# A First Step Toward Chromosome Analysis by Compression Algorithms *†

E. Rivals
J-P. Delahaye
M. Dauchet
L.I.F.L.
Université de LILLE I
Villeneuve d'Ascq FRANCE 59655
email: rivals@lifl.fr

O. Delgrange
Service Informatique
Université de Mons-Hainaut
Mons BELGIQUE 7000
delgrang@lifl.fr

**Abstract**

*In this paper, we use Kolmogorov complexity and compression algorithms to study DOS-DNA (DOS: defined ordered sequence). This approach gives quantitative and qualitative explanations of the regularities of apparently regular regions. We present the problem of the coding of approximate multiple tandem repeats in order to obtain compression. Then we describe an algorithm that allows to find efficiently approximate multiple tandem repeats. Finally, we briefly describe some of our results.* ***Area:*** *DNA sequence processing.*

## 1 Introduction

Complex genomes contain numerous repeated sequences, the biological significance of which remains obscure. Recently, it has been shown that several human diseases are the result of changes in such sequences, thus it has become urgent to undertake a systematic study of their properties [10]. Certain diseases are the results of amplifications in the sequences which contain bases organised according to symmetrical elements, the DOS-DNA (DOS: defined ordered sequence, see [20]). DOS-DNA is characterized by the presence of palindromes, multiple tandem repeats, mono or oligonucleotide repetitions. Distortions may occur in these regular motifs. Experimental ([3], [7]) and computer ([19], [12], [18], [15], [9], [11], [5]) methods have been used to detect DOS-DNA. These methods use statistical (chi-square like) or combinatorial (matching like) methods.

In this paper, we use Kolmogorov complexity and compression algorithms as tools to study DOS-DNA (also see [6], [16]). This approach gives quantitative (a near-objective measure) and qualitative explanations of the regularities of apparently regular regions. The fact that this method is based on an approximation of the absolute notion of the Kolmogorov complexity, allows us to consider it as an alternative of purely statistical methods.

Kolmogorov complexity of a sequence $s$, $K(s)$ (see [14], [21] and [4]) is by definition the length of the shortest program (for a fixed computer) that prints $s$. This definition is invariant within a constant if one changes the universal computer. This is why $K(s)$ is considered as the absolute measure of the information content of $s$. Unfortunately $K(s)$ is not computable (no algorithm can compute the function $s \to K(s)$). But every compression algorithm actually gives an approximation of $K(s)$: the length $K_C(s)$ of the compressed sequence of $s$ by a compression algorithm $C$ is an upper bound of $K(s)$ and can thus be considered as a tentative measure of the information content of $s$ (if the compression algorithm C is improved, the measure $K_C(s)$ converges to K(s)). With this idea, we can consider that DOS-DNA is DNA that is compressible, and that identification and localisation of DOS-DNA can be achieved by compression algorithms.

Some repeated sequences such as $s = CGACGACGACGACGA$ are clearly compressible (the use of the fact that $s$ is 5 times CGA can be coded in a form that is shorter than $s$). Some other sequences such as $s' = CTACGAGACGATCGA$ (the same as $s$ with one substitution, one deletion and one insertion) are not really interesting since the coding of their descriptions takes more digits than to copy them. Compression algorithms thus give an objective

*Partially supported by the "GDR 1029 Informatique et Genomes" and by the "Groupement de Recherches et d'Etudes sur les Génomes"

†Published in N.G. Bourbakis, editor, *First International IEEE Symposium on Intelligence in Neural and Biological Systems*, pages 233–239, Whashington DC, USA, may 1995. IEEE Computer Society Press.

criterion to distinguish real regularities from pseudo-regularities, and thus to distinguish DOS-DNA from other DNA. The best notion of random sequence is based on the complexity of Kolmogorov (see [14], [4]), the idea is roughly speaking: a random sequence is a sequence such that $K(s)$ is approximately the length of s.

In the first part, we present the problem of the coding of approximate multiple tandem repeats in order to obtain compression. In the second part, we describe an algorithm that allows to find approximate multiple tandem repeats efficiently. In the third part, we briefly describe some of our results. Another paper will give a more detailed biological interpretation of our results [8].

## 2 Coding Scheme for the Sequence Windows

Under the general title of the Coding Scheme, we present here the detail of the structure of the compressed sequence, in order to show that it is possible to restore the original sequence from it (i.e. that we do not forget any information) and also to provide an intuitive meaning of the regularities detected by the algorithm.

### 2.1 General Presentation

We address the problem of finding all the **approximate tandem repeats (ATR)** of a motif $m$ within a text $S$. Here, *tandem repeats* means *side to side repetition of a particular motif m of a definite length.* We design several algorithms for different motif lengths. Here we explain the algorithm for motifs of length 3. The sequence $S$ in which we want to detect Approximate Tandem Repeats (ATR) and the motif $m$ of those ATR are both parameters of the algorithm.

The complete compression algorithm is divided in two main procedures. First, **the detection procedure** investigates the sequence to find the subwords that best match $m^p$ for any $p$ greater than a limit. The algorithm detects all subwords (streches of contiguous letters in $S$) of the sequence that resemble a tandem repeat of $m$, and aligns each of them with the word $m^p$ to identify all differences. It then produces a list of differences' positions and types between the subword and $m^p$. The value of $p$ is guessed (to each tandem repeat-like subword corresponds a different value of $p$).

Then, the coding procedure uses the result of the detection procedure to output a compressed version of the original sequence **without loss of information**. The goal of this part of the algorithm is to produce two files: the first one contains a list of codes to describe all the tandem repeats found in the sequence, the second is the concatenation (in the order of the sequence) of all characters that do not belong to the tandem repeats. This two file organization is adopted for experiment purpose: it is interesting to iterate compressors on the non coded part of the sequence, in order to look for regularities of a second order. Assume that the ancestor sequence of $S$ enclosed some regularities, later some tandem repeats could have been independently inserted in those regularities, making them harder to locate. This is what we call second order regularities.

The definition of similarity is of major importance to understand the results of the algorithm. It is formalized in the next section, but we give an informal presentation of it now. A subword of $S$ would be said similar to $m^p$, if it adopts the tandem repeats structure. In other words, if it has a subword that is similar to $m$ nearly every $|m|$ [1] characters. This is quite different than to say that two words are similar because they have a specific character at position $i$, another specific character at position $i+1$, and so on ... without relation between positions. So, in our similarity definition, the errors are defined, their amount is limited relatively to a motif and not to the whole sequence. Notice that this specificity also simplifies the algorithm.

**When do we consider a subword of a sequence as similar to $m^p$?**

If we call the subword T, we shall view $T$ itself as a succession of smaller subwords $T = t_1 \cdots t_p$. $T$ is said to be similar to $m^p$ iff:

- all $t_i$ have either length $|m|$ or $|m|+1$ or $|m|-1$
- all $t_i$ ought to be equal to $m$ corrected by at most one substitution or [2] one deletion, and eventually followed by an insertion
- it begins and ends with an exact motif i.e. $t_1$ and $t_p$ are exactly equal to $m$ (this is later explained by technical consideration).

For instance, if $m = ACG$, either $ACT$ or $AT$ or $ACGT$ are qualified as similar to $m$ but neither $CGA$ nor $CCC$ would.

This approach is motivated by the belief that specific regularities (like tandem repeats) need specific similarity definition, and ought not be searched with generally used and useful common distances. Both, the algorithm and its results are simplified by this view-point.

[1] $|m|$ denote the length of the text $m$

[2] It is an exclusive or.

Each type of possible motifs has an objective coding cost: a motif with one substitution or deletion costs 7 bits, and a motif with a substitution/deletion plus an insertion costs 14 bits.

## 2.2 Detail of the Coding Scheme

To achieve the traduction of the sequence in its compressed format, the coding procedure takes into account the following informations output by the detection procedure:

- the basic motif $m$
- the number of tandem repeats like subwords, later called **zones**
- their positions in the original sequence
- for each zone
  - the value of $p$, in the word $m^p$ to which they best match
  - the optimal list (i.e. of lowest cost) of errors: position and type

Here comes an example of a window taken out of the 11th chromosome of Saccharomices cerevisiae from position 576,000 to 576,499, to illustrate our coding scheme. In this window, two ATRs of the motif CTG were found. The first one has been matched with the word made of 29 consecutives copies of CTG, and 9 errors were identified. A second zone of CTG tandem repeat-like was found, it is only 9 bases long and is error-free. Their respective beginning positions in the window are 112 and 235.

In the following schema, the beginning and the end of the zones are marked by $<$ and $>$, while errors are denoted by $*$.

```
ATGAGTATTTTTCCAGAAATTACAAAACACAAATAAG

TAGAATGTGTTTAGTGTATTTGTAATTCATAGACTAT

TTATTGATGCTTATCTATTATACCTGGGGTTTCTGCT

TCGTCCTCGTCCTCGTCATCATCGTCGTCGTCGTCGT
<    *     *     *  *
CGTCGTCGTCGTCGTCGTCGTCGTCATCGTCGTCATC
                         *        *
GTCATCATCATCGTTGTTGTTTTCGTTCTTATCTTCT
   *  *  *    >
GTATCTTCTTCTTCGTCGTCGTCTTCTTCCTCATCCT
            <         >
CATCATCGTCCTCTTCCTCGTTATTTTTTGGGTACCC
ACCCAATTTCATAACGATTTTGTTAACAATCTCATTC
CAGTCCATCATACCCCCGAGACCAATATCACCATTTA
TATCCATAACTTTCTCAGACGGCTTGAAAACAGTGAC
CCAGGACATTTCCTCCTTTATCGTTTGTAGTTGCTTC
TTTGTCATCATAGAATTAAAAGTGCTACTCACCATAG
AAGGTGCCAAAAGAATGGG
```

The output code takes this format:

- one file contains the remaining sequence: the concatenation of the subwords that are not zones.

```
ATGAGT ... CTGCTT +
TGTTGT ... CTTCTT +
CTTCTT ... AATGGG
```

- the second file contains a structured list of binary codes that we write hereafter in comprehensive manner [3]. Several lines are used for a presentation purpose, because they correspond to natural "parts" of the complete code. The first part (the first line) tells us about the organization of the complete code: the motif and his length, the number of zones and so the number of parts of the code. The second part takes two lines to describe the first zone stucture, while the last part tells us about the second zone in the same way. A zone description begins with general informations about the zone followed by the list of descriptions of each error.

```
3; CGT; 1; 1;
length of the motif, the motif itself,
number of zones minus one,
choice between 3 or 4 bits code

112; 29; 9;
zone 1: relative window position,
#occurrences of m, #errors

(2,12);(2,12);(2,11);(2,11);
(14,11);(3,11);(2,11);(1,11);
(1,11);
zone 1 errors list:
(relative pos. of 1's motif,
error 1 type); etc.

36; 3; 0;
zone 2: relative window position
(235 -112 - 29 * 3),
#occurrences of m matched,
#errors (without errors list)
```

[3] All codes are given in integer or string format, followed by the ';' which means that we know where they end.

The major issue in the design of the code is to set the length of the codes for all the items. We just give a few interesting examples.

Some items' length are "arbitrarily" defined. For instance, the number of possible zones in a 500 bases long window takes two bits. To be compressed the sequence must contain at least one zone, so only the number of zone minus one is coded. So the number of coded zones is limited to 4.

The item position in window takes 9 bits and can code from 0 to 511, i.e. more than the length of the sequence. The number of repeats takes 8 bits because the $3 * 256 > 500$ and also because it is possible to find very long tandem repeats. On the contrary, the item number of errors has not a limited length but depends on the number of repeats. Actually, a short zone cannot afford too much errors, contrary to a long zone. Under 3 repetitions of $m$ there is no way to gain any bit with or without any error. Until 5 repeats you cannot allow any error. Nearly under 8 repeats, the maximum error number is 3, etc. So, the length of the item's code is always determined dynamically depending on the number of repeats.

*Each error is associated with a motif*, so we give the relative position of the edited motif depending on the preceding error. The default code length is 3 bits, but sometimes like here, the interval between 2 errors exceeds 8 motifs. So we use a 4 bits code and indicate that choice by a 1-bit code in the first part of the code (choice between 3/4 bits code).

In our example, the gain worked out for the first zone is 80, for the second it is 0, so for the whole window it is: $80 + 0 - 11 = 69$, which corresponds to a compression rate of 7%.

# 3 An Heuristic Algorithm

In this section, we present the algorithm we design to locate the zones of ATR. This is a practical algorithm based on some heuristics and it produces satisfactory results from a biological point of view. We begin by a presentation of the specific problem, its restrictions and then we give a succint description of the algorithm.

## 3.1 Problem Definition

Let $S = s_1 s_2 \dots s_n$ be a text and $m = m_1 m_2 \dots m_l$ be a pattern. Both are words over the same alphabet $A$. We denote the length of a word $w$ by $|w|$. Thus $|S| = n$ and $|m| = l$. We denote as $w[i, j]$ the subword of $w$, starting at position $i$ and ending at position $j$. The $i^{th}$ character of $w$ is denoted by $w[i]$.

A *Perfect* Tandem Repeat (PTR) of a word $m$ is a concatenation of $m$, $p$ times ($p > 0$). We denote it by $m^p$. An ATR of a word $m$ is a word "similar" to a PTR of $m$. Thus $u$ is an ATR of $m$ if it exists $p > 0$ so that $u$ is "similar" to $m^p$. We clarify later what we mean by "similar".

Given the text $S$, the problem consists in locating non overlapping zones of ATR in $S$. Thus we must find a sequence of positions $(i_1, j_1, \dots, i_z, j_z)$ with $1 \leq i_1 < j_1 < \dots < i_z < j_z \leq n$ such that the words $S[i_1, j_1]$, $S[i_2, j_2]$, …, $S[i_z, j_z]$ are ATR of some words $w_1, \dots, w_z$.

Most of the other combinatorial algorithms already developped are exhaustive approaches based on dynamic programming. Their time complexity rises up to $O(n^2 \log(n))$ [17], even for limited motif lengths [1], while our algorithm is in linear time complexity. Moreover, some methods only search for patterns repeated twice [17], [13].

## 3.2 Restrictions

We restrict the problem to the following points:

1. The alphabet for the text $S$ and the patterns $w_1, \dots, w_z$ is $A = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ since the text is a subword of a DNA contig.

2. The text is short enough to consider that all the zones we search are ATR of a same pattern $m$. Thus given the text $S$, we must find the pattern $m$ and the sequence of positions $(i_1, j_1, \dots, i_z, j_z)$ such that each zone $S[i_k, i_k]$ is an ATR of $m$.

   In practice, a long DNA sequence is splited into short length windows and the algorithm is applied independently on each window. Therefore, the zones of independent windows may be considered as ATR of different patterns.

3. The length $l$ of the pattern is known at the beginning of the algorithm. In practice, the patterns are very short ($l \leq 5$) and the algorithm is tried for each possible pattern length.

## 3.3 Similarity Between a Word and a Tandem Repeat

Here, we define what we mean exactly by a word $u$ is similar to the PTR $m^p$. The word $u$ is similar to the PTR $m^p$ if it can be written as $u = u_1 u_2 \dots u_p$, each subword $u_k$ being obtained by a limited set of mutations from the word $m$. The length of the pattern $m$ determines the set of authorized mutations. In the following, we present the algorithm for patterns of length 3, it can be easily adapted to another length.

The list of authorized mutations comes from experimental observations. If $m = \alpha\beta\gamma$, with $\alpha, \beta, \gamma \in A$, then the authorized forms of $u_k$ are:

1. The word $m$ itself:
   $u_k = \alpha\beta\gamma \Rightarrow |u_k| = 3$

2. The word $m$ with the deletion of one base:
   $u_k = \alpha\beta$ or $u_k = \alpha\gamma$ or $u_k = \beta\gamma \Rightarrow |u_k| = 2$

3. The word $m$ with the substitution of one base:
   $u_k = \mu\beta\gamma$ or $u_k = \alpha\mu\gamma$ or $u_k = \alpha\beta\mu \Rightarrow |u_k| = 3$

4. One of the three above forms plus the insertion of one base after the word:
   $u_k = \alpha\beta\gamma\rho \Rightarrow |u_k| = 4$
   $u_k = \alpha\beta\rho$ or $u_k = \alpha\gamma\rho$ or $u_k = \beta\gamma\rho \Rightarrow |u_k| = 3$
   $u_k = \mu\beta\gamma\rho$ or $u_k = \alpha\mu\gamma\rho$ or $u_k = \alpha\beta\mu\rho \Rightarrow |u_k| = 4$

Notice that all these forms keep at least 2 bases from the original word $m$.

## 3.4 Framework of the Algorithm

The goal of the algorithm is to detect the ATR zones in order to produce a compressed version of the sequence. Therefore, we apply some heuristics to obtain maximal compression gains.

The complete algorithm is decomposed into four steps:

1. The choice of the pattern $m$.
2. The localisation of all occurences of $m$ in the text.
3. The construction of all PTR zones. This can be done from the informations collected at step 2.
4. The construction of the ATR zones using the localisations of the PTR zones.

We develop each of these steps.

### 3.4.1 Choice of the pattern $m$

For the experience, we only looked for one pattern: this restriction only applied to these experiments. To compress the sequence, we suppose that each ATR zone is equal to $m^p$ modulo a set of mutations. Therefore we only code the set of mutations. In order to maximize the number of perfect matches between the text and the pattern in the ATR, we choose the pattern which has the maximum number of occurrences in the text.

Thus, the algorithm considers all subwords of length 3 of the text and count the number of occurrences of the 64 possible patterns. The pattern with the maximal count is choosen.

### 3.4.2 Localisation of all occurrences of the pattern

Each ATR zone must begin and end with a perfect match of the pattern. This is deduced from technical considerations: if a zone $u = u_1u_2\ldots u_p$ does not begin (resp. end) with a perfect match, i.e. $u_1 \neq m$ ($u_p \neq m$), then the ATR $u' = u_2\ldots u_p$ ($u' = u_1u_2\ldots u_{p-1}$) produces a better compression gain. For this we suppose that the coding of a mutation is more expensive than the coding of the word $u_1$ ($u_p$), for all possible $u_1$ ($u_p$).

To locate all the potential beginnings of ATR zones, we locate all the occurrences of the pattern.

### 3.4.3 Construction of PTR zones

At this stage, the algorithm gathers together all side by side occurrences of the pattern in order to construct the PTR zones. Since we want to maximize the compression gain, we always consider all PTR as included in some ATR. Thus, we must locate them.

### 3.4.4 Construction of the ATR zones using the PTR zones

The fourth step tries to join the PTR together, using several words $u_k \neq m$ which have the form described in subsection 3.3. Thus, all consecutive PTR zones are considered two by two, from left to right in order to construct longer ATR zones.

**Given $S[i_a, j_a]$ and $S[i_b, j_b]$ two PTR zones, how do we join them together ?**

The idea is to lengthen the zone $S[i_a, j_a]$ to the right while it is possible. Thus, at each step we try to increase the right limit $j_a$ of the first zone. When $j_a = i_b - 1$ then the two zones have been joined.

At each step, we denote by $u_k$ the word $S[j_a+1, i_b-1]$ which separates the first zone from the second one and by $l_{sep} = i_b - j_a - 1$ its length. Three cases must be considered:

**Case 1:** $l_{sep} < 2$

The two zones cannot be joined because their separation is not long enough. There is no mutation of $m$ that would produce a word of length 1.

**Case 2:** $2 \leq l_{sep} \leq 4$

If the word $u_k$ approximately matches to $m$, the two zones are joined in one step. If there are several kinds of mutations which can produce $u_k$ from $m$, then the cheapest one (from a compression point of view) is chosen. If $u_k$ cannot be obtained by any kind of mutation then the two zones cannot be joined.

**Case 3:** $l_{sep} > 4$

The two zones cannot be joined in one step. Therefore, among the 13 authorized mutations of $m$, the algorithm selects those which produce a prefix of $u_k$.

If such mutations do not exist, then the two zones cannot be joined.

Otherwise, the algorithm chooses the mutation which produces the greatest gain, i.e. the mutation which generates the longest word $u_k[1, o]$ (with $o < l_{sep}$) for the lowest cost. Thus the first zone is lengthened and becomes the subword $S[i_a, j_a + o]$.

When two zones cannot be joined, then the first one is restored in its original form, i.e. in the longest for which it begins and ends with a perfect match of $m$.

## 4 Experiments and Results

With the exhaustive sequencing of many complete eukaryotic chromosomes, questions about chromosomal organisation and evolution are now possible to address. Long after the pioneer steps of the discovery of isochores by Bernardi [2], compositional biases and more complex sequence inhomogeneities issues have already been approached by statistical analysis [11].

The approach of compression algorithms brings a new class of tools to detect specific types of regularities with an informational significance given by the compression rate. The study of distribution of regularities and their site-specificity are relevant to both organizational chromosomal constraints and large size evolutionnary events.

In these experiments, we use algorithms to identify ATR based on small motifs (mono-, di- and trinucleotides) on complete chromosomes of Saccharomyces cerevisiae. As ATR are local regularities, the experience applies the 3 algorithms to 500-base long contiguous windows [4], *only the best compression result on each window is given.* Here we focus only on the 11th chromosome of SC for place limitation and also because other ones entail quantitatively similar results.

**Calculus of the compression gain**: in an obvious way one base needs 2 bits to be coded, so each 500-base long window has an initial length of 1000 bits. Two files result from a compression:

- one contains the remaining sequence of length $n \leq 500$ that is encoded in $2n$ bits
- the code-file of length $p$ in bits, includes the description of the removed ATR.

[4] also called segments.

So the compression gain in bits is: $1000 - (2n + p)$ and the compression (or reduction) rate is given by: $\frac{100*(1000-(2n+p))}{1000}$. All the results are now expressed by compression gain in bits.

Chromosome 11 is divided into 1332 windows, 113 of which have been compressed by at least one algorithm. The mean gain is about 15 bits. We recapitulate the windows count as well as the mean and max gain values by algorithms.

**Windows count on chromosome 11 of Saccharomyces cerevisiae**:

| Motif Length | # | % | Gain Avg | Gain Max |
|---|---|---|---|---|
| 1 | 79 | 69 | 7 | 46 |
| 2 | 14 | 12 | 19 | 89 |
| 3 | 20 | 17 | 41 | 292 |

Most of the time, a motif of length 1 gives the best compression (79 windows, i.e. 69 % of the compressed segments) with a mean gain of 7 bits ; while algorithms for di- and trinucleotides motifs perform fewer but better compressions: respectively 12 and 17% of the compressed windows with an average gain of 19 and 41 bits and a maximum gain up to 89 and 292 bits. They probably do not reveal the same sort of biological signals.

Hereunder, the figure 1 displays at each beginning position of a segment along the sequence, a vertical line from the x-axis of height: the gain of the corresponding segment.

The ATR zones avoid no cluster of the sequence, and are distributed from $3'$ all the way down to $5'$ end. Occurrence sites of the tandem repeats are found in all sort of regions:

- either in telomeric extremities of the chromosome: e.g. a telomeric tandem repeat of $AC$ of gain 32 (37 repeats with 11 mutations)
- or bording ORF or protein coding regions: an ATR of $AT$ is positionned near dihydroororate dehydrogenase locus with gain 29 (19 repeats with 3 mutations)
- or inside coding regions: in the locus of YKL202w hypothetical protein a 114 repeats of $TTC$ with 51 mutations of gain 292 (which can also be partly seen as a 154 poly-T with 55 mutations of gain 6).

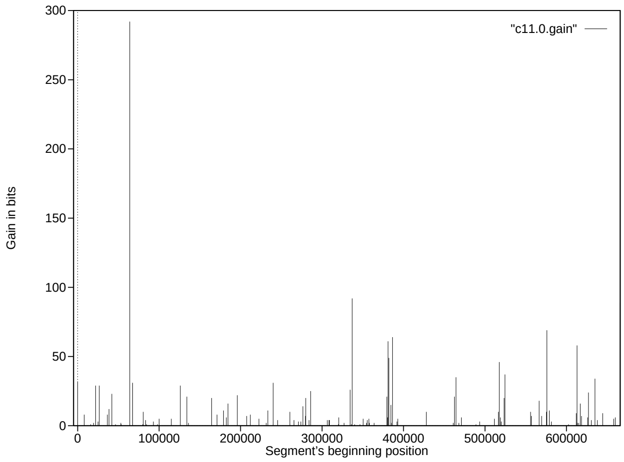


Figure 1: Position and gain of the compressed segments along Chromosome 11

## References


[1] G. Benson and M.S. Waterman. *A Method for Fast Database Search for All k-nucleotide Repeats.* 1994.

[2] G. Bernardi. The Isochore Organization of the Human Genome. *Journal of Molecular Evolution*, Vol. 28, pp. 7-18, 1989.

[3] A. Behn-Krappa, J. Mollenhauer, and W. Doerfler. Triplet Repeat Sequences in Human DNA Can Be Detected by Hybridization to a Synthetic (5'-cgg-3')17 Oligodeoxyribonucleotide. *FEBS Letters*, Vol. 333, pp. 248-250, 1993.

[4] J-P. Delahaye. *Information, complexité et hasard.* Hermès, Paris, 1994.

[5] E. Gilson, W. Saurin, D. Perrin, S. Bachelier, and M. Hofnung. *Nucleic Acids Research*, Vol. 19, pp. 1375-1383, 1991.

[6] Stéphane Grumbach and Fariza Tahi. Compression of DNA Sequences. In *Data Compression Conference*, 1993.

[7] H. Hummerich, S. Baxendale, R. Mott, S.F. Kirby, M.E. MacDonald, J. Gusella, H. Lehrach, and G.P. Bates. Distribution of Trinucleotide Repeat Sequences accross a 2Mbp Region Containing the Huntington's Disease Gene. *Human Molecular Genetics*, Vol. 3, pp. 73-78, 1994.

[8] A. Henaut, M.O. Delorme, E. Ollivier, E. Rivals, O. Delgrange, J-P. Delahaye, and M. Dauchet. Extended DOS-DNA Analysis of Yeast Chromosomes. unpublished.

[9] J. Han, C. Hsu, Z. Zhu, J.W. Longshore, and W.H. Finley. Over-representation of the Disease Associated (CAG) and (CGG) Repeats in the Human Genome. *Nucleic Acids Research*, Vol. 22, pp. 1735-1740, 1994.

[10] A. Henaut and E. Ollivier. DOS-DNA Occurs every 35kb in Chromosomes of Saccharomyces Cerevisiae. unpublished.

[11] S. Karlin, B.E. Blaisdell, R. Sapolsky, L. Cardon, and C. Burge. Assessments of DNA Inhomogeneities in Yeast Chromosome III. *Nucleic Acids Research*, Vol. 21, pp. 703-711, 1993.

[12] M-Y. Leung, B.E. Blaisdell, C. Burge, and S. Karlin. An Efficient Algorithm for Identifying Matches with Errors in Multiple Long Molecular Sequences. *Journal of Molecular Biology*, Vol. 221, 1367-1378, 1991.

[13] G.M. Landau and J.P. Schmidt. *An Algorithm for Approximate Tandem Repeats.* Proc. 4th Symp. Combinatorial Pattern Matching. Springer-Verlag, LNCS, Vol. 648, pp. 120-133, 1993.

[14] M. Li and P.M.B. Vitanyi. *Introduction to Kolmogorov Complexity.* Springer-Verlag, 1993.

[15] A. Milosavljeviċ and J. Jurka. Discovering Simple Dna Sequences by the Algorithmic Significance Method. *CABIOS*, Vol. 9(4), pp. 407-411, 1993.

[16] E. Rivals, O. Delgrange, M. Dauchet, and J-P. Delahaye. Compression and Sequence Comparison. DIMACS Workshop on Sequence Comparison, nov. 1994.

[17] J.P. Schmidt. *All Shortest Paths in Weighted Grid Graphs and its Application to Finding All Approximate Repeats in Strings.* 1994

[18] P. Salomon, J.C. Wootton, A.K. Konopka, and L. Hansen. On the Robustness of Maximum Entropy Relationships for Complexity Distributions of Nucleotide Sequences. *Computer Chem.*, Vol. 17, pp.135-148, 1993.

[19] D. Tautz, M. Trick, and G.A. Dover. Cryptic Simplicity in DNA is a Major Source of Genetic Variation. *Nature*, Vol. 332 pp. 652-656, 1986.

[20] R.D. Wells and R.R. Sinden. *Genome Rearrangement and Stability*, chapter Defined Ordered Sequence DNA, DNA Structure, and DNA-directed Mutation. Genome Analysis. Cold Spring Harbor Laboratory Press, 1993.

[21] P.H. Yockey. *Information Theory and Molecular Biology*. Cambridge Univ. Press, 1992.