Lossless Compression of biological sequences with evolutionary metadictionaries

Oscar Herrera[1] and Angel Kuri[2]

[1] Universidad Autónoma Metropolitana Azcapotzalco
Departamento de Sistemas
oha@correo.azc.uam.mx
[2] Instituto Tecnológico Autónomo de México
Departamento de Computación
akuri@itam.mx

Abstract. This work presents a method for lossless data compression based on the identification of patterns in sequences called metasymbols. A metasymbol is a subsequence consisting of regular alphabet symbols and the gap symbol $*$ that matches any alphabet symbol. A first task of the method focuses on discovering metasymbols in single sequences and a second task aims to identify a metadictionary that contains a minimal set of common metasymbols in multiple sequences. Since these tasks represent NP-complete problems we propose, for the first of them, a heuristic method to discover metasymbols which is guided by the minimization of a compression function defined according to the Kolmogorov complexity and the Minimum Description Length principle, and for the second we appeal to a genetic algorithm that aims to identify the metadictionary. As our compression method is based on evolutionary metadictionaries it can be applied to highly entropic sequences and to others where classic compressors do not have a good performance. Biological and multimedia sequences have these properties so we propose a case study to identify a metadictionary in Saccharomyces cerevisiae's proteins sequences. In a future work we expect to apply this method to multimedia data and to deal with larger sequences.

Key words: Lossless Data compression, Information Theory, Genetic Algorithm, Pattern Discovery, Biological Sequences.

1 Introduction

In the last decades several methods have been proposed for lossless data compression (for instance: Huffman [9], Arithmetic [25], Lempel-Ziv (LZ) [26], and Prediction by Partial Matching (PPM) [4]). These methods, which we will refer to as classic, exhibit good performance on those files[3] which include plain text, computer generated graphics, and others that: *i*) involve a reduced number of ASCII symbols, *ii*) their symbols follow a non-uniform distribution, *iii*) their

[3] File, string, indexed symbol array or sequence will be used indistinctively

symbols follow predefined gramatic rules, or *iv*) have local correlation between adjacent symbols. Lossless data compression has been typically interpreted as "text compression" and since classic methods have similar performance they are often compared with two corpus: Calgary [2] and Canterbury [22] which are representative of the kind of files whose features have been listed previously. But a plethora of sequences, such as those that encode DNA (deoxyribonucleic acid), RNA (ribonucleic acid) and proteins, cannot be successfully treated with classic methods despite the fact that they can be expressed as letter sequences [16][17].

In the case of DNA the symbols are A, T, G, and C corresponding to the four bases (nucleotides) adenine, cytosine, guanine, and thymine respectively; in the case of RNA symbols are A, C, G and U for uracil. In the case of proteins the aminoacids[4] can be expresed as upper case letters A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, and Y, where each letter encodes a triplet of nucleotides (codons).

Multimedia sequences are another case where classic methods cannot be successfully applied. Since a certain amount of information loss is tolerable, lossy methods have been proposed. Such is the case of MP3 [8] and JPEG [18].

The Incompressibility Theorem (IT), which is revisited in Section 2, plays an important role in the lossless data compression research area. The IT states the non-existence of any method that compresses all sequences. An important result that follows from the IT is that all lossless compression methods converge to the same performance over a sufficiently large set of files (see [7]) and, given that we typically do not compress a large set of files[5], we should have a variety of compressors that represent alternatives to face the challenge of compressing different kinds of them. But classic methods were not designed under this approach and they focus on the same kind of files. In this work we explore a new approach for lossless data compression where the main idea is that is possible to identify metasymbols in multiple sequences and it allows us to encode them by capturing the redundancy that is not exploited by classic methods [12][13].

In Section 3 we describe the heuristic that allows us to discover metasymbols in single sequences.

In Section 4 we describe how to build a metadictionary from the metasymbols discovered by the heuristic applied to single sequences. With the metadictionary is possible to encode several sequences and achieve compression without loss of data.

In Section 5 we present experiments and their results for a case study of compressing Saccharomyces cerevisiae's protein sequences [23] using the heuristic and applying a highly efficient non-traditional genetic algorithm to build a metadictionary.

In Section 6 we present conclusions and comment on future work.

---

[4] The twenty standard or primary aminoacids.

[5] Given a file that stores a sequence of n symbols, a large subset means $\sim 2^n$ different files.

2 Incompressibility and Compression

An information source generates infinite sequences $s_{i1}s_{i2}s_{i3}...$ with symbols s_i taken from a finite alphabet $\Sigma = \{s_1, s_2, ...s_N\}$ with an associated probability distribution $P_{i=1}^{N} = \{p_i\}$, in the case of finite sequences each symbol probability is estimated from its relative frequency.

Let A_n be any sequence of n bits, $n \geq 1$, and let F_n be the set of all 2^n different A_n sequences. Given an original sequence A_n it is mapped to other A_i that uses i bits. If $i < n$ we say A_n is compressed to A_i, if $i = n$ then we say that there is no compression and if $i > n$ we say A_n was expanded to A_i. The compression rate r (smaller is better) is defined as:

$$r = \frac{i}{n}. \tag{1}$$

A lossless method achieves compression with a mapping funtion f that must be injective in order to warrant the decompression through the inverse function f^{-1}.

Incompressibility Theorem: For all n, there exists an incompressible string of length n. Proof: There are 2^n strings of length n and less than 2^n descriptions that are shorter than n.
For $F_1 = \{0, 1\}$ it is obvious that each file is incompressible because both of them (0 and 1) have the minimum one bit length and f can interchange $0 \rightarrow 1$ and $1 \rightarrow 0$ or keep them unchanged. Now consider the restriction of not to expand any file. The elements of $F_2 = \{00, 01, 10, 11\}$ cannot be compressed because the only way to do this is mapping F_2 to F_1 but they were already used; as the same occurs for $n \geq 2$ we have shown that there is no method that promises to compress at least once and never to expand. The only way to achieve compression is to allow some sequences to be expanded, so elements in F_n can be assigned to elements of F_j for $j = 1, 2, .., n-1$. Now let us expand the families $F_1, F_2, ...F_{n-1}$ in order to compress the sequences of F_n, the average compression rate for $2^n - 2$ compresed sequences of F_n is

$$\overline{r} = \frac{\sum_{i=1}^{n-1} 2^i \frac{i}{n}}{2^n - 2} \tag{2}$$

which can be simplified to

$$\overline{r} = \frac{2^n(n-2) + 2}{(2^n - 2)n} \tag{3}$$

for $n \gg 2$, $\overline{r} \rightarrow 1$ meaning that if we compress $2^n - 2$ different files of F_n the average compression will always tend to one in spite of the f we use. Note that in this approach some files of F_n will be mapped to F_1 and achieve the lowest compression rate but other files will be mapped to families which provide undesirable high compression rates.

The success of a compression method relies in the mapping of one sequence to another with minimum length. But what is the best method? Does this method

exist? What about random sequences? If a given method fails then we should try another (different enough) method aiming to fit the data model but this is tipically unknown.

Because all compressor functions have the same average compresion rate over a large set of files ($\sim 2^n$) and typically we do not compress all of them, the mapping function must be chosen to fit the data model in order to achieve low compression rates, for example $n = 1024$ bits produce $2^{1024} = 16^{256}$ different files (more than a googol) so we will not compress all but rather a subset of them.

Classic methods focus on the same subset of files where high correlation between adjacent symbols and non-uniform distributions are assumed (see [7]), they intend to be as fast as is possible, to read the data few times, and to keep the *simmetry* which implies the use of comparable times in compression and decompression.

However, this situation does not address our current needs. Internet traffic and storage systems involve other kind of files such as audio, image, video, biological sequences, climatic, and astronomic data that cannot be successfully treated with classic methods. Internet servers have more processing capabilities than clients and files compression can be achieved offline. Once a file is compressed it is published on the Web or distributed many times in CD/DVD's. In these environments non-symmetric systems may be better appreciated than classic methods. Such is the case of image fractal compression [1] used in Microsoft's Encarta encyclopedia [27].

We propose two models for lossless data compression based on the identification of repeated patterns called *metasymbols*. A first approach uses a heuristic to discover metasymbols and consequently to encode single sequences considering: *i*) the metasymbols description and *ii*) the sequence description based on these metasymbols. With these two elements it is possible to express the sequence in the most compact way [11, 6].

A second approach goes beyond and requires to identify a minimum set of common metasymbols from all those proposed by the heuristic applied to single sequences. The set of common metasymbols is called *metadictionary* and allows to describe (compress) several sequences by capturing global redundancy. With this goal in mind we perform multiple passes over the data in order to extract and learn as much as is possible from it, and to build an ad-hoc model.

At this point we may think that this model has no a general application. However, consider the case where many users share files, have copies of the same software packages, use predefined formats to encode multimedia (header files) or the case of biological sequences which have common ancestors which exhibit conserved regions (motifs)[24]. Therefore we may use a very large but finite set of files. The premise is that is possible to identify their global metasymbols. If the metadictionary is representative of these kind of sequences we may expect to compress sequences that were not explored previously (generalization).

3 Lossless Compression of single sequences

Pattern based lossless data compression for a single sequence aims to minimize the number of bits required to describe the discovered metasymbols $\Delta = \{M_1,\ M_2,\ \ldots,\ M_J\}$ plus the number of bits of the encoded sequence using these metasymbols. This approach is consistent with the MDL principle [21]. The MDL principle is based on the simple idea that the best way to capture regular features in data is to construct a model in a certain class which permits the shortest description of the data and the model itself.

In order to achieve maximum compression the best metasymbols must be identified. Here "best" means those metasymbols that allow to reexpress a sequence in the most compact way. This takes us to a cyclic dependence: to discover the best metasymbols the sequence must be compressed and to compress the sequence we must know the best metasymbols. In this respect we propose a heuristic method that explores several metasymbols and selects only one of them. The rest of the sequence is updated and the algorithm proceeds to search the next metasymbols. The algorithm stops when there are not enough repeated metasymbols or the metasymbols do not provide a reduction in the number of bits of the encoded sequence. As the i-th selected metasymbols covers (where by "cover" we mean the inclusion of all the original symbols) the sequence partially, in order to reduce the size of the search space, the metasymbols are required to be *exclusive*, that is, they consist of those patterns which do not share symbols. These considerations are crucial for future selections of metasymbols.

Metasymbols have the next properties:

1. Length $|\mathrm{M}_i|$.
2. Frequency $|\mathrm{f}_i|$.
3. Gaps $\{g_i\}$.
4. Offsets $\{o_i\}$.

These properties define the structure of the metasymbols and, of course, require the use of a certain number of bits to be encoded. Our goal is to minimize the number of bits required plus the number of bits of the encoded sequence using these metasymbols.

Unlike other parametric methods for gapped pattern (metasymbols) discovery such as [19] and [20] that apply a penalty criterion for maximum gaps, predefine a minimum frequency for patterns or fix their maximum length, our algorithm performs an unsupervised search of metasymbols in a sequence guided only by goal of estimating its algorithmic complexity[10] by minimizing a compression function C [11, 6] that counts the necessary bits to fully encode the compressed sequence, as follows.

Let $|a_i|$ be the number of bits of each symbol a_i in Σ.

The maximum gap length for a sequence S_i with length $|S_i|$ is given by $g_i = \log_2 \lceil |S_i| \rceil$ (in bits). As the metasymbol has $|M_i| - 1$ gaps then the number of bits used for all gaps is $(|M_i| - 1)g_i$.

The maximum offset for a metasymbol requires $o_i = \log_2 \lceil |S_i| \rceil$ bits. Since the metasymbol requires f_i offsets, the number of bits used for offsets is given by $f_i o_i$.

The number of bits to describe the contents is $|M_i||a_i|$.

Symbols that are not covered by the $i-th$ metasymbol are joined in the sequence S_{i+1} requiring $(|S_i| - f_i|M_i|)|a_i|$ bits.

The number of bits for S_i expressed with the i-th metasymbol is given by the compression function

$$C(S_i) = (|M_i| - 1)g_i + |M_i||a_i| + f_i o_i + (|S_i| - f_i|M_i|)|a_i|. \tag{4}$$

The description of the i-th metasymbol uses

$$|M_i - 1|g_i + |M_i||s_i| + f_i o_i \tag{5}$$

bits and $(|S_i| - f_i|M_i|)|a_i|$ means the number of bits for the residual sequence S_{i+1}.

The number of bits of the contents of the i-th metasymbol in the original sequence is $|M_i|f_i|a_i|$. We define the discriminant d_i as

$$d_i = \frac{(|M_i| - 1)g_i + |M_i||s_i| + f_i o_i}{|M_i|f_i|s_i|}. \tag{6}$$

A minimum d_i is desired to select the best metasymbol from all the possible metasymbols in the search space. The condition expressed in (6) means that we achieve compression if

$$d_i < 1. \tag{7}$$

After a metasymbol was selected, the residual sequence S_{i+1} is compressed again, and the algorithm stops when there are no more metasymbols or (7) is not satisfied. Then the residual symbols are lumped in the so-called filler metasymbol.

Now we can view the sequence compression for S_0 (original sequence at level 0) as a transformation process where the length of the compressed sequence at the i-th level ($i = 1, \ldots, J$) is given by:

$$C(S_i) = \begin{cases} |S_i|, & \text{if } d_i \geq 1 \\ I + C(S_{i+1}|M_i), & \text{if } 0 < d < 1 \end{cases} \tag{8}$$

Algorithm 1 shows how to search metasymbols to compress a sequence.[6] Note that, by definition, metasymbols in a sequence correspond to those groups of symbols which repeat themselves frequently and are as large as is possible. Therefore, they are representative of the redundancy and structure of the sequence.

[6] More detailed information of this algorithm can be reviewed at [6] and [7].

Algorithm 1 Pattern discovery by sequence compression

$T(S_i)$: INPUT(SEQUENCE S_i) OUTPUT(PATTERN M_i)
$FreeSymbols \leftarrow |S_i|, d_i = 0$
Mark all symbols in S_i as free symbols
$LEN = 1$
$\{a_1, a_2, ..., a_k\}$ is the set of different symbols in S
for $j = 1$ to k **do**
  $f_{jLEN} \leftarrow$ Highest frequency of all the symbols located at g_{LEN} positions of $\sigma_{j0} = a_j$
  while $f_{jLEN} > 1$ **do**
    Identify the symbol σ_{jLEN} with highest frequency f_{jLEN} nearest to σ_{j0}, located at g_{LEN} positions. Mark all the instances of σ_{jLEN} as non-free symbols.
    LEN = LEN+1
    Evaluate d_i
    Store the metasymbol$\{\sigma_{j0}\ \sigma_{j1}\sigma_{j2}...\sigma_{LEN}\}$ with minimal d_i
  end while
end for
Select the metasymbol$\{\sigma_0\ \sigma_1\ \sigma_2...\sigma_{LEN}\}$ with minimal d_i
if $d_i < 1.0$ **then**
  Mark symbols of S as *non_free_symbols* for each symbol of the metasymbol instances.
  FreeSymbols ← Free symbols in S
else
  $M_{filler} \leftarrow$ all free symbols in S
end if

4 Data compression of multiple sequences

Definitions.

Covering. Is the number of symbols that the selected metasymbols comprises. Its maximum value is $L = \sum_{i=1}^{N} l_i$.

Noncovered. Is the number of symbols that are not covered by the selected metasymbols. The value is given by $(L - covering)$.

Overlapping. Is the number of times that a symbol is covered more than once by the selected metasymbols.

UsedMetasymbols. Is the number of metasymbols used to encode a sequence.

AllMetasymbols. Is the number of metasymbols obtained from compressing single sequences.

Given N sequences with length l_i symbols each, the problem of discovering common metasymbols consists in identifying those with minimal redundance and maximum covering. The minimal redundance criterion implies that we get minimal overlapping. An overlap is generated when more than a metasymbol uses the same symbol. Notice that metasymbols were selected by compressing individual sequences and there is no overlap for single sequences (intrasequence metasymbols) but when we search for repeated metasymbols in other sequences there

may be overlapped symbols (intersequence metasymbols). In order to achieve maximum covering we must guarantee that the number of non-covered symbols ($filler$) must be minimized. If we have M metasymbols obtained by compressing individual sequences there are 2^M possibilities to choose the metadictionary. To tackle this decision problem we appeal to a non-traditional Vasconcelos GA or VGA [14] where each individual is a string decision of M bits and where the fitness function for the individuals is evaluated maximizing (9)

$$Fitness = Covering - |Noncovered + Overlapping| - \frac{UsedMetasymbols}{AllMetasymbols} \quad (9)$$

Basically what the fitness measure of (9) does is to apply a penalization when a symbol is not covered as well as when a symbol is covered more than once, and to tending to achieve the identification of the minimal metadictionary. The optimal value is given for a full covering, when common metasymbols do not overlap and when the number of common metasymbols is minimal. That means $-\infty < Fitness_{opt} < L$.

Once a set of m common metasymbols are proposed by the VGA to conform the metadictionary we proceed to index each metasymbol with $\lceil log_2 m \rceil$ bits, so that, if a single sequence required J metasymbols and the j_{th} metasymbol is repeated f_j times at positions pointed by the offsets which requires $\lceil log_2 l_i \rceil$, then the length of the compressed sequence is given by $l_i(S|m)$ where

$$l_i(S|m) = J\lceil log_2 m \rceil + \sum_J (f_j \lceil log_2 l_i \rceil) + bitsfiller \quad (10)$$

Here, $bitsfiller$ counts the number of bits of the non-covered symbols in the sequence. Note that, under this approach, the length of the metadictionary is not considered because it is assumed that the decompressor program has a copy of the metadictionary.

5 Experiments

According to our lossless compression method a first experiment consists in the next steps:

1. Select N sequences from the Saccharomyces cerevisiae database [23].
2. Compress each sequence using the heuristic described in Algorithm 1.
3. Store all the metasymbols from individual sequences.
4. Identify possible repeated metasymbols (each one must be included only once).
5. Apply a VGA to identify the metadictionary. Each metasymbol of the metadictionary is encoded with $log_2 m$ bits, where m is the number of common metasymbols or metadictionary length.
6. Optimize the number of metasymbols (J) required to encode individual sequences ($i = 1, 2, \ldots, N$). For this purpose the VGA is executed for each sequence.

```
MSISFPKMQHLIVMTTIGDKKVNNNIILFL
                 G******N
          K****************N
               I***********N
                 MT
```

Fig. 1. Four metasymbols G_6N:(17), $K_{17}N$:(6), $I_{11}N$:(11), and M_0T:(13) taken from the evolutionary metadictionary for YAL037C-A

7. Evaluate the length of each encoded sequence using the J metasymbols from the metadictionary.

In the experiments we considered $N = 88$ protein sequences stored in text files where every aminoacid requires one byte. The average length of the analyzed sequences is 494 with a standard deviation of 353.

The number of different metasymbols obtained by compressing individual sequences is 3319 (209 are repeated from 3528). Those metasymbols model the internal redundancy of single sequences (intrasequence redundancy).

After applying the VGA with 300 generations, a population of $P = 40$ individuals with crossover probability $P_c = 0.9$ and mutation probability $P_m = 0.01$, the evolutionary metadictionary yields $m = 242$ metasymbols that embody the intersequence redundancy.

Each metasymbol of the metadictionary can be referenced with entries of $\lceil log_2 242 \rceil = 8$ bits. Then we proceed to identify the optimal subset of the metadictionary required to encode individual sequences. For this purpose we use the same fitness function of (9) but using the metadictionary as source of metasymbols.

The average number of required metasymbols to encode $N = 88$ sequences is 106, with a standard deviation of 40. As a simple example consider the shortest sequence of "YAL037C-A" that describes the sequence of aminoacids MSISFPKMQHLIVMTTIGDKKVNNNIILFL. It uses four metasymbols G_7N:(17), $K_{18}N$:(6), $I_{12}N$:(11), and M_1T:(13) of the metadictionary, where the subindices denote the gaps and the content of the parenthesis denotes the offsets. From Fig. 1 we see that the two first metasymbols overlap its second aminoacid "N" In this case, we need $4 * 8 = 32$ bits to cover 7 aminoacids from a total of 30. This represents a cover of 23%.

Our experimental results show that for $N = 88$ sequences the average for the cover is 65% with a standard deviation of 8%. This is achieved with an average of 106 metasymbols with a standard deviation of 40. Additionally, notice that the heuristic applied to this sequence identifies the metasymbol $T_4K_3N_1I$ but as it is not common to other sequences the genetic algorithm does not accept it as a member of the metadictionary. (see Fig 2)

For $N = 88$ sequences we require an average of $\overline{J} = 106$ metasymbols. The repetitions average is $\overline{f_j} = 281$ and the average length of the filler is $\overline{bits filler} = 1259$ bits. Consequently we require 298 bytes. This represents an

```
MSISFPKMQHLIVMTTIGDKKVNNNIILFL
          T****K***N*I
           T****K***N*I
```

Fig. 2. The metasymbol $T_4K_3N_1I$ for the sequence YAL037C-A

Table 1. Evolutionary metadictionary applied to non-preprocessed sequences

Name	Length (bytes)	Elapsed time (seconds)	Metasymbols (Max 242)	%Cover
YAL035W	1002	69.26	143	64
YAL051W	1062	74.46	148	65
YAL019W	1131	82.94	146	67
YAL001C	1160	77.53	141	66
YAL002W	1176	87.75	158	69
YAL063C	1322	86.65	139	61
YAL026C	1355	98.69	155	68
YAL017W	1356	104.90	153	65
YAL024C	1435	109.94	152	63
YAL029C	1471	106.59	155	64

average compression ratio of $\frac{298}{494} = 0.6$. A second experiment uses the metadictionary to explore whether these metasymbols are repeated in other sequences that were no previously analyzed. The results are reported in Table 1. The first column describes the name of the sequence, the second column its length, the third the VGA execution time, the fourth column lists the execution time of the genetic algorithm, and the last column describes the cover.

6 Conclusions

The Incompressibility Theorem plays an important role in the lossless compression research area. It proves the non existence of a general optimal compressor and suggests that a set of different compressor methods should be used to tackle the challenge of compressing different kind of files. As an alternative to the classic compression methods (which were designed to compress low entropic files with local correlation) we study another compression method which focuses on high entropic sequences; such as biological and other types characteristic of the multimedia age.

A first approach to compress single sequences intends to discover the sequence's metasymbols using a heuristic that minimizes a compression function C defined according to Kolmogorov's complexity and the MDL principle.

Given a set of sequences, there exists the possibility that their metasymbols are repeated in other sequences. Hence, we state that is possible to optimize a

metadictionary that aims to reach: a) The maximum cover, b) The minimum overlapping between metasymbols and c) To reduce the number of required metasymbols which allows multiple sequences encoding.

Our experiments show that for a set of protein's sequences it is possible to apply a genetic algorithm to build an evolutionary metadictionary from which we can express these sequences without loss of data. They show that the metadictionary is representative enough and allows to encode another non-explored sequences.

As this method is not based on the entropy of the sequences it can be applied complementarily with classic methods. Future work, therefore, will be aimed at determining an evolutionary metadictionary on multimedia files such as MP3 and JPG and to provide a faster alternative method to lossless data compression.

The use of evolutionary metadictionaries has immediate applications on fields such as pattern recognition, biological sequence alignment and sequence classification among others. They will be explored in a future work.

One very interesting issue has to do with the fact that, as of today, there is no recognized method yielding properly compressed re-expression of protein sequences. So much so, that in [17], a hard conclusion is that no such scheme exists. But the foregoing work shows conclusively that it is not the case. For the mere fact of being able to re-express a protein as a set of covering metasymbols, guarantees the possibility of a more economic re-expression of the sequence in accordance to the MDL principle. More importantly, it uncovers the immediate possibility of analyzing genetic trends both operational and phylogenetic. This last issue should not be underestimated. Such possibility has been the explicit aim of various researchers [5][3] and remains to be fully studied. In fact, it is a corollary of the preceding discussion that metasymbolic analysis as described above is tantamount to numerically approaching Kolmogorov's complexity [15].

References

1. **Barnsley, M.**, "Fractals Everywhere", Morgan Kaufmann Pub; 2nd. Sub edition, June 1993.
2. *http://links.uwaterloo.ca/calgary.corpus.html*–July 14, 2008. Compression Algorithms", Proceedings of the Conference on Data Compression, IEEE Computer Society, 1997.
3. **Cavalli-Sforza, L.** and **Edwards, A.**, "Phylogenetic analysis: models and estimation procedures", Evolution 32: 550-570 (also Amer. J. Human Genetics 19: 233-257), 1967.
4. **Cleary, J.** and **Witten, I.**, "Data compression using adaptive coding and partial string matching", IEEE Transactions on Communications, Vol. 32, No. 4, 396-402, April 1984.
5. **Edwards, A.** and **Cavalli-Sforza, L.**, "Reconstruction of evolutionary trees", Phenetic and Phylogenetic Classification, ed. V. H. Heywood and J. McNeill, Systematics Association Volume No. 6. 67-76, London. 1964.
6. **Herrera, O.**, "Lossless data compression using metasymbols", Doctoral thesis, Center for Computing Research, IPN, Mexico, 2005.

7. **Herrera, A.** and **Zaragoza, F.**, "Incompressibility and lossless data compression: An approach by pattern discovery", Computación y Sistemas, Vol. 13, No. 1, 45-60, ISSN 1405-5546, IPN, Mexico, 2009.
8. **Hacker, S.**, "MP3: The definitive guide, 1st ed., Sebastopol Calif., O'Reilly, 2000.
9. **Huffman, D.**, "A method for the construction of minimum-redundancy codes", Proc. Inst. Radio Eng. 40, 9 (.), 1098-1101, Sept 1952.
10. **Hutter, M.**,**Merkle, W.** and **Vitányi, P.**, "Kolmogorov Complexity and Applications", Internationales Begegnungs-und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Dagstuhl Seminar Proceedings, Vol. 06051, Germany, 2006.
11. **Kuri, A.** and **Herrera, O.**, "MSIM: A Pattern Based Lossless Data Compressor", Advances in Artificial Intelligence Applications, Research on Computing Science, ISSN 1665-9899, Vol. 17, 183-192, noviembre 2005, Monterrey, México.
12. **Kuri, A., Ortiz M.**, "A New Approach to Sequence Representation of Proteins in Bioinformatics", Lecture Notes in Artificial Intelligence: No. 3789, Springer-Verlag, 880-889, Editors: Gelbukh, A., Albornoz, A., Terashima-Marín, H., ISBN 3-540-29896-7, ISSN 0302-9743, 2005.
13. **Kuri, A., Ortiz M.**, "A New Approach for Representation in Biological Sequences", WSEAS Transactions in Biology and Biomedicine, WSEAS Press, Issue 1, Vol 3, 31-36, ISSN 1109-9518, 2006.
14. **Kuri, A.**, "A Comprehensive Approach to Genetic Algorithms in Optimization and Learning, Vol. 1: Foundations", Instituto Politécnico Nacional, 1-270, ISBN 970-18-2220-X, 1999.
15. **Kuri, A.**, **Galaviz, J.** and **Herrera, O.**, "Practical estimation of Kolmogorov Complexity using highly efficient compression algorithms", Advances in Artificial Intelligence: Applications, Gelbukh, A., Monroy, R., ISBN 165-9899, 193-201, Mexico, 2005.
16. **Nelson, M.** and **Gailly, J. L.**, "The Data Compression Book", 2nd Edition, MT Books Redwood City, CA, 1995.
17. **Nevill-Manning, C. G.** and **Witten, I.**, "Protein is incompressible", Data Compression Conference (DCC '99), 257, 1999.
18. **Pennebaker, W.B.** and **Mitchell, J.L.**, "JPEG: Still Image Data Compression Standard", ITP Inc., 1993.
19. **Rigoutsos, I.** and **A. Floratos**, "Combinatorial Pattern Discovery in Biological Sequences: the TEIRESIAS Algorithm", Bioinformatics, 14(1), January 1998.
20. **Rigoutsos, I.** and **A. Floratos**, "Motif Discovery Without Alignment Or Enumeration", Proceedings 2nd Annual ACM International Conference on Computational Molecular Biology (RECOMB '98), New York, NY. March 1998.
21. **Rissanen, J.**, "Modelling by shortest data description", Automatica, 14, 465-471, 1978.
22. **Ross, A.** and **Tim, B.**, "A Corpus for the Evaluation of Lossless Compression", Data Compression Conference (DCC'97), 1997.
23. *http://www.yeastgenome.org/*–July 14, 2009.
24. **Waterman, M.**, "General methods of sequence comparison", Bulletin of Mathematical Biology Vol. 46, No. 4, 473-500, 1984.
25. **Witten, I.**, **Neal, R.** and **Cleary J.**, "Arithmetic coding for data compression", Communications of the ACM 30(6): 520-540, 1987.
26. **Ziv, J.**, and **Lempel, A.**, "A Universal Algorithm for Sequential Data Compression", IEEE Trans. on Inf. Theory IT-23, v3, 337-343, 1977.
27. *http://encarta.msn.com/encyclopedia_761568021/fractal.html*–July 14, 2009.