論 文

OCM방법을 이용한 Compact Code의 구성에 관한 연구

正會員 金 慶 泰* 正會員 閔 勇 植**

A Study on the Composition of Compact Code using Octal-Compact Mapping Technique

Kyung Tae KIM* *and* Yong Sik MIN**, *Regular Members*

요 약 데이터 통신의 급격한 발전으로 인하여 각종 정보를 손쉽게 이용할 수 있게 되었다. 이같이 얻은 정보들을 송·수신함에 있어서 가능한 한 비트수를 최소화하여 정보를 전송하는 것이 필요로 하게 되었다. 즉 8개의 symbol을 다룬 경우의 entropy가 평균적으로 3.5bytes로서 다른 방식에서 사용된 것보다 적어도 1 byte이상 축약되는 OCM(Octal-Compact Mapping)방식을 제시함과 이같이 최소화로 인해 redundancy가 상당히 감소되며 동시에 기존에 사용한 data compression방식보다 효율성이 좋은 방식을 본 논문에서 제시하였다.

ABSTRACT According to rapid developments in data communication, we have being used every information with ease. In receiving and transmitting the information acquired, it is being needed to transmit it with minimizing bits if possible. Therefore this paper suggests the efficient coding system, that is, OCM(Octal- Compact Mapping) technique. In case of average-case, it has 3.5bytes in entropy with 8 symbols. This means it is compressed more than at least 1 byte as compared with another coding techniques. It decreases the redundancy of data and is superior to another data compression techniques.

1. 서 론

수송과 통신기술이 발명되기 이전의 오랜 기간 동안에는 발생정보와 사용정보의 비율은 완만한 움직임 밖에 없었다. 그러나 컴퓨터의 출현과 통신기술의 진보는 많은 사람들에게 많은 양의 정보를 제공하게 되었다. 이렇게 제공된 정보의 양이 급증하므로 현재 우리는 여러 가지 측면에서의 정보증가가 가져오는 영향을 이해하는 실마리조차 잡을 수 없는 형편이다. 이렇게 발생된 각가지 종류의 정보를 간단하게 공급되게 하는 일이야 말로 새로이 당면한 과제이다.

비록 데이터를 저장하는 장치들의 가격이 급속도로 감소되고 있지만 물리적인 기억장소에 더 많은 데이터를 넣기 위해서는 여전히 많은 문제들이 존재하고 있다.

현재 data compression의 이용은 정보를 처리함에 따라 보내야 할 데이터 비트수가 감소함으로써 데이터의 전송을 개선시키는데 중요한 개선책을 제공한다.

따라서 본 논문은 이같이 얻은 정보들을 이용하여 이 정보들을 가능한 최소의 비트수로서 송·수신함으로써 전송선로상에서도 보다 많은 양의 데이터를 송·수신할 수 있으며 속도도 빠르게 될 수가 있게 되었다. 또한 이같은 최소화로

** 光云大學電子計算學科
Dept. of computer sciece, Kwangwoon university, Seoul, 132 Korea
* 松源實業專門大學電子計算科
Dept. of computer science, Songwon Vocational Junior College, Kwangju-Shi 500 Korea
論文番號：84－13 (接受 1984. 6. 2)

표 1 각 Symbol당 통계숫자와 확률
Symbol frequency analysis & count.

SYMBOL	COUNT	PROBABILITY
	32955	.164779883
,	1679	8.39515814E-03
#	39	1.95236236E-04
%	117	5.85708708E-04
"	234	1.17141742E-03
'	390	1.95236236E-03
-	390	1.95236236E-03
.	1522	7.6142132E-03
0	195	9.76181179E-04
1	546	2.7333073E-03
2	39	1.95236236E-04
3	117	5.85708708E-04
4	78	3.90472472E-04
5	39	1.95236236E-04
6	39	1.95236236E-04
7	78	3.90472472E-04
8	156	7.80944943E-04
9	273	1.36665365E-03
A	13080	.065404139
B	2577	.0128855916
C	5154	.0257711831
D	5349	.0267473643
E	19367	.09683717
F	3084	.0154236626
G	5310	.0265521228
H	7262	.0363139399
I	11948	.0597422882
J	195	9.76181179E-04
K	1522	7.6142132E-03
L	5427	.0271378368
M	4021	.0201093323
N	13666	.0683326825
O	11479	.0573994533
P	3358	.0167903163
Q	78	3.90472472E-04
R	10308	.0515423663
S	10308	.0515423663
T	15814	.0790706755
U	3904	.0195236236
V	1561	7.80944943E-03
W	2616	.0130808278
X	312	1.56188989E-03
Y	3279	.0163998438
Z	117	5.85708708E-04

인해 redundancy를 제거시킴과 동시에 다른 data compression방식보다 좀 더 나은 compact code인 OCM방법을 제시하는데 목적이 있다.

2. Data Compression방법

현재 컴퓨터 시스템에서는 여러 가지의 source symbol들을 EBCDIC나 ASCII와 같은 fixed length의 code로 표시하지만 전송선로상에서는 code의 평균길이를 가능한 한 짧게 하는 것이 fixed-length로 하는 것보다 경제적이다.

일반적으로 source symbol $S=\{S_1, S_2, S_3, \cdots S_q\}$에 대한 code의 평균길이는 다음식으로 표시된다[2].

$$L=\sum_{i=1}^{q} P(S_i)\cdot I(S_i)$$

여기서 $P(S_i)$는 symbol S_i의 확률이고 $I(S_i)$는

$$I(S_i)=\log_2\frac{1}{P(S_i)}\ \text{bits}$$

이다. 여기서 평균 정보량은

$$H(S)=\sum_{i=1}^{q} P(S_i)\cdot\log_2\frac{1}{P(S_i)}\ \text{bits}$$

이다. entropy는 1개 symbol이 갖는 평균 정보량이므로 다양한 source symbol의 확률에 의존한다. 즉

$$H(S)\leqq\log_2 q\ (q:\ \text{source symbol수})$$

여기서 알 수 있듯이 entropy가 적은 code를 설계하기 위해 자주 사용되는 문자에 길이가 짧은 code를 할당하고 사용빈도가 낮은 것에 긴 code를 할당하는 variable length code를 채택하게 되면 code의 평균길이가 축소될 것이므로 효율적인 code라는 측면에서 바람직한 방법이다.

data compression의 여러 방법 중 두 가지만 살펴 보자[1].

bit-mapping기법은 space-compression의 대표적인 예로 source symbol에 공백이 많은 경우에 커다란 압축효과를 얻을 수 있다. 즉 source symbol $\{S_1, S_2, \cdots, S_q\}$에서 한 자리씩 조사하여 공백인 경우 0을, 아닌 경우 1을 각 문자와 대응시켜 bit-mapping을 1 byte씩 형성한다. 공백이 아닌 문자 즉 bit-mapping이 1인 문자에 대해 EBCDIC을 각각 오른쪽에 첨가하여 생성한다. 마지막 symbol까지 계속 이같이 처리하고 마지막의 문자가 8문자가 아닌 경우는 나머지는 공백으로 간

주한다.

예로서 AB␣␣␣CD를 bit-mapping기법으로 하면

11000011 11000001 11000010 11000011 11000100
bit-map A B C D

로 된다. 원래 source symbol을 표시하기 위해서는 8 byte가 필요하나 여기서는 원래길이보다 62.5%가 축약된 5 byte로 전송이 된다. 그러나 공백이 없는 경우에 전송되는 데이터는 원래의 수만큼 차지한다. bit-mapping의 compression율은

$$\frac{\text{bit-mapping code 길이}}{\text{EBCDIC길이}} \times 100$$

이다.

bit-mapping은 fixed-length로 하는 반면 variable length로 표현한 방법 즉 각 문자에 대한 빈도수를 조사하여 정보 전송시 확률이 높은 문자에 code길이를 짧게, 확률이 낮은 문자에 code길이를 길게 부여하여 전송효율을 높이는 binary compact code방식이 있다. 앞의 전송 데이터 AB␣␣␣CD를 binary compact code로 표시하면

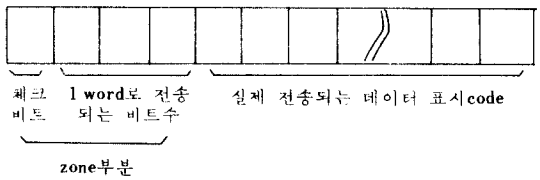
0110 101101 000 000 000 000 10111 10100
A B ␣ ␣ ␣ ␣ C D

이다.

이같이 32비트가 필요하다. 그러나 EBCDIC나 ASCII는 64비트가 소요되며 bit-mapping은 40비트가 소요됨을 앞에서 알 수가 있다. 즉 원래길이보다 50%가 축약됨을 알 수 있다. 이같이 variable length로 표시하므로 전송정보의 길이를 줄일 수가 있게 되었다. 그러나 이 방식도 발생빈도수가 낮은 문자가 많거나 공백이 많은 symbol의 전송에 있어서는 오히려 많은 자리를 차지하게 되는 단점이 있다. 이 외에도 여러 방법이 있으나 아직까지 binary compact code방식이 좋은 것으로 나타나 있다.

3. Octal-Compact Mapping기법

각 문자에 대응하기 위해 각 symbol들의 출현확률을 파악할 필요가 있다. 이러한 목적에 부합되는 통계자료를 문자 100만자를 임의 추출하면 평균 1 word당 약 8 개의 symbol로 구성되어 있음을 알 수가 있다. 즉 1 word를 나타내기 위해서는 최소 3 비트($2^3=8$)만으로 표시가 가능하다. 다음 그림과 같이 표시된 것이 1 word를 표시하기 위한 것이며 본 논문에서는 이것을 zone 부분이라 칭한다.



zone부분은 1 word를 나타내기 위하여 3 비트와 체크 비트를 위한 1 비트로 구성되어 있다. 즉 체크 비트에서 0 인 경우는 문자들이 다음 3 비트에 나타난 갯수(2^4)만큼 공백임을 뜻한다. 1인 경우는 다음 3 비트에 존재하는 갯수만큼 symbol들을 표시됨을 의미한다.

이같은 구조는 space-compression의 장점을 지닌 bit-mapping방법과 각 symbol들의 사용빈도를 고려하여 사용빈도가 낮은 symbol에 긴 code를, 빈도가 높은 symbol에 짧은 code를 할당하는 variable length를 지닌 binary compact code를 이용한 것이다. 그러나 binary compact code에서 ␣(blank)에 대한 code가 존재하나 여기서는 code를 부여할 필요가 없다.

앞에 살펴본 예 AB␣␣␣CD를 OCM으로 표시하면 다음과 같다.

1010 0110 101101 0100 1010 10111 10100
zone부분 A B zone부분 zone부분 C D

이같이 32비트가 소요된다. 다른 방식을 이용한 것보다 좋고 binary-compact code와는 같은 자리를 차지함을 알 수가 있다. 즉 worst-case인 경우는 binary compact code보다 항상 4비트(zone 부분)가 더 많이 차지하게 되며 average-case인 경우는 OCM방식이 훨씬 적은 비트수 즉 binary compact code의 평균 정보량은 36비트인 반면[6] OCM방법은 28비트임을 알 수가 있다. 또한 문장이 길어지는 경우와 공백의 수가 많이 존재하는 문장에서는 OCM방법이 훨씬 좋음을 알 수가 있다. 이같은 특성을 지닌 OCM방법을 해결하는 알고리즘은 다음과 같다.

1. 각 source symbol들의 사용빈도를 고려하여 다음과 같은 과정을 통해 그림 1 의 binary code

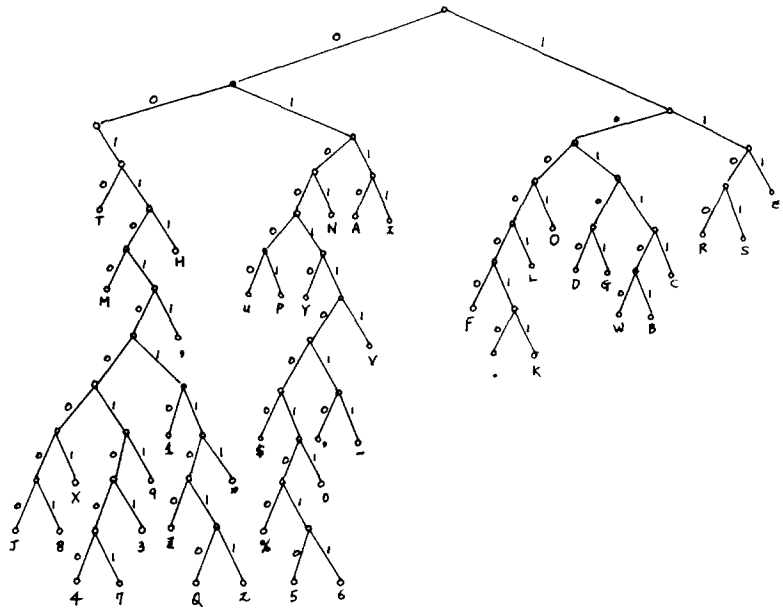


그림 1 문자들을 위한 binary code tree
Binary code tree for characters.

표 2 binary tree code 생성
Creating the binary tree code.

	PROBABILITY.	CODE		PROBABILITY	CODE
E	.09684	111	.	7.61E-03	1000010
T	.07907	0010	K	7.61E-03	1000011
N	.06833	0101	1	2.73E-03	001101010
A	.0654	0110	#	2E-03	010011000
I	.05974	0111	'	1.95E-03	010011000
O	.0574	1001	-	1.95E-03	010011011
R	.05154	1100	X	1.56E-03	0011010001
S	.05154	1101	9	1.37E-03	0011010011
H	.03631	00111	"	1.17E-03	0011010111
L	.02714	10001	Q	9.8E-04	0100110011
D	.02675	10100	J	9.8E-04	00110100000
G	.02655	10101	8	7.8E-04	00110100001
C	.02578	10111	3	5.9E-04	00110100101
M	.02011	001100	Z	5.9E-04	00110101100
U	.01952	010000	%	5.8E-04	01001100100
P	.01679	010001	4	3.9E-04	001101001000
Y	.0164	010010	7	3.9E-04	001101001001
F	.01542	100000	0	3.9E-04	001101011010
W	.01308	101100	2	2E-04	001101011011
B	.01289	101101	5	2E-04	010011001010
,	8.39E-03	0011011	6	2E-04	010011001011
V	7.81E-03	0100111			

tree를 형성한다.

㉮ 사용빈도의 크기순으로 symbol들을 다시 순서적으로 배열한다(blank제외).

㉯ 사용빈도가 가장 낮은 symbol과 다음으로 낮은 symbol에 대하여 각각 1과 0의 code alphabet를 부여한다.

㉰ 위의 2개 symbol을 합성하여 하나의 새로운 symbol로 간주한다.

㉱ source symbol의 수효가 점차로 감소하여 1개로 될 때까지 위의 방법을 반복한다.

2. 이같이 생성된 binary code tree를 다음과 같은 조건으로 string(symbol)들을 처리한다.

㉮ 읽은 string이 blank string인지 문자string인지를 식별해야 한다.

㉯ 각각 string들이 1 word로 주어진 갯수를 조사하여서 2^i의 형태로 변환하여 i값을 택한 후 zone부분에 set시킨다.

㉰ 읽혀진 string들이 문자 string이면 체크 비트(zone부분)에 1을, 아니면 0을 set하고 zone부분 이외의 부분에 문자string인 경우만 해당 문자에 대응되는 binary code를 2^i갯수만큼 set시킨다.

㉱ 마지막의 문자까지 읽혀질 때 앞의 과정을 반복처리하면 된다.

4. 결 론

8문자를 기준으로 하여 각각의 경우의 entropy를 계산하면 다음과 같다.

bit-mapping에서는 worst-case가 9 bytes, best-case는 1 byte이며 average-case는 5 bytes이다. binary compact code는 worst-case가 12bytes, best-cast가 3 bytes이고 average-case가 4.5bytes이다. 이에 반해 OCM방식은 worst-case가 13bytes, best-case는 1 byte이고 average-case는 3.5bytes가 된다.

이같은 방법으로 미루어 볼 때 본 연구에서 제시한 방법이 가장 효율성이 있는 compression방법임을 알 수가 있다. 즉 보통길이의 code로 된 8개 문자를 다룬 경우의 entropy가 적어도 1 byte 이상 심지어는 2.3bytes까지 축약이 가능함을 나타내고 있다.

또한 본 연구에서는 최소화된 정보의 비트수를 전송선로를 통하여 모든 데이터를 전송함으로써 장거리 전송을 통하여 정보를 전송하는 경우 적은 용량을 가지고도 많은 양의 정보를 송·수신이 가능하게 될 수 있으며 효율성도 다른 것에 비해 좋음을 알 수 있다. 그러나 본 연구에서는 전송상의 error detection문제에 대해서는 전혀 다루지 않았으며 또한 한글에 적용하는 경우에도 상당한 효과가 있으리라고 기대된다.

참 고 문 헌

(1) Cobin, Harold, "An introduction to data compression," Byte Publication, pp. 218~250, April 1981.

(2) Abramson, Norman, "Information theory and coding," Mc Graw-Hill, 1963.

(3) R. Tropper, "Binary-coded text, a text compression method," Byte Publication, pp. 398~413, April 1982.

(4) J. Pieter, M. Schalkwijk, "An algorithm for source coding," IEEE Trans. Information Theory, vol. 18 no. 3, May 1972.

(5) 김경태, 이균하, "한글정보의 경제적 처리를 위한 부호화에 관한 연구," 한국정보과학회지, vol. 5, no. 2, pp. 99~104, Dec. 1979.

(6) 문경혜, "효율적 자료전송을 위한 compression code방식에 관한 연구," 광운대학대학원 석사청구논문, 1984.

金 慶 泰 (Kyung Tae KIM) 正會員

1929年 5 月15日生

1956年 2 月：延世大學校數學科卒業

1958年 2 月：延世大學校大學院數學科卒業(理學碩士)

1972年 ：캐나다 Dalhousie 大學校大學院應用數學科卒業

1974年～現在：光云大學電子計算學科教授

1982年～1984年：韓國情報科學會副會長

1983年～1984年：캐나다 Carleton大學客員教授

閔 勇 植 (Yong Sik MIN) 正會員

1958年 1 月19日生

1981年 2 月：光云工科大學電子計算學科卒業

1983年 2 月：光云大學大學院電子計算科卒業(理學碩士)

1984年現在：松源實業專門大學電子計算科專任講師