# Modified Golomb-Rice Codes for Lossless Compression of Medical Images

Roman Starosolski [(1)], Władysław Skarbek [(2)]
[(1)] Silesian University of Technology
[(2)] Warsaw University of Technology

**Abstract**

Lossless image compression algorithms are used for images that are documents, when lossy compression is not accepted by users, or when we have no knowledge whether lossy compression is allowed or not. Lossless algorithms are especially important for systems transmitting and archiving medical data, because on the one hand lossy compression of medical images used for diagnostic purposes is forbidden by law, on the other hand the number and sizes of medical images stored or transmitted grows rapidly.

Image compression researchers usually focus on results obtained for typical data, however for real life systems results for non-typical data should also be considered. For a compression algorithm the worst case image data results in data expansion. In applications, the results of compressing non-typical data should be known and the data expansion, unavoidable while processing incompressible data, should be minimized by the rule: primum non nocere.

In the paper we analyze the Golomb-Rice (GR) family of codes, infinite family of prefix codes optimal for encoding symbols of exponential probability distribution. The GR family is used in predictive lossless image compression algorithms since the probability distribution of symbols encoded by those algorithms for typical images is close to exponential. The modified GR family of limited codeword length is used in the JPEG-LS algorithm recently included in the DICOM standard. In the paper we analyze effects of using both the GR and the limited codeword length GR codes for encoding actual images, where the set of encoded symbols is finite and the probability distribution is not exactly exponential. We also analyze a special case of encoding incompressible data. As a result we suggest new family of modified GR codes. We compare the efficiency of code families experimentally. Using the suggested family in the JPEG-LS algorithm we reduce the expansion of incompressible data from 0.5 bits per pixel to 0.166 bpp, and in a certain simpler algorithm (based on the FELICS algorithm) we reduce the expansion from 0.5 bpp to 0.002 bpp.

Keywords: multimedia systems; legal aspects; standards; medical image coding; lossless image compression; Golomb-Rice codes; data expansion; JPEG-LS; real life systems; non-typical data.

## 1 Introduction and overview

Lossless image compression algorithms are used for images that are documents, when lossy compression is not accepted by users, or when we have no knowledge whether lossy compression is allowed or not. Lossless algorithms are especially important for systems transmitting and archiving

medical data, because on the one hand lossy compression of medical images used for diagnostic purposes is forbidden by law, on the other hand the number and sizes of medical images stored or transmitted grows rapidly.

In the paper we research the Golomb-Rice (GR) family of codes which is the infinite family of prefix codes optimal for encoding symbols of exponential probability distribution. The GR family is used in predictive lossless image compression algorithms since the probability distribution of symbols encoded by those algorithms for typical images is close to exponential. A modified GR family of limited codeword length is used in the JPEG-LS, which is a new international standard of lossless image compression recently included in the DICOM standard.

Image compression researchers usually focus on results obtained for typical data, however for real life systems results for non-typical data should also be considered. For a compression algorithm the worst case image data results in data expansion. In applications, the results of compressing non-typical data should be known and the data expansion, unavoidable while processing incompressible data, should be minimized—primum non nocere.

Specific classes of data are incompressible for various algorithms, however there is a special class of data, that is incompressible for every compression algorithm. The sequence of symbols generated by the memoryless source with uniform probability distribution is incompressible by any compression algorithm. If the alphabet size equals $2^N$, then we cannot encode the sequence more efficiently then using the $N$-bit natural binary code. If we encode such an incompressible sequence using GR codes then we get the data expansion greater or equal to 0.5 bits per symbol, i.e. we get the average encoded symbol length greater or equal to $N + 0.5$ bits.

In the paper we analyze effects of using both the GR and the limited codeword length GR codes for encoding the real life images, where the set of encoded symbols is finite and the probability distribution is not exactly exponential. We also analyze a special case of encoding incompressible data. As a result we suggest new family of modified GR codes. We compare the families experimentally for the JPEG-LS algorithm and for a simple predictive image compression algorithm based on the data model of the FELICS algorithm.

This paper is organized as follows: In section 2, after introducing basic definitions we describe the GR family and the limited codeword length GR family. In section 3 we analyze properties of the families described in section 2 used for encoding actual images, where the set of encoded symbols is finite and the probability distribution is not exactly exponential. In section 4 we suggest new family of modified GR codes. In section 5 we present experimental results of compressing images using families described in sections 2 and 4. In section 6 we shortly summarize the paper.

## 2 Definitions

### 2.1 Coding and compression

The full introduction to the domain of coding, compression and image compression exceeds the scope of this paper, therefore we describe here basic definitions and ideas that are referred to in this paper only. An overview of methods and standards of digital image representation may be found in monographs [4, 5], the description of the FELICS algorithm in [2, 10] and the JPEG-LS description in [3, 11, 12].

In a general case we encode messages generated by the information source. Messages are sequences of symbols from the alphabet called the source alphabet. A code is a set of codewords assigned to individual symbols of the source alphabet. Codewords are sequences of symbols from the code alphabet. For the binary code the code alphabet is {0, 1}.

Let's assume:

- $S = \{s_0, s_1, \ldots, s_{n-1}\}$ is the source alphabet, $s_i$ is the symbol from the source alphabet, $\|S\| = n$ is the size of the source alphabet. Whenever a numerical value assigned to a symbol $s_i$ is required, we assume that it is a nonnegative integer $i$, $i < \|S\|$.
- $P = \{p_0, p_1, \ldots, p_{n-1}\}$ is the probability distribution of symbols from the source alphabet. The source generates symbol $s_i$ with the probability $p_i$. Depending on the assumed model of the source, the $p_i \in P$ may be constant for the whole message (memoryless source), or vary depending on the context of symbol $s_i$ and on the position of the symbol in the message.
- $K = \{\kappa_0, \kappa_1, \ldots, \kappa_{n-1}\}$ is the code, that assigns codewords to symbols of the source alphabet. The codeword $\kappa_i$ is assigned to the symbol $s_i$.

If all the codewords in a code consist of the same number of symbols, then it is the fixed length code. The fixed length natural binary code $K_n^b$ is used for encoding nonnegative integers less than $n$. To numbers $i \in \{0, 1, \ldots, n-1\}$, i.e. integers in range [0, $n$ – 1], it assigns codewords that are sequences of bits $\kappa_i = b_{N-1} \ldots b_1 b_0$, where $N = \lceil \log_2 n \rceil$, $b_j \in \{0, 1\}$ and $2^0 b_0 + 2^1 b_1 + \ldots + 2^{N-1} b_{N-1} = i$. This code is also called the $N$ -bit natural binary code.

The lengths of codewords in the variable length code are not equal. Variable length codes assign codewords to symbols of alphabets of limited or unlimited sizes. For example the unary code is defined for the set of nonnegative integers. To the given number $i$ it assigns a codeword consisting of $i$ ones and single zero.

The code is uniquely decodable if each sequence of codewords may be decoded unambiguously, i.e. there is only one way of dividing the sequence into individual codewords. A code is called the prefix code if it does not contain any codeword that is a prefix of other codeword in this code. Each prefix code is uniquely decodable. Code *K* is complete if for each sequence *D* of symbols that is a prefix of a codeword in *K*, the sequence $D' = D\alpha$, where $\alpha$ is a symbol from the code alphabet, is either a prefix of a codeword in *K* or a codeword in *K*.

The natural binary code is complete when the source alphabet size is an integer power of 2. Adjusted binary code $K_j^a$ is complete for each source. To integers in range [0, $j$ – 1] it assigns codewords, that are sequences of $\lfloor \log_2(j) \rfloor$ or $\lceil \log_2(j) \rceil$ bits (table 1). Let's assume $N = \lceil \log_2(j) \rceil$ and $n = 2^N$ ($2^{N-1} < j \le n$). If $i < n - j$ then the codeword of *i* in adjusted binary code is identical to the codeword of $i$ in $N-1$-bit natural binary code. In the opposite case ($i \ge n - j$) the codeword of *i* in adjusted binary code is identical to the codeword of $i + n - j$ in $N$-bit natural binary code. Adjusted binary code becomes the natural binary code when *j* is an integer power of 2.

Table 1

Adjusted binary codes

| **Integer** | **Range** | | | |
|---|---|---|---|---|
| | **[0, 4]** | **[0, 5]** | **[0, 6]** | **[0, 7]** |
| *i*=0 | 00 | 00 | 00 | 000 |
| *i*=1 | 01 | 01 | 010 | 001 |
| *i*=2 | 10 | 100 | 011 | 010 |
| *i*=3 | 110 | 101 | 100 | 011 |
| *i*=4 | 111 | 110 | 101 | 100 |
| *i*=5 | | 111 | 110 | 101 |
| *i*=6 | | | 111 | 110 |
| *i*=7 | | | | 111 |

For the probability distribution *P* the average code *K* length $L_K$ is:

$$L_K = \sum_{i=0}^{n-1} p_i \left| \kappa_i \right| ,$$

where $\left| \kappa_i \right|$ is the length of the codeword assigned to the symbol $s_i$ by the code *K*. Code *K* is optimal for the probability distribution *P* if for this probability distribution the average length of code *K* is minimal among all the codes. Optimal prefix code is complete. For a specific *P* more than one optimal code may exist.

The data compression may be defined as coding in order to minimize the encoded message length. Results of compression are measured using the compression ratio and expressed in bits per symbol: $z/u$, where $u$ denotes length (number of symbols) of the encoded message, $u > 0$, $z$—length (in bits) of the encoded message. For the compression ratio expressed in bits per symbol, the compression is better if the ratio is smaller. The data expansion occurs if after encoding symbols from the alphabet of size $2^N$ we have $uN < z$. We express the data expansion in bits per symbol:

$$\frac{z}{u} - N .$$

Unless indicated otherwise in the remainder of this paper we assume that the alphabet size is a positive integer power of 2.

If we compress images, then the message being encoded consists of pixels, therefore we express the compression ratio and the data expansion in bits per pixel [bpp]. In predictive image compression algorithms we do not encode actual pixels, instead we predict values of pixels using a prediction function. Then we encode the sequence of prediction errors, which is called residuum. The process of generating residuum using the prediction function is called decorrelation. In statistical compression algorithms we store information of the probability distribution of symbols being encoded in the data model.

### 2.2 The Golomb-Rice family (GR family)

The Golomb-Rice (GR) family is an infinite family of prefix codes. It is a subset of a family described in 1966 by S.W. Golomb (Golomb family), rediscovered independently by R. F. Rice [1]. GR codes are optimal for encoding symbols from an infinite alphabet of exponential symbol probability distribution (for some parameters of the exponential distribution). However for a finite alphabet GR codes are neither optimal nor complete.

Each code in the GR family is characterized by the nonnegative integer rank $k$. In order to encode nonnegative integer $i$ using the GR code of rank $k$, we first encode the codeword prefix: $\lfloor i/2^k \rfloor$ using unary code, then the suffix: $i \bmod 2^k$ using $k$-bit natural binary code. Sample codewords in table 2 contain a separator between prefix and suffix of the codeword, it is not a part of the codeword. Generating codewords from GR codes is very simple, since instead of division $i/2^k$ we just shift $i$ right $k$ bits, instead of modulo reduction we simply take $k$ least significant bits of $i$ and output them directly.

The probability distribution of sequences of symbols being encoded in predictive image compression algorithms, e.g. JPEG-LS and FELICS, is close to exponential. Furthermore the use of the family of codes significantly simplifies the compression algorithm since for a specific symbol, using information stored in the data model, we just select a code in the code family and output a codeword assigned to this symbol in the selected code.

Table 2

Golomb-Rice (GR) codes

| Integer | Code | | | |
|---|---|---|---|---|
| | $k = 0$ | $k = 1$ | $k = 2$ | $k = 3$ |
| 0 | 0• | 0•0 | 0•00 | 0•000 |
| 1 | 10• | 0•1 | 0•01 | 0•001 |
| 2 | 110• | 10•0 | 0•10 | 0•010 |
| 3 | 1110• | 10•1 | 0•11 | 0•011 |
| 4 | 11110• | 110•0 | 10•00 | 0•100 |
| 5 | 111110• | 110•1 | 10•01 | 0•101 |
| 6 | 1111110• | 1110•0 | 10•10 | 0•110 |
| 7 | 11111110• | 1110•1 | 10•11 | 0•111 |
| 8 | 111111110• | 11110•0 | 110•00 | 10•000 |
| 9 | 1111111110• | 11110•1 | 110•01 | 10•001 |
| 10 | 11111111110• | 111110•0 | 110•10 | 10•010 |
| 11 | 111111111110• | 111110•1 | 110•11 | 10•011 |
| 12 | 1111111111110• | 1111110•0 | 1110•00 | 10•100 |
| 13 | 11111111111110• | 1111110•1 | 1110•01 | 10•101 |
| 14 | 111111111111110• | 11111110•0 | 1110•10 | 10•110 |
| 15 | 1111111111111110• | 11111110•1 | 1110•11 | 10•111 |
| 16 | 11111111111111110• | 111111110•0 | 11110•00 | 110•000 |
| 17 | 111111111111111110• | 111111110•1 | 11110•01 | 110•001 |

### 2.3 The limited codeword length GR codes used in JPEG-LS

In the JPEG-LS algorithm a modified GR family—the limited codeword length GR family is used. The limiting of the codeword length was introduced in order to limit the data expansion when we select improper code using the data model. In the worst case the code of rank $k = 0$ is used to encode the symbol $s_{||S||-1}$, i.e. last symbol from alphabet *S*. In this case the codeword length becomes equal to the size of the alphabet. The data expansion for such symbol is $||S|| - \lceil \log_2 ||S|| \rceil$ bits, that is 248 bits for 256 symbol alphabet and over $6.5 \cdot 10^4$ bits for the alphabet size $2^{16}$. Limiting the codeword length we limit the negative effects of imperfect data modeling.

Codewords in the JPEG-LS family are constructed in a following way: we encode integer *i*, *i* is in the range $[0, 2^N - 1]$, the codeword length is limited to $l_{max}$ bits, the code rank *k* is in range $[0, N - 1]$. If $i < (l_{max} - N - 1) \cdot 2^k$ then we encode symbol *i* using *GR* code of rank *k*. In the opposite case we output a prefix: $l_{max} - N - 1$ using unary code, and a suffix: integer $i - 1$ using $N$-bit natural binary code $K^b_{2^N}$.

Some codewords in tables 3, 4 and 5 are underlined. For a specific code the underlined codeword and codewords above the underlined one are identical to equivalent codewords in the GR code.

Among other things the implementation issues decided that in the JPEG-LS algorithm unary coding is realized by outputting a sequence of zeroes and a single one instead of outputting a sequence of ones and single zero. To make the presentation of code families consistent the prefixes of codewords in table 3 are sequences of ones and a single zero.

As compared to GR codes only some of the JPEG-LS codes differ. JPEG-LS codes are not complete. The disadvantage of method of limiting the codeword length used in the JPEG-LS family is that the codes containing no codewords longer than $l_{max}$ are unnecessarily modified. See example in table 3. Some codewords in the code $k = 2$ are 8 bits long, while equivalent GR codewords are 6 bits long.

Table 3

The JPEG-LS family
for integers in range [0, 15], codeword length limited to 8 bits

| **Integer** | **Code** | | | |
|---|---|---|---|---|
| | ***k* = 0** | ***k* = 1** | ***k* = 2** | ***k* = 3** |
| 0 | 0• | 0•0 | 0•00 | 0•000 |
| 1 | 10• | 0•1 | 0•01 | 0•001 |
| 2 | 110• | 10•0 | 0•10 | 0•010 |
| 3 | 1110•0010 | 10•1 | 0•11 | 0•011 |
| 4 | 1110•0011 | 110•0 | 10•00 | 0•100 |
| 5 | 1110•0100 | 110•1 | 10•01 | 0•101 |
| 6 | 1110•0101 | 1110•0101 | 10•10 | 0•110 |
| 7 | 1110•0110 | 1110•0110 | 10•11 | 0•111 |
| 8 | 1110•0111 | 1110•0111 | 110•00 | 10•000 |
| 9 | 1110•1000 | 1110•1000 | 110•01 | 10•001 |
| 10 | 1110•1001 | 1110•1001 | 110•10 | 10•010 |
| 11 | 1110•1010 | 1110•1010 | 110•11 | 10•011 |
| 12 | 1110•1011 | 1110•1011 | 1110•1011 | 10•100 |
| 13 | 1110•1100 | 1110•1100 | 1110•1100 | 10•101 |
| 14 | 1110•1101 | 1110•1101 | 1110•1101 | 10•110 |
| 15 | 1110•1110 | 1110•1110 | 1110•1110 | 10•111 |

## 3 Properties of Golomb-Rice codes for encoding symbols from finite alphabets

### 3.1 Motivation of further modifying the GR codes

As we already mentioned GR codes are not optimal when the set of encoded symbols is finite and when the probability distribution is not exactly exponential. For that reason in recent algorithms the modified GR family is used. An example of such a modification is the limited codeword length GR family described in subsection 2.3.

It is obvious, that for the finite alphabet we should use only a finite subset of the infinite family. Considerations concerning the number of codes in the family are subject of subsection 3.2. In subsection 3.3 we discuss a special case of the uniform symbol probability distribution.

### 3.2 Number of codes in the family

In the known JPEG-LS and FELICS implementations for encoding symbols from $2^N$ symbol alphabet a GR family of $N$ codes (code ranks: 0, 1, … , $N-1$) is used [13, 14]. If the probability distribution of symbols being encoded in those algorithms is exponential then the family of $N-1$ codes (code ranks: 0, 1, … , $N-2$) should be used instead. It can be shown, that for encoding symbols of non increasing symbol probability distribution the average length of GR code of rank $k = N-1$ is greater or equal to the average length of code $k = N-2$.

We encode the message of symbols from alphabet of size $2^N$, the alphabet symbols occur with probabilities $p_i$, $i \in [0, 2^N - 1]$. By $R_k$ we denote the GR code of rank $k$, by $R_k(i)$ we denote the codeword assigned to integer $i$ by the $R_k$, $|R_k(i)|$ denotes the codeword $R_k(i)$ length.

**Theorem 1**

If $L_K$ is the average length of code $K$ then

$$L_{R_{N-1}} - L_{R_{N-2}} = \sum_{i=0}^{2^{N-2}-1} \left( p_i - p_{i+3\cdot 2^{N-2}} \right)$$

**Proof:**

For codes $R_{N-1}$ and $R_{N-2}$ we have:
1) for $n = 0, \dots, 2^{N-1} - 1 : |R_{N-1}(i)| = N$;
2) for $n = 2^{N-1}, \dots, 2^N - 1 : |R_{N-1}(i)| = N + 1$;
3) for $n = 0, \dots, 2^{N-2} - 1 : |R_{N-2}(i)| = N - 1$;
4) for $n = 2^{N-2}, \dots, 2 \cdot 2^{N-2} - 1 : |R_{N-2}(i)| = N$;
5) for $n = 2 \cdot 2^{N-2}, \dots, 3 \cdot 2^{N-2} - 1 : |R_{N-2}(i)| = N + 1$;
6) for $n = 3 \cdot 2^{N-2}, \dots, 2^N - 1 : |R_{N-2}(i)| = N + 2$;

Let's now compute the code length differences $|R_{N-1}(i)| - |R_{N-2}(i)|$ in integer subintervals as follows:

$$\left|R_{N-1}(i)\right| - \left|R_{N-2}(i)\right| = \begin{cases} 1 & \text{for} \quad 0 \le i < 2^{N-2} \\ 0 & \text{for} \quad 2^{N-2} \le i < 3 \cdot 2^{N-2} \\ -1 & \text{for} \quad 3 \cdot 2^{N-2} \le i < 2^{N} \end{cases}$$

Hence

$$L_{R_{N-1}} - L_{R_{N-2}} = \sum_{i=0}^{2^{N-2}-1} p_i - \sum_{i=0}^{2^{N-2}-1} p_{i+3\cdot 2^{N-2}} = \sum_{i=0}^{2^{N-2}-1} \left(p_i - p_{i+3\cdot 2^{N-2}}\right).$$

□

If we now assume, that the probability distribution is non increasing, i.e. $i < j$ implies $p_i \ge p_j$, then the theorem 1 implies:

$$L_{R_{N-1}} \ge L_{R_{N-2}}.$$

Using the $R_{N-2}$ instead of the $R_{N-1}$ for encoding symbols of non increasing probability distribution will not increase the average code length. Using the GR family of $N-1$ codes, i.e. removing from the family of $N$ codes the code $R_{N-1}$, we may simplify the data structures of the compression algorithm and the compression algorithm itself without worsening the compression ratio.

A special case of non increasing symbol probability distribution is the uniform distribution, for the uniform symbol probability distribution we have $L_{R_{N-1}} = L_{R_{N-2}}$. If non increasing distribution is not uniform, then $L_{R_{N-1}} > L_{R_{N-2}}$. If the alphabet size $y$ is not a power of 2 ($2^{N-1} < y < 2^{N}$), then we may treat the message as a message of symbols from alphabet $S = \{s_0, s_1, \ldots, s_{y-1}, s_y, \ldots, s_{n-1}\}$, where $n = 2^{N}$ and if $i \ge y$ then $p_i \equiv 0$. In this case, if the probability distribution is non increasing, then from the theorem 1 we have $L_{R_{N-1}} > L_{R_{N-2}}$.

Assuming the non increasing probability distribution is sufficient to show that $L_{R_{N-1}} \ge L_{R_{N-2}}$, however it is not a necessary condition. From theorem 1 we have the following sufficient and necessary condition for $L_{R_{N-1}} \ge L_{R_{N-2}}$:

$$\sum_{i=0}^{2^{N-2}-1} p_i \ge \sum_{i=0}^{2^{N-2}-1} p_{i+3\cdot 2^{N-2}}.$$

Removing the $R_{N-1}$ from the code family used by a compression algorithm for encoding symbols of non increasing probability distribution may improve the compression ratio. However the ratio may also be not affected, since the compression algorithm does not have to use all the codes from the family.

### 3.3 Case of the uniform probability distribution

The probability distribution of symbols encoded by predictive lossless image compression algorithms for typical images is close to exponential. For typical images we successfully use the GR codes. In the case of uniform probability distribution the use of GR codes results in the data expansion greater or equal to 0.5 bits per pixel. As mentioned before, in real life systems, including medical systems, the data expansion should be minimized.

The most straightforward method, that allows encoding the incompressible data without the data expansion is a method of inserting a code optimal for uniform symbol probability distribution to the code family. A natural binary code is optimal for uniform probability distribution. With the natural binary code we may replace the GR code of rank $R_{N-1}$. For typical images such a new family is as suitable as the unmodified GR family (see subsection 3.2). In the case of incompressible data, when use of the GR codes results in the data expansion, the natural binary code may be selected by the data model. Using this code we avoid the data expansion. Regardless of the probability distribution, which may be uniform, increasing or any other, the symbols being encoded may be simply copied to the output without the data expansion.

## 4 Suggested code family

Arguments mentioned earlier imply, that the code family designed for robust encoding of symbols from alphabet of size $2^N$, based on the GR family, should consist of $N-1$ limited codeword length GR codes and the $N$-bit natural binary code. Below we suggest such a family.

Codewords in the suggested family are constructed in a way similar to JPEG-LS family. For each code in the family we define the threshold $\pi_k$. We encode integer $i$, $i$ is in the range $[0, 2^N - 1]$, the codeword length is limited to $l_{max}$ bits, the code rank $k$ is in range $[0, N-1]$. If $i < \pi_k$ then we encode integer $i$ using $GR$ code of rank $k$. In the opposite case we output the prefix: $\pi_k / 2^k$ ones, and the suffix: $i - \pi_k$ encoded using natural binary code $K^b_{2^N - \pi_k}$ (table 4). The threshold $\pi_k$ is the smaller value selected from two following: the $(l_{max} - N) \cdot 2^k$ and the $2^N - 2^k$.

All the codes in the suggested family differ from the codes in the GR family and the code $k = N-1$ becomes the $N$-bit natural binary code. Lengths of all the codewords are either equal or shorter than equivalent codewords in the JPEG-LS family.

Some codes are complete (for example, codes $k = 1 \ldots 3$ from table 4). Remaining codes may be made complete using method described in [6], i.e. for $i \geq \pi_k$ we encode the codeword suffix using the adjusted binary code instead of natural binary code. The example of the complete variant of the suggested family (suggested-c family) is presented in table 5.

Table 4

The suggested family
for integers in range [0, 15], codeword length limited to 8 bits

| **Integer** | **Code** | | | |
|---|---|---|---|---|
| | ***k* = 0** | ***k* = 1** | ***k* = 2** | ***k* = 3** |
| 0 | 0• | 0•0 | 0•00 | 0•000 |
| 1 | 10• | 0•1 | 0•01 | 0•001 |
| 2 | 110• | 10•0 | 0•10 | 0•010 |
| 3 | 1110• | 10•1 | 0•11 | 0•011 |
| 4 | 1111•0000 | 110•1 | 10•00 | 0•100 |
| 5 | 1111•0001 | 110•1 | 10•01 | 0•101 |
| 6 | 1111•0010 | 1110•0 | 10•10 | 0•110 |
| 7 | 1111•0011 | 1110•1 | 10•11 | 0•111 |
| 8 | 1111•0100 | 1111•000 | 110•00 | 1•000 |
| 9 | 1111•0101 | 1111•001 | 110•01 | 1•001 |
| 10 | 1111•0110 | 1111•010 | 110•10 | 1•010 |
| 11 | 1111•0111 | 1111•011 | 110•11 | 1•011 |
| 12 | 1111•1000 | 1111•100 | 111•00 | 1•100 |
| 13 | 1111•1001 | 1111•101 | 111•01 | 1•101 |
| 14 | 1111•1010 | 1111•110 | 111•10 | 1•110 |
| 15 | 1111•1011 | 1111•111 | 111•11 | 1•111 |

Method of selecting the $\pi_k$ and coding of $i$ for $i \geq \pi_k$ results in limiting the length of codewords for some codes to less than $l_{max}$. For example the code $k = 1$ from table 4 (and table 5 as well) contains codewords not longer than 7 bits in spite of the $l_{max} = 8$ limit.

Coding images we deal with alphabets of sizes up to $2^{16}$. For coding symbols from those alphabets we use families of modified or unmodified GR codes containing no more than 16 codes. Knowing the alphabet size and the codeword length limit we may calculate thresholds $\pi_k$ for all the codes in the family once (before the coding starts) and not calculate the $\pi_k$ threshold each time the code of rank $k$ is used. Using the described modifications of GR family during compression of typical images almost all pixels are encoded using codewords identical to equivalent codewords in the unmodified GR family. Therefore for typical images we do not expect to get a significant improvement in the average code length by using the JPEG-LS family or the suggested family instead of the unmodified GR family.

Table 5

The suggested-c family
for integers in range [0, 15], codeword length limited to 8 bits

| Integer | Code | | | |
|---|---|---|---|---|
| | $k = 0$ | $k = 1$ | $k = 2$ | $k = 3$ |
| 0 | 0• | 0•0 | 0•00 | 0•000 |
| 1 | 10• | 0•1 | 0•01 | 0•001 |
| 2 | 110• | 10•0 | 0•10 | 0•010 |
| 3 | 1110• | 10•1 | 0•11 | 0•011 |
| 4 | 1111•000 | 110•1 | 10•00 | 0•100 |
| 5 | 1111•001 | 110•1 | 10•01 | 0•101 |
| 6 | 1111•010 | 1110•0 | 10•10 | 0•110 |
| 7 | 1111•011 | 1110•1 | 10•11 | 0•111 |
| 8 | 1111•1000 | 1111•000 | 110•00 | 1•000 |
| 9 | 1111•1001 | 1111•001 | 110•01 | 1•001 |
| 10 | 1111•1010 | 1111•010 | 110•10 | 1•010 |
| 11 | 1111•1011 | 1111•011 | 110•11 | 1•011 |
| 12 | 1111•1100 | 1111•100 | 111•00 | 1•100 |
| 13 | 1111•1101 | 1111•101 | 111•01 | 1•101 |
| 14 | 1111•1110 | 1111•110 | 111•10 | 1•110 |
| 15 | 1111•1111 | 1111•111 | 111•11 | 1•111 |

## 5 Experimental comparison of the GR families

### 5.1 Procedure

A test image set described in the subsection 5.2 was used to perform experiments with all the families described in this paper: GR, JPEG-LS, suggested and suggested-c. For all the families, but the GR family, the codeword length was limited to 32 bits (the alphabet size was 256). The experiments were performed for two following algorithms: JPEG-LS algorithm and the simpler algorithm based on the data model known from the FELICS algorithm. The FELICS data model was used to adaptively select, in the code family, the code for encoding residuum symbols of images decorrelated using the default prediction function of the Lossless JPEG algorithm. The decorrelation procedure is described in a more detailed manner in [9]. In the research an implementation of JPEG-LS by I. R. Ismaeil and F. Kossintini [14] was used. Compression algorithm based on the FELICS data model was implemented in the C language, the decorrelation procedure was implemented earlier [8]. The parameters of the FELICS data model were the same as in the *mg* system [10, 13].

### 5.2 Test image set

The large set of test images was used, the set is described more thoroughly in [7, 9]. The set consists of some disjoint groups of images and some images. All the images are 8 bit grayscale images. The set contains groups of images of significantly different sizes and resolutions, noisy images and image containing nothing but the noise. The set contains:

*funet* —group of 9 well known images ("*lena*", "*bridge*", "*boats*" etc.) often used in the image compression research, size: 64–405 kB.

*corel* —8 images from Corel Professional Photos library, size: 384 kB.

*i_300* —5 images which include 3 photographs scanned at 300 dpi, size: 1994 kB, and 2 images composed using all *corel* images, size: 3072 kB.

*i_150* —above scaled 50%, size: 486–768 kB.

*i_75* —above scaled 50%, size: 121–192 kB.

*i_37* —above scaled 50%, size: 30–48 kB.

*i_18* —above scaled 50%, size: 7–12 kB.

*noise* —8 noisy images created using two images from *i_150* group ("*big_150*" and "*ph2_150*") by adding the Gaussian noise (mean 0 and variances 1, 4, 16 and 64).

"*random*" —random pixel intensities (incompressible), one image, size 4247 kB.

The set is large to permit analyzing average results for groups of images instead of, arguable, results for individual images. From all the set an additional group of typical images was selected: *normal*. The group *normal* consists of all the images from: *funet*, *corel*, *i_300*, *i_150* and *i_75*.

### 5.3 Results

Results for the JPEG-LS algorithm are presented in table 6, for the other algorithm in table 7. Tables 6 and 7 contain average compression ratios for disjoint image groups, for the "*random*" image and for the *normal* group.

For all the groups but the *noise* group, groups of smallest images and the "*random*" image, average code lengths for all the families do not differ significantly. Relative effects of using individual families are similar for both algorithms, however greater differences may be noticed for the algorithm based on the FELICS data model.

For typical images replacing the GR family by the limited codeword length family improves the average (group *normal*) compression ratio by about 8‰ for the FELICS data model, for the JPEG-LS algorithm we gain about 0.4‰ only. Improvement in the average codeword length is greater for smaller images. Relatively simple limiting of the codeword length, used in the JPEG-LS family, improves a little average code length for typical images. Further improvement using the suggested families is smaller.

Table 6

Compression results for the JPEG-LS algorithm [bpp]

| Group (image) | Code family | | | |
|---|---|---|---|---|
| | **GR** | **JPEG-LS** | **Suggested** | **Suggested-c** |
| *funet* | 4.578 | 4.576 | 4.576 | 4.576 |
| *corel* | 3.025 | 3.025 | 3.024 | 3.024 |
| *i_18* | 5.084 | 5.068 | 5.066 | 5.064 |
| *i_37* | 4.491 | 4.484 | 4.483 | 4.482 |
| *i_75* | 3.894 | 3.891 | 3.891 | 3.890 |
| *i_150* | 3.169 | 3.168 | 3.168 | 3.168 |
| *i_300* | 2.717 | 2.717 | 2.716 | 2.716 |
| *noise* | 5.804 | 5.803 | 5.770 | 5.770 |
| "*random*" | 8.505 | 8.505 | 8.166 | 8.166 |
| *normal* | 3.572 | 3.571 | 3.570 | 3.570 |

Table 7

Compression results for the FELICS data model [bpp]

| Group (image) | Code family | | | |
|---|---|---|---|---|
| | **GR** | **JPEG-LS** | **Suggested** | **Suggested-c** |
| *funet* | 5.002 | 4.945 | 4.945 | 4.943 |
| *corel* | 3.532 | 3.513 | 3.512 | 3.512 |
| *i_18* | 6.324 | 5.814 | 5.811 | 5.805 |
| *i_37* | 5.210 | 5.068 | 5.067 | 5.064 |
| *i_75* | 4.483 | 4.441 | 4.440 | 4.438 |
| *i_150* | 3.800 | 3.779 | 3.779 | 3.777 |
| *i_300* | 3.300 | 3.291 | 3.291 | 3.291 |
| *noise* | 6.001 | 5.984 | 5.947 | 5.946 |
| "*random*" | 8.508 | 8.501 | 8.002 | 8.002 |
| *normal* | 4.100 | 4.068 | 4.067 | 4.066 |

The difference in results obtained for the non complete and complete variant of the suggested family is practically negligible for tested images.

In the case of noisy images *noise* and the incompressible "*random*" image results for the GR family and the JPEG-LS family are similar. For those images better results are obtained using the suggested families, significantly better for the incompressible data. Better average code lengths are obtained for the suggested families thanks to the natural binary code present in those families (the code $k = N - 1$). For the suggested families greater expansion of

the incompressible data is observed for the JPEG-LS algorithm (0.166 bpp vs. 0.002 bpp for the FELICS model) that for normal images significantly outperforms the FELICS model. The reason is that in the JPEG-LS the unmodified GR family is modeled regardless of the family actually used, as opposed to modeling the actual family in the FELICS data model.

Based on the results we conclude that limiting the GR codeword length allows a little improvement of compression results for typical images. It also bounds the local data expansion what may be important in practical applications. Further modifications of GR family that result in inserting the natural binary code to the family improve significantly the compression results for the incompressible data and for noisy images. The expansion of the incompressible data was about 0.5 bpp for the families GR and JPEG-LS used in both the compression algorithms. Using the suggested family in the JPEG-LS algorithm we reduce the expansion of incompressible data 3 times, and in a compression algorithm based on the FELICS data model we reduce the expansion over 200 times.

Generating codewords from the introduced code families is, as generating codewords from the JPEG-LS family, a little more complicated than generating codewords from unmodified GR family. Generating codewords from the complete variant of the suggested family is a little more complicated than for a not complete variant of the suggested family. Since differences in the results for those two variants are practically negligible, the not complete suggested family may be a better choice for practical applications. As compared to the GR family for both examined algorithms the suggested family improves average code length for both typical and non-typical data.

## 6 Conclusions

In the paper we analyze effects of using both the GR and the limited codeword length GR codes for encoding actual images, where the set of encoded symbols is finite and the probability distribution is not exactly exponential. We also analyze a special case of encoding incompressible data. As a result we suggest new family of modified GR codes that contains natural binary code.

We compare the families experimentally for the JPEG-LS algorithm and for the algorithm based on the data model known from the FELICS algorithm. The most significant improvement in compression results is observed for the incompressible data. Using the suggested family in the JPEG-LS algorithm we reduce the expansion of incompressible data from 0.5 bits per pixel to 0.166 bpp, and in the algorithm based on the FELICS data model we reduce the expansion from 0.5 bpp to 0.002 bpp. In real life systems that use the data compression the suggested family may significantly improve results of processing non-typical data.

**Acknowledgment:** The research supported by grant BK-279/Rau-2/2002 was carried out at the Institute of Computer Science, Silesian University of Technology.

## References

1. Golomb, S.W.: Run-Length Encodings. IEEE Transactions on Information Theory, IT-12, pp.: 399-401, July 1966.
2. Howard, P.G.; Vitter, J.S.: Fast and efficient lossless image compression. Proceedings DCC '93. Data Compression Conference, IEEE Comput. Soc. Press Los Alamitos, CA, USA, pp. 351-60.
3. ISO/IEC JTC1/SC29 WG1 FCD 14495, public draft: Lossless and near-lossless compression of continuous-tone still images (JPEG-LS). ISO Working Document ISO/IEC JTC1/SC29/WG1 N522, July 1997.
4. Skarbek, W.: Metody reprezentacji obrazów cyfrowych. Akademicka Oficyna Wydawnicza PLJ, Warsaw 1993.
5. Skarbek, W. red.: Multimedia algorytmy i standardy kompresji. Akademicka Oficyna Wydawnicza PLJ, Warsaw 1998.
6. Starosolski, R.: Fast, robust and adaptive lossless image compression. Machine Graphics and Vision, Warsaw 1999, Vol. 8(1) pp. 95-116.
7. Starosolski, R.: Fast and adaptive lossless grayscale image compression using the LZW algorithm. Archiwum Informatyki Teoretycznej i Stosowanej, Tom 11 (1999), z.2, Katowice 1999, pp. 171-93.
8. Starosolski, R.: Szybkie, bezstratne oraz adaptacyjne metody kompresji obrazów w odcieniach szarości. Zeszyty Naukowe Politechniki Śląskiej, Informatyka z.37, Gliwice 1999, pp. 121-45.
9. Starosolski, R.: Bezstratne algorytmy kompresji obrazów. PhD dissertation, Institute of Computer Science, Silesian University of Technology, Gliwice 2001.
10. Witten, I.H.; Moffat, A.; Bell, T.C.: Managing Gigabytes. Van Nostrand Reinhold, second edition, USA 1999.
11. Weinberger, M.J.; Seroussi, G.; Sapiro, G.: LOCO-I: A low complexity, context-based, lossless image compression algorithm. Proceedings DCC '96. Data Compression Conference, IEEE Comput. Soc. Press Los Alamitos, CA, USA 1996, pp. 140-9.
12. Weinberger, M.J.; Seroussi, G.; Sapiro, G.: The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. IEEE Trans. Image Processing, August 2000, Vol 9(8), pp. 1309-24.
13. Bell, T.C.; Moffat, A.; Witten, I.; Zobel, J; Inglis, S.; Nevill-Manning, C.; Sharman, N.; Shimmin, T.: The MG Information Retrieval System (version 1.2) and documentation. Dept. of Computer Science & Software Engineering, University of Melbourne, 1995, ftp://munnari.oz.au/pub/mg.
14. Ismaeil, I.R.: Kossintini, F.: JPEG-LS Lossless Image Compression Standard (C version 1.0) program and documentation. Dept. of Electrical and Computer Engineering, University of British Columbia, 1997, http://spmg.ece.ubc.ca/research/jpeg/jpeg_ls/jpegls.html.