MSIM: A Pattern Based Lossless Data Compressor

[1]Angel Kuri-Morales, Oscar Herrera-Alcántara

[1]Instituto Tecnológico Autónomo de México.
Río Hondo No.1, Tizapán San Angel, C.P. 01000, MÉXICO.
akuri@itam.mx

Abstract. We present a lossless data compression method based on the identification of unbounded patterns within a message. The patterns are called *metasymbols* and consist of groups non-necessarily neighboring symbols. Metasymbols are selected under the criterion that the original message can be re-expressed in the most compact way using an efficient coding scheme. We have shown that the problem of discovering metasymbols is NP-Hard and as an approach to the solution we developed an algorithm called MSIM. The performance of MSIM was tested on a set of messages that contains known metasymbols. Then we use a reference compressor (REF) which yields a compression function K to evaluate the theoretical limit of compression and then compare the performance of MSIM relative to REF. MSIM has a statistically significant very similar performance to REF and represents not only a compressor but also an excellent tool to discover unknown patterns in arbitrary messages.

1 Introduction

During the last decades several methods for lossless data compression have been proposed such as RLE[1], Huffman Coding[2], Arithmetic Coding[3], LZ[4], BWT[5], and PPM[6]. We present a method for lossless data compression where an original message *m* is re-expressed in the most compact way by identifying patterns. Patterns are called *metasymbols* and groups non-necessarily symbols in *m*.

Compressing with metasymbols has two main motivations:

1. To achieve maximum compression for *m*.
2. To identify high order structures in arbitrary sets of data.

The problem of identifying the metasymbols that allow us to re-express *m* in the most compact way is NP-Hard[7]. Considering this we appeal to soft computing techniques and particularly to a Genetic Algorithm called MSIM that identifies the already mentioned metasymbols and achieves maximum compression. In section 2 we describe MSIM, in section 3 we discuss its efficiency to discover metasymbols, in section 4 we present some experimental results. Finally, in section 5 we offer our conclusions. The performance of MSIM as a compressor and as a tool for metasymbolic discovery is supported statistically analyzing a set of 10,500 messages. Experimental results are satisfactory and show that MSIM does not depends of the content of the messages and is able to identify structural patterns. We show that MSIM performance is similar to the theoretical compressor REF which has *a priori* knowledge of the metasymbols. From the analysis we confirm that, if there are metasymbols in

© A. Gelbukh, R. Monroy. (Eds.)
Advances in Artificial Intelligence Theory
Research on Computing Science 16, 2005, pp. 183-192

the message, MSIM will discover them with statistical certainty. Future applications include protein sequences analysis and time series classification based on their plausible compressibility.

We define compression as

$$\text{Compression} = \frac{\text{length of } m}{\text{length of compressed message}} \quad (1)$$

2 Compression with metasymbols

We model a message *m* as an one-dimensional array of symbols; metasymbols are groups of these symbols and have four basic properties:

1. Order $|M_i|$. The number of symbols that the i-th metasymbol includes.
2. Frequency $|f_i|$. The number of repetitions of the i-th metasymbol in *m*.
3. Gaps. The spaces between the symbols of a metasymbol.
4. Positions. The indices of the first symbol of all the instances of a metasymbol.

As an example consider an hypothetical message *m* = "$A_0a_1D_2b_3E_4c_5F_6d_7e_8B_9f_{10}$-$g_{11}C_{12}h_{13}i_{14}j_{15}k_{16}D_{17}l_{18}E_{19}m_{20}F_{21}n_{22}a_{23}o_{24}A_{25}p_{26}A_{27}q_{28}r_{29}s_{30}t_{31}u_{32}v_{33}B_{34}w_{35}B_{36}C_{37}x_{38}C_{39}$ $D_{40}y_{41}E_{42}D_{43}F_{44}E_{45}z_{46}F_{47}$" with $|m| = 48$ and where the positions of the symbols are denoted by subindices. To ease visualization we include a two-dimensional matrix for *m* as we show in figure 1.

	0	1	2	3	4	5	6	7
0	A	a	D	b	E	c	F	d
8	e	B	f	g	C	h	i	j
16	k	D	l	E	m	F	n	a
24	o	A	p	A	q	r	s	t
32	u	v	B	w	B	C	x	C
40	D	y	E	D	F	E	z	F

Fig. 1. A bi-dimensional representation of a message

We can appreciate a first metasymbol ("ABC") at positions 0, 25 and 27 with gaps of size 9 and 3 respectively. In what follows we will use the representation $M_1=A_9B_3C$: 0, 25, 2 which is read: There is an "A" at position 0, a "B" 9 positions away from "A" and a "C" 3 positions away "B"; the metasymbol is repeated at position 25 and 2 away from 25 (25+2=27). A second metasymbol is $M_2=D_2E_2F$: 2, 15, 23, 3. The symbols that do not form repeated patterns are lumped in a pseudo-metasymbol we call the *filler*. One possible efficient encoding depends on the fact that the filler is easily determined as the remaining space once the other metasymbols are known. Hence

$$M_{filler}=abcdefghijklmnaopqrstuvwxyz$$

Metasymbolic search consists not only in the identification of repeated patterns but also in the selection of those which imply the minimum number of bits to re-express *m*. In the last example we assume, for simplicity, that metasymbols M_1, M_2 and M_{filler} comply with this criterion but, as we pointed out above, this problem is NP-hard. Therefore, we have developed an efficient Genetic Algorithm to tackle it.

We describe the proposed encoding scheme:

Let λ be the number of bits of each symbol (in the example λ=8)

Let μ be the number of metasymbols (in the example μ=3)

We use nibble[1] 'a' to indicate the value of λ in binary[2]. In the example $a=1000_2$.

We use nibble 'b' to indicate the number of bits required to code the value of μ in binary. In the example $b=0010_2$ and μ is coded as $\mu=11_2$.

Likewise, the maximum gap for M_1 is 9, and we need $g_1=\log_2\lceil 9+1\rceil=3$ bits to encode this value in binary. The other gaps for M_1 will be encoded with g_1 bits.

We use nibble 'c_1' to indicate the number of bits required by the g_1 value. In the example $c_1=0011_2$.

The maximum position for M_1 is 25 and we need $p_1=\log_2\lceil 25+1\rceil=5$ bits to encode this value in binary. The positions of the other instances of M_1 will be encoded with p_1 bits.

We use nibble 'd_1' to indicate the number of bits required by the p_1 value. In the example $c_1=0101_2$

$|M_1|=3$ and we need 3λ bits to encode its content.

The maximum gap for M_2 is 2, and we need $g_2=\log_2\lceil 2+1\rceil=3$ bits to encode this value in binary. The other gaps for M_2 will be encoded with g_2 bits.

We use nibble 'c_2' to indicate the number of bits required by the g_2 value. In the example $c_2=0011_2$.

The maximum position for M_2 is 23 and we need $p_2=\log_2\lceil 23+1\rceil=5$ to encode this value in binary. The positions of the other instances of M_2 will be encoded with p_2 bits.

We use nibble 'd_1' to indicate the number of bits required by the p_2 value. In the example $c_2=0101_2$.

$|M_2| = 3$ and we need 3λ bits to encode this content.

Also, $|M_{filler}|=30$ and is simply an enumeration of values, Therefore, we need 30λ bits to encode it.

The number of bits for *m* expressed as metasymbols is given by

$$K = a + b + \mu + \sum_{i=1}^{M-1} (|M_i - 1|\, g_i + c_i + d_i + |M_i|\,\lambda + f_i p_i) + \tag{2}$$

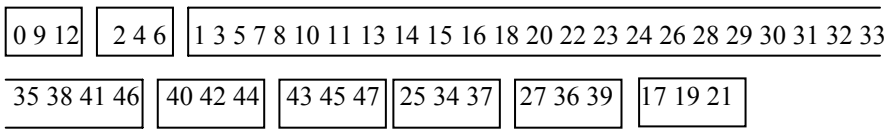
$$+ \lambda\,(|m| - \sum |M_i|\, f_i)$$

[1] By "nibble" we mean 4 consecutive bits

[2] We simply put "binary" when referring to weighted binary

2.1 Metasymbolic search

The GA developed for metasymbolic Search (MSIM) is based on the Vasconcelos's Genetic Algorithm[8] which codes individuals as binary strings and has three operators: selection, mutation and crossover. VGA's mutation assumes a binary coding of the individuals, mutation replaces a bit's value by its complement with probability P_m. VGA Crossover requires two individuals and interchanges their genetic material with probability Pc. Selection is deterministic and elitist.
In the case of MSIM, the individuals are permutations of the array of indices of *m* in the interval [0, |*m*|-1] and each index is coded in binary requiring of $\lceil \log_2 |m| \rceil$ bits. A metasymbol's span is gotten from the indices of an individual by looking for an ascending order and reading from left to right. This representation warrants that each symbol belongs exclusively to one metasymbol as shown in Figure 2.



M_1 $A_0B_9C_{12}$, A_{25} B_{34} C_{37}, A_{27} B_{36} C_{39}
M_2 D_2 E_4 F_6, D_{17} E_{19} F_{21}, D_{40} E_{42} F_{44}, D_{43} E_{45} F_{47}
M_{Filler} $a_1b_3c_5d_7e_8f_{10}g_{11}h_{13}i_{14}j_{15}k_{16}l_{18}m_{20}n_{22}a_{23}o_{24}p_{26}q_{28}r_{29}s_{30}t_{31}u_{32}v_{33}w_{35}x_{38}y_{41}z_{46}$

Fig. 2. A hypothetical individual of MSIM

A= 9 8 4 | **5** 6 7 | 1 3 **2** 10
B = 8 7 1 | **2** 3 10 | 9 **5** 4 6
A'= 9 8 4 | **2** 3 10 | 1 6 **5** 7
B'= 8 10 1 | **5** 6 7 | 9 **2** 4 3

Fig. 3. Crossover operator for MSIM

MSIM does not work on the binary genotype as VGA does; it has special operators that work on the phenotype (indices represented in binary) and maintains the individuals in the feasible region. MSIM has 3 operators: mutation, crossover and catastrophe. Mutation interchanges two indices of an individual and crossover is an operator that acts on two individuals (A, B) and uses PMX[9] to interchange the indices from one individual to the other. It ensures that there are no repeated indices in the first individual and that the same indices are missing in the other. Fig. 3 Illustrates PMX.

2.1.1 Catastrophe. An inconvenience of K is that it considers the bits used by the re-expressed message if the metasymbols are known; to know the metasymbols, on the other hand, we need to minimize K. This leads to a vicious circle whose way out is described next. We propose an on-line method for metasymbol searching. Note that the first 3 terms a, b and μ of ec. 2 may be neglected (they represent a few bits) and that the fourth term (the summation) represents the number of bits used by all metasymbols. Finally, the fifth term represents the number of bits required by the filler. From these considerations we define a discriminant d_i that corresponds to the number of bits required by the i-th metasymbol. Hence the fourth term can be expressed as

$$\sum_{i=1}^{M-1} d_i = \sum_{i=1}^{M-1} (|M_i - 1| g_i + c_i + d_i + |M_i| \lambda + f_i p_i) \quad (3)$$

where

$$d_i = |M_i - 1| g_i + c_i + d_i + |M_i| \lambda + f_i p_i \quad (4)$$

The main advantage of d_i is that it does not depend of the other metasymbols and, consequently, we can search metasymbols iteratively by comparing their d_i values and selecting the one with minimum value. The catastrophe operator is in charge of this task: it encodes the metasymbols as individuals of MSIM and inserts them as potential individuals into the population. The algorithm for the catastrophe algorithm is as follows:

Let i = 1
Mark all the symbols of *m* as free nodes, FreeNodes = |*m*| and LEN = 1
Do
  D= Number of different symbols of *m*
  For j=1 **to** j=D
    s_{j0} = j-th different symbol of *m*
    Mark as used all instances of s_{j0}
    Do
      Identify the symbol s_{jLEN} with the highest frequency (f_{jLEN}) nearest to any s_{j0} (to g_{LEN} positions) and mark all the instances of s_{jLEN} as free nodes. If there is more than one symbol with the same frequency, select s_{jLEN} randomly.
      Free the symbols s_{jk} with $0 \leq k \leq$ LEN. If s_{jLEN} is not at g_{LEN} positions of any instance of sj_0 then make LEN = LEN+1
      Evaluate the compression function and do d_{jLEN} = K
    While $f_{jLEN} > 1$
  next j
  Select the metasymbol $M_j = s_{j0}s_{j1}s_{j2} ...s_{jLEN}$ such that it has minimum d_{jLEN}
  Mark as used nodes the symbols used by the M_i metasymbol in all its instances
  Update the number of free nodes of *m*
  i=i+1
While FreeNodes>0 and $d_{jLEN} > 1.0$
Free nodes are grouped in the filler
The number of metasymbols is M = i+1

The fitness function for the individuals of MSIM is K and the individuals evolve to the best solution with MSIM's three operators.

3 Experiments

A set of experiments was performed to:

1. Measure the efficiency of MSIM as a compressor.
2. Set a confidence level for the expected lower bound of compression achieved with MSIM.
3. Measure the efficiency of MSIM as a tool to discover the metasymbols that allow us to re-express a message in the most compact way.
4. Measure the efficiency of MSIM as tool for discovering metasymbols in spite of the content of the messages.

To this effect we built a set messages that contain metasymbols. The algorithm to generate the set is:

1. Select the length of the message $|m|$ with a value in the interval [a, b], Set i=0
2. **While** there are non-used indices in *m* and for a maximum number of iterations
3. i=i+1
4. Generate a random value for $|M_i|$ in the interval [1, $|m|$]
5. Propose $|M_i|$-1 gaps for the symbols of the metasymbol that is being generated
6. Generate a random value for f_i in the interval [1, $|m|$]
7. For j=1 to f_i
   8. Generate a position p_i for a instance of the current metasymbol
   9. Try to put the instance of the metasymbol at p_i whenever it does not overlap with another metasymbol.
   10. If the number of tries is exceeded do f_i= number of successful outcomes and go to step 2

   next j

Wend

11. If there remain free indices in *m*, group the symbols in the filler
12. i=i+1
13. M expresses the final number of metasymbols in *m*.
14. Store in a database the properties of all the M metasymbols.

Note that this algorithm simply determines the structure of the metasymbols but does not assign them contents; contents are determined later in the process.

We define a compressor called REF that represents our theoretical limit of compression. It takes the information stored in the database and, from ec. 2, determines the number of bits (K) in the corresponding compressed message. Now, we can measure the efficiency of MSIM relative to REF as long as MSIM yields the compression of REF. If MSIM reaches a performance similar to the hypothetical one of REF we rest assured that:

MSIM is able to reach the "theoretical" limit of compression (goal 1).
MSIM is able to identify the metasymbols hidden in *m* (goal 2).

We compared the compression achieved by MSIM against the compression of other 4 compressors: LZ77, LZW, Arithmetic and PPM.

Our third goal reflects our interest on the search for structural patterns which are independent of the content of the message. Hence, we generate messages with the same metasymbolic structure but different contents. First, we store the properties of the sets

of metasymbols (order, positions and gaps) then, for a given set of metasymbols, we explore 5 variations of the same message by filling the metasymbols with contents determined from the experimental PDF from a) Spanish text file, b) English text file, c) Audio compressed file, d) Image compressed file and e) Uniformly distributed data. Because the REF compressor is fully based on the structure of the metasymbols and is impervious to its contents it was used to measure the performance of MSIM as a metasymbolic identification tool.

We want to estimate the worst case compression value achieved for each of the compressors and for each of the 5 distributions. We need to determine the values of the mean (μ) and the standard deviation (σ) of unknown distributions of the compression rates for the messages (call them distributions "A"). We will appeal to the Central Limit Theorem [10] by generating the corresponding distributions where each mean is the average of n objects from distributions A (call them distributions "B"). If X represents an object from one distribution A for a given type of file (Spanish, for example) then

$$\overline{X}_i = \frac{\sum_{i=1}^{n} X_i}{n} \tag{5}$$

represents an object of the corresponding distribution B.

The mean $\mu_{\overline{X}}$ for any of the B distributions is given by:

$$\mu_{\overline{X}} = \frac{\sum_{i=1}^{Z} \overline{X}_i}{Z} \tag{6}$$

Where Z is the number of objects of the B distributions.

The standard deviation $\sigma_{\overline{X}}$ for distribution B is given by:

$$\sigma_{\overline{X}} = \sqrt{\frac{\sum_{i=1}^{Z} (\overline{Xi} - \mu_{\overline{X}})^2}{Z}} \tag{7}$$

We generate as many messages (objects of A distributions) as were necessary until the next two conditions were fulfilled:

1. For each distribution A we require that there are at least 16.6% of Z samples in each sextil in the associated distribution B.
2. The value of the means of both distributions A and B are similar enough

$$\frac{(\mu - \sigma_{\overline{X}})}{\mu_{\overline{X}}} > 0.95$$

In our experiments we needed 2,100 for each of the different distributions: i.e. we analyzed a total of 10,500 messages.

Then we may estimate the values of the mean μ and the standard deviation σ for the A distributions since:

$$\mu = \mu_{\overline{X}} \tag{8}$$

$$\sigma = \sqrt{n}\sigma_{\overline{X}} \tag{9}$$

4 Results

As we see in table 1, MSIM provides consistent values of compression (see μ and σ) for the 5 explored types of file and, as expected, regardless of the contents of the metasymbols.

Table 1. Values for the parameters μ and σ of the distributions A for different compressors

	Message	English	JPG	MP3	Spanish	Uniform	Average
Compressor							
REF	μ	1.92	1.92	1.92	1.92	1.92	1.92
	σ	0.7	0.7	0.7	0.7	0.7	0.7
MSIM	μ	1.58	1.66	1.66	1.58	1.66	1.62
	σ	0.35	0.4	0.4	0.35	0.4	0.35
PPM	μ	1.66	1.21	1.21	1.71	1.21	1.4
	σ	0.9	0.75	0.75	0.9	0.75	0.8
LZW	μ	1.33	1.05	1.05	1.35	1.05	1.16
	σ	0.7	0.75	0.8	0.7	0.75	0.7
LZ77	μ	1	0.94	0.94	0.99	0.94	0.96
	σ	0.35	0.35	0.35	0.35	0.35	0.35
ARIT	μ	1.55	1.25	1.25	1.57	1.25	1.37
	σ	0.6	0.35	0.35	0.6	0.35	0.45

Once we have estimated the values of the parameters for the five distributions A, we estimate the value of the compression for the worst case X_{worst} of the different compressors appealing to the Chebyshev's Theorem

$$P(\mu_X - k\sigma \le X \le \mu_X + k\,\sigma) \ \ge 1 - \frac{1}{k^2} \tag{10}$$

where *k* is the number of standard deviations. Assuming that the distributions A are symmetric we find, from ec. 10 that

$$P(\mu_X - k\sigma \le X) \ge 1 - \frac{1}{2k^2} \tag{11}$$

and then is possible to estimate the expected compression value X_{worst} for all distributions. It immediately follows that $P(\mu - k\sigma \le X) \ge 0.7 \rightarrow k = 1.3$. Table 2 shows the experimental results.

Table 2. Values of the worst case compression rates

Compressor	Message	English	JPG	MP3	Spanish	Uniform	Average
REF	Xworst	1.01	1.01	1.01	1.01	1.01	1.01
MSIM	Xworst	1.12	1.14	1.14	1.12	1.14	1.13
PPM	Xworst	0.49	0.23	0.23	0.54	0.23	0.34
LZW	Xworst	0.42	0.07	0.01	0.44	0.07	0.20
LZ77	Xworst	0.54	0.48	0.48	0.53	0.48	0.50
Arithmetic	Xworst	0.77	0.79	0.79	0.79	0.79	0.78

From table 2 we know that in 70% of the cases MSIM achieves better performance than the other compressors and that its behavior is not a function of the contents of the messages.

Finally, we analyze the likelihood between the distributions B of REF and the distributions B of MSIM. Remember that REF does not depend of the metasymbol contents so its five B distributions are identical. In table 3 we can appreciate how MSIM distributions display similar behavior as REF distributions. A direct conclusion is that MSIM also does not depend of the contents. A goodness test of fit was used to test this hypothesis. We can see, from table 3, that the χ^2 values for MSIM are small enough. In contrast see, for example, the cases of PPM ($\chi^2 > 22$), JPG, MP3 and Uniform. They reveal strong dependency between their performance and the content of the messages. Other compressors (LZW, LZ77, Arithmetic, Huffman, etc.) behave similarly but we are unable to show the corresponding numbers for lack of space.

Table 3. χ^2 values for 35 distributions B

Compressor	Message	Sextil1	Sextil2	Sextil3	Sextil4	Sextil5	Sextil6	χ^2
REF	English	13	10	12	16	23	10	0.00
	JPG	13	10	12	16	23	10	0.00
	MP3	13	10	12	16	23	10	0.00
	Spanish	13	10	12	16	23	10	0.00
	Uniform	13	10	12	16	23	10	0.00
MSIM	English	12	9	16	19	14	14	7.19
	JPG	10	12	11	20	18	13	4.16
	MP3	10	11	12	20	18	13	3.78
	Spanish	12	8	18	18	14	14	8.85
	Uniform	10	14	12	15	19	14	4.65
PPM	English	12	12	16	16	15	13	5.49

	JPG	11	20	18	14	9	12	22.48
	MP3	10	21	18	14	9	12	24.96
	Spanish	13	9	16	19	14	13	6.42
	Uniform	11	20	18	14	9	12	22.48

5 Conclusions

The performance of MSIM has been compared with other compressors by compressing large sets of files that contain the same structure but different contents. Statistical validation allows us to obtain lower bounds on the compression ratio of the acknowledged "best" compressors and MSIM stands out in all cases. The application of MSIM, however, relies on relatively large execution times and, until we are able to identify a faster algorithm, is more suitable for archival purposes, where the emphasis is on optimal compression and multiple accesses rather than on real time applications.

On the other hand, the principle of minimum description length, which underlies MSIM, translates in an unsupervised pattern recognition method where deep similarities between apparently dissimilar sets of data are likely to be found. At present this characteristic is being applied to the discovery of functional relationships in biological genomic sequences.

References

1. Nelson, M., Gailly, J. L., *The Data Compression Book*, Second Edition, M&T Books Redwood City, CA (1995).
2. Huffman, D. A., "A method for the construction of minimum-redundancy codes", *Proc. Inst. Radio Eng.* 40, 9 (.), 1098–1101, Sept 1952.
3. Witten, I. H., R. Neal, and J. G. Cleary. 1987. "Arithmetic coding for data compression", *Communications of the ACM* 30(6): 520-540.
4. Ziv, J., and Lempel, A., "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. on Inf. Theory* IT-23, 3 (May 1977), 337-343.
5. Burrows, M., and Wheeler, D. J., "A block-sorting lossless data compression algorithm", *Digital Syst. Res. Ctr.*, Palo Alto, CA, Tech. Rep. SRC 124, May 1994.
6. Cleary, J. G. and Witten, I. H., "Data compression using adaptive coding and partial string matching", *IEEE Transactions on Communications*, Vol. 32, No. 4, 396-402, April 1984.
7. Kuri, A. and Galaviz, J., "Pattern-based data compression", *Lecture Notes in Artificial Intelligence* LNAI 2972, 2004, pp. 1-10.
8. Kuri, A., "Solution of Simultaneous Non-Linear Equations using Genetic Algorithms", *WSEAS Transactions on Systems*, Issue 1, Vol. 2, 2003, pp. 44-51.
9. Goldberg, D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley Publishing, 1989.
10. Feller, W., An Introduction to Probability Theory and Its Applications, pp. 233-234, John Wiley, 2nd. Edition, 1968.