으로 수행하면서 그 모델에 기초한 확률분포를 이용하여 부호화하는 방법

(2)현재의 부분계열이 처리가 끝난 이전의 데이타 계열의 어느 부분과 일치하고 있는가를 나타내는 정보를 이용하여 알고리즘적으로 부호화하는 방법

(1)의 구체적인 부호화를 위하여 산술부호(arithmetic codes)를 이용하는 방법[6]과 Dynamic Huffman부호를 이용하는 방법[7]이 있다. (2)에는 Ziv-Lempel부호[8], Recency Rank부호[9] 및 Interval 부호[10]등이 있다. 산술부호는 주어진 문맥모델하에서의 빈도 분포가 설정되면 엔트로피까지 압축이 가능하므로 관심의 촛점은 어떻게 좋은 문맥모델(context model)을 작성하느냐에 집중된다. 또한 적응사전(adaptive dictionary)을 이용하는 (2)의 방법은 먼저 데이타 계열을 적응사전을 이용하여 자연수의 계열로 변환하고, 다음에 그 자연수를 2진계열로 부호화하는 2단계로 이루어진다. 문맥모델을 이용하는 (1)의 방식과 적응사전을 이용하는 (2)의 방식은 전혀 다르게 보이나 정보원 모델 구축의 입장에서 보면 서로 밀접한 관계를 갖는 것으로 간주할 수 있다. 일반적으로 산술 부호의 최적성에 의해 문맥모델을 이용하는 방식이 높은 압축 효과를 얻을 수 있으나, 부호화·복호화의 고속성이나 소요기억용량의 면에서 적응사전을 이용하는 방식이 경제적이므로[11] 실제에는 (2)의 방식이 널리 이용되고 있는 실정이다.

이 논문에서는 위에 언급한 각종 유니버샬 부호의 기본원리와 그 알고리즘을 중심으로 다루고 이에 관련된 최근의 연구동향에 대하여 기술한다.

Ⅱ. Dynamic Huffman부호

Huffman부호는 평균 부호의 길이를 최소로 하는 최단부호(compact codes)로 잘 알려져 있으나 이 부호를 이용한 데이타 압축 알고리즘의 구성에는 두가지 방법이 있다.

2.1 이중 주사 방식(two pass method)[3]

길이n의 데이타 계열 $X_n=x_1x_2x_3\cdots x_n$으로부터 각 기호의 출현빈도를 구하여 그 빈도분포에 기초한 Huffman tree를 작성한다. 부호화는 먼저 Huffman tree 또는 빈도표를 부호화한 후, 작성된 Huffman tree를 이용하여 X_n을 Huffman 부호화 한다.

2.2 단일 주사 방식(one pass method)

X_n의 한 기호를 처리할때 마다 그때까지의 빈도분포에 의해 결정되는 tree가 Huffman tree가 되도록 어떤 정해진 방법으로 tree를 매회 갱신하면서 Huffman부호화를 실시한다. Dynamic Huffman부호는 Faller[12]와 Gallager[13]에 의해 독립적으로 제안되었으며, 그후 Knuth[7]와 Vitter[14]에 의해 상세한 알고리즘이 제시되었다.

Dynamic Huffman부호의 code tree 수정 및 부호화의 일예를 Kunth 알고리즘에 따라 그림1에 나타낸다. 지금 계열 X_3=abc에 대해 그림1(a)와 같이 tree가 작성되어 있다고 하자. 각 단점(terminal nodes)에는 지금까지 출현한 기호와 출현빈도가 표시되어 있고 각 내부절점(internal nodes)에는 그 절점에 예속되는 절점의 빈도 합이 기록되어 있다. 이때 다음문자 x_4=c라면 101이 c에 대응하는 부호어(codeword)가

되고, 아직 출현하지 않은 기호 x_4=d가 나타나면 $100(d)_2$를 부호어로 한다. 단, $(d)_2$는 기호 d를 2진수로 나타낸 것이다. Tree의 갱신을 위해 출현 기호에 대응하는 절점의 빈도 값을 1 증가시키나, 만약 증가전의 절점과 같은 빈도를 갖는 절점이 고순위(순위는 위 일수록 높고 같은 레벨이면 오른쪽일수록 높다)에 있으면 sub-tree를 서로 교환한다. 그림 1(a)의 tree를 x_4=c 및 x_4=d로 갱신한 후의 tree를 그림1(b),(c)에 나타낸다.

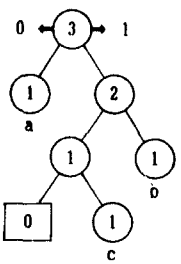


Fig 1. (a) Dynamic Huffman tree after processing "abc".

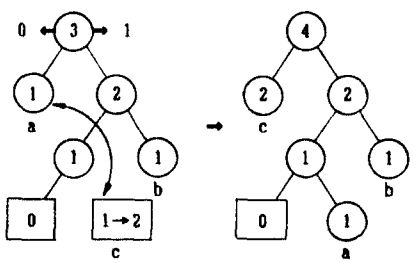


Fig 1. (b) Dynamic Huffman tree after processing "abcc".

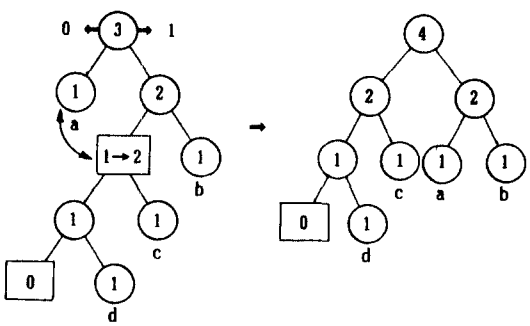


Fig 1. (c) Dynamic Huffman tree after processing "abdc".

Knuth의 알고리즘을 이용하면 tree가 항상 Huffman tree로 되기 때문에 평균 부호의 길이는 최단이 된다. 그러나 서로 교환하는 절점이 동일 빈도의 최하위 순위로 되기 때문에 최대 부호의 길이가 최단이 되지 않는다. Vitter는 Schwarz의 알고리즘[15]을 이용하여 최대 부호의 길이와 전체 부호의 길이가 최소가 되고, sub-tree 의 정렬 회수를 감소시키는 알고리즘을 제시하였다.[14] Dynamic Huffman부호를 이용한 응용 가운데 Knuth의 알고리즘에 기초한 UNIX 의 COMPACT command[16]등 압축용으로서 널리 이용되고 있으며, 한글 데이타의 표현[17] 및 데이타 압축과 암호화를 결합시키는등의 연구[18]가 이루어지고 있다.

Ⅲ. 자연수의 유니버샬 표현

적응사전을 이용하는 Dictionary법[19]은 현재의 데이타 계열이 처리가 끝난 계열의 어느 부분에 일치하고 있는가를 나타내는 인덱스(자연수) $i \in N^+$를 부호어로서 이용한다. 따라서 자연수 i를 얼마나 효율좋게 2진표현 하느냐가 부호화 성능에 큰 영향을 미치게 된다. 아래의 자연수의 2진표현법은 모두 부호어 사이의 구별이 자동적으로 이루어지는 Comma free조건을 만족하게 된다. 이하, $(i)_2$는 최상위 비트가 1인 2진수이며, $(i)^p$는 p비트로 나타낸 i의 2진수 표현이다. 또한 각 표현법의 i의 2진 표현을 B(i)로 나타내며 이때 필요한 비트수를 L(i)로 표시한다. 이하 *log*의 밑은 2이다.

3.1 자연수 i가 유한일때($1 \leq i \leq N$)

(1) i가 같은 확률로 발생하는 경우

$\lceil \log N \rceil$ 비트의 고정 길이로 간단히 표현(이하, B_1법)할 수 있으나, code-tree가 complete tree로 되어 있지 않기 때문에 ($\lceil \log N \rceil - \log N$) 비트의 손실이 발생하게 된다. 여기서 $\lceil x \rceil$는 x보다 큰 최소 정수이며, $\lfloor x \rfloor$는 x보다 작은 최대 정수이다. 이 결점을 개량한 방법[20]은 압축율의

개선 효과가 상당히 크므로 처리 속도가 중요시되지 않는 경우에는 유효한 방법으로 기대된다. 이때 $B_1(i|N)=(i-1)^{\lceil \log N \rceil}$으로 표기된다.

(2) i값이 작을수록 높은 확률을 갖는 경우 : $P(i-1)\geq p(i)$

i의 값이 작을수록 짧은 비트로 표현할 필요가 있는데, Willems[10]는 N개의 자연수를 몇개의 그룹으로 나누어 i를 G번째 그룹의 M번째의 멤버로 나타내어 G, M를 2진 부호화하는 방식을 제안하였다. Willems부호의 표현방법(이하, B_2법)과 그 일예를 다음에 보인다.

$2^G\leq i\leq 2^{G+1}-1$, $G=0,1,\cdots,L=\lfloor \log N \rfloor$에 대해,

$$B_2(i|N)=\begin{cases}(G)_{\lceil \log_2(L+1)\rceil}\cdot (M)^G, M=i-2^G, & i\neq N\\ (G)_{\lceil \log_2(L+1)\rceil}, & i=N\end{cases}$$

이므로 소요되는 비트수 $L_2(i)=\lceil \log(\lfloor \log N \rfloor+1)\rceil+\lfloor \log i \rfloor$, $i\neq N$으로 된다. N=8의 경우,

$B_2(1|8)=00$, G=0 $B_2(5|8)=10\ 01$, G=2
$B_2(2|8)=01-0$, G=1 $B_2(6|8)=10-10$, G=2
$B_2(3|8)=01-1$, G=1 $B_2(7|8)=10-11$, G=2
$B_2(4|8)=10-00$, G=2 $B_2(8|8)=11$, G=3

로 되어 i값이 작을수록 짧은 길이로 표현할 수 있게 된다. -는 그룹G와 멤버M을 식별하기 위한 가상의 기호이다.

3.2 i가 유한이 아니고, i값이 작을수록 높은 확률을 갖을때

(1) Zero Run법[21]

이 방법은 자연수 $i\in\{1,2,\cdots\}$를 comma free 조건을 만족하도록 $(i)_2$의 길이 $\lfloor \log i \rfloor+1$를 나타내기 위해 $\lfloor \log i \rfloor$ 비트의 Zero Run을 $(i)_2$의 선두에 붙이는 표현법(이하, B_3법)이다. 따라서, $B_3(i)=\underline{0\cdots0}\cdot\underline{(i)_2}$, $B_3(9)=000\cdot1001$이므로 $L_3(i)=1+2\lfloor \log i \rfloor$로 된다.

(2) Doubly Zero Run법[21]

Zero Run법의 0런의 길이에 대해 또다시 Zero Run법을 적용시키는 방법(이하, B_4법)으로서 i의 값이 클때 효과적이다. 즉, $B_4(i)=\underline{0\cdots0}\cdot\underline{(1+\lfloor \log i \rfloor)_2}\cdot\underline{(i)_2}$로 되며 $(i)_2$의 MSB는 항상 1로서 생략 가능하므로 $L_4(i)=1+\lfloor \log i \rfloor+2\lfloor \log(1+\log i)\rfloor$로 된다. $B_3(i)$는 유니버살 부호로 최적성이 보장되지 않으나, $B_4(i)$는 점근적으로 최적이 되는 부호이다. 한편, $(i)_2$의 길이를 나타내는 부분을 재귀적으로 정의한 $B_5(i)$는 다음과 같다.

$B_{5\text{-}L}(i)=R_L(i)\cdot 0$, $B_{5\text{-}2}(20)=\underline{10}\ \underline{100}\ \underline{10100}\cdot 0$,
$B_{5\text{-}3}(20)=\underline{100}\ \underline{10100}\cdot 0$.

따라서 $L_5\fallingdotseq 1+\log i+\log\log i+\cdots\cdots$로 된다. B_5의 밑줄 친 각 블록의 MSB가 반드시 1임을 이용하여 최후에 0을 붙여 부호의 끝을 나타내고 있다. 그러나, 반복되는 부분의 비트를 반전시킨 $B'_{5\text{-}2}(20)=\underline{01}\ \underline{011}\ 10100$로 되므로 최후의 0을 생략할 수 있어 한 부호당 1 비트 절약할 수 있다.[22]

(3) Fibonacci code[23]

Elias에 의한 표현 방법에서는 부호어를 서로 구별하기 위해 $(i)_2$의 길이를 나타내는 prefix를 이용하였으나, Fibonacci code에서는 부호간의 구별을 위한 flag를 suffix에 부가하고 있다. Fibonacci number의 차수가 r일때 j번째의 수는 r개 이전의 Fibonacci number의 합으로 나타낸다. 즉,

$F_j=F_{j-1}+F_{j-2}+\cdots+F_{j-r}$, $j\geq 1$, $F_0=\cdots=F_{1-r}=1$.

자연수 i의 Fibonacci 표현 R(i)는

$$R(i)=\sum_{j=0}^{k} d_j\cdot F_j,\ d_j\in\{0,1\},\ k<i$$

의 d_j계열로 나타낸다. 차수 r=2의 Fibonacci code는 $F(i)=\sim R(i)\cdot 1$로 되며 $\sim R(i)$는 R(i)의 비트 순서를 역순으로 나타낸 계열이다. r=2의 Fibonacci code의 일예를 표1에 나타낸다.

표 1. Fibonacci number에 의한 표현
Table 1. Fibonacci representations and cooles.

i	R(i)	F(i)=~R(i)·1
1	1	1 1
2	1 0	0 1 1
3	1 0 0	0 0 1 1
4	1 0 1	1 0 1 1
5	1 0 0 0	0 0 0 1 1
6	1 0 0 1	1 0 0 1 1
7	1 0 1 0	0 1 0 1 1
8	1 0 0 0 0	0 0 0 0 1 1
16	1 0 0 1 0 0	0 0 1 0 0 1 1
32	1 0 1 0 1 0 0	0 0 1 0 1 0 1 1
	F_6 F_5 F_4 F_3 F_2 F_1 F_0	
	21 13 8 5 3 2 1	

Fibonacci code는 prefix code를 구성하지는 않으나 최후미의 1에 의해 comma free code의 조건을 만족한다. 또한 점근적 최량성은 보장되지 않지만, 유니버샬 코드로서 i<514,228에서는 Elias code에 비해 우수한 성능을 보이고 있어 유한 길이의 데이타 압축에 많이 이용되고 있다.[24]

Ⅳ. 적응사전(Adaptive Dictionary:AD)을 이용한 방식

사전을 이용한 데이타 압축법은 오래전부터 사용되어 왔으나, 효율이 좋은 실용적인 적응사전을 이용한 방식의 연구는 Ziv & Lempel의 논문[8][25][26]이 계기가 되어 활발한 연구가 이루어지게 되었다. AD방식은 데이타 계열을 자연수 계열로 변환한 후 그것을 2진 계열로 부호화하게 된다. 데이타 계열로부터 자연수 계열로의 부호화는 다음의 3단계로 이루어지게 된다.

(1) 검 색 : 압축하고자 하는 데이타 기호(열)을 사전 내에서 검색.

(2) 부호화 : 데이타 열을 사전 내의 순위등을 이용하여 자연수의 계열로 부호화

(3) 갱 신 : 부호화 / 복호화의 사전을 갱신.

Ziv-Lempel이 포인터형의 유니버샬 부호(이하 ZL-77)[25]를 제안하여 데이타 압축에 적용시킨 이래 수많은 연구가에 의해 응용되어 압축 유틸리티[27]로써 널리 사용되게 되었다.

4.1 ZL-77방식

Ziv-Lempel부호의 기본원리는 원 데이타 계열의 이미 출현한 기호열로부터 기호 복제를 거듭하여 새로운 기호열을 만들면서 압축계열로 치환하는 것이다. 기호열 **X**=xxy에 대해, **X**[a,b]는 **X**의 a의 위치에서 시작되는 길이 b의 기호열의 생성을 의미한다. 즉 **X**[1,1]=x, **X**[2,2]=xy 및 **X**[2,4]=xyxy로 복제된다. 또한 새로운 기호의 출현에 대응하기 위하여 새로운 기호를 추가하는 규칙을 도입한다. **Y**=xxyxyz일때, **X**로부터 **Y**를 직접은 복제할 수 없으나 **X**[2,2]로부터 xyz를 생성하기 위해

$$\mathbf{X}[2,2]=xy=\rangle^{z}xy=xy\cdot z$$

라는 생성규칙을 이용하면 복제 가능하다. 따라서, 모든 임의의 계열은 기호(열)의 생성을 반복함으로서 서로 다른 부분계열로 분해된다. 이것에 바탕을 둔 ZL-77방식의 알고리즘은 다음과 같다.

(1) 버퍼를 초기화한다.

길이 p의 참조부 버퍼 p와 길이 q의 부호화부 버퍼 Q를 이용하여 기호복제를 수행한다. 최초에는 P에 아무 데이타도 들어 있지 않는 공백의 상태이고, Q에는 원 데이타의 선두 부분이 설정되어 있다.

(2) 부분계열로 분해한다.

생성규칙을 이용하여 P로부터 Q를 부분계열로 분해한다. 즉 계열 X=xxyxyz는 $(\lambda)=\rangle^{x}x=\rangle^{y}x\cdot y=\rangle^{z}xy\cdot z(x\cdot xy\cdot xyz)$의 세개의 부분계열로 분해된다. i번째의 버퍼의 상태를 $B_i=P\cdot Q$라 하면 Q에 대해 부분계열 $X_i=Q(1,b_i)$를 아래와 같이 구한다.

$X_1=Q(1,b_1)=Q(1,b_1-1)\cdot Q(b_1)=P(a_1,a_1+b_1-2)\cdot Q(b_1)=P[a_1,b_1-1]\cdot Q(b_1)$

$Q(1,b_1-1)$은 이미 부호화가 끝난 계열로부터 복제되는 부분으로서 b_1-1가 0인 경우는 공계열(λ)을 복제함을 의미한다. 여기서 P(a,b)는 계열 P의 a에서 b까지의 부분계열을 나타내며 P[a,b]와 구별됨에 주의 바란다.

(3) 부분계열을 부호화한다.

부분계열 X_1에 대한 부호어 $C_1=C_{11}C_{12}C_{13}$로 이루어진다. 여기서 $C_{11}=a_1-1$, $C_{12}=b_1-1$ 및 $C_{13}=Q(b_1)$에 해당한다.

(4) 버퍼를 갱신한다.

버퍼의 내용을 부호화 처리가 이루어진 데이타의 수 만큼 좌로 이동시키고 Q버퍼에는 이동된 기호의 수만큼 새로운 데이타가 들어오게 된다. (2)에서 (4)의 절차를 원 데이타가 전부 처리될 때까지 반복한다. 아래에 ZL 77방식에 의한 부호화의 일예를 보인다.

〈전제조건〉

원 데이타 : X=xxyxyzyzyzx……
=x·xy·xyz·yzyzx……

버퍼 P와 Q의 크기 : p=q=8 byte

C_{11}과 C_{12}의 크기 : $\lceil \log_2 p \rceil$=3 bit, $\lceil \log_2 q \rceil$=3 bit

①버 퍼 : $B_1=P\cdot Q$, P=::::, Q=xxyxyzyz
분 해 : $X_1=x=Q(1,0)\cdot x=P[1,0]\cdot x$
부호화 : $C_{11}=000$, $C_{12}=000$, $C_{13}=x$

②버 퍼 : $B_2=P\cdot Q$, P=::::::x, Q=xyxyzyzy
분 해 : $X_2=xy=Q(1,1)\cdot y=P[8,1]\cdot y$
부호화 : $C_{21}=111$, $C_{22}=001$, $C_{23}=y$

③버 퍼 : $B_3=P\cdot Q$, P=:::::xxy, Q=xyzyzyzx
분 해 : $X_3=xyz=Q(1,2)\cdot z=P[7,2]\cdot z$
부호화 : $C_{31}=110$, $C_{32}=010$, $C_{33}=z$

④버 퍼 : $B_4=P\cdot Q$, P=::xxyxyz, Q=yzyzx--
분 해 : $X_4=yzyzx=Q(1,4)=P[7,4]\cdot x$
부호화 : $C_{41}=110$, $C_{42}=100$, $C_{43}=x$

ZL 77방식의 주된 계산량은 일치하는 최대 길이의 기호열을 P버퍼로부터 탐색하기 위한 패턴매칭(pattern matching) 작업이다. 이를 위해 KMP 방식이나 BM방식등의 실시간 패턴매칭 알고리즘[28]을 이용한 방식이 있다.[29] ZL-77방식에 관한 많은 개량 방식이 연구 및 실용화 되어 전자우편, BBS 등 비롯한 PC통신 분야와 각종 데이타의 효율적인 보관을 위해 널리 사용되고 있다.[30]

4.2 ZL-78방식

Ziv & Lempel은 ZL 77방식의 유니버샬 부호 이외에 IL(Information Lossless)부호기에 대한 부호화 정리를 증명하기 위하여 증분분해(incremental parsing)을 이용한 유니버샬 부호(이하 ZL-78방식)를 제안하고 있다.[26] ZL-78방식의 기본개념인 증분분해는 원 데이타 계열 $X_n=x_1x_2x_3\cdots x_n$을 $X=X_0X_1X_2\cdots X_N$,$(N<n)$으로 분해할 때, 각 부분계열 $X_i(0\leq i\leq N)$는 다음 조건을 만족한다.

(1) $X_0=\lambda$(길이 0의 Null string)

(2) X_N을 제외한 모든 $X_i(1\leq i<N)$는 서로 다르다.

(3) X_i의 최후의 문자 x_i를 제거하면 일치하는 $X_j(0\leq j<i)$가 반드시 하나 존재한다.

각 X_i는 (3)의 성질에 의해 $X_i=X_j\cdot x_i$를 이용하여 (j,x_i)를 부호화하면 된다. ZL-78방식의 구체적인 부호화 알고리즘은 다음과 같다.

(1) 기호열을 부분계열로 증분분해한다.
$X_i=X_j\cdot x_i$, $j<i$

(2) 부분계열을 부호화한다.
부호어 $C_i=j\cdot r+x_i$, $x_i<r$(radix)·로 이루어지며, r=2일때 C_i의 upper bound는 $C_i=2i-1$로 되어 X_i를 부호화하기 위해 필요한 비트수 L_i는 $\lceil \log i \rceil+1$로 된다. 아래에 X_9=001010100에 대한 부호화의 일예를 보인다.

(1) 증분분해

$X_0=\lambda$

(1) $X_1=X_0\cdot 0\rightarrow 0$

(2) $X_2=X_1\cdot 1\rightarrow 01$

(3) $X_3=X_2\cdot 0\rightarrow 010$

(4) $X_4=X_0\cdot 1\rightarrow 1$

(5) $X_5=X_1\cdot 0\rightarrow 00$

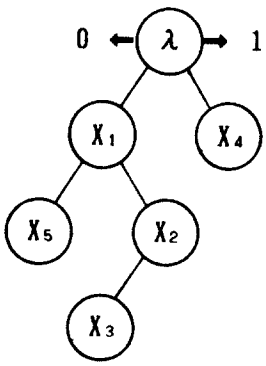


Fig 2. parsing tree

② 부호화

(1) $X_1=X_0\cdot 0=0$, $C_1=0\cdot 2+0=0$, $L_1=\lceil \log 1\rceil+1=1$, $b_1=0$

(2) $X_2=X_1\cdot 1=01$, $C_2=1\cdot 2+1=3$, $L_2=\lceil \log 2\rceil+1=2$, $b_2=11$

(3) $X_3=X_2\cdot 0=010$, $C_3=2\cdot 2+0=4$, $L_3=\lceil \log 3\rceil+1=3$, $b_3=100$

(4) $X_4=X_0\cdot 1=1$, $C_4=0\cdot 2+1=1$, $L_4=\lceil \log 4\rceil+1=3$, $b_4=001$

(5) $X_5=X_1\cdot 0=00$, $C_5=1\cdot 2+0=2$, $L_5=\lceil \log 5\rceil+1=4$, $b_5=0010$

따라서, 원 데이타 $\mathbf{X}_9=0\cdot 01\cdot 010\cdot 1\cdot 00$는 $\mathbf{Y}_{13}=0\cdot 11\cdot 100\cdot 001\cdot 0010$의 압축계열로 변환된다. 위의 예에서 알 수 있듯이 ZL방식의 결점은 입력 데이타의 길이 n이 충분히 길지 않으면 압축의 효과를 기대할 수 없다는 점에 있다. 그 이유는

(1) 각 부분계열의 마지막 기호 x_i가 압축되지 않는 형태로 부호화 되고,

(2) i가 커짐에 따라 더 이상 참조되지 않는 부분계열 X_j가 급격히 증가하기 때문이다. (1)의 문제점은 부분계열 X_i로 분해할때 최후의 기호 x_i를 X_{i+1}의 선두 기호에 중첩시킴으로써 j만을 부호화하는 LZW방식에서 해결하고 있다.[31] UNIX의 Compress Command[32]는 LZW 방식에 근거하고 있다. (2)의 문제점은 2진계열의 경우 중분분해는 완전나무(complete tree)를 구성하여 외부 절점에만 참조번호를 할당시켜 해결할 수 있다.[20] 이때 decoding tree에서는 외부 절점의 번호에 의해 데이타 계열을 복호하기 때문에 외부 절점에서 루트 절점의 방향으로 LIFO순서로 데이타가 복원된다. 그러나, Enumerative Code[33]의 개념을 도입하여 외부 절점에 번호를 할당하면 FIFO형식으로 동시에 데이타 길이 n에 비례하는 O(n)의 속도와 소요용량으로 복호된다.[34]

또한 위의 알고리즘에서는 적응사전을 구성하기 위한 소요 기억 용량에는 제한을 두지 않고 있다. 그래서, 일종의 self-organizing list를 준비하여 사전의 용량이 포화가 되었을 때 LRU (Least Recently Used) 계열을 제거하게 되면 효율적으로 사용할 수 있다.[35] 이외에도 LFU (Least Frequently Used)계열을 제거하거나, 사전을 Reset, Freeze, Swap하는 등의 방법이 있다.[36]

ZL-78방식에서는 Tree나 Hash함수등을 이용하여 부호화하기 때문에 일반적으로 부호화 및 복호화의 처리속도가 고속이나, 증분분해의 부분계열로만 새로운 단어가 사전에 등록되기 때문에 초기에는 압축효율이 좋지 못한 결점이 있다. 한편, ZL-77방식은 과거의 모든 계열을 부호화에 이용할 수 있으나 고속으로 검색, 등록, 소거 등의 처리가 가능한 데이타 구조를 만들기 어렵다. 그래서, 사전의 처리가 고속으로 이루어지도록 Hash함수나 Trie등을 이용하고 동시에 ZL-77방식보다 사전에 등록되는 단어의 수가 증가하도록 고려한 방식이 여러 가지 제안되어 있다.[37][38][39]

V. 산술부호(arithmetic codes)

산술부호는 [0,1)의 數直線 구간을 정보원 기호의 출현확률에 비례하게 순차적으로 분할하여, 정보원 계열(source sequences)을 확률 구간상의 점의 좌표로 부호화하는 방법으로 해석할 수 있다. 산술부호의 기본 개념은 C.E.Shannon[2]의 논문에서 찾을 수 있으며, Elias의 이상부호(ideal code)[40]가 제기가 되어 Rissanen과 Pasco에 의해 실용적인 형태로 발전하게 되었다. 그 후 많은 연구가에 의해 개선 발전되어 텍스트 데이타를 비롯하여 영상 데이타의 압축에 이르기까지 다양하게 응용되고 있다.

산술부호는 기존의 잘 알려진 Huffman Codes[3]와는 달리 블록부호(block code)가 아니며 i.i.d (independently and identically distribution) 정보원에 대해서도 높은 압축율을 달성할 수 있어 장치화의 문제점으로 지적되어 왔던 정보원 확대(alphabet extention)를 수반하지 않는 장점이 있다. 특히, 재귀적 수법(recursive operations)에 의한 알고리즘의 구성이 간단하며, 정보원 모델링과 부호화를 명확히 구분할 수 있어 정보원의 변화에 대한 적응성이 뛰어나므로 다양한 분포의 정보원에도 적용할 수 있는 잇점이 있다.[41]

5.1 Binary Arithmetic Codes(BAC)

텍스트 중심인 정보원의 팩시밀리 통신에는 **MH**(Modified Huffman), **MR**(Modified READ) 및 **MMR**방식이 주로 사용되어 왔으나, 최근의 문서통신의 다양화에 따라 등장된 몇가지 방식중에 산술부호를 이용한 방식이 주목되고 있다. 여기서는 **BAC** 알고리즘을 중심으로 소개하고[42] 그 응용에 대하여 알아 본다.

길이 n의 정보원 계열 $\mathbf{X_n}=x_1x_2\cdots x_n$, $x_i\in\{0,1\}$의 부호화를 생각한다. 기호 0,1의 출현확률이 p(0), p(1)일때, MPS(More Probability Symbol)를 0, LPS(Less Probability Symbol)를 1로 가정한다. 산술부호의 확률구간 [0,1)을 아래와 같이 q 비트의 2진 소수로 나타낸다.

$$A(\lambda)=0.\underbrace{111\cdots\cdots1}_{q\text{bits}} \qquad (1)$$

길이 n-1의 계열 $\mathbf{X_{n-1}}=x_1x_2\cdots x_{n-1}$의 부호화가 끝난 상태에서 $\mathbf{X_n}$의 부호화를 위한 부분구간은 (2)식에 의해 분할된다.

$$\begin{aligned} A(\mathbf{X}_{n-1}\cdot 1)&=A(\mathbf{X}_{n-1})\times p(1)\\ A(\mathbf{X}_{n-1}\cdot 0)&=1-A(\mathbf{X}_{n-1}\cdot 1) \end{aligned} \qquad (2)$$

이때 계열 $\mathbf{X_n}$의 부호어 $C(\mathbf{X_n})$은 (3)식에 의해 구해진다.

$$C(\mathbf{X}_n)=\begin{cases} C(\mathbf{X}_{n-1}\cdot 0)=C(\mathbf{X}_{n-1}), & x_n=0\\ C(\mathbf{X}_{n-1}\cdot 1)=C(\mathbf{X}_{n-1})+\\ A(\mathbf{X}_{n-1}\cdot 0), & x_n=1 \end{cases} \qquad (3)$$

한편, 계열 $\mathbf{X'}_{n-1}$의 복호가 완료된 상태에서 $\mathbf{X'}_n=\mathbf{X'}_{n-1}\cdot x'_n$는 x'_n는

$$\begin{aligned} &C(\mathbf{X'}_n)-C(\mathbf{X'}_{n-1})<A(\mathbf{X'}_n\cdot 0)\text{이면, } x'_n\rightarrow 0\\ &\qquad\qquad\qquad\qquad\qquad\qquad\text{아니면, } x'_n\rightarrow 1 \end{aligned} \qquad (4)$$

로 복호화가 이루어진다.

이상에서 본 바와 같이 산술부호의 계산량은 구간 분할에 필요한 곱셈에 지배되며, 순차적으로 구간폭 A가 좁아지게 되므로 이진소수의 精度(precision)가 증가하는 문제가 발생한다. 이것을 해결하기 위해 Rissanen은 LPS의 확률을 2^{-Q}(Q : integer)로 근사화시켜 곱셈을 쉬프트 조작으로 대신하는 방법을 제안하고 있다.[43]

$$p(1)\doteq p(1)'=2^{-Q},\ 1:\text{LPS},\ Q:\text{skew number}$$

$$\begin{cases} A(\lambda)=0.111\cdots\cdots1,\ q\text{ bits precision}\\ C(\lambda)=0.000\cdots\cdots0 \end{cases} \qquad (5)$$

$$A(\mathbf{X}_{n-1}\cdot 1)=\langle A(\mathbf{X}_{n-1})\cdot 2^{-Q}\rangle_{q-Q} \qquad (6)$$

$$A(\mathbf{X}_{n-1}\cdot 0)=A(\mathbf{X}_{n-1})-A(\mathbf{X}_{n-1}\cdot 1) \qquad (7)$$

여기서, $\langle X\rangle_q$는 2진수 X의 유효 비트수 q비트로 절삭함을 의미한다. 길이 8인 계열 $\mathbf{X_8}=01001$

101,(Q=2, q=4, p(1)=1/4, p(0)=3/4)의 산술부호화의 과정을 표2에 보인다.

표 2. BAC에 의한 산술 부호화의 일례
Table 2. Arithmetic coding example using BAC.

x_n	$A(X_{n-1})$	$A(X_{n-1}\cdot 0)$	$A(X_{n-1}\cdot 1)$	$C(X_{n-1}\cdot x_n)$
0	.1111	.1100	.0011	.0000
1	.1100	.1001	.0011	.1001
0	.001100	.001001	.000011	.1001
0	.001001	.000111	.000010	.1001
1	.0001110	.0001011	.0000011	.1010011
1	.0000001100	.0000001001	.0000000011	.10101010 1
0	.00000001100	.00000001001	.00000000011	.10101010 1
1	.00000001001	.00000000111	.00000000010	.10101011011

산술부호의 효율(efficiency)은 최종 부분구간의 좌단점 $C(\mathbf{X}_n)$을 나타내기 위해 필요한 비트수로 주어지는 평균부호 길이(average code length) 로 평가된다. $C(\mathbf{X}_n)$의 유효 자릿수는 $A(\mathbf{X}_n)$의 자릿수와 동일하며, 정보원 계열 $\mathbf{X}_n$에서 LPS가 출현하면 Q만큼 증가하게 된다. $(1-2^{-q})^d \leq 1/2$을 만족하는 d에 대해, $\mathbf{X}_n$중에 MPS가 d번 나타나면 유효 자릿수는 한 자릿수 증가한다. 길이가 충분히 큰 n에 대하여 LPS는 np, MPS는 n(1−p)회 출현하므로 $A(\mathbf{X}_n)$의 소숫점 이하의 자릿수는,

$$npQ+\frac{n(1-p)}{d} \quad =npQ-n(1-p)log(1-2^{-q}) \qquad (8)$$

로 된다. 따라서 한 기호당 평균부호길이 $L_n=pQ-(1-p)log(1-2^{-q})$로 되어, 유효 자릿수 q를 크게 취하고 $n\to\infty$일때, $L_n\to H(\mathbf{X}_n)$에 접근하는 점근적 최량성(asymptotical optimality)이 보장되는 부호이다. 여기서, $H(\mathbf{X}_n)$은 계열 $\mathbf{X}_n$의 엔트로피이다.

BAC를 이용한 B/W FAX 영상의 압축효과는 G3 FAX의 표준방식인 MR(Modified READ) 에 비해 20~30%의 압축율이 향상됨을 보이고 있어 많은 응용이 전개되고 있다.[43] 특히, CCITT와 ISO를 중심으로 한 JPEG(Joint Picture Expert Group) 및 JBIG(Joint Bi-level Image Group)의 표준화 활동에 의해 크게 발전하게 되었다. JPEG와 JBIG의 엔트로피 부호화(entropy coding)을 위해 IBM 중심의 Q-coder와 MITSUBISHI의 MEL coder 그리고 AT&T의 Minimax coder의 난생을 보게 되었다.[44] 이들은 모두 산술부호의 기본 개념을 기초로 하여 실용화를 위해 설계된 부호들이다. Q-coder[45]는 기호의 출현확률을 평가하여 확률 적응화를 실행하고 있다. Q coder에서는 LPS의 출현확률인 Q의 대표점이 미리 준비되어 있어서 이 Q를 가리키는 인덱스를 적응적으로 움직임으로써 동적 부호화(dynamic coding)가 가능하도록 한 특징을 갖고 있다.

5.2 Multi-level Arithmetic Codes(MAC)

FAX의 적용분야가 확대됨에 따라 종래의 문자나 그림과 같은 2진영상(binary image)만을 취급해온 FAX도 사진(photographic)과 같은 멀티 레벨 및 칼라의 영상 데이타를 취급하기에 이르렀다. 이와 같은 FAX신호의 부호화에는 자연 화상의 부호화와는 다른 무잡음형 부호화(noiseless coding)가 주류를 이루고 있으며, 정보원의 국소적인 성질(locally statistics)을 잘 반영하는 산술부호의 이용이 좋은 효과를 얻을 수 있다.

멀티 레벨 산술 부호는 BAC의 확장으로 쉽게 실현할 수 있다. 정보원 계열 $\mathbf{X}_n=x_1x_2\cdots x_n$, $x_i \in A=\{0,1,\cdots,k-1\}$, $p(0)\geq p(1)\geq\cdots\geq p(k-1)$에 대한 부호화는 아래와 같이 이루어 진다.

$$A(\lambda)=0.11\cdots1,\ q \text{ bits precision} \qquad (9)$$

$$A(x_i)=A(\lambda)\cdot p(x_i),\ x_i\in A \qquad (10)$$

$$C(x_i)=\begin{cases}\sum_{l=1}^{i-1} A(l), & x_i\neq 0 \text{ (1st symbol cordword)}\\ 0, & x_i=o\end{cases} \qquad (11)$$

$$A(\mathbf{X}_n)=A(X_{n-1}\cdot x_n)=A(X_{n-1}\cdot 1)\cdot p(x_n), \qquad (12)$$

$$C(X_n)=\begin{cases}C(X_{n-1})+\sum_{l=0}^{k-1}A(X_{n-1}\cdot l), & x_n\neq 0\\ C(X_{n-1}), & x_n=0\end{cases} \quad (13)$$

여기서 (10), (12)식의 곱셈은 다음과 같이 쉬프트 조작으로 대체하여 연산을 고속화 시킬 수 있다.

$$A(l)=\langle A(\lambda)\cdot 2^{Ql}\rangle_{q-Ql},\ 0<l\leq k-1 \quad (14)$$

$$A(0)=A(\lambda)-\sum_{l=1}^{k-1}A(l) \quad (15)$$

$$A(X_{n-1}\cdot 1)=\langle A(X_{n-1})\cdot 2^{Ql}\rangle_{q-Ql} \quad (16)$$

$$A(X_{n-1}\cdot 0)=A(X_{n-1})-\sum_{l=1}^{k-1}A(X_{n-1}\cdot l) \quad (17)$$

따라서, 멀티 레벨 산술부호의 평균부호 길이 L은 (18)식으로 주어진다.

$$L\fallingdotseq\sum_{l=1}^{k-1}\{p(l)\times Q_l\}-p(0)\log(1-\sum_{l=1}^{k-1}2^{Ql}) \quad (18)$$

산술부호를 실제로 구현함에 있어서의 문제점은 精度의 증가 및 자리올림(carry over)의 문제가 일어나므로 이를 위한 해결법이 강구되어야 한다. 精度증가의 문제는 쉬프트 조작(shift operation)에 의해 간단히 해결되며, 자리올림문제는 그림3의 비트 삽입법(bit stuffing)[43]으로 해결할 수 있다. 자리올림은 $C(X_{n-1})$ 및 $A(X_{n-1}\cdot x_n)$을 q개의 비트로 표시하여 덧셈 조작을 할 때 일어난다. $C(X_{n-1})$과 $A(X_{n-1}\cdot x_n)$의 덧셈으로 얻어지는 부호 계열(code string)을 상위 레지스터 V 및 M을 거쳐 전송로로 보내지게 되나, 그림3과 같이 V레지스터의 비트가 전부 1로 세트되어 있다면 그 상위의 M레지스터까지 자리올림의 영향을 주게되어 많은 비트 반전을 요구하게 된다. 더욱 파급되는 자리 올림은 전송이 끝난 부호 계열에까지 영향을 미치게 되므로 정상적인 정보전달이 불가능 하게 된다. 이를 위한 해결 방안의 하나로 Rissanen은 비트 삽입법을 제안하였고, MORITA[46]는 레지스터 V와 M에 걸쳐서 생기는 자리올림을 해결하기 위하여

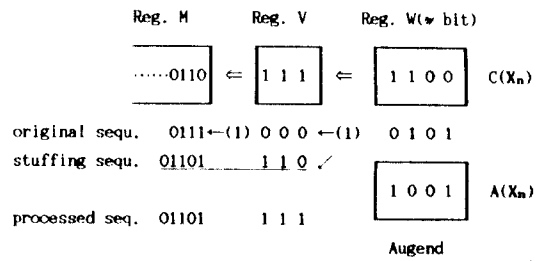


그림 3. 비트 삽입법의 일예
Fig 3. bit stuffing example.

비트 삽입법을 확장하는 방법을 제시하였다.

5.3 MAC의 고속화

데이타 압축의 또다른 목표는 알고리즘의 고속화이다. 높은 압축효율을 얻을 수 있는 산술부호의 계산량은 입력 데이타의 길이 n에 선형적으로 비례하는 $O(n)$으로 개선되어 실용적으로 응용 가능하게 되었다. 그러나, 많은 산술 인식 배분에 적응 사전법의 유니버셜 압축법에 비하여 처리속도가 대단히 느린 결점이 있다.[11] 한편, 산술부호와 같은 모델 베이스 유니버셜 압축법에서는 적응적으로 확률 파라메타를 갱신해야 하기 때문에 모델링과 부호화를 함께 고려함이 바람직하다.

차분모델[47]과 산술 부호의 일종인 Pasco의 FV부호(Fixed to Variable codes)[48]를 이용하여 gray level image를 압축시의 산술부호화를 고속화하는 한 방법을 보인다.[47] Pasco의 산술부호는 Elias부호의 $O(n^2)$의 계산량을 $O(n)$으로 개선시킨 것으로 알고리즘이 간단하며 Elias부호를 가장 충실히 실현하는 부호이다. 길이 n의 정보원 계열 $X_n=x_1x_2\cdots x_n$에 대하여 각 기호 $x_l\in A=\{0,1,\cdots,k-1\}$의 출현확률 $p(x_l)$를 J비트 이하의 2진 小數로 나타낸다. 이때 $p(x_l)$는 다음의 조건을 만족해야 한다.

$$0\leq p(x_l)<1,\ \Sigma p(x_l)=1 \quad (19)$$

또한 각 기호 x_l에 대한 누적확률 $c(x_l)$는

$$c(\lambda)=0 \tag{20}$$

$$c(x_i)=\sum_{j=1}^{i-1} p(x_j) \tag{21}$$

$$c(x_i)+p(x_i)\leq 1 \tag{22}$$

로 된다. Pasco의 산술부호화 알고리즘은 다음과 같다.

〈encoding process〉

(1) $F\rightarrow 0$

(2) $T\rightarrow 1$

(3) For i=1 to n do

(3.1) Load the $p(x_i)$ and $c(x_i)$

(3.2) $F_i\rightarrow F_{i-1}+T_{i-1}\times c(x_i)$

(3.3) $T_i\rightarrow T_{i-1}\times p(x_i)$, (truncated to q siginificant digits)

(4) $L\rightarrow r+1(T_n=t_0t_1t_2\cdots t_{q-1}\times b^r,\ t_0\neq 0)$, $\lceil -log_b T_n\rceil+1)$

(5) $F_n\rightarrow F_n+b^L$

(6) transmit the L digits of F_n. (code string $Y_L=y_1y_2\cdots y_L$)

위 알고리즘에서 각 변수(F,T,c,p)는 小數이지만 산술연산을 고속으로 하기 위하여 모두 정수를 이용한다. 이를 위해 출현확률 $p(x_i)$와 누적확률 $c(x_i)$를 2^{16}배 하므로 (19)식의 총합은 2^{16}이 된다. 또 T와 F의 값은 부호화와 동시에 점점 작아지나 T의 유효 자릿수를 q=16으로 하여 F와 함께 순차적으로 shift out한다. 그러나 T와 F는 알고리즘(3.2)에서 곱셈과 덧셈이 이루어지므로 32비트의 정수를 이용한다. F로부터 shift out되는 상위의 비트가 해당 부호어로써 출력되지만 자리 올림 문제를 해결하기 위하여 비트 삽입법을 부가하고 있다.

또한, 멀티 레벨 산술 부호의 실제의 연산속도는 encoding process의 (3)에 크게 의존한다. (3.1)의 $p(x_i)$와 $c(x_i)$는 정보원 모델링의 갱신과 함께 동적으로 계산되는 량이므로 거의 일정한 파라메타이나, F와 T는

$$F_i=\begin{cases} F_{i-1}+\sum_{j=0}^{y-1} T_{i-1}\times p(j), & y\neq 0 \\ F_{i-1}, & y=0 \end{cases} \tag{23}$$

$$T_i=T_{i-1}\times p(y),\quad y: x_i\text{의 gray-level value} \tag{24}$$

로 구해진다. 적응 산술부호화에 필요한 연산량은,

(1) 출현확률 $p(x_i)$를 위한 나눗셈

(2) 부분구간 F_i를 위한 곱셈 및 덧셈

(3) 각 부분구간을 누적하기 위한 덧셈 및 뺄셈

등이다. 따라서 상대적인 연산량을 줄이기 위해서는 높은 확률의 기호에 해당하는 부분구간일수록 분할의 시점 가까이에 위치 시키면 된다. 이 착상을 근접 화소의 차분값이 y=0 부근에 집중하는 차분모델에 적응시키는 것은 용이하다. 즉 차분값을 확률구간의 시작점으로부터 $0,-1,1,\cdots,-(k-1),(k-1)$의 순서로 설정하면 필요한 연산량이 대폭 줄어들게 되므로 고속처리가 가능하게 된다.

Ⅵ. 결 론

이 논문에서는 다양한 정보원에 적용할 수 있는 유니버셜 부호의 기본 개념을 보이고 그 응용을 중심으로 한 최근의 연구 동향에 대하여 알아보았다. 문서통신의 표준방식으로 사용되어 온 MH부호의 기본인 Huffman부호를 적응적으로 처리하는 새로운 방식의 등장과 자연수를 효과적으로 표현하기 위한 유니버셜부호는 데이타 압축시스템의 통합적인 구축을 위해 효과적인 방법이 될 것이다.

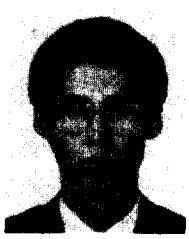
적응 사전법의 대표적인 방식인 Ziv-Lempel 부호를 알고리즘 중심으로 나타내었으며 각종 개량방식 및 실용화 방식에 대하여 알아 보았다. 현재 PC통신의 압축 유틸리티의 주류를 이루고 있는 PKARC, LZH, ZIP 및 LHARC는 Ziv Lempel부호의 응용인 점을 주시하면 보다 활발한 연구 환경의 조성으로 우리의 고유

한 알고리즘의 개발이 시급한 실정이라 하겠다. 또한 FAX를 비롯한 영상 데이타의 압축을 위한 모델과 산술부호에 대하여 그 기본 원리로부터 확장 및 응용에 이르기까지 알아보았다. 그러나 여기서 소개한 많은 부호들 가운데 그 성능에 관하여 이론적으로 또는 시뮬레이션등으로 자세하게 해석 평가된 것은 극소수에 지나지 않는다. 따라서 각 부호의 개량법등에 비교적 쉽게 얻어질 수도 있으므로 이 분야의 중요자료로 활용될 것을 기대한다.

참 고 문 헌

1. T. Berger., "Rate distortion theory", prentice Hall(1971).
2. C.E.Shannon., "the mathematical theory of communication", University of Illinois Press, Urbana (1949).
3. D.A.Huffman., "A method for the construction of minimum-redundancy codes", Proc. IRE, 40, 9, pp. 1098-1101 (Sept. 1952).
4. R.E.Blahut., "Principles and practice of information thoery", Addison-wesley.
5. D.A.Lelewer and D.S.Hirschberg., "Data Compression", ACM Computing surveys, 19, 3, pp.261-296 (Sep. 1987).
6. J.Rissanen., "Generalized Kraft inequality and arithmetic coding", IBM J.Res.Dev.20,pp.198-203 (May 1979).
7. D.E.Knuth., "Dynamic Huffman coding", J.Algorithms, 6,2,pp.163-180 (Jun, 1985).
8. A.Lempel, J.Ziv., "On the complexity of finite sequences", IEEE Trans, IT-22, 1, pp.75-81 (Jan, 1976).
9. L.Bentley et al., "A locally adaptive data compression scheme", CACM 29, 4, pp.320-330(Apr.1986).
10. F.M.Willems., "Universal data compression and repetition times", IEEE Trans. IT-35, 1,pp.54-58 (Jan, 1989).
11. J.H.PARK and H.IMAI., "Data compression", KITE 15, 1,pp.99-105 (2-1988).
12. N.Faller., "An adaptive system for data compression", Record of the 7-th Asilomar Conf. on Circuits, System and Computers, pp.593-597, (1973).
13. R.G.Gallager, "Variations on a theme by Huffman", IEEE Trans. IT-24, 6,pp.668-675 (Nov.1978).
14. J.S.Vitter., "Design and analysis of dynamic Huffman coding", JACM 34, 4,pp.825-845 (Oct.1987).
15. E.S.Schwartz., "An optimum encoding with minimum longest code and total number digits", Inf. Contorl, 7,1,pp.37-44 (mar. 1964).
16. C.L.McMaster., "Documentation of the compact command", In UNIX User's Manual, 4.2 BSD, Virtual VAX-11 Version, Univ.of Califo, Berkeley (Mar. 1984).
17. 김경내, 박용석., "효율적인 한글코드화에 관한 연구", 한국통신학회논문지 14, 6, pp.633-641 (12.1989)
18. 김현찬 외 3인., "허프만 부호를 이용한 압축 및 암호 결합 알고리즘", 한국정보과학회 '91 봄학술발표논문집 18.1 (4.1991).
19. J.A.Storer., "Data compression : methods and theory", Compu. Sci. Press(1988).
20. H.Yamamoto et al., "Ziv-Lempel符號の改良とシミュレーションによる性能評價", IEICE Tech. Report CS84-135, pp.1-8 (1.1985).
21. P.Elias., "Universal codeword sets and representations of the integers", IEEE Trans IT-21, 2, pp.194-203 (Mar.1975).
22. J.H.PARK et al., "An adaptive data compression using self organizing rule", IEICE Trans J72-A, 8, pp.1353-1359 (8.1989).
23. A.Apostolico et al., "Robust transmission of unbounded strings using Fibonacci representations", IEEE Trans IT-33, 2, pp.238-245 (Mar.1987).
24. H.Yamamoto et al., "有限 正整數に對する2値符號化法について", SITA 12, 2, pp.559-564 (12.1989).
25. J.Ziv and A.Lempel., "An universal algorithm for sequential data compression", IEEE Trans IT-23, 3, pp.337-343 (May. 1977).
26. J.Ziv and A.Lempel., "Compression of individual sequences via variablerate coding", IEEE Trans. IT-24, 5, pp.530-536 (Sept.1978).
27. PKARC FAST., "PKARC FAST File Archival Utility", Version 3.5 PKWARE, Inc. Glendale, Wis. (1987).
28. D.E.Knuth et al., "Fast pattern matching in strings", SIAM. J. of comput., Vol.6, No.2 pp.323-350 (June. 1977).

29. M.Rodeh et al., "Linear algorithm for data compression via string matching", J. of ACM, 28, 1, pp. 16-24 (Jan.1981).
30. J.H.PARK et al., "Practical Data Compression Methods using Pattern Matching and Arithmetic Codes", IEICE Trans J71-A, 8, pp.1615-1623 (1988. 8).
31. T.A.Welch., "A technique for high-performance data compression", IEEE computer, pp.8-19 (June.1984).
32. S.W.Thomas et al., "Compress(version 4.0) program and documentation", available from joepetsd. UUCP.
33. T.M Cover, "Enumerative source coding", IEEE Trans IT-19.1,pp.73-77 (1.1973).
34. T.Kawabata et al., "A new implementation of Ziv-Lempel incremental parsing algorithm", submitted to IEEE Trans Information Theory.
35. R.N.Horspool., "A locally adaptive data compression scheme", CACM, 30,9, pp.792 (Sept.1987).
36. J.A.Storer., "Parallel algorithms for On-line data compression", IEEE ICC'88,pp.385-389 (1988).
37. E.R.Fiala et al., "Data Compression with finite windows", CACM, 32, 4, pp.490-505 (April.1989).
38. 村島,中村,廣田., "文字列登録を強化したLZW法によるソースファイルの壓縮", IEICE Trans J73-A, 9, pp. 1529-1533 (Sept.1990).
39. 森田,小林., "再生可能な文字列分解にもとづくユニバーサルデータ壓縮について", SITA, 13, 2(1.1991).
40. N.Abramson., "Information theory and coding", New York McGraw-Hill, 1963.
41. J.Rissanen and G.G.Langdon., "Universal modeling and coding", IEEE Trans. IT-27, 1, pp.12-23 (Jan. 1981).
42. G.G.Langdon., "An introduction to arithmetic coding", IBM J.Res. Dev., 28, 2, (1984).
43. G.G.Langdon and J.Rissanen., "Compression of B/W images with arithmetic coding", IEEE Trans. COM-29, 6, pp.858-867 (1981).
44. H.Yasuda., "International sandardization of high efficency coding", J, ITE, 43, 10, pp.1011-1019, (1989).
45. Q-Coder Special, "Q-Coder adaptive binary arithmetic coder", IBM J. Res. Dev. 32, 6, pp.715-840 (Nov. 1989).
46. H.Morita et al., "On efficiency of binary arithmetic codes", J. IPA, 25, 4, pp.622-631 (1984).
47. J.H.Park and H. Imai., "Arithmetic coding for gray-level image data compression", PCSJ 89, 7-18, (1989).
48. Pasco R.C., "Source coding algorithm for fast data compression", Ph.D Disseration Stanford Univ. (1976).



朴 志 煥(Ji Hwan PARK) 正會員
1958年 2月 6日生
1987年 3月 : 日本電氣通信大學 應用電子工學科(工學修士)
1990年 3月 : 日本北海道大學 電子情報工學科(工學博士)
1990年 3月~現在 : 釜山水産大學校 電子計算學科 專任講師

陳 庸 玉(Yong Ohk CHIN) 正會員
1943年 3月 21日生
1968年 2月 : 延世大學校 電氣工學科 卒業(工學士)
1975年 2月 : 延世大學校 大學院 電子工學科(工學碩士)
1981年 2月 : 延世大學校 大學院 電子工學科(工學博士)
1975年~1978年 : 光云工科大學 通信工學科 教授
1980年~現在 : 慶熙大學校 電子工學科 教授
1980年 通信技術士