特別寄稿 · 日本支部

유니버샬 데이타 압축 (Universal Data Compression)

朴志煥, 今井 秀樹
(正 會 員)
日本 橫浜國立大學 工學部

Ⅰ. 서 론

情報化 社會의 發展과 더불어 다양하고도 대량의 데이타를 효율좋게 보관하거나 傳送하기 위한 데이타 압축의 必要性이 크게 要求되고 있다. Shannon의 情報理論으로부터 出發한 데이타 압축의 基本原理는, 情報源 系列(source sequence)에 포함된 redundance를 제거하여 보다 짧은 系列로 변환시키는 符號化 技法이다.

압축된 系列로부터 원래의 계열을 復元할 때, 復元 error가 zero가 되는 無雜音壓縮(noerror compression)과 어느정도 error를 허용하는 有雜音壓縮(compression wrth distortion)으로 분류된다. 여기서 紹介하는 壓縮法은 計算機코드처리에 유효한 前者의 경우에 한하여 고찰하고자 한다.

最適符號化法으로 잘 알려진 Huffman符號[1]는 입력 데이타의 出現빈도를 조사하여 그 빈도가 높은 記號일수록 짧은 符號를, 빈도가 낮은 記號에는 길이가 긴 符號를 割當하여 平均符號의 길이 (average length of a code)를 보다 짧게하는 이른바 情報源의 統計的 性質을 利用하는 符號이다.

그러나, 情報源의 確率統計的인 性質을 前提로 하지 않고도 그 理論的 限界인 entropy壓縮이 가능한 유니버샬 데이타 압축법이 最近에 주목을 받고 있다. 이와같은 符號의 硏究는 Lynch,[2] Davission,[3] Schalkwijk[4], Lawrence[5] 로부터 Ziv-Lempel 및 Rissanen에 이르고 있다.

Ziv-Lempel 符號는, 記號列의 random性(complexity)를 나타내는 情報量의 計算法을 基本으로 하고 있다. 그 압축 algorithm은 記號列의 producible性을 利用하여 入力데이타系列을 서로 다른 部分列로 分解함으로써 符號化가 이루어지는 간단하고도 고속성을 갖고 있는 符號이다.

이 符號는 임의의 情報源에 적용 가능한 만능성(universality)를 갖고 있으며, 입력데이타가 ergodic 및 stationary한 경우, 그 압축율은 漸近的 最良性(入力데이타의 길이 n이 충분히 클 때 entropy까지 압축이 가능)이 証明되어 있다.[6]

한편, Rissanen符號는 context gathering과 context selection를 利用한 모델을 設定하여, 선택된 context의 確率을 算術符號化(arithmetic coding)하는 方法을 취하고 있다.

이 두 符號는 전혀 無關係한 서로 다른 방법을 취하고 있는 것 같지만, 入力데이타의 構造를 學習한 結果를 基本으로 符號化를 實行하는 점에서 關係가 깊은 모델을 共通의 基礎로 하고 있다고 볼 수 있다.

여기서는, 2장에서 Ziv-Lempel符號의 두가지의 구체적인 algorithm을 중심으로 고찰함과 더불어, 有限길이의 입력에 대해 압축성능이 우수한 改良된 algorithm을 소개한다. 또한 3장에서는, Rissanen 符號를 適應的(adaptive)으로 實現하여, Ziv-Lempel 符號와의 관계를 壓縮率 및 complexity를 中心으로 고찰한다.

Ⅱ. Ziv-Lempel 符號

A. Lempel과 J. Ziv는 系列의 情報量을 서로 다른 最大의 部最大의 部分列로서 정의하여, 그것을 實現(implement)하기 위하여 copy를 이용하고 있다. 系列의 copy를 이용하여 各系列 고유의 情報量을 設定하여, 어느 意味에서 確率 모델을 고려하지 않고도 entropy 압축이 가능함을 보였다.[1]

具體的인 實現法으로서 universal algorithm[2]과

incremental algorithm[3]이 있다. 前者가 buffer위의 「임의의 位置로부터 임의의 길이」에 producible을 利用하는 반면, 後者는 정해진 位置로부터 이미 분해된 과거의 部分列에 대해서만 producible을 적용시키는 制限된 방법을 사용한다. 따라서, 壓縮率은 前者가 좋지만, 장치화 및 complexity에 있어서는 後者가 우수하다고 생각된다.

1. Universal Algorithm

1) Ziv-Lempel의 情報量

이 情報量를 정의하기 위한 重要한 두가지 性質(reproducible과 producible)을 간단히 소개한다.

記號列 R=0010101에 대하여, S=001, Q=0101로서 R=SQ라고 하자. String Q는 string S의 부분열(substring) 01를 2回 반복하여 copy함으로써 얻어진다. 일반적으로, R=SQ에서 Q가 S로부터 copy가능하면 string R은 S로부터 reproducible이라고 한다. 한편, Q가 S로부터 copy가 불가능하지만 列Q의 가장 오른쪽의 記號를 제거한 stringQ′가 S로 부터 copy가능하다면, 列R은 S로부터 producible 이라고 한다. 따라서, 임의의 系列은 길이가 zero인 空列(null string)로부터 producible을 반복하는 것에 의해서로 다른 부분열로 분해가 가능하다.

Lempel과 Ziv는 이와같이 얻어진 部分列의 最小個數로서 記號列의 random性을 나타내는 情報量 C_{LZ}를 定義하고 있다. 그 理論的 特性은 문헌[6, 7]에 상세하게 解說되어 있다.

2) VF(variable-to-fix)符號와 VV(variable-to-variable)符號

C_{LZ}를 實現시키기 위한 具體的인 algorithm은 앞에서 說明한 두가지 방법이 있다. 時期的으로 먼저 提案된 universal algorithm은, producible을 利用하여 입력데이타를 부분열(substring)로 분해하여 각 부분열을 生成하기 위한 情報〔copy위치, copy길이, 다음의 記號〕를 고정길이(fixed length)의 符號語(codeword)로 변환한다. 그것을 實現하기 위한 符號器의 구성은 그림 1과 같이 길이가 p, q인 buffer P와 Q를 利用한다.

符號器는 Q buffer의 선두에서 시작되는 記號列

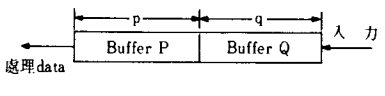


그림 1. 符號器의 構成

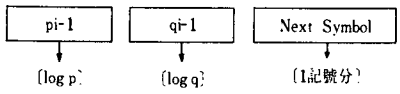


그림 2. VF 符號語의 format

과 producible 관계에 있는 最長(maximum length)의 記號列을 P buffer로부터 search한다. 이때 사용되는 壓縮符號의 format은 그림 2와 같이 주어진다.

단, 「x」는 x이상의 最小整數를 나타내는 ceiling function이다. 여기서, p_i는 時點 i에 있어서 pattern matching의 start position이며, q_i는 producible에 의한 string matching의 길이이다. 따라서, q_i-1은 reproducible에 의한 길이임에 주의가 必要하다. 이상의 결과, producible를 利用한 符號法에서는 「위치」, 「길이」, 「다음의 記號」의 3개의 要素로 구성되는 VF의 고정 format이 항상 必要하게 된다.

이와같은 producible에 의한 VF符號의 redundance를 살펴 보고자 한다.

① Pattern matching이 역효과(초기의 계열)임에도 불구하고, 그림 2의 고정 format을 사용함에 의해 出力系列의 길이가 오히려 팽창하게 된다.

② 「다음의 記號」는 「길이」가 0이외의 경우는 意味가 없다.

③ 「다음의 記號」의 shift에 의해 pattern matching의 기회가 없어진다

④ 符號化의 초기에는, 「위치」를 나타내기 위한 상위 bit가 필요없다.

上記의 缺點을 解決하기 위한 方法으로서, reproducible을 利用한 VV(variable-to-variable) 符號의 format을 그림 3에 나타냈다.

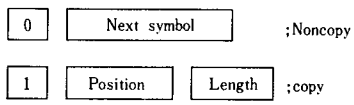


그림 3. VV 符號語의 format

1 bit flag를 設置하여 pattern matching을 실시할까 하지 않을까를 표시한다. 즉, pattern matching에 의한 역효과의 경우에는 flag를 0으로하여 초기의 압축율을 1 bit만의 손실로 제한할 수 있으며, 또한 P buffer상의 記號의 增加에 따라 「위치」를 나타내는

bit數를 可變시켜 ④의 문제를 해결할 수 있다.

이상의 결과, Ziv-Lempel이 강조한 分解部分列의 數는 VF符號法에 비해 증가하지만, variable length를 사용함으로써 表現上의 장점을 살릴 수있다. VF符號와 VV符號의 性能를 評價하기 위하여 computer simulation하여, 各種 file〔C source program, excutable program, english text〕에 대해 壓縮性能을 조사하였다. 일부를 그림 4 에 表示하였다.

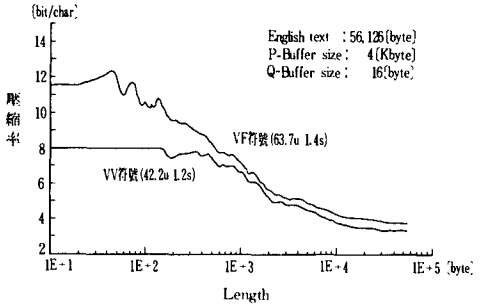


그림 4. 圧縮率의 變化

P, Q buffer를 각각 4 kbyte, 16byte로 구성하여, 56, 126byte의 english text에 대한 壓縮率의 變化를 나타냈다. 여기서, 壓縮率은 bit/character이며 符號化 및 復號에 소요된 時間은 PDP-11/73 상에서 實現한 時間이다. U와 S는 각각 user time과 system call을 μs로 表示하였다.

한편, 情報源系列(source sequence)이 binary인 경우는 Ziv-Lempel의 VF符號로 實現해도 다음의 기호(next symbol)는 送信할 必要가 없음이 명백하다.

2. Incremental Algorithm[8]

이 符號化法은 前節의 VF符號의 pattern matching을 간략화한 방법으로서, 이미 分解된 部分列에 1記號를 추가시켜 새로운 部分列를 形成하는 제한된 分解法을 利用하고 있다.

1) Incremental parsing

入力데이타 $x=x_1, x_2 \cdots x_n$을 部分列 $x=X_1, X_2 \cdots X_P$ $(n \gg P)$로 분해할 때, 새로 만들어지는 j번째의 部分列을 X_j라고 하면, 다음의 성질이 성립한다.

① X_j는 과거의 部分別 $X_0, \cdots, X_{j-1}$ 중의 어느 것과도 일치하지 않는다.

② X_1의 길이는 1이다(단, X_0는 길이가 0인 空列임)

③ X_j'(X_j의 우측의 기호 $x(n_j)$를 제거한 계열)와 일치하는, 과거의 부분열 $X_i (i<j)$가 반드시 1개 존재한다.

이상의 결과, 各部分列 X_j는 다음과 같이 表現된다.

$$X_j = X_i \cdot x(n_j) \qquad (1)$$

이와같은 incremental parsing은 分解木(parsing tree)을 사용하면 간단히 실현할 수 있다.[9]

「예를들어, 2進 입력데이타 x=001010100의 分解과정을 2進分解木으로 나타내면 그림 5와 같다.

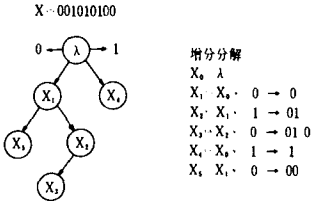


그림 5. 2進分解木과 分解과정

2) 外符號化(outter coding)

Incremental parsing으로부터 얻어진 各部分列 X_j에 대한 2進符號 C_j는 參照部分列 X_i와 우측의 기호 $x(n_j)$로부터 다음과 같이 주어진다.

$$C_j = 2 \cdot i + x(n_j), \quad i \leq j-1, \quad x(n_j) \in \{0, 1\} \qquad (2)$$

C_j의 upper bound는 $C_j = 2 \cdot j - 1$로 되어 j번째의 부분열 X_j를 符號化 하기 위해 필요한 bit수 L_j는 (3)式으로 주어진다.

$$L_j = \lceil \log j \rfloor + 1 \qquad (3)$$

따라서, 符號길이 L_j는 分解部分列의 番號 j와 더불어 단계적으로 증분하는 incremental 符號이다.

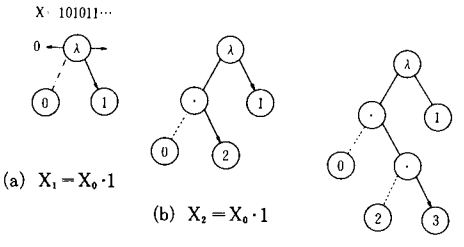
Incremental 符號는 分解木을 사용함으로서 고속으로 符號化, 復號가 이루어지어지며, 壓縮率에 있어서는 漸近的 最良性이 보장되어 있지만, 有限데이타에서의 特性은 만족할 만한 결과를 얻지 못하고 있다. 따라서, 실용성의 관점에서 압축율을 개선하고자 하는 改良 Algorithm을 소개한다.

3) 擴張分解木—改良A

Incremental parsing의 성질②를 만족시킬 때, 分

解木의 내부 node는 절대로 참조되지 않기 때문에 그곳에는 參照番號를 붙이지 않음으로써 압축율을 개선할 수 있다.

구체적인 방법으로서, parsing에 의해 얻어진 X_j에 대하여, 우측의 $x(n_j)$를 반전시킨 $X_i \cdot \overline{x(n_j)}$도 동시에 출현했다고 간주하는 확장된 parsing法을 사용한다.[10] 입력데이타 x=101011에 대한 parsing의 결과를 그림 6 에 나타낸다.



(a) $X_1 = X_0 \cdot 1$ (b) $X_2 = X_0 \cdot 1$ (c) $X_3 = X_2 \cdot 1$

그림 6. 擴張分解木

이상의 결과, 外部 node에만 參照番號 i가 할당되어지게 되며, 分解部分列의 數가 감소되어 특히 初期의 壓縮率의 향상의 효과를 期待할 수 있다.

4) 參照番號 符號化－改良B

한편, 外符號化에 필요한 bit數는 (3)式으로 주어지지만, 參照番號 i를 表現하기 위하여 「log j」digit의 bit가 필요한 것은 j가 2의 整數乘일때 만이다.

즉, 「log j」－log j의 loss가 존재하게 된다. 그 손실을 방지하기 위한 algorithm은 다음과 같다.[11]

$j \neq 2^k$일때 $(k=1, 2, \cdots)$

① $j^* > i^*$이면,

參照番號 i를 길이 「log j」digit의 2進 符號化하여 그 逆順으로 出力한다.

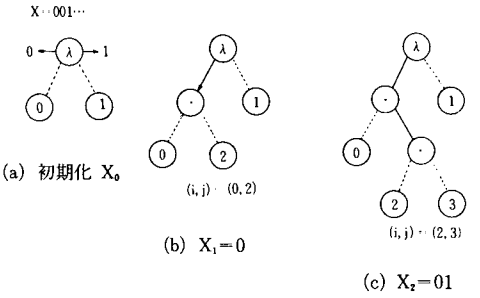
② $j^* \leq i^*$이면,

i^*를 길이 「log j」－1 digit의 2進 符號化하여 그 逆順으로 出力한다.

단, X^*는 정수 X의 「log j」digit 表現에서 최상위 bit〔MSB〕를 제거한 정수를 의미한다.

5) 修正Z-L符號－改良C

Incremental 符號의 外符號化에 있어서, 각 部分列의 최후의 記號 $x(n_j)$를 송신하지 않는 방법을 취하고 있다.[12] 이 방법을 실현하기 위하여 修正 Z-L 木을 사용한다. Tree의 作成은 그림 7 (a)를 초기상태로 하여, 내부 node에는 이미 出現한 既部分列를 나타내고, 外部 node에는 다음에 出現可能한 部分列를 대응시킨다. (그림7(b, c))



(a) 初期化 X_0 (b) $X_1 = 0$ (c) $X_2 = 01$

그림 7. 修正 ZL木

修正ZL木으로부터 j번째 얻어진 參照番號 i를 다음의 條件에 따라 符號化한다.

① $0 \leq i \leq 2^{\lceil \log(j+1) \rfloor} - j - 1$ 이면,

i를 「log j」digit로 2進 符號化한다.

② $2^{\lceil \log(j+1) \rfloor} - j \leq i \leq j - 1$ 이면,

$2^{\lceil \log(j+1) \rfloor} - j + 1$을 「log j」+1 digit로 2進 符號化한다.

6) 計算機 simulation에 의한 性能評價

Incremental algorithm을 포함한 3가지의 改良 algorithm에 대한 simulation 結果의 일부를 그림 8에 나타냈다.[13] 入力데이타는 理論的 限界(entropy)의 計算이 용이한 1重 markov sequence를 사용하였다. 그 천이확률 및 상태천이도를 그림 9 에 표시했다. 여기서, 압축율은 出力데이타의 길이와 入力데이타의 길이의 比로서, 이것을 될 수 있는 한 작게 하는 것이 데이타압축의 基本問題이다.

各 改良 algorithm에 의한 압축율은 Incremental algorithm(Z-L)에 비해, 특히 초기단계에서 많은 향상을 보이고 있으며, 改良에 의한 計算量의 증가는 그다지 크지 않기 때문에 별로 문제가 되지 않음을 알 수 있다.

改良A는 內符號化에 있어서의 部分列의 數가 줄어든 結果이며, 改良B는 部分列의 分解의 數는 改良A와 같지만, 外符號化의 表現의 最適化를 실시한 효과이다. 改良C의 경우는, 分解는 incremental parsing이지만 다음에 出現可能한 부분열만을 대상으로

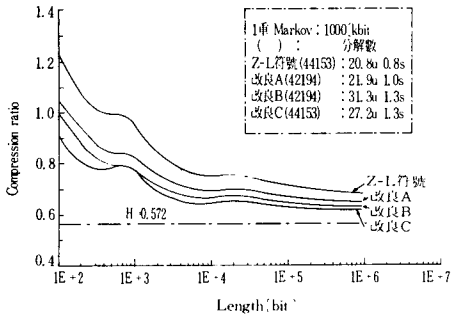


그림 8. 圧縮率의 比較

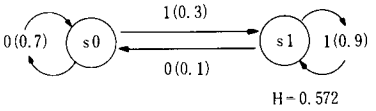


그림 9. 實驗 data의 遷移確率

함으로써, 결과적으로 1개의 部分列에 對하여 1記號 節約된 效果를 보이고 있다.

Ⅲ. Rissanen符號

Rissanen符號는 incremental parsing에 文脈收集(context gathering)의 學習能力을 도입한 情報源의 model化를 基本으로 하고 있다.[4] Model化에 의해 얻어진 conditional probability는 情報源의 擴大없이 高能率壓縮이 可能한 算術符號와의 組合에 의해 그목적을 달성할 수 있다.

한편, 구체적인 algorithm의 實現에는 모델링하면서 符號化를 실시하는 adaptive한 方法[15]과 1 pass에서 情報源의 性質을 파악하여 parameter를 推定한 후, 그 parameter를 利用하여 2 pass에서 符號化를 실시하는 nonadaptive한 방법[16]으로 分類된다.

여기서는 實用的으로 重要한 adaptive한 방법으로 rissanen符號를 構成했다.

壓縮의 性能을 좌우하는 modeing은, 入力데이타에 따라 文脈木(context tree)을 만드는 文脈收集(context gathering)과 만들어진 文脈木으로부터 conditional probability를 구하는 文脈選擇(context gathering)으로 이루어진다.

1. Context Gathering

Ziv-Lempel符號의 parsing tree가 本質的으로, 임의의 入力데이타 계열에 대하여 學習을 하고 있음을 지적한 Rissanen은 그 學習技能을 적극적으로 이용한 context gathering을 정의하였다. 즉,完全 2進木을 利用하여 각 context Z의 出現回數 C(Z) 및 그 context Z의 바탕아래 記號 u= {0, 1}의 發生回數 C(U, Z)의 計算을 효과적으로 表現하고 있다.

각 node는 count pair[C(0, Z), C(1, Z)]를 갖으며, context Z란 root node로부터 그 node에 이르는 pass $Z_1 \cdots Z_n$을 나타내는 string이다. 入力데이타 계열을 x(1), x(2)…x(n)로 했을 때, context tree T(i)는 다음의 algorithm에 의해 만들어진다.

1) 初期化 : λ C(0, λ) = C(1, λ) = 1

2) 現在의 入力 x(i) = u, u= {0, 1}에 대하여,

u의 context Z(=x(i−1) x(i−2)…x(2), x(1))에 따라, root node로부터 context tree T(i−1)를 node의 count가 1인 node W에 도달할 때까지 search한다. 이때, search된 각 node의 count를 1씩 증가시킨다.

(1) W가 내부 node이면, C(U, W1)와 C(U, W1)을 1씩 증가시킨다.

(2) W가 外部 node이면, Wϕ, W1의 새로운 node를 만들어 각 count를 다음과 같이 更新한다.

C(U, Wϕ) = C(U, W1) = 1,

C($\bar{U}$, Wϕ) = C($\bar{U}$, W1) = 0

단, $\bar{U}$는 U의 complementary이고, Wϕ, W1는 각각 node W에 記號 0과 1을 concatenation한 node에 해당한다.

2. 確率의 表現

文脈 Z에 對하여, Zϕ와 Z1를 Z에 ϕ과 1를 각각 concatenation한 것이라면, C(Z)는 C(0, Z)와 C(1, Z)의 합이 된다.

① Context Z에 있어서 記號U의 conditional probability;

$$P(U/Z) = C(U, Z)/C(Z) \quad (4a)$$

C(U, Z) = 0 일때,

$$P(U/Z) = 1/C(Z) + 1 \quad (4b)$$

② Conditional probability에 의한 binany entropy;

$$H(U;Z) = -\sum_{u} P(U/Z) \log P(U/Z) \quad (5)$$

③ Context Z의 出現確率;

$$P(Z) = C(Z)/C(\lambda) \quad (6)$$

3. Context Selection

Context gathering으로부터 얻어진 count pair를 利用하여, MDL(minimum description length)을 만족하는 context가 결정되면 그 context의 conditional probability를 算術 符號化한다.

① Contidonal entropy;

$$H(U/Z)=\sum P(z)H(U;Z) \quad (7)$$

② Conditional entropy의 增分;

$$\triangle(t,z)=P(z)H(U;z)-P(z\phi)H(U;z\phi)-P(z1)H(U;z1) \quad (8)$$

③ Selection rule

記號 $U=x(t)$에 대한 context $Z^*(t)$는 다음의 條件을 만족하는 node를 선택한다.

$$\triangle(t,z)>\frac{(\log t)}{2t} \quad (9a)$$

$$|Z|\leq\beta*(\log t) \quad (9b)$$

$$\min\{C(z\phi),C(z1)\}\geq\frac{2\alpha t}{\sqrt{\log t}} \quad (9c)$$

여기서, α와 β는 有限列의 入力에 대하여 search되어야 할 node의 범위를 결정하는 parameter로서 非負整數이다. 만약, 이 조건을 만족하는 node가 존재하지 않으면 root node를 선택한다. 따라서, 계열 $S=x(1),x(2)\cdots x(t)$에 대한 모델링에 의해 달성가능한 最適符號길이(optimum code length)는 선택된 context $Z^*(t)$의 condional probability $P((x(t)/Z^*(t))$에 의해 주어진다.

$$L=-\log P(S)=-\sum_{i=1}^{t}\log P(x(i)/z^*(i)) \quad (10)$$

4. 性能比較

Rissanen 符號의 性能을 評價하기 위하여 m重(m=0, 1, 2, 3) Markov sequences의 入力데이타에 대하여 simulation를 실시하였다.[17] 모델에 의해 달성가능한 압축율을 incremental 符號의 改良C(mod Z-L)와 비교하여 表1에 나타냈다. 그 결과 길이 10kbit의 입력으로써 entropy에 상당히 접근했으며, 改良C(mod Z-L)에 비하여 壓縮率이 훨씬 양호함을 알 수 있다. 선택되는 context의 次數를 나타내기 위하여 m重 Markov source에 대한 局所次數의 推定(estimation of local order) 관계를 表2에 表示 하였다.

表2의 各數字는 context tree에서 선택된 node의 depth를 나타내는 것으로 入力계열의 길이가 증가함에 따라 그 계열에 대한 充分한 學習이 이루어져, 局所狀態의 次數가 正確히 推定되어 감을 알 수 있다.

표1. 圧縮率의 比較

m	Entropy	Rissanen符號	mod Z-L符號	비 고
0	0.8813	0.8869	0.9146	Length
1	0.7840	0.7908	0.8215	=10 Kbit
2	0.7429	0.7529	0.7968	α=e-0.005t
3	0.9119	0.9233	0.9667	β=0.5

표2. 選擇된 文脈의 次數

길이 / m	100	400	1600	6400	12000	20000[bit]
0	0	0	0	0	0	0
1	0	0	1	1	1	1
2	0	0	1	1	2	2
3	0	1	2	3	3	3

한편, 그림10에는 入力의 길이를 변화시킴에 따른 壓縮特性을 mod Z-L과 더불어 나타냈다. 그 결과, 壓縮率에 있서는는 현저한 향상을 보이고 있으나(充分한 學習이 이루어진 상태로 부터)計算量의 증가를 보이고 있음을 알 수 있다.

그 원인으로 ① incremental algorithm의 complexity가 O(n)임에 비하여 Rissanen 符號는 O(n log n)인 本質的인 차이 ②各 node에 있어서의 確率計算 및 selection등을 생각할 수 있다.

그 해결책으로서 context gathering에 따른 tree의 깊이를 어느정도 제한함으로써 불필요한 node의 成

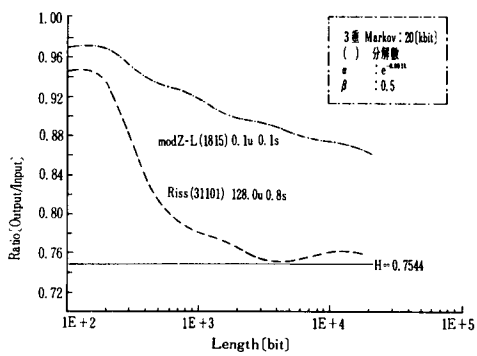


그림10. 圧縮率의 변화

長를 막을 수 있으며, 약간의 壓縮率 손실을 감수하면서 selection rule〔(9b), (9c)〕를 생략하는 것도 가능하리라 본다.

Ⅳ. 結　　論

情報源의 統計的 事前 知識을 前提로 하지 않고도 entropy壓縮이 가능한 유니버샬 데이타 壓縮法를 몇가지 紹介하였다.

各 符號의 理論的 考察과 함께 計算機 simulation에 의한 性能評價를 ① 壓縮率의 收束速度 ② algorithm의 complexity (3) Universality를 中心으로 실시하였다. 實用的인 面에서 重要한 file圧縮의 경우 Ziv－Lempel의 VF符號를 VV符號로 확장함으로서 양호한 性能을 얻을 수 있음을 보였다.

한편, Rissanen符號를 Adaptive로 構成하여 Ziv－Lempel符號와의 관계를 明示하였다. 즉, Source coding의 문제는 modeling에 크게 의존함을 보였으며, incremental parsing을 context gathering으로 發展시킨 점에서 두 방법은 상당히 관련성이 큰 model을 基礎로 하고 있다고 볼 수 있다.

결과적으로, Ziv－Lempel符號는 비교적 길이가 긴 입력 데이타에 대하여 고속성이 요구되는 경우 유효성이 있으며, Rissanen符號는 計算時間에 구속되지 않는 高圧縮率이 要求되는 경우에 적용성이 크다고 볼 수 있다. 일반적으로 데이타 압축 시스템을 구축함에 있어서, 圧縮率과 計算量 간에는 trade-off가 존재하여, 여하히 양자를 만족시키는 最良의 algorithm을 개발 할 것인가는 흥미있는 과제가 될 것이다.

參 考 文 獻

[1] D.A. Huffman, "A method for the construction of minimum redundancy codes," Proc. IRE, 40, 9, pp. 1098-1101, Sept. 1952.

[2] T.J. Lynch, "Sequence time coding for data compression," Proc. IEEE, 54, pp. 1490-1491, Oct. 1966.

[3] L.D. Davisson, "Comments on sequence time coding for data compression," Proc. IEEE, 54, pp. 2010, Dec. 1966.

[4] J.P.M. Schalkwijk, "An algorithm for source coding," IEEE Trans Inf. Theory, 1T-18, 3, pp. 395-399, May 1972.

[5] J.C. Lawrence, "A new universal coding scheme for the binary memoryless source," IEEE Trans. 1T-23, 4, pp. 466-472, July 1977.

[6] A. Lempel and J. Ziv, "On the complexity of finite sequences," IEEE Trans. 1T-22, no. 1, pp. 75-8, 1976.

[7] J. Ziv and A. Lempel, "An universal algorithm for sequential data compression," IEEE Trasn.1T-23, no. 3, pp. 337-343, 1977.

[8] J. Ziv and A. Lempel, "Compression of individual sequences via variablerate coding," IEEE Trans. 1T-24, no. 5, pp. 530-536, 1978.

[9] G.G. Langdon, Jr, "An note on the Ziv-Lempel model for compressing individual sequences," IEEE Trans. 1T-29, no. 2, pp. 284-287, 1983..

[10] 中田, 山本, "Ziv-Lempel符號の改良とシミュレーションによる性能評價" 情報理論とその應用研究會, 第7回シンポジウム, pp. 138-142, 1984.

[11] 山本, 中田, "Ziv-Lempel符號の改良とシミュレーションによる性能評價Ⅱ" 信學技報, cs 84-135, 1985.

[12] 横尾英俊, "ユニバーサル情報源符號のための修正Ziv-Lempel符號," 信學論(A), J68-A, no. 7, pp. 664-671, July 1985.

[13] 朴志煥 他, "インクリメンタルデータ圧縮の性能評價について" 昭和61年度 電子通信學會通信部門全國大會, 1-17.

[14] J. Rissanen, "A universal data compression svstem," IEEE Trans. Inf. Theory, 1T-29, 5, pp. 656-664, 1983.

[15] 横尾英俊, "MDL規準による適用的データ圧縮法のモデリルグ," 信學論(A), J69-A, no. 8, pp. 974-982, Aug. 1986.

[16] 伊藤, 川端, "パラメタ分散推定量を用いたユニバーサルデータ圧縮アルゴリズム," 情報理論とその應用研究會, 第8回シンポジウム, pp. 239-244, 1985.

[17] 朴志煥 他, "モデリングによるアダプテイブユニバーサルデータ圧縮," 昭和62年度 電子情報通信學會總合全國大會, 1376, 1987. ＊