Memoirs of the Scientific Sections of the Romanian Academy
Tome XXXIV, 2011

ON SOME CHARACTERISTICS OF A NOVEL LOSSLESS DATA COMPRESSION ALGORITHM BASED ON POLYNOMIAL CODES

ADRIANA SÎRBU and IOAN CLEJU

Technical University "Gh. Asachi" of Iasi
Faculty of Electronics, Telecommunications and Information Technology
E-mail: icleju@etti.tuiasi.ro
Corresponding author: asirbu@etti.tuiasi.ro

A new lossless data compression algorithm based on some characteristics of polynomial codes, namely polynomial codes data compression (PCDC), is introduced. In this scheme, the binary sequence to be compressed is divided into blocks of n bits length. Exploiting the property of a codeword to be multiple of a certain polynomial, one can decide to shorten the length of the given sequence by sending only the quotient polynomial. This approach needs an extra bit to signal the performed shortening. The performance of the PCDC algorithm is analyzed and the results obtained are presented. Comparison between the proposed algorithm and similar ones are commented. A possible hardware implementation is also described.

Key words: lossless data compression, polynomial codes, codewords, hardware implementation.

1. INTRODUCTION

Data compression has wide application in terms of information storage and transmission. In recent years, in order to reduce the size of the data so that less disk space for storage and/or less bandwidth for transmission on data communication channels are required, several data compression algorithms were designed [7], [8], [11].

Data compression algorithms are generally classified into either lossless or lossy [14]. Lossless data compression allows the exact original data to be reconstructed from the compressed data, while lossy data compression, allows an approximation of the original data to be reconstructed, with the benefit of better compression ratios.

Recently, a special attention was paid to data compression algorithms evolving from coding theory [2], [3], [6]. So, for example, low-density parity-check codes (LDPC) [4], Fountain codes [5], punctured turbo codes [12], or Hamming codes [1], have been used for this purpose.

In [1] the author proposes a novel lossless binary data compression scheme based on error correcting Hamming codes. The basic idea is to divide the sequence to be

compressed into blocks of *n* bits, in order to take advantage of the properties of Hamming code codewords that consists of *p* parity bits and *d* data bits (*n*= *d* + *p*), keeping the original notations. Each block is tested to check if it represents a valid or a non-valid codeword. For valid blocks, only the *k* data bits are retained to be stored or transmitted, while for non-valid blocks, the whole block (*n* bits) remains unchanged. In order to distinguish the two cases, an additional bit is inserted: *1* for valid Hamming codewords and *0* for non-valid Hamming codewords.

The present work proposes a similar approach for data compression, exploiting this time the properties of polynomial codes, generated by a generator polynomial, g(x). The resulted algorithm proves to be more efficient as compared with similar data compression schemes, mainly the algorithm in [1].

The remainder of this paper is organized as follows: in Section 2 we describe the principle of the method. In section 3 we derive the formula and the resulted bounds for the compression ratio. In section 4 one comments a possible hardware implementation of the proposed algorithm. Finally, conclusions and further research goals are presented.

2. PRINCIPLE OF THE METHOD

In what follows, we propose a new compression algorithm, inspired from the basic principle of the method described in [1], using this time polynomial codes as a mean to obtain data compression.

For these codes, a sequence of n binary symbols $a_{n-1}\ a_{n-2}\ \dots\ a_1\ a_0$ is described as a polynomial $a(x)= a_{n-1}x^{n-1} + a_{n-2}\ x^{n-2} + \dots + a_1\ x+a_0$. The generator polynomial of the code is $g(x)= x^m + g_{m-1}\ x^{m-1} + \dots + g_1\ x + 1$, where m is the degree of the polynomial and it represents the number of the parity-check symbols.

One can obviously write :

$$n = k + m \tag{1}$$

where *n* is the total number of the symbols of a codeword and *k* is the number of information symbols.

The polynomial a(x) of degree *n* - 1 is a codeword if and only if it is divisible by the generator polynomial g(x). In literature, [9], it is proved that if a(x) is divisible by g(x), that is:

$$a(x)= g(x)q(x) \quad (2)$$

then the polynomial q(x) represents exactly the information symbols or can be uniquely mapped to it.

That is why, in what follows, we will consider, without loosing generality, that q(x) represents exactly the information symbols.

The compression method we propose consists of the following operations.

The original sequence of symbols is divided in groups of n symbols, treated as polynomials in what follows :

$$a^1_{n-1}a^1_{n-2}\ldots a^1_0\ a^2_{n-1}a^2_{n-2}\ldots a^2_0\ldots a^i_{n-1}a^i_{n-2}\ldots a^i_0\ldots a^j_{n-1}a^j_{n-2}\ldots a^j_0\ldots \quad (3)$$

Each polynomial which corresponds to such a group is divided by the generator polynomial.

According to the polynomial division theorem, one can write :

$$a(x)=g(x)q(x)+r(x) \quad (4)$$

where:

- $q(x)= q_{k-1} x^{k-1} + \ldots + q_1 x+q_0$ represents the quotient polynomial, associated with *k* symbols.

- $r(x)= r_{m-1} x^{m-1} + \ldots + r_1 x+r_0$ represents the remainder polynomial, associated with *m* symbols.

Taking into account the statements above, based on the properties of polynomial codes, it implies that:

- if r(x)=0 then a(x) is a codeword and q(x) represents the information symbols

- if r(x)≠0 then a(x) is a not a codeword

We propose that, for the case when a(x) is a codeword, one transmits only the information symbols, that is only the quotient polynomial q(x).

If a(x) is not a codeword, then one transmits both the quotient q(x) and the reminder r(x), that is *n* symbols (i.e. the same number of bits as the initial sequence a(x)).

The two cases, a(x) - codeword or a(x) not codeword, are distinguished by means of a marker flag, *mf*, as shown in Figure 1.

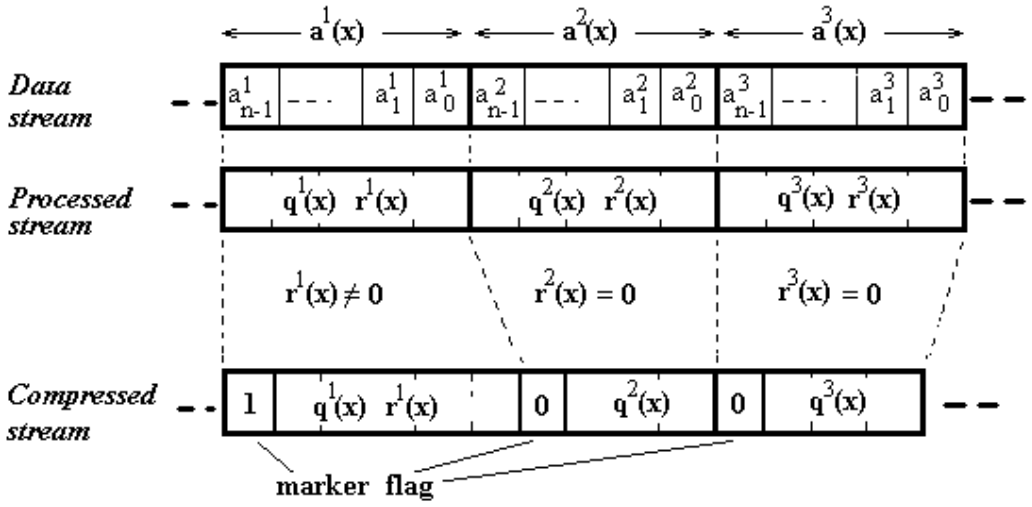

Fig. 1 - Principle of the proposed lossless data compression algorithm

3. DERIVATION OF THE COMPRESSION RATIO FORMULA

In what follows we will derive a relation that can be used to compute the compression ratio C as a function of the size of the initial data sequence *n* and the degree of the generator polynomial, *m*.

We consider the original sequence of symbols S_0 divided into *b* blocks of n symbols, that is $S_0 = b* n$. Let us suppose that among the *b* blocks, *v* are codewords and *w* are not codewords, that is $b = v + w$.

Then, the compressed sequence has the length S_c :

$$S_c = w(n+1) + v(k+1) \tag{5}$$

Taking into account (1) and (2), one can derive :

$$S_c = n*b + b - v*m \tag{6}$$

and further on, the compression ratio:

$$C = \frac{S_0}{S_c} = \frac{n \cdot b}{n \cdot b + b - v \cdot m} = \frac{n}{n + 1 - p \cdot m} \tag{7}$$

where we denoted by p the probability of a sequence of length *n* to be a codeword, that is $p=v/b$.

From equation (7) it is obvious that for p=1 (meaning that all the sequences of n bits are codewords) one gets the maximum compression ratio C_{max}.

On the other hand, for p=0 (meaning that none of the sequences of n bits is a codeword), it implies that $C_{min}=n/(n+1)$, obviously a subunitary value.

We denote by $p_{crit} = 1/m$ the value of the probability for which the compression ratio is unitary (C=1 in (7)). It is obvious that for $p > p_{crit}$, the proposed method allows obtaining a compression of the initial data sequence, while for $p < p_{crit}$, one gets an inflation, that is literally, the resulted sequence after applying the proposed method is in fact larger than the initial one.

In Table 1 we present the values of C_{max} for several sequence lengths, n, and polynomial degrees, m. In Table 2, we present the values of C_{min} as function of n (it does not depend on m).

Table 1

Values of Cmax

n	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
m=3	2	1.67	1.5	1.4	1.33	1.28	1.25	1.22	1.20	1.18	1.16	1.15	1.14	1.13	1.12	1.12	1.11
m=4		2.5	2	1.75	1.60	1.5	1.42	1.37	1.33	1.30	1.27	1.25	1.23	1.21	1.2	1.19	1.18
m=5			3	2.33	2	1.8	1.66	1.57	1.50	1.44	1.40	1.36	1.33	1.3	1.29	1.27	1.25
m=6				3.5	2.66	2.25	2	1.83	1.71	1.62	1.55	1.50	1.45	1.42	1.38	1.36	1.33
m=7					4	3	2.5	2.2	2	1.86	1.75	1.67	1.6	1.55	1.5	1.46	1.43
m=8						4.5	3.33	2.75	2.4	2.17	2	1.88	1.78	1.7	1.64	1.58	1.54
m=9							5	3.67	3	2.6	2.33	2.14	2	1.89	1.8	1.73	1.67
m=10								5.5	4	3.25	2.8	2.5	2.29	2.13	2	1.9	1.82

Table 2

Values of Cmin

n	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
C_{min}	0.8	0.833	0.857	0.875	0.888	0.9	0.909	0.916	0.923	0.928	0.933	0.937	0.941	0.944	0.947	0.95	0.952

One can choose different generator polynomials g(x), of different degrees, n =3,4,.. We have used, for generating data in Table 1 and Table 2 well known primitive polynomials, used in cyclic codes, [13] :

- for m=3, we have proposed $g(x) = x^3+x+1$
- for m=4, we have proposed $g(x) = x^4+x+1$
- for m=5, we have proposed $g(x) = x^5+ x^2+1$
- for m=6, we have proposed $g(x) = x^6+x+1$
- for m=7, we have proposed $g(x) = x^7+x+1$

- for m=8, we have proposed g(x) = $x^8+ x^7+x^2+ x+1$
- for m=9, we have proposed g(x) = $x^9+ x^4+1$
- for m=10, we have proposed g(x) = $x^{10}+x^3+1$

It is extremely important to notice the fact that equation (2) holds for any polynomials a(x) and g(x). Tables 1 and 2 have been constructed taking into account this aspect. This fact also implies that there is no imposed correlation between the length of the initial sequence, *n*, and the degree of the generator polynomial, *m*. In this respect, we consider that our method has a greater efficiency as compared with the approach proposed in [1], where the length *n* and the value of *m* were restricted to be $n = 2^m - 1$.

In Figure 2 we present the variation of C_{min} and C_{min} as functions of *n*, for different values of *m*.

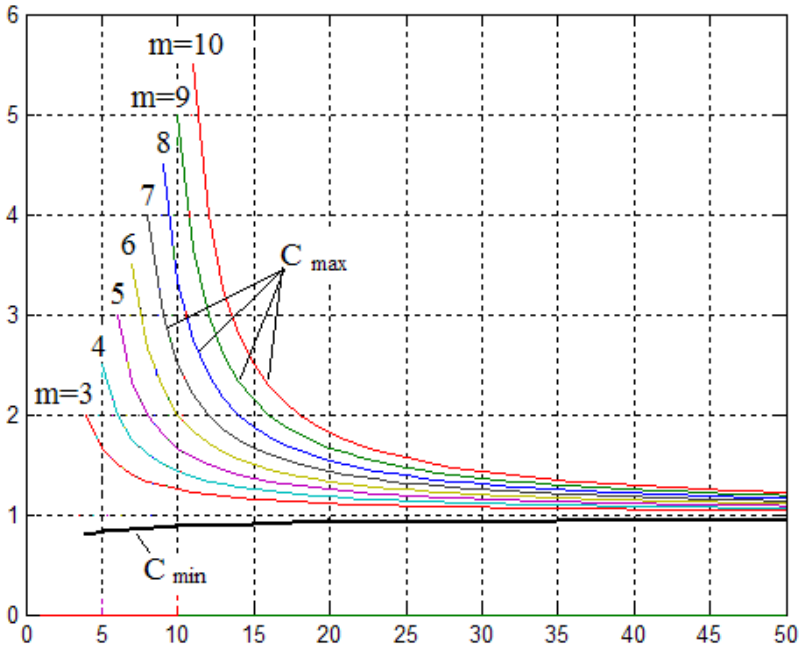

Fig. 2 - Cmin and Cmin variation as functions of n, with m as parameter

While Tables 1 and 2 have been completed for values of *n* up to 20, the graphics in Figure 2 have been drawn for values up to 50. One can notice that as *n* increases, both C_{min} and C_{max} (the last one for all the values of *m*) approach unity, that is unitary compression ratio. Though, taking a closer look to C_{max} , it is to be observed that for

larger values of *m* one gets a larger compression ratio. Therefore, one can state a practical rule: even though there is no connection between the values of *n* and *m*, as one selects a larger value for *n*, it is recommended to choose a larger value for *m*, the degree of the generator polynomial, in order to obtain a larger compression ratio.

Intensive simulation studies have confirmed the performances of the proposed data compression algorithm. In Figure 3 we present the structure of the simulation set-up.

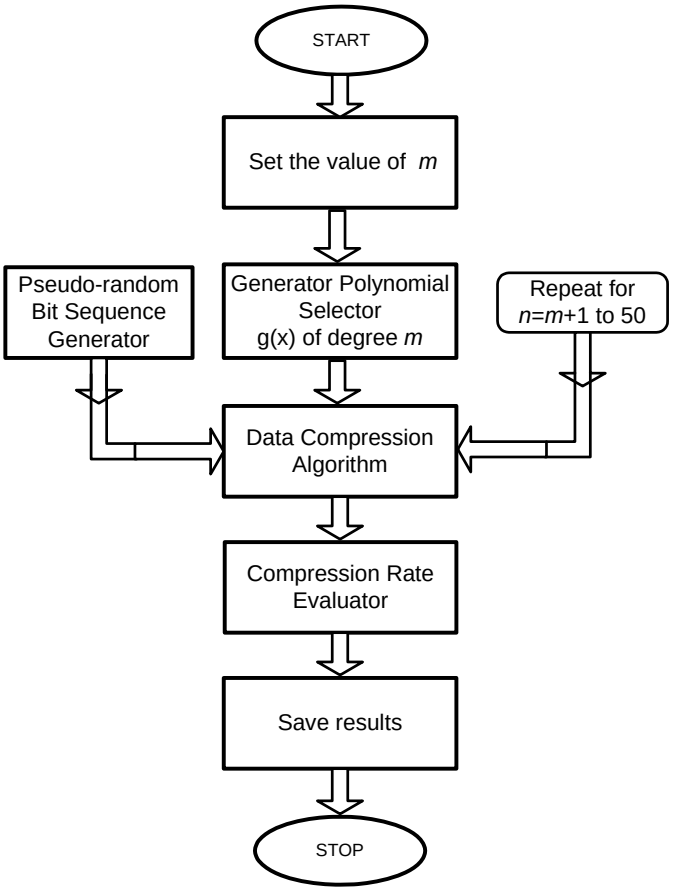

Fig. 3 - Structure of the simulation set-up

The simulation software includes a pseudo-random bit sequence generator implemented according to [10]. For each generated (*n,m*) pair, an appropriate generator polynomial, g(x), of degree *m*, is selected from an available database. The resulted compression rates are saved in a file on disk.

In order to outline the advantages of the proposed algorithm as compared to the inspiring one in [1], we assert the following remarks:

Remark 1

The restriction $n = 2^m - 1$, taken into consideration in [1], derives from the constraint proper to error-correcting codes (the necessity to correct errors imposes the *n* should not be larger than $2^m - 1$). On the other hand, n can be smaller than $2^m - 1$, condition which provides the so called "shortened codes". For the case of compression, what is important is the identification of a codeword only from its information symbols, but this is possible even if the chosen code has not error-correction properties. In other words, for the PCDC case we can increase the value of *n*, al least theoretically, to infinity.

Remark 2

In principle, even in [1] one can give up the restriction $n = 2^m - 1$and choose a larger *n*, but this will imply using other control matrices H, which will provide other parity relationships. In our case, we can increase the value of n, the number of the symbols that constitute a group, without modifying the generator polynomial, g(x), that is, without modifying either the algorithm, or the implementation. From this point of view, using the polynomial codes as "compression vehicle" seems a more flexible idea.

Remark 3

The idea of modifying dimension *n* allows us developing compression algorithms of "multi-length" type, which will suppose repeating the current compression algorithm for several values of *n* and then choosing the best solution.

Remark 4

In the case of "multi-length" compression, as argued above, one can modify the value of *n* arbitrarily. Based on the original data flow one can choose the most convenient value for n from the compression ratio point of view. However, Table I shows that the most convenient compression ratio is obtained for a generator polynomial g(x) with maximum possible degree.

4. IMPLEMENTATION SOLUTION

In Figure 4 we present the block diagram of a possible hardware implementation of the proposed algorithm.

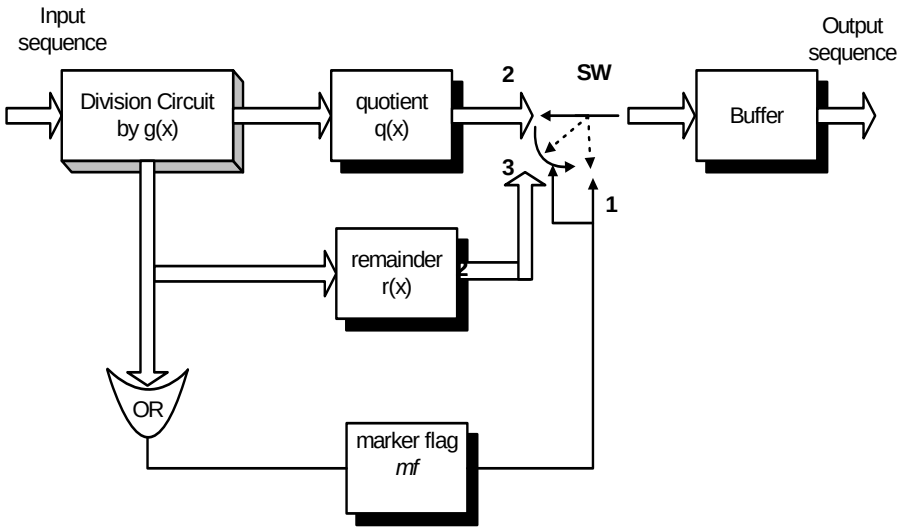

Fig. 4 - Block diagram of the suggested hardware implementation

The sequence to be compressed is processed by a division circuit, able to provide the quotient and the remainder (or an equivalent) for the division of the input polynomial to the chosen generator polynomial. The OR circuit detects the null remainder and, implicitly, the value of the marker flag bit, *mf*. Apart from being transmitted, the marker flag bit itself controls the flow of the output sequence, preventing the transmission of the zero bits sequences corresponding to null remainders.

In detail, for the case when the marker flag *mf* has the value 1, the switch SW is positioned first on position 1 and bit 1 is inserted into the compressed data sequence. Then, the switch SW is placed successively on position 2 and 3 respectively, allowing the quotient and the reminder provided by the division circuit to be inserted in the compressed sequence. For the case when the marker flag *mf* has the value 0, the switch SW is positioned first on position 1 and bit 0 is inserted in the compressed data sequence. Then the switch SW is placed only in position 2, inserting only the quotient of the division process (in this way, the sequence of *m* zero value bits corresponding to the null remainder is not inserted, and so, data compression is obtained).

For real time compression, in order to speed up the process of compressing, two such circuits, connected in parallel, can process alternatively blocks of n bits.

In Figure 5 we present a possible implementation of the division circuit using shift registers and XOR circuits.

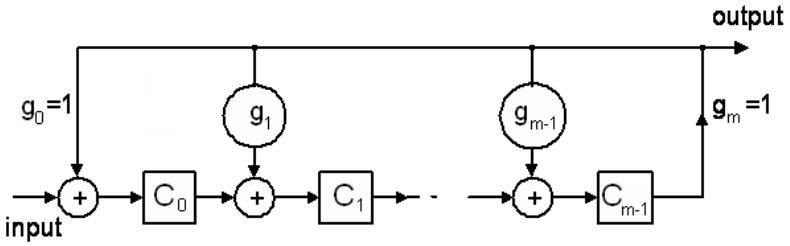

Fig. 5 - Block diagram of the division circuit

The implementation variant having the XOR circuits placed between the cells of the shift register provides at output the quotient of the polynomial division, while in the cells of the shift register one gets the remainder of the division process. The division circuit for $m \leq 10$ can be easily implemented using the PAL 22 V10 circuit. The solution is extremely attractive as it requires only one circuit for the whole division circuit.

For the decompression process, the reverse operation of multiplication by g(x), can be also implemented using only one PAL circuit adequately programmed.

5. CONCLUSIONS

A new lossless data compression algorithm was proposed based on the properties of the polynomial codes, generated by a given polynomial g(x). The main advantage of the scheme is that it offers a greater flexibility, as the values of the parameters n and m are uncorrelated. The method is also attractive due to the simplicity of the operations involved (multiplication/division of polynomials). A hardware implementation is also described.

Authors contributions: The authors have equally contributed to the elaboration of the paper.

REFERENCES

1. AL-BAHADILI H., *A novel lossless data compression scheme based on the error correcting Hamming codes*, Computers and Mathematics with Applications, 2008, **56**, 143–150.

2. CLAIRE G., SHAMAI S., VERDU S., *A New Data Compression Algorithm for Sources with Memory based on Error Correcting Codes*, ITW 2003, Paris, 291-295.

3. CLAIRE G., SHAMAI S., VERDU S., *Lossless Data Compression with Error Correcting Codes*, ISlT 2003, Yokohama, Japan, 2003.

4. CLAIRE G., SHAMAI S., VERDU S., *Universal Data Compression with LDPC Codes,* Third International Symposium On Turbo Codes and Related Topics, Brest, France, September 1-5, 2003.

5. CLAIRE G., SHAMAI S., SHOKROLLAHI A., VERDU S., *Fountain Codes for Lossless Data Compression*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 2005, **68**, 1-18.

6. COVER T. M., JOY A. THOMAS, *Elements of Information Theory*, John Wiley & Sons, 2006.

7. HANKERSON D.R., *Introduction to Information Theory and Data Compression*, CRC Press, 2003.

8. HELD G., MARSHELL T. R., *Data and Image Compression*, John Wiley & Sons, 1996.

9. MORELOS-ZARAGOZA R. H., *The Art of Error Correcting Coding*, John Wiley, 2006.

10. PRESS W., TEUKOLSKY S., et al., *Numerical Recipes. The Art of Scientific Computing*, 3rd Edition, Cambridge University Press, 2007.

11. SAYOOD K., *Introduction to Data Compression*, Morgan Kaufmann Publishers, 2000.

12. ZHAO Y., GARCIA-FRIAS J., *Data Compression of Correlated Non-Binary Sources Using Punctured Turbo Codes*, Proceedings of the Data Compression Conference, 2002, 242-246.

13. ZIVKOVIC M., *A table of primitive binary polynomials*, Math. Comp., 1994, **62**, 385-386.

14. ⁂ http://www.data-compression.com/lossless.shtml, 2010

Received December 9, 2010