논 문
58P-3-8

새로운 무손실 유니버셜 데이터 압축 기법

A New Method of Lossless Universal Data Compression

김 성 수† · 이 해 기*
(Sung-Soo Kim · Hae-Kee Lee)

Abstract - In this paper, we propose a new algorithm that improves the lossless data compression rate. The proposed algorithm lessens the redundancy and improves the compression rate evolutionarily around 40 up to 80 percentile depending on the characteristics of binary images used for compression. In order to demonstrate the superiority of the proposed method, the comparison between the proposed method and the LZ78 (LZ77) is demonstrated through experimental results theoretical analysis.

Key Words : Lossless, Data Compression, Binary Image Compression

1. Introduction

The exploration of data created in the recent decades has brought us the need of efficient data procession in various areas, especially in the digital communications and signal processing areas. Coming into the age of information, the importance of information becomes too much important to be emphasized. The expansion of data size has been the monstrous huddle to overcome for communications even though the technology of coding methods in the source and the channel has been developed in great amount, which still one of the most desperate subjects to be resolved [1-3].

The technologies that are developed so far may be categorized in two prospects, such as the lossless universal data compression and the lossy data compression. For the purpose of applications, the approximation of data is employed to compress the data of various forms. JPEG is one of the most well known lossy data compression method. However, data of text documents that need complete recovery require the lossless codes [4, 5]. In order to overcome this problem, several compressing techniques, such as Huffman-based codes, have been studied for decades. Universal lossless data compression is the codes that performs without the information on the statistics of the system. The proposed universal binary image data compression method is different new universal codes since Lempel-Ziv's proposal. There are several different universal lossless codes for binary images such as the arithmetic code and the structured grammar-based code. The structured grammar-based code consists of the two stage. The first stage transforms the original data into the grammar-based data, The second stage compresses the transformed data using arithmetic codes [6, 7]. Then through the overall process, the data is transformed from the stationary ergodic data source to the lossless universal codes. In this paper, a structured grammar-based lossless data compression algorithm is mentioned in Section 2 and a new method of the lossless code for binary images with revising, which is developed based on the conventional universal compression and the property of multi-layer methods, is proposed to resolve the redundancy like Lempel-Ziv's method in Section 3.

Section 4 describes the modified grammar-based universal algorithm theoretically by introducing the techniques and properties of proposed method. The superiority of the proposed method is presented through the theoretical analysis and simulations in Section 5 followed by conclusions in Section 6.

2. A lossless universal data compression

Great amount of work has accomplished on the universal lossless source codes (ULSC) which was initiated in the late of 1960's. Lempel-Ziv, which was proposed in the late 1970's, is one of the ULSC's that has been widely used in various applications. Lately, the ULSC that uses grammar-based transformation scheme was proposed by E. Yang and J. Kieffer. The ULSC

† 교신저자, 정회원 : 충북대학교 전기공학과 교수・공박
* 정 회 원 : 충청대학 전기전자학부 교수・공박
E-mail: sungkim@chungbuk.ac.kr
접수일자 : 2009년 5월 15일
최종완료 : 2009년 6월 11일

using grammar-based transformation compresses source data in the form of a string x through the grammar, G_x, that consists of production rules. Fig. 1 shows the structure of the encoder that employes the ULSC with grammar-based as a preprocessing.

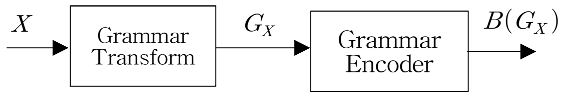

Fig. 1 Structure of grammar-base code encoding.

A data string X is transformed via the grammar obtaining a transformed source G_x, and it becomes a binary code $B(G_x)$ through encoding process. The actual compression is carried out on the transformed data string G_x instead of x itself. Here the grammar G is chosen to admissible such that it satisfy $L(G) = \{x\}$. The set of the different symbols, $V(G)$, is arranged by the production rule in the order of left to right following the grammar that is independent of the data contents where reduction rule is suggested to obtain the effective compression. Some of the grammar-based transformations are Lempel-Ziv and bisection. As a criterion of the effectiveness of grammar-based transformed codes $x \rightarrow G_x$ is theoretically analyzed by the upper bound with maximum piecewise redundancy. In this paper, in addition to the rules for compression [7, 8], the additional five rules for better compression are added to yield the SEQUITUR algorithm [9, 10, 11] that is different from the longest matching substring algorithm and the conventional SQUITUR algorithm. Also, as one of the lossless data compression codes using the classification methods based on the pattern matching, the multi-level pattern matching (MPM) was introduced recently by J. Kieffer and E. Yang, whose redundancy, $O(1/\log n)$ for the data of length n, is improved compared to previously developed codes. In this paper, we have proposed a new codes that has superior redundancy per sample by modifying the grammar based codes with the constraints of structure in the encoding procedure.

3. A new lossless data compression codes

The proposed new lossless data compression codes minimize the redundancy so that we can maximize the compression rates. In Fig.2 and 3, the block diagram of encoding and decoding is described where the information on the code tree is obtained by top to down by bisection. Some of the terminologies are described below.

A code tree for a string x is constructed by bi-sectioning the string as described in [7], which is called the reduction rule. Each partitioned string is defined as a symbol where the collection of the same length symbols constructs a level, such that ordering the elements of a set at each level constructs branches of a tree. The process of compression is established by eliminating the same symbols that appears repeatedly and just using the index of symbols. The main part of encoding can be described as the strategy that how we represent the symbols by the index not by the contents itself.

For each branch each branch of the code tree is classified into two categories: one is a collection of symbols that do not repeat and the other collection of symbols that repeat. We assigned 1 to unrepeated symbols and the first repeated symbol such that the collection of symbols with index 1 construct a unique tree called the unique structure of source data. However, the structured tree built is different in the sense that the repetition of symbols appear only in the same level while the structured tree in [7] does not. Another difference of the proposed codes in this paper is that the encoding process begins at the level at which the partition occurs first time not from the beginning of the code tree information, such that the lowest level of the code tree adaptively chosen, which is not the case for the conventional ones [7]. The purpose of choosing the last level adaptively is to handle the problem that the code tree uses bits which is longer than the symbols at a lower level. Since the main point how the compression occurs strongly depends on the way of representing the structure information of source data, representing a shorter length symbols with longer indexing structure information cannot be used for compression purpose. The adaptively chosen last level is turned out as the most effective one. The code tree is assigned as the first component of encoding process. In addition to this, for the complete reconstruction using decoder, we need to keep the first level where the partition first occurs and the last level. The second stage of encoding builds a unique code that consists of the total collection of unique symbols in the form of a binary string. The last three steps in encoding are the most important part of encoding. Since the code tree has been constructed by eliminating those symbols appear more than once from top to bottom, the eliminated symbol must be indexed in the shorter length presenting information on the location and contents. Thus if there exist more than one symbols that have the same symbols in the same level, we need to index the order of different symbols with different number of repeats.

Again, in this step we need to exam whether we can represent the length of index must be shorter than the length of symbol to be represented because the purpose

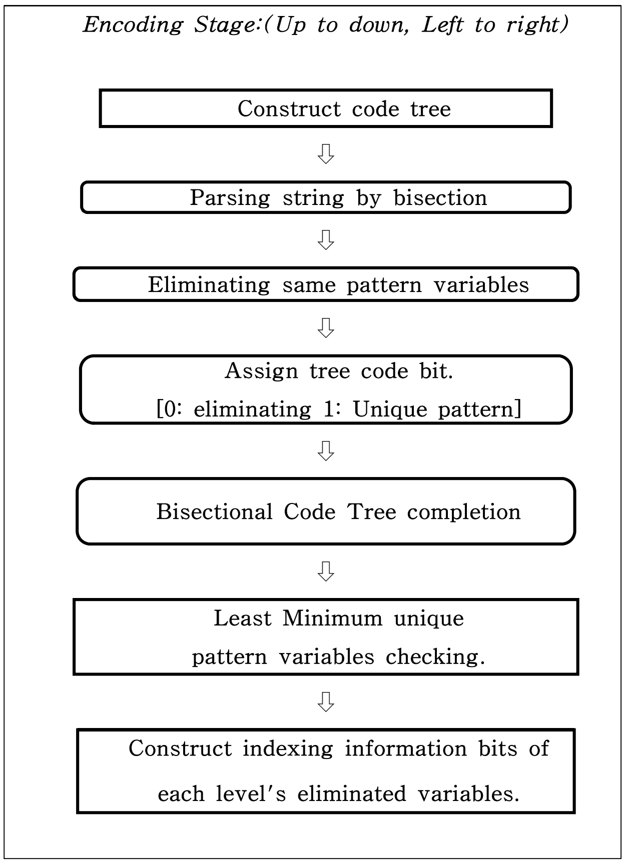
Encoding Stage:(Up to down, Left to right)
Construct code tree
Parsing string by bisection
Eliminating same pattern variables
Assign tree code bit. [0: eliminating 1: Unique pattern]
Bisectional Code Tree completion
Least Minimum unique pattern variables checking.
Construct indexing information bits of each level's eliminated variables.

Fig. 2 Flowchart of Encoding

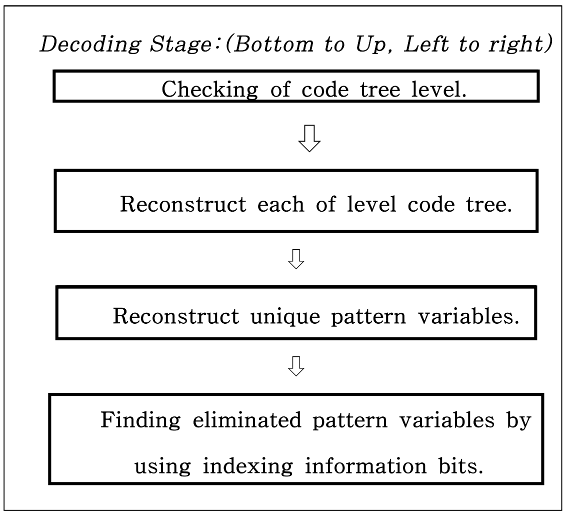
Decoding Stage:(Bottom to Up, Left to right)
Checking of code tree level.
Reconstruct each of level code tree.
Reconstruct unique pattern variables.
Finding eliminated pattern variables by using indexing information bits.

Fig. 3 Flowchart of Decoding.

of compression is to minimize the redundancy. Therefore, the process of encoding consists of three parts. The first part is the head bits that represents the first level of partition and the last level the partition ends. Decoding procedure is the reverse of the encoding procedure. The encoding is carried out from the top to bottom, such that the decoding is carried out by bottom to top. The initial level and last level information are used for identifying the whole structure of levels. Then using the collection of symbols and the collection of the structure index, we can reconstruct the original source data. The total process of encoding in Fig. 2 and decoding is presented in Fig. 3.

4. Modified Structured Grammar-Based Codes for Universal Lossless Data Compression

In this section we present a new algorithm that modifies the structured grammar-based codes presented by Kieffer and Yang for universal lossless data compression. The aim of this work is set not only to correct Lemma and Theorems presented previously, but also to propose a method that finds the bounds of the largest value $J(n.D)$ for an arbitrary string of length n and a finite set D with at least two elements.

Data compression methodology is a great concern to many researchers for several decades, especially the lossless data compression has been explored by various pattern matching schemes [11] ever since the Lempel-Ziv coding algorithms (LZ77, LZ78 [4, 5])were published in the late 1970's after L. Davissin's work in 1973 on universal coding. In 2000, the Universal Lossless Data Compression (ULDC) algorithm based on pattern matching called the Multilevel Pattern Matching code (MPM code) and an efficient ULDC algorithm based on a greedy sequential grammar transform were proposed by J. C. Kieffer and E. H. Yang. After two years, in 2002, they also introduced the structured grammar-based coding method for ULDC that reduces the maximal redundancy/sample.

Fundamentally, the structured grammar-based coding scheme in [7] consists of analysis and synthesis procedures. In the analysis stage, the context free parsing scheme is decomposed into the data content and the structure of parsing tree, which are used in both the encoding and the decoding procedures. The analysis stage is regarded as the most important part that constructs the way of parsing, such that the information of parsing is mapped into the tree structure.

Consequently, the main subject is how to parse a string. There are various methods for this purpose such as Lempel-Zip, MPM, ULDC, and etc.. This subject is addressed in [7] by introducing the least upper bound of the maximum length of a distinct parsed string via Theorem 1 that was proved using Lemma 2. In this paper, we will correct some mistakes made in Lemma 2 and Theorems proved based on the Lemma 2 in [7].

Corrections on Structured Grammar- Based Codes for USDC: In this section, the proof of Lemma 2 appeared in the appendix of [7] will be reviewed and errors, if there exists any, will be corrected. Besides, the theorems developed based on Lemma 2 will be discussed also. Lemma 2, whose definition is presented below, is introduced as the first step to prove Theorem 1 from 2 which other theorems are derived. The following statements are Lemma 2 and Theorem 1 presented in [7].

Theorem 1. Let n be any integer 2 and let G be any representational grammar which represents an A-string of length n. Then

$$|G| \leqq 72\beta(G^*)^2(|A|+2)^2\log(|A|+2)(\frac{n}{\log n}) \tag{1}$$

Lemma 2. Let n be any integer at least 2 and let D be a finite set with at least two elements. Let $J(n,D)$ be the largest number of distinct strings in $D+$ which are of total length at most n. Then

$$J(n,D) \leqq 4|D|^2\log|D|(\frac{n}{\log n}) \tag{2}$$

Fist let us look at the proof of Lemma 2. The authors [7] seem to tried to derive the least upper bound of the $J(n,D)$ using the mathematical inequalities and equalities. However, unfortunately, the proposed Lemma 2 needs to be fixed for finding the least upper bound of the $J(n,D)$. For the proof of Lemma 2 in [11], they let n be any integer at least 2 and let D be a finite set with at least two elements, and let $J(n,D)$ be the largest number of distinct strings in $D+$ which are of total length at most n. Then,

$$d+d^2+d^3+\ldots+d^j \leqq J(n,D) < d+d^2+\ldots+d^{j+1} \tag{3}$$

$$d[\frac{jd^j}{d-1}+\frac{1-d^j}{(d-1)^2}] \leqq n < d[\frac{(j+1)d^{(j+1)}}{d-1}+\frac{1-d^{(j+1)}}{(d-1)^2}]. \tag{4}$$

From (3), a relation was derived in [11] such as

$$J(n,D) < d(\frac{d^{j+1}-1}{d-1}) \leqq d^2(\frac{d^j}{(d-1)}). \tag{5}$$

They also have proposed

$$\frac{(j-1)d^j}{d-1} \leqq n < (d^2)^{j+1}. \tag{6}$$

based on the facts

$$d[\frac{jd^j}{d-1}+\frac{1-d^j}{(d-1)^2}] \geqq \frac{(j-1)d^j}{d-1} \tag{7}$$

and

$$d[\frac{(j+1)d^{(j+1)}}{d-1}+\frac{1-d^{j+1}}{(d-1)^2}] \leqq \frac{(j+1)d^{j+2}}{d-1} \leqq (d^2)^{j+1}. \tag{8}$$

However, the equality on the right side of (5) obviously does not hold since $d>2$. Also, (6) does not hold either since the equalities in (7) and (8) are not true. This implies that Lemma 2 fails to satisfy the requirement of equality for determining the greatest lower bound (GLB) of $J(n,D)$. Specifically, it is unfortunate that the proof of Lemma 2 was derived in [11] based on

$$J(n,D) \leqq n \leqq 8\log d(\frac{n}{\log n}) < 4d^2\log d(\frac{n}{\log n}). \tag{9}$$

concluding that (9) implies $J(n,D) \leqq 4|D|^2\log|D|(\frac{n}{log n})$, which is incorrect and implies that Lemma 2 needs to be modified. In this paper, in order to remove the problems mentioned previously, we propose a method that modifies Lemma 2 by setting the interval where $J(n,D)$ is to be determined according to the values of j and d. By determining the lower bound and the upper bounds of a given string of length n and the number of alphabets d using the relation shown in (4), we can obtain the information on the bounds of $J(n,D)$. In other words the proposed method extracts information on the bounds of $J(n,D)$ from the variables d, n only. In the first stage, a proper interval with the least upper and the greatest lower bounds of $J(n,D)$ can be obtained by calculating the length n which is defined as $n=\sum_{k=1}^{k=j} kd^k$ with increasing the integer value of j. The procedure stops when the total length of string n satisfies the following condition in (10).

$$d[\frac{jd^j}{d-1}+\frac{1-d^j}{(d-1)^2}] \leqq n < d[\frac{(j+1)d^{(j+1)}}{d-1}+\frac{1-d^{(j+1)}}{(d-1)^2}]. \tag{10}$$

In the second stage, with the value j obtained in the previous stage, we can find the interval to which $J(n,D)$ belongs by calculating the following relation in (11), which is the rearranged form of (3).

$$d[\frac{d^j-1}{d-1}] \leqq J(n,D) < d[\frac{d^{j+1}-1}{d-1}]. \tag{11}$$

From (11), once a proper j is found, then the Least Upper Bound(LUB) and the Greatest Lower Bound (GLB) of (11) are set by the following relation

$$GLB \leqq J(n,D) < LUB \tag{12}$$

where $GLB \cong \left\lceil d\left[\frac{d^j-1}{d-1}\right]\right\rceil$, and $LUB \cong \left\lfloor d\left[\frac{d^{j+1}-1}{d-1}\right]\right\rfloor$

are integers.

Besides finding the proper bounds for $J(n,D)$, from the procedure mentioned above, we also can justify the relation between n and d. In [11], the relationship between n and d is assumed in the form of $n > d^8$, which is later removed at the end of the proof of Lemma 2 by the following statement: For $2 \leqq n \leqq d^8$, Lemma 2 is also true because

$$J(n,D) \leqq n \leqq 8\log d(\frac{n}{\log n}) < 4d^2\log d(\frac{n}{\log n}). \qquad (13)$$

However this statement is not sufficient to support the equation in Lemma 2 just because of the misuse of equality in Lemma 2. If (13) is true, the equality in Lemma 2 cannot be hold obviously. Suppose that the relation in (13) is true, then it is necessary to prove that the rightmost term of (13) can be used as the *LUB*. In addition to this, Since the equality in the right most term of (13) does not exist, the proof of Theorem 1 is wrong since the proof of Theorem 1 is established based on Lemma 2. Specifically, the statement in Theorem 1 [11], "applying Lemma 2 with $D = A \cup \{[,]\}$ and $|D| = |A| + 2$, we obtain $s \leqq 36K(|A|+2)^2\log(|A|+2)(\frac{n}{\log n})$," is not true, such that $|G| \leqq 72\beta(G^*)^2(|A|+2)^2\log(|A|+2)(\frac{n}{\log n})$ cannot be true. Hence the Theorems derived from the Theorem 1 with equality are not true either. In Table 1, the *GLB* and the *LUB* of $J(n,D)$ are obtained as increasing j until (11) is satisfied for a string of an arbitrary length n. Then the *GLB* and *LUB* of $J(n,D)$ are determined by (3). Therefore, for a string with n, d, we can obtain the interval which $J(n,D)$ belongs to. 6

Table 1 *GLB* and *LUB* of $J(n,D)$ with respect to the integer j , with $d = 2$.

j	2	3	4	5	6	7	8	9	10	11	12
GLB	6	14	30	62	128	254	510	1022	2046	4094	8190
LUB	13	29	61	125	253	509	1021	2045	4093	8189	16381

As a consequence of this paper, we have examined the relation between any integer $n > 2$ and a finite set D, and found that Lemma 2 does not hold. When we remove the equality in Lemma 2 in [11], Lemma 2 fails to provide the least upper bound of $J(n,D)$. Therefore, we claim that the theorems that are developed based on the Lemma 2 must be restated or redefined. However, in this paper, we have proposed a new method that determines the least upper bound and the greatest lower bound of $J(n,D)$, the largest number of distinct strings in $D+$, where $D+$ denotes the set of all strings $d_1d_2d_3\cdots d_k$ in which $d_1, d_2, \cdots, d_k$ are $1 \leqq k < \infty$ entries from set D.

5. Experimental Results

In experiments the compression rate of the proposed lossless universal compression codes is examined by comparison between the conventional lossless compression codes. Several bmp formated binary images of several different sizes, 256 by 256, 512 by 512, and 1024 by 1024 are used. For the purpose of simulations, the original images are transformed into binary format, such that the data format of an 256 by 256 binary image consists of 65536 bits. The two different binary images of 65536 bits show two different compression rates, 54.2 and 58.2 percentiles compared to the compression rate 13.6 percentile by Gzip, where the compression rates are improved by 3.98 and 4.28 times compared to the compression rate of Gzip respectively. The algorithm in [3] presents that the greedy sequential grammar transform for memoryless binary source data compression performs all better than the Unix Compress and Gzip algorithms on average roughly 26% more efficient than the Unix Compress and 37% more efficient than Gzip. However, the algorithm we proposed in this paper outperforms the conventional compression algorithm as shown in Table 2, in the range between 180% and 427% depending on the images used for experiment. As it is shown in the Table 2, the compression rate depends on the size of data and the characteristics of data.

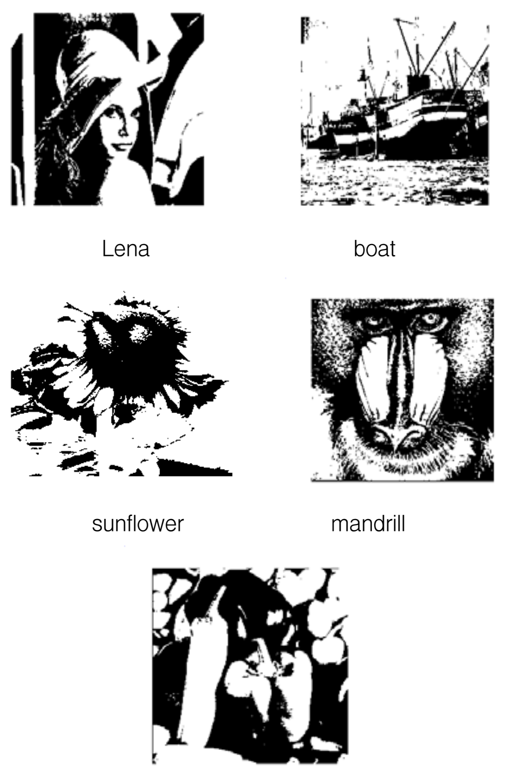

Fig. 4 Binary images used in experiments.

Table 2 Compression rates for binary images in Fig. 4.

Test images	Compression rate of proposed algorithm (%)	Compression rate of Gzip (%)	Improvement (SJ01 / Gzip)
Lena (256×256)	54.2	13.6	398.5%
Pepper (256×256)	58.2	13.6	427.9%
Lena (512×512)	64.4	33.3	193.4%
Baboon (512x512)	40.9	20.9	197.7%
BlueBG (512x512)	68.3	29.6	230.7%
Boat (512x512)	60.0	33.3	180.2%
Lena (1024x1024)	84.2	44.6	188.8%
BlueBG (1024x1024)	80.6	39.0	206.7%

6. Conclusions

Even though the area of lossless universal data compression attracts great amount of concern by many researchers lately, There are very limited number of new algorithms after Lempel-Ziv's work on data compression. One of the newly proposed algorithms is the structured grammar-based codes for universal lossless data compression, whose compression rate is reported the best up to now.

In this paper, we have analyzed the structure grammar-based codes and found some defects of it. The great accomplishment of this work is not only just removing the defects of the existing methods, but also surprisingly developed a new system for improving the lossless data compression rate. Base on the results of analysis, we have proposed a new algorithm, the lossless universal data compression algorithm with grouping and bi-sectioning. The proposed code lessens the redundancy and improves the compression rate evolutionarily, which is in the range of around 40 up to 80 percentile depending on the images used for compression. The proposed method guarantees the optimal compression adaptively depending on the characteristics of images. Experimental results are presented in Table 1.2, which shows the compression rates and the redundancies of the images used. The superiority of the proposed algorithm is proved and demonstrated by comparing with the conventional compression methods, such as LZ78, LZ77, Unix Compress, Gzip, and the structured grammar-based method through analysis and simulations for several different images.

References

[1] A. Lempel and J. Ziv, " On the complexity of finite sequences," *IEEE Trans, Inform. Theory* , vol IT-22, pp. 75-81, Jan. 1976.

[2] C. Cook, A. Rosenfeld, and A. Aronson, "Grammatical inference by hill climbing", *Inform*, Sci., vol. 10, pp. 59-80, 1997

[3] C. evill-Manning and I. itten, "Identifying hierarchical structure in sequences: A linear-time algorithm," *J. Artificial Intell. Res.*, vol. 7, pp. 67-82, 1997

[4] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. on Information Theory*, Vol. IT-23, no. 3, May 1977.

[5] J. Ziv, A. Lempel, "Compression of Individual Sequences via Variable Rate," *IEEE Trans. on Information Theory*, Vol. 24, no. 5, pp. 530-536, Sept., 1978.

[6] Y. Hershkovits and J. Ziv, "On Sliding-Window Universal Data Compression with Limited Memory," *IEEE Trans. Information Theory*, Vol. 44, no.1, Jan 1998.

[7] J. C. Kieffer, E. H. Yang,"Grammar-Based Codes: A New Class of Universal Lossless Source Codes", *IEEE Trans. on Information Theory*, Vol. 46, no. 3, pp. 737-754, 2000.

[8] E. Yang and J.C. Kieffer, "Efficient Universal Lossless Data Compression Algorithms Based on a Greedy Sequential Grammar Transform-Part One : Without Context Models," *IEEE Trans. on Information Theory*, Vol. 46, no. 3, May 2000.

[9] J.C. Kieffer, E. Yang, G.J. Nelsom, and P. Cosman, "Universal Lossless Compression Via Multilevel Pattern Matching," *IEEE Trans. on Information Theory*, Vol. 46, no. 4, July 2000.

[10] E. Yang , A. Kaltchenko , J.C. Kieffer, "Universal Lossless Data Compression With Side Information by Using a Conditional MPM Grammar Transform," *IEEE Trans. on Information Theory*, Vol. 47, no. 6, Sep 2001.

[11] J.C. Kieffer and E. Yang, "Structured Grammar-Based codes for universal lossless data compression," *Communication in Information and Systems*, Vol. 2, no.1 , pp. 29-52, June 2002.

저 자 소 개

김 성 수 (金 聖 洙)

1997년 12월 Univ. of Central Florida(공학박사). 1999년 3월~2001년 8월 우석대학교 전기공학 조교수. 2001년 9월~현재 충북대학교 전자정보대학 전기공학과 교수
<관심분야> 디지털통신, 인공지능, 신호처리

이 해 기 (李 海 基)

1981년 충북대학교 공업교육(공학사). 1985년 성균관대학교 전기공학(공학석사). 1990년 성균관대학교 전기공학 (공학박사). 1991년 ~ 현재 충청대학 전지전자학부 교수
<관심분야> 신호처리, 전력제어, 씨퀸스제어