ANALYSIS AND COMPARISON OF SEQUENCES : AN INFORMATIONAL APPROACH

Olivier.Delgrange@umh.ac.be

UNIVERSITE DE MONS-HAINAUT
Service Informatique

Laboratoire d'Informatique Fondamentale de Lille

Introduction

Usually :

Use of *compression programs* to reduce the size of data files: saving of storage space or transmission time.
Efficiency: compression performance, speed and reliability.

⇒ **Data Compression is used as a black box**

Genetic Sequence Analysis :

We must know how a compression program has managed to reduce the file size

Classical Genetic Sequence Analysis Tools :

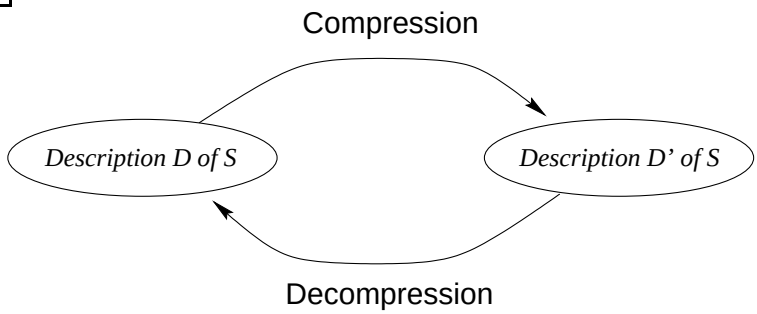
Difficult to know if the regularities detected are relevant or not

Example :

ACTA **GACATTT** GGGA **GACATTT** CTGATCGAGGCA

Without any biological knowledge, *data compression* tries to give an answer

Compression Cycle



The original data are not lost! : the original file can be recovered by the *decompression program*.

Input of the compression program: the description D of an object S
Output of the compression program: the description D' of the same object S

The compression program tries to compute a **shorter description :** $|D'| < |D|$

The compression program deletes some regularities and replace them by a code: a *synthetic representation of the regularity*: a kind of "explanation".

Example :

direct repeat is a kind of regularity in DNA sequences.
A repeated segment in a DNA sequence is a regularity.

ACTA **GACATTT** GGGA **GACATTT** CTGATCGAGGCA

ACTA**GACATTT**GGGA[5,7]CTGATCGAGGCA

It "explains" that the origin of the segment is the duplication of another segment.

The new description seems to be a right explanation if it is a shorter one

⇒ Given a compression algorithm and **knowing the kind of regularities it encodes**, one may study the presence of those regularities in the genetic sequence.

One can locate regular segments and understand why they are regular by looking at their code.

Moreover, the *compression rate* gives us a quantitative measure of the regularities

Data Compression

Two different classes of compression methods :

1. lossy compression methods (image compression, sound compression,...)
2. **lossless compression methods**: used when loosing a character in the data may change its meaning.

In the framework of genetic sequences analysis (DNA, RNA, proteins), **we only consider lossless text compression methods**: the compressed version of the sequence must enclose all the information contained in the original sequence.

The sequences are texts over a given alphabet :

DNA : {A, C, G, T}

RNA : {A, C, G, U}

Proteins : {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}

How can be measured the size reduction ?

compression gain = $original\ size - compressed\ size$

To compute the compression gain, the two sequences must be written over the same alphabet $\{0, 1\}$

For DNA or RNA : each base can be written over 2 bits : A=00, C=01, G=10 and T(or U)=11

Thus the length in bits for a nucleotidic sequence is twice its length in bases.

For proteic sequences the length in bits is $\log_2 20 \times n$ where n is the length in amino acids.

compression rate = $\frac{original\ size - compressed\ size}{original\ size}$

What is compressible ?

In a compressible sequence, some parts of the sequence must be regular enough to be described shortly.

Examples of compressible and incompressible sequences

The output code may be the answer of an observer which is prompted for a description of the sequences.

s_0 = ACGCGCACCATCGGCCAGCTCCGCCTTCCCGGGCACGCCT

Type of regularity: statistical regularity: statistical bias in nucleotidic usage.

Code:
01 10 00 110 0 10 0 10 01100011011101010001101001110010001111110001010100110010001111

Explanations : 01,10 and 00 give the most common base used : C, G and A.
According to this a new code is given to each nucleotide: A=110,C=0,G=10 and T=111

s_0 reveals a statistical regularity: 50% C, 25% G, 12.5% A and 12.5% T.

The new coding allocates short codes to frequent characters and longer codes to rare characters.

This is a Huffman coding, it ensures that an unique decompression is possible.

$original\ size$: 80 bits, $compressed\ size$: 76 bits $\Rightarrow$ $compression\ rate = 5\%$

Statistical compression views the sequence as it has been produced by a random process that chooses each character in respect to a probability law.

The *model* of the compression method looks at the character frequencies.

The more realistic the model is, the more compressed the sequence will be.

Statistical regularities are at the origin of the Shannon's *Theory of Information (1949)*.

They are well studied in genetic sequence (example : nucleotidic composition of isochores).

But genetic sequences are not produced from such a random process.
The statistical model does not care about the order of the bases in the sequence.

$\Longrightarrow$ ***algebraic regularities***

They are defined by properties on the order of the bases in the sequence (repeats, genetic palindromes, etc). They are local regularities whereas statistical regularities are global ones.

The models used are *algorithmic* instead of random processes.

These are the central concepts of the *Algorithmic Theory of Information*, also called the *Theory of Kolmogorov Complexity (1969)*.

Algebraic compression algorithms are able to point out regular segments of the sequence and to suggest an algorithmic process to produce these segments (*explanations*).

They do no propose any biochemical explanation of the presence of the segment.

But if biologically relevant models are used in the compression algorithms, they results would have biological implications.

$s_1 =$ `CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC`

Type of regularity: runs of nucleotides.

Code: `R,40,C` (in binary format, we have to code `R,40` and `C`)

$original\ size$: 80 bits, $compressed\ size$: 17 bits $\Rightarrow compression\ rate = 78\%$

s_1 is extremely compressible.

$s_2 =$ `ACGACGACGACGACGACGACGACGACGACGACGACG`

Type of regularity: runs of polynucleotides of any size

Code : `R,13,ACG` (in binary format, we have to code `R,13`, the length `3` of the repeated polynucleotide and `ACG`)

$original\ size$: 80 bits, $compressed\ size$: 21 bits $\Rightarrow compression\ rate = 74\%$

s_2 is extremely compressible.

$s_3 =$ CGCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCGCCCC

Type of regularity: runs of nucleotides with point mutations

Code : R,40,C,(2,G),(36,G)

s_3 is compressible but less than s_1

It requires a more sophisticated compression algorithm

The more the segment contains mutations, the less it would be compressed.

Until which number of mutations is the sequence compressible ?

Answer : It depends of the size of the recoding of the sequence:

- size of the sequence
- the number of mutations
- the positions of each mutation

The fact that the algorithm computes the compression rate helps to distinguish between runs of nucleotides altered by mutations and "random" segments

$s_4 =$ AGCGGCTATTGCTCTACGTGTGACCGTAGTTCCACGACAC

Type of regularity ?

Code : P,AGCGGCTATTGCTCTACGTGTGACCGTAGTTCCACGACAC

The algorithm cannot find any shorter description of s_4.
The only code it can output contains s_4 itself.

For this algorithm s_4 is incompressible: it is "random".

$s_5 =$ TGCTCCTGCTGCTCTGCTGCTGCTGCTAGCTGC $s_6 =$ TCTGCTACTGCTGGTCTGTGAGCTGCGTGCTGGTGC

Type of regularity: runs of polynucleotides of any length with mutations

Code:

R,12,TGC,S(5,C),D(9),I(17,A)

R,12,TGC,D(2),S(6,A),S(7,G),D(3),
I(4,A),I(5,G),S(6,G)

Are s_5 and (or) s_6 compressible ?

$compression\ rate = \frac{72-42}{72} = 42\%$

$compression\ rate = \frac{72-77}{72} = -7\%$

$\Rightarrow s_5$ is more regular than s_6

$s_7 =$ AGTACAT**ATAGTCGCATAC**GCTGCA**ATAGTCGCATAC**ATG

Type of regularity: exact direct repeats

Code: `1,(26,8,12),AGTACATATAGTCGCATACGCTGCAATG`

s_7 is compressible because the repeated subword is long enough: $compression\ rate = \frac{80-76}{80} = 5\%$

All the preceding segments could have be inferred by a human observer using the Occam's Razor principle: "**the sortest descriptions are the more probable**".

Every description is sufficient to reconstruct the original sequence.

In fact, the encoded version is a "program" to compute the original sequence.

Compression relatively to a model = understanding

A compression algorithm is fixed and **completely independent of the input sequence**. It tries to compute a compressed version of the sequence.

Sometimes it reachs this goal, sometimes it fails!

A compression algorithm leans upon a model of a regular sequence and includes a synthetic code to describe a regular segment that fits the model.

Two steps :

1. it search the regular segments that fit the model
2. it computes the encoding of the sequence using the regularities detected. This code gives the specific data that allows to recover the sequence relatively to the model.

If the sequence fi ts the model, the resulting code is shorter than the original sequence and compression is achieved.

If the sequence does not fi t the model, the compression would output a longer code than the sequence.

For biological experiments, the model is originated by a biochemical hypothesis or property.
The correct conception of the model from the hypothesis requires an in-depth cooperation between biologists and computer scientists;
It often asks to refine the biochemical hypothesis.

Compression implies understanding because:

- The compression algorithm analyses how the sequence fits the model
- the compression rate evaluates, in term of information contents, how much the knowledge of the model allows to reduce the sequence description: it gives a quantitative and comparable measure of the validity of the hypothesis on a sequence.

Thus compression is a tool to test properties on genetic sequences

Notice:

In practice, the compressed sequence in binary format is not required. We just have to compute the **compression rate** and to give the localisations and the types of all regularities detected.

Applications: examples of compression algorithms

Catalog of compression algorithms, dedicated to DNA sequence analysis, developped in the BioInformatics team (LIFL).

Widely used text compression algorithms (LZ77, LZ78, ...), implemented in `pkzip,gzip,compress,...`, are not able to compress genetic sequences !

Indeed, the repeats in natural texts are quite small and the occurrences occur not far from each other.

$\Longrightarrow$ Need of dedicated compression methods

1.*Cfact* (É. Rivals)

Cfact considers repeats that might be very long and with occurrences far away from each other.

It can objectively and automatically detect significant direct repeats and measure the “repetitiveness” of a sequence.

Model of DNA sequence evolution: the sequence is constructed by segment duplications.

Remark:
Sequences only contain mutated repeats, but mutated repeats must include exact repeats.
Example:
The gene for human growth hormone (66495 bp) contains one huge approximate repeat (approx. 20000 bp). Inside this approximate repeat, we find long exact repeats.

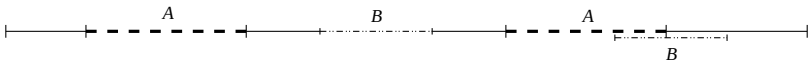
Numerous pattern matching algorithms exists to find repeats in strings but *Cfact* select some of them according to their compressibility.

Cfact allows:

- to locate repeated segments in a sequence
- to measure their quantitative importance by the compression rate
- to assert a significative presence of repeats if the sequence has been compressed.

Two phases:

1. search of all repeats in the sequence:
   Judicious use of the *Suffix Tree* of the sequence (compact index of all subwords of the sequence)
2. encoding of the detected repeats:
   A choice must be made about the repeats:
   - it is not possible to consider all repeats because of overlapping occurrences.



   - too short repeats do not lead to effective local compression: the corresponding code is longer than the occurrence itself.

Heuristic:

All repeats are considered in decreasing length order and a repeat is selected only if it can be encoded shortly.

$\Longrightarrow$ guarantee of compression:

If the sequence contains at least one repeat that can be encoded shortly,
it is effectively compressed by *Cfact*

$\Longrightarrow$ using the compression rate values, one may classify any set of sequences upon the repetitiveness criterion.

For each selected repeat, the following pointer to the 1st occurrence take place of the 2nd (3rd, 4th,...) one: $(pos\ 2nd, pos\ 1st, length)$.

Results:

Cfact has been applied to the whole genome of *Haemophilius Influenzae* and 210 exact repeats have been found (length from 19 bp to 5563 bp).

2.*Cpal* (É. Rivals)

Cpal does, for genetic palindromes, the same work as *Cfact* did for direct repeats.

What is a genetic palindrome? AC**TAAGCT**ACCTGT**AGCTTA**CCTCG

A couple of subwords (f, g), g (resp. f) being the reverse and complementary copy of f (resp. g).

In a genetic palindrome, the 2nd subword (g) can be replaced by the triplet of integers identifying f.

$\Longrightarrow$ Same coding scheme as *Cfact*

Remarks:

- only two subwords are involved in a genetic palindrome whereas more subwords (occurrences) could be concerned by a direct repeat
- *Cfactpal* is a combination of *Cfact* and *Cpal*. At each step, it chooses either to code a direct repeat either a genetic palindrome.

3.*ATR* (É. Rivals)

Compression of sequences that contain *Approximate Tandem Repeats (ATRs)* of any short motif (mono-, di- or trinucleodides).

Exact Tandem Repeat (ETR): ACGTG ATC ATC ATC ATC ATC ATC ATC AT GAC
Approximate Tandem Repeat (ATR): GGA ATg ATC AC AgC AgTC ggC AGACA

Matter of interest because the amplification of ATRs was discovered to cause genetic diseases.

Model of DNA sequence evolution: generation of tandem repeats by amplification of a single motif.
Later point mutations (substitutions, deletions, insertions).

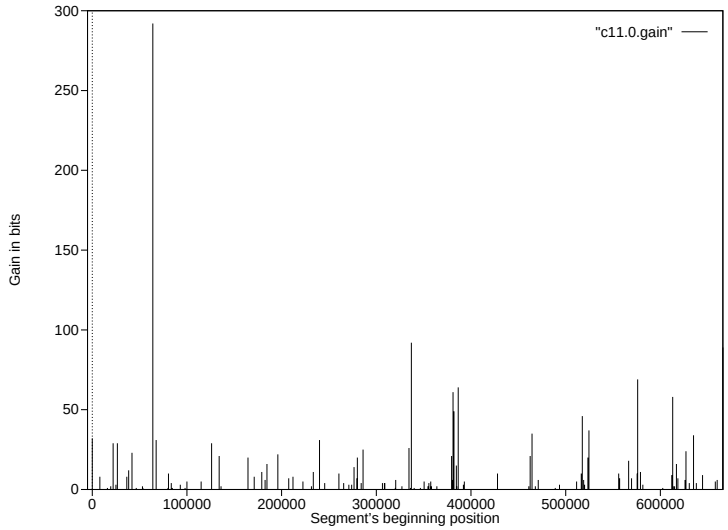
Given a window of a sequence and a motif length (1,2 or 3), *ATR* does:

- choice of a motif u: the most frequent in the window
- location and encoding of all ATRs of u in the window.
  A list of possible degenerated forms of u is known by the program.
  Encoding (in binary format): $(i, n, (mutation\_list...))$ where:
  - i is the beginning position of the ATR in the window
  - the motif u has been replicated n times
  - the corresponding segment has been altered by the list $mutation\_list$ of point mutations.

If the sequence is effectively compressed by *ATR*, then this generation process is a better explanation than a random process.

Experiments: study of the repartition of ATR zones along already (4) known yeast chromosomes.

Each chromosome was cut up into 500bp long adjacent windows. The method has been applied to all windows for each motif length.



Results:

- ATRs appear in the same proportion over each studied chromosome
- 9% of the windows contain significant ATRs
- Discovered ATR zones invalidate hypothetical mechanisms about their generation (blocking of DNA polymerase movement during the replication by a specific space conformation of the molecule): this space conformation is not possible for the detected ATRs.

4. TURBOOPTLIFT (O. Delgrange)

This is not a new compression method.

Goal: the location of repetitive regions in sequences by optimizing an existing compression method.

We want to study a local property P in a sequence.

Where does the property P occur "by chance"
and where does it occur as the result of a specific process ?

Principle:

1. The **whole sequence** is compressed by exploiting the property P

2. Then the sequence is decomposed into alternating regions that are :

- either **repetitive regions** for the property P
- or **non repetitive regions**: it is preferable to keep these regions as they are instead of compressing them.

Given:

- a sequence S
- a lossless compression method which exploits the property P

The location of repetitive regions (for P) consists in:

> Computing an optimal decomposition of S
> into regions to compress and regions to keep as they are
> in order to maximize the global compression gain

However, additional informations must be coded with each copy to allow the decipering of the new compressed sequence.

In compressed regions, the property is said to be relevant.
In copy regions, it is said not to be so.

The algorithm TURBOOPTLIFT computes an optimal decomposition in time $O(n \log n)$

> Application: Location of Approximate Tandem Repeats
> of a given pattern M in a DNA sequence S

Implied in some genetic diseases

Exact Tandem Repeat (ETR): ACGTG ATC ATC ATC ATC ATC ATC ATC AT GAC
Approximate Tandem Repeat (ATR): GGA ATg ATC AC AgC AgTC ggC AGACA

Problem: Even if the basic pattern of the ATR is given, everything is ATR !

- How can we locate ATR zones ?
- Objective criterion to accept/reject and delimit an ATR zone ?

We present a new location method that requires the knowledge of the pattern but do not need any threshold or any restrictive definition of an ATR.

Example : $M =$ TTC and S is the 1000-bases long fragment of yeast chromosome 11 starting at position 63700:

```
TAATAGCATTAAGATTAAATGTATGTTTGATTAATTGCAATATCATGATGTTATAGAGTAACGTGGTATAATTACCACGA
TTCCGTCGAATTGTTTGCCACTTATCACTGGCGCCAAAATGCAGCAGTATGTAAAAAATGACGGTAATAAATATATAATA
CAAATGCATAGAGGAAAGGCTATAGAAATGAAGAGATTCATGAATTTTCAGTCAGGTTCTCCGACTTTTAATTGCTGTGC
CCCTCCTCCTGCTTCTTTTTTTCCTCTTCTTCGTTCTTCTTCTTTTCTTCCTCTTCCTGTTTCTTCTTTTCTTCATCTTC
ATTCTTCTTGTTTTTCTCTTCCTGCTTCTTTTTTTCCTCTTCTTCGTTCTTCTTCTTTTCTTCATCTTCATTCTTCTTAT
TTTCTTCATCTTCATTCTTCATCTTTTCTCCTTCTTCCTGCTTCTTCTTCTCTTCTTCTTCCTTCTTCTTTTTCTCCTCT
TCCTCCTGCTTCTTCTTTTCTTCTTCTTCCTTCTTTTTCTTCTCTTCTTCTTCCTTCTTTTTCTTCTCTTCCTCTTCCTT
CTTTTTCTTTTCTTCTTCTTCCTTCTTCTTCTTTTCCTCTTCTTCCTTCTTCTTCTTCTCCTCTTCCTCCTTCTTCTTCT
TCTCCCTCCGCTTTCTTTCTTCAAGTAACTTGGCATAAGCATAGTCCTCGTATACAATAATAGGGTCCTTATCAAACATT
TGAATACCCTTTCTGTTAACGTCGGATAAAGTTGGCGCAAAAAGGTCTAAACCATATGATTTTAAATCCAAAAAGGAGTT
GCTACTAAAGTTTAAATTTGATTTAAAATCCACAGGATTTACTTCAACAGTATTAGCAGGAGTATAATCTTCTGTGTTAT
CATCTAGAGTTGTTCCAGTTTCATCAAAAGTAATGTTGAAAAGTGAAAAATTCCTGTATTTCAAACTTTCGGACAGATCA
ATAATTTTTCCGACATATTCTTCATGAACAATCTTGGTAA
```


The Compression Method for ATRs

We consider that the whole sequence S is an ATR of M and we try to compress S using this property.

For this, we use the *Wraparound Dynamic Programming (WDP)* technique
(Fischetti, Landau, Schmidt and Sellers - 1992):
The WDP computes the optimal alignment of the sequence S with the infinite ETR of M, i.e. with M^∞.

The alignment is computed in $O(np)$ time where $|S| = n$ and $|M| = p$.

It computes the minimal list of point mutations which transforms an ETR of M into S.

```
      T--T--C-TTc---TTct-TcT-TcTTc--TTc-TT-C--T-TC-T--TcTTcT----T--CtTc-T-Tc-TT-CttC--TTCt-TC---TTcTT--CttCTTcTt-CTt
    1 TaaTagCaTTaagaTTaaaTgTaTgTTtgaTTaaTTgCaaTaTCaTgaTgTTaTagagTaaCgTggTaTaaTTaCcaCgaTTCcgTCgaaTTgTTtgCcaCTTaTcaCTg

      -CttCt---T-CttC--T-TcT------T--C--T--Tc--T-TcT--T-Ct--T-CtT----------CT-Tct---Tc------TTC-T---TcTTCt-TC---TTCT
  111 gCgcCaaaaTgCagCagTaTgTaaaaaaTgaCggTaaTaaaTaTaTaaTaCaaaTgCaTagaggaaaggCTaTagaaaTgaagagaTTCaTgaaTtTTCagTCaggTTCT

      tCttCTTcTTc-TT-CT-T-CttCTtCTtCTtCTTCTTcTTcTTCtTCTTCTTC-TTCTTCTTCTTcTTCTTCtTCTTCtT-CTTCTTCTTcTTCTTCtTCTTC-TTCTT
  221 cCgaCTT-TTaaTTgCTgTgCccCTcCTcCTgCTTCTT-TTtTTCcTCTTCTTCgTTCTTCTTCTT-TTCTTCcTCTTCcTgCTTCTTCTT-TTCTTCaTCTTCaTTCTT
                                      ------------------------------------------------------------------------ --------

      CTTcTTcTTCtTCTTCtT-CTTCTTcTTcTTCtTCTTCTTC-TTCTTCTTCTTcTTCTTCtTCTTC-TTCTTCTTcTTcTTCTTCtTCTTC-TTCTTCtTCTTcTTCTTtC
  327 CTTgTTtTTC-TCTTCcTgCTTCTT-TTtTTCcTCTTCTTCgTTCTTCTTCTT-TTCTTCaTCTTCaTTCTTCTTaTT-TTCTTCaTCTTCaTTCTTCaTCTT-TTCTcC
      ------------------------------------------------------------------------------------------------------ --------

      TTCTTCtT-CTTCTTCTTCtTCTTCTTCTTC-TTCTTCTTcTTCT--TCTTCtTCtT-CTTCTTCTTcTTCTTCTTCTTC-TTCTTcTTCTTCtTCTTCTTCTTC-TTCT
  432 TTCTTCcTgCTTCTTCTTC-TCTTCTTCTTCcTTCTTCTTtTTCTccTCTTCcTCcTgCTTCTTCTT-TTCTTCTTCTTCcTTCTTtTTCTTC-TCTTCTTCTTCcTTCT
      ------------------------------------------------------------------------------------------------------ --------

      TcTTCTTCtTCTTCtTCTTC-TTCTTcTTCTTcTTCTTCTTCTTC-TTCTTCTTCTTcTTCtTCTTCTTC-TTCTTCTTCTTCT--TCTTCtTC-TTCTTCTTCTTCTtC
  539 TtTTCTTC-TCTTCcTCTTCcTTCTTtTTCTT-TTCTTCTTCTTCcTTCTTCTTCTT-TTCcTCTTCTTCcTTCTTCTTCTTCTccTCTTCcTCcTTCTTCTTCTTCTcC
      ------------------------------------------------------------------------------------------------------ ------

      tTCttCTT-CTT-CTTCt--T--CTT--CtT---C-T--TCtTC-T-T-Ct-Tc-T----TC-TTcTt---CtTcT---T-C--TT-CT-T---CtTCt-Tc---TT--
  646 cTCcgCTTtCTTtCTTCaagTaaCTTggCaTaagCaTagTCcTCgTaTaCaaTaaTagggTCcTTaTcaaaCaTtTgaaTaCccTTtCTgTtaaCgTCggaTaaagTTgg

      Ct--------TCTt--Ct-TcT---TcT---TC----------TT-CTtCT----TcT---TcT--TcT----TCttC----TT--CTTCttCt-TcTT-Ct----TcT
  756 CgcaaaaaggTCTaaaCcaTaTgaaTtTaaaTCcaaaaaggagTTgCTaCTaaagTtTaaaTtTgaTtTaaaaTCcaCaggaTTtaCTTCaaCagTaTTagCaggagTaT

      --TCTTCT-TcTTcT--TCTtc--TTcTTC---TT-CtTC-----T--TcTTct----Tc-----TTC-T-TcTT-Ctt-CTT-Ctt-Ct--TCt-Tc-TTcTTCtt-Ct
  866 aaTCTTCTgTgTTaTcaTCTagagTTgTTCcagTTtCaTCaaaagTaaTgTTgaaaagTgaaaaaTTCcTgTaTTtCaaaCTTtCggaCagaTCaaTaaTTTcTTCcgaCa

      TcTTCTTCtT---Ct-TCTTctTc-
  976 TaTTCTTCaTgaaCaaTCTTggTaa
```

To compress the sequence, we just have to make a cheap coding of the list of point mutations.

We use a coding scheme that favours consecutive matches in the alignment.

We consider that the alignment is a sequel of groups of consecutive matches, each group separated from the following by one point mutation.
Thus we code the sequel:

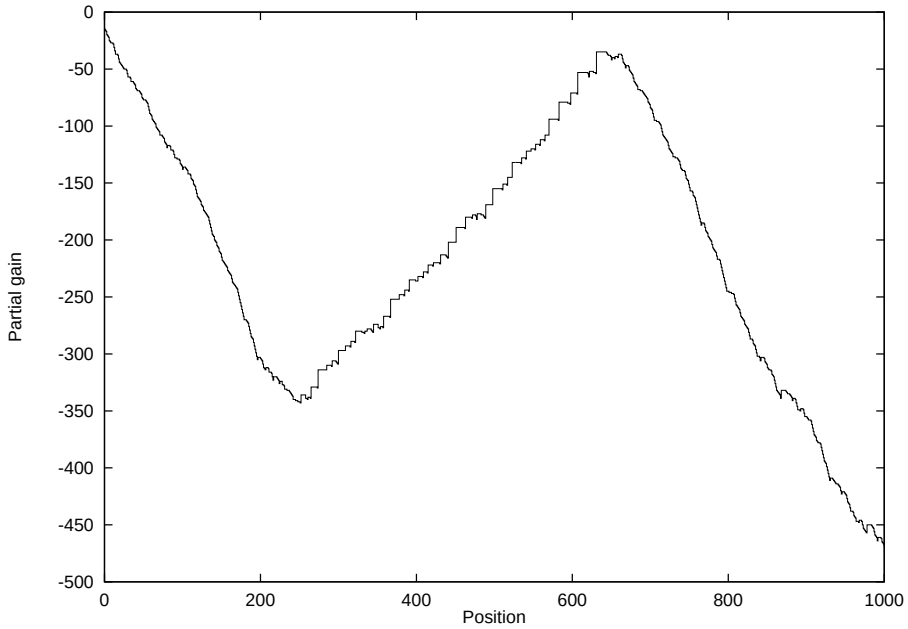
1. The length of M
2. M (2 bits per nucleotide)
3. the number of matches in the first group
4. the 1^{st} point mutation
5. the number of matches in the second group
6. the 2^{nd} point mutation
7. etc...

Each point mutation is coded over 3 bits.

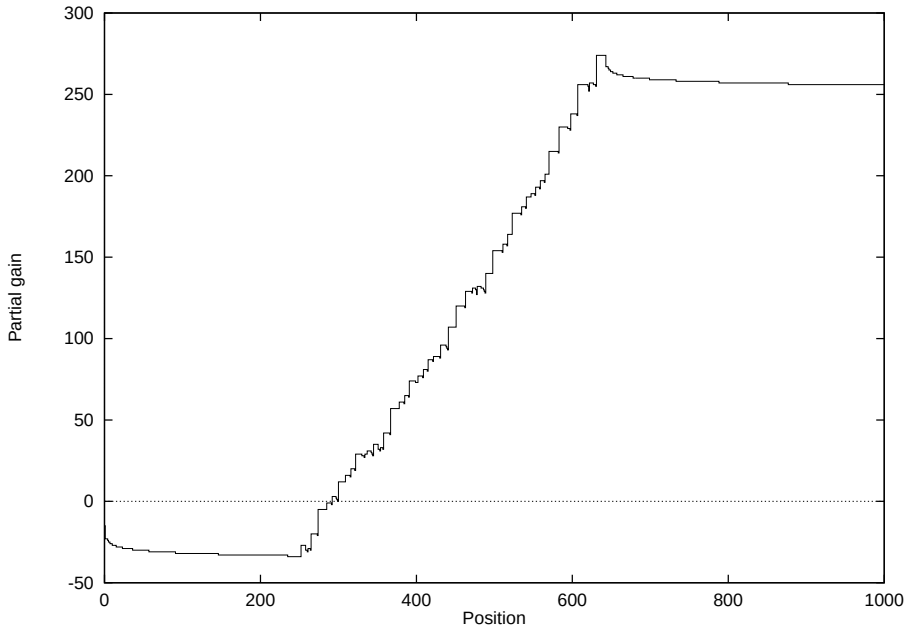
All these informations are enough to reconstruct the initial sequence S.

Optimization Principle: Use of the Compression Curve

Compression Curve: For each position i in the sequence, the ordinate is the partial gain obtained by the compression of the prefix of S ending at i.



Improve of the Curve by *Liftings*

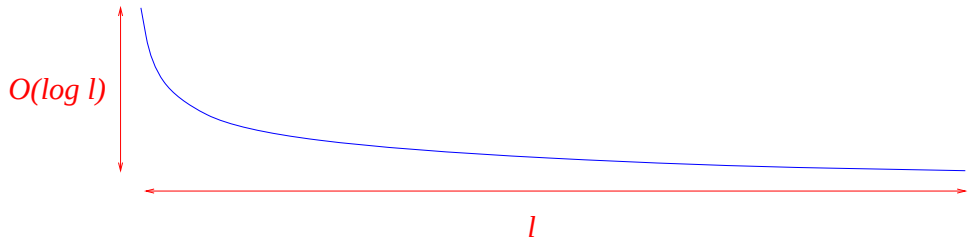


Application of a *rupture*

General Optimization Problem

Given:

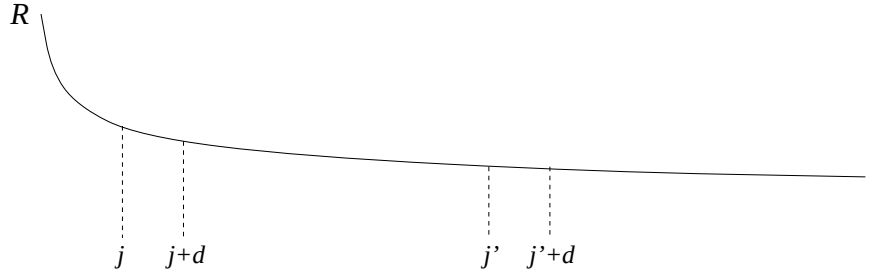
- A modular compression method $\mathcal{C}$ which allows the use of a rupture flag.
- A sequence S of length n that we compress using $\mathcal{C}$.
- A fixed rupture curve.



Find the optimal decomposition of the sequence into segments that must be compressed and subwords that must be copied as they are in order to maximize the global compression gain.

Technical Constraint

The rupture curve must be decreasing and concave ('reversed logarithm curve').

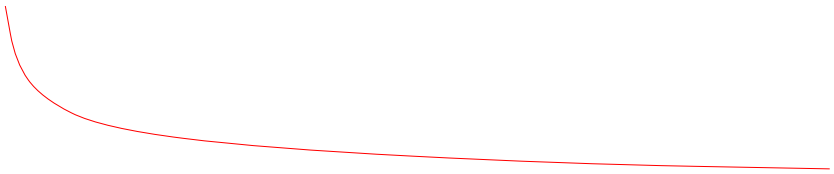


$$R(j) - R(j+d) \geq R(j') - R(j'+d) \text{ if } j < j' \text{ and } d > 0$$

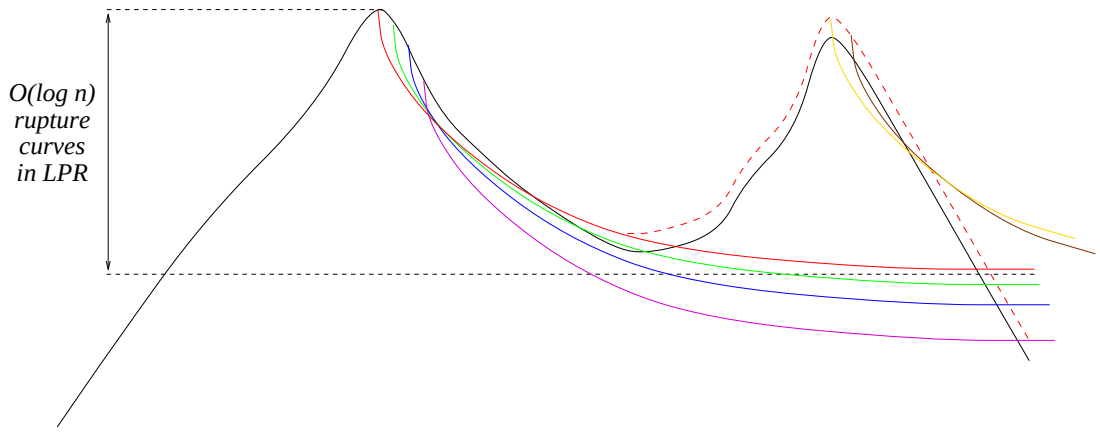
$\Longrightarrow$ Use of this specific shape of the rupture curves: precise crossing point of two rupture curves.

TURBOOPTLIFT Algorithm

Provides one optimal curve in time $O(n \log n)$



TURBOOPTLIFT Algorithm



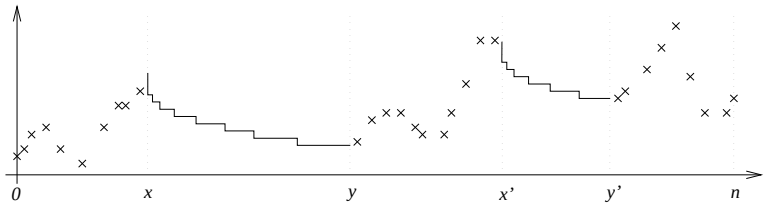
The curve is processed from left to right.
A List of Potential Ruptures is maintained (LPR).
For each position, at most $O(\log n)$ potential ruptures are present in LPR.
The greatest one is applied if it improves the partial curve.

$\Rightarrow$ **Unique optimal curve and minimal in rupture computed in time** $O(n \log n)$.

Use of TURBOOPTLIFT to Locate Repetitive Regions

We must have at our disposal a modular lossless compression method $\mathcal{C}$ which tries to compress by exploiting the particular kind of repetitive regions.

The choice of the initial compression method is very important.



3 repetitive regions: $S_{1..x-1}$, $S_{y+1..x'-1}$ and $S_{y'+1..n}$

2 non repetitive regions: $S_{x..y}$ and $S_{x'..y'}$ (ruptures)

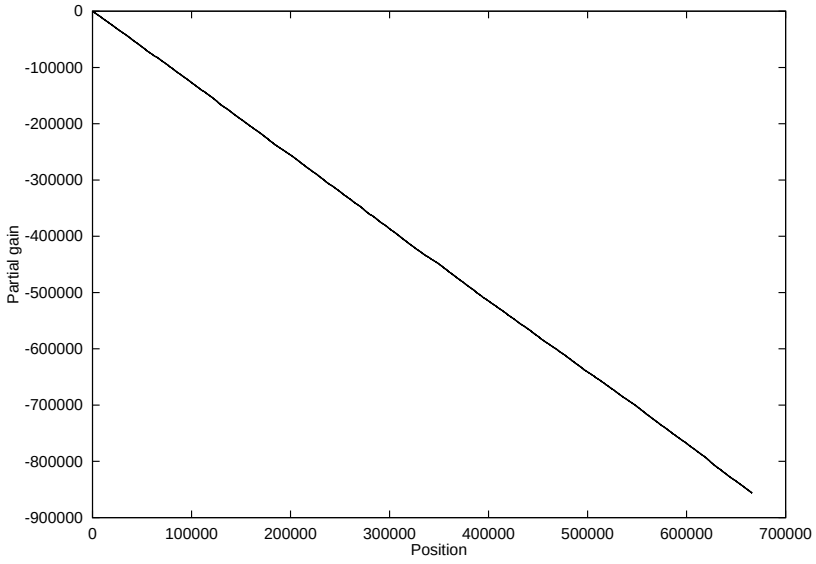
The repetitive regions are significant at the scale of the whole sequence:
If a repetitive subword is not significant enough at the scale of the whole sequence, its local gain will be absorbed in a longer rupture.

Another compression method may be applied to non repetitive regions.

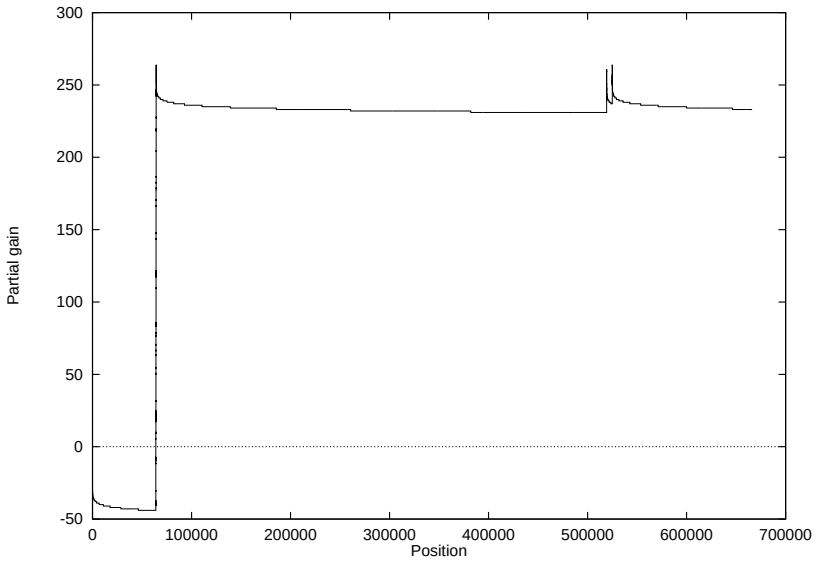
Back to the Location of ATR in DNA sequences

Let S be the whole yeast chromosome 11 (666448 bases) and $M = \texttt{TTC}$.

Computation of the minimal list of point mutations: takes 8 seconds (Sparcstation 20).



Optimization by liftings: takes 12 seconds



3 ATR and 4 ruptures

Three ATR located:

1. Begin: 63951, length: 393, gain: 308

```
       CTTCTTcTTcTTCtTCTTCTTC-TTCTTCTTCTTcTTCTTCtTCTTCtT-CTTCTTCTTcTTCTTCtTCTTC-TTCTTCTTcTTcTTCtTCTTCtT-CTTCTTcTTcTTC
63951  CTTCTT-TTtTTCcTCTTCTTCgTTCTTCTTCTT-TTCTTCcTCTTCcTgCTTCTTCTT-TTCTTCaTCTTCaTTCTTCTTgTTtTTC-TCTTCcTgCTTCTT-TTtTTC
```

```
       tTCTTCTTC-TTCTTCTTCTTcTTCTTCtTCTTC-TTCTTCTTcTTcTTCTTCtTCTTC-TTCTTCtTCTTcTTCTtCTTCTTCtT-CTTCTTCTTCtTCTTCTTCTTC-
64056  cTCTTCTTCgTTCTTCTTCTT-TTCTTCaTCTTCaTTCTTCTTaTT-TTCTTCaTCTTCaTTCTTCaTCTT-TTCTcCTTCTTCcTgCTTCTTCTTC-TCTTCTTCTTCc
```

```
       TTCTTCTTcTTCT--TCTTCtTCtT-CTTCTTCTTcTTCTTCTTCTTC-TTCTTcTTCTTCtTCTTCTTCTTC-TTCTTcTTCTTCtTCTTCTTCTTC-TCTTCTTCTT
64162  TTCTTCTTtTTCTccTCTTCcTCcTgCTTCTTCTT-TTCTTCTTCTTCcTTCTTtTTCTTC-TCTTCTTCTTCcTTCTTtTTCTTC-TCTTCTTCTTCcTTCTTtTTCTT
```

```
       cTTCTTCTTCTTC-TTCTTCTTCTTcTTCtTCTTCTTC-TTCTTCTTCTTCT--TCTTCtTC-TTCTTCTTCTTCT
64269  -TTCTTCTTCTTCcTTCTTCTTCTT-TTCcTCTTCTTCcTTCTTCTTCTTCTccTCTTCcTCcTTCTTCTTCTTCT
```

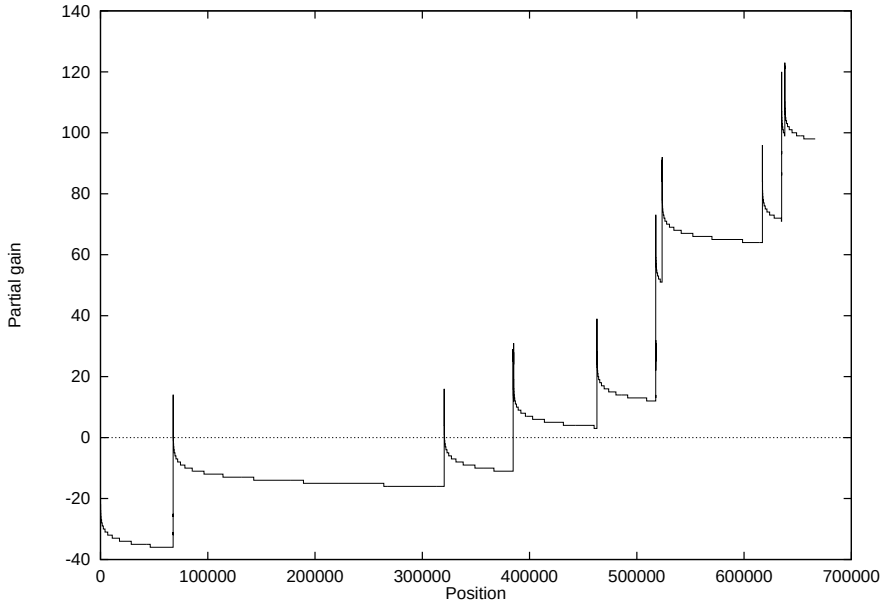
2. Begin: 519431, length: 45, gain: 30

```
       TCTTCTTCtTCtTCTTCtTCTTCtTCtTCTTCtTCTTCTTCtTCT
519431 TCTTCTTCcTCcTCTTCcTCTTCcTCcTCTTCcTCTTCTTCcTCT
```

3. Begin: 525127, length: 38, gain: 27

```
       TCTTCTTCTTCtTCTTCtTCtTCtTCttCTTCTTCTTC
525127 TCTTCTTCTTCgTCTTCgTCcTCgTCcgCTTCTTCTTC
```

Example: $M = \mathsf{T}$, S: whole yeast chromosome 11.



9 ATRs

Location of Microsatellites in the genome of the Archea *Methanococcus jannaschii*

- Genome length: 1664970 bp
- All possible patterns M with length in $[1, 6]$, but:
  - only primitives roots (AT but not ATAT)
  - only Lyndon words (lexicographicaly first circular permutation : AGTG but not TGAG).

Results

- 41 ATRs with length in $[14, 161]$, average length of 43 bp
- most of them are related to hexanucleotide patterns
- the most frequent patterns are nearly periodic: AAAAAG,...
- 18 microsatellites are located in intergenic regions
- 17 microsatellites are located inside a gene
- 6 microsatellites span over gene boundaries

Output of all detected zones because some small ones are part of larger, but composite, microsatellites (several interleaved zones concerning different patterns).
Since we do not have to give an arbitrary level of authorized imperfectness in the ATR, the method seems to be a consistent tool for microsatellites automatic annotation.

Application: Location of Locally Repetitive Regions (LRR)

LRR: a lot of short or long repeats with close occurrences

$\Longrightarrow$ no precise definition

Notice: ATR belong to LRR

Example:

```
AAAAACAAGAAAGAAAGAAAGAAGGAAGGAAGGAAAGAAGGAAGGAAGGAAGGAAGAAAGGAAGG
AAGGAAAGAAAGAAAGAAAGAAAAAGAAAGAAAGAAGGAAGGAAGGAAAGAAGGAAGGAAGGAAG
AAAGGAAGGAAGGAAAGAAAGAAAGAAAGAAAAAGAAAGAAAGAAAGAGAAAGAAAAAGGAAAGC
AAGAAAGAAAAGAAAGAAAGAAAGAAAAAGAAAGAAGGAAAGAAAAGAAACTAAAA
```

Solution: Application of TURBOOPTLIFT to the result of the compression algorithm LZ77

Usual compression programs use LZ77: `pkzip, gzip, arj,...`

Principle of LZ77: a sliding window runs trough the sequence from left to right.
It contains two parts:

1. a dictionnary (left)
2. a lookahead buffer (right)

At each step :

The dictionnary is used to compress the beginning of the buffer: a code, describing the repeat, is output and then the window slides to the right to skip the repeat.

Example: the DNA segment:

AAAAAGAACAGAAACAGAAAAACATAGAACAGAAACAGC

will be compressed to (in binary format):

$$(0,0,\mathsf{A})(1,4,\mathsf{G})(3,2,\mathsf{C})(5,4,\mathsf{A})(6,6,\mathsf{A})(8,3,\mathsf{T})(21,13,\mathsf{C})$$

If there are only a few repetitions, the compression will be disastrous!

TURBOOPTLIFT can be used to optimize the result of LZ77

$\Longrightarrow$ Precise location of LRR

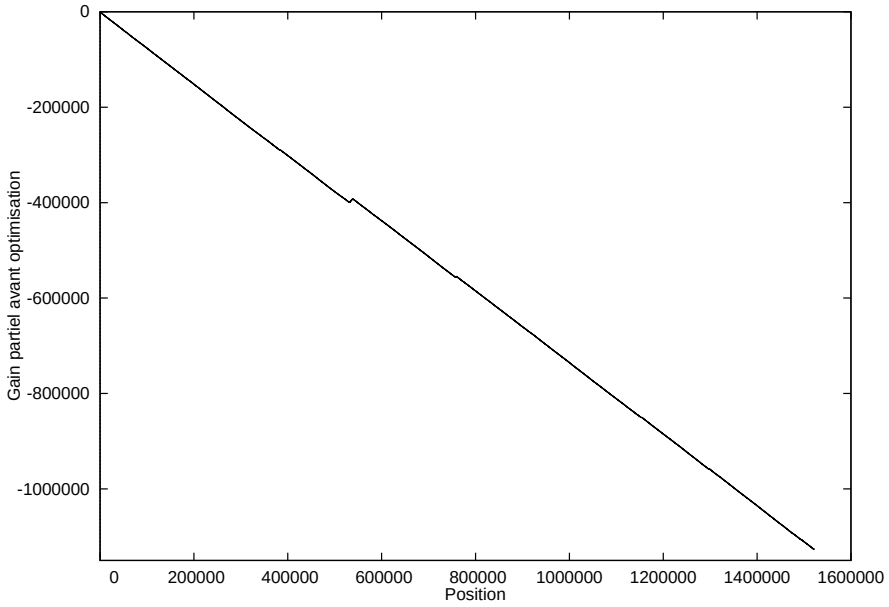
Allows the location of ATR without the knowledge of the pattern!

Notice:
The shape of the rupture curve is very important, it determines the sensitivity of the decomposition.

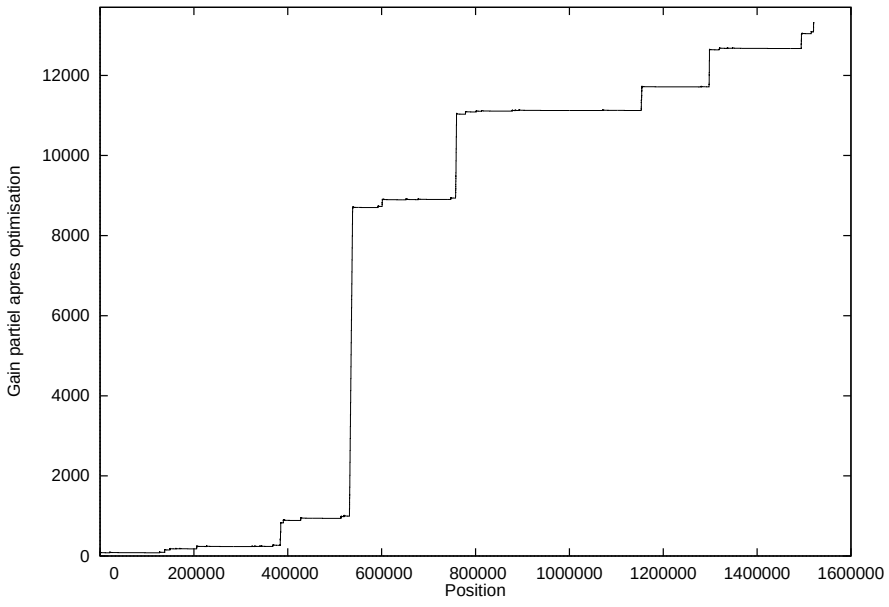
From now on, it is still difficult to **characterize** the detected zones because of our imprecise definition of LRR.

Example: Yeast chromosome 4 (1 522 191pb)

Before optimization:



After optimization:



96 LRR

Examples of detected LRR:

CACACCCACACCACACCCACACACACCACACCCACACACCACACCCACACCCACACACCCACACCCACA
CACCACACACCACACCACACCACACCCACACCCACACCACACCCACACCCACACACCACACCCACACCC
ACACACC

ATAAATAAATAAATAAATAAATAAATAAATAAATAAATAAATAA

TGAAGTAGTAGAGGATGAAGTAGTAGAGGATGAAGTAGTAGAGGATGAAGTAGTAGAGGATGAAGTAGT
AGAGGATGAAGTAGTAGAGGATGAAGTAGTAGAGGATGAAGTAGTAGAGGATGAAGTAGTAGAGGATGA
AGTAGTAGAGGATGAAGTAGTAGAGGATGAAGTAGTAGTGGATGAAGTAGTAGTGGATGAAGTAGTAGT
GGATGAAGTAGTAATGGATGAAGTAGTAATGGATGAAG

Interleaved repeats can also be detected.

5.Comparison of sequences (J-S. Varré)

Data compression can be used to compare genetic sequences.

Relative compression of a sequence S provided another sequence T: sequence T is used as a "dictionary" to construct S.

Example:
$T =$ATT**CATTA**GGATGGAT
$S =$ATT**CATTACATTA**GGATGGAT
Code: $DUP(3, 5)$

Model of DNA sequence evolution: generation of a *target sequence* from a *source sequence* by moving or duplicationg complete segments of the source sequence.

This approach does not impose the same order of corresponding segments in the two sequences.
$\Longrightarrow$ it better fits the biological reality than the classical models (alignments).

Transformational script: sequel of segment operations (move or duplication) to construct a target sequence from a source sequence.

Length of a transformational script: the length of its code (in binary format).

Transformational distance between S and T: $TD(S, T)$ is the length of the shortest transformational script that constructs S from T.

Theorem: $TD(S, T)$ can be computed in polynomial time.

Experiments:

Use of the transformational distance to construct phylogenetic trees.

Data: 46 genes of mitochondrial rRNA 16S in *crustacea isopoda*.

Results: phylogenetic analysis in agreement with known phylogenetic results.

Benefit:

This phylogeny inference does not need the initial construction of a multiple alignment

References

1. Delgrange O.,
   "Un algorithme rapide pour une compression modulaire optimale. Application à l'analyse de séquences génétiques"
   PhD Thesis, Université de Mons-Hainaut,1997 Available at
   `ftp://ftp.umh.ac.be/pub/ftp_info/Delgrange/pdf/publications/ThDelgrange.pdf`
2. Delgrange O., Dauchet M. and Rivals E.,
   "Location of Repetitive Regions in Sequences by Optimizing a Compression Method"
   Proc. *Pacific Symposium on Biocomputing (PSB' 99)*, january 4-9 1999, Big Island of Hawaii.
3. Li M. and Vitànyï PMB.,
   "Introduction to Kolmogorov Complexity and Its Applications"
   Springer-Verlag, 1993.
4. Milosavljević A. and Jurka J.,
   "Discovering simple DNA sequences by the algorithmic significance method"
   CABIOS, 1993, vol. 9 num 4, PP. 407-411.
5. Nelson M.,
   "The Data Compression Book"
   M&T Publishing Inc., 1991.

6. Rivals E., Dauchet M., Delahaye J-P. and Delgrange O., “Compression and Genetic Sequences Analysis”
BioChimie, 1996, vol. 78 num. 4, pp. 315-322.

7. Rivals E., Delgrange O., Delahaye J-P., Dauchet M., Delorme M-O., Hénaut A. and Ollivier E., “Detection of significant patterns by compression algorithms : the case of Approximate Tandem Repeats in DNA sequences”
CABIOS, 1997, vol. 13 num. 2, pp. 131-136.

8. Rivals E., Dauchet M., Delahaye J-P. and Delgrange O,
“Fast Discerning Repeats in DNA Sequences with a Compression Algorithm”
Proc. *The Eighth Workshop on Genome Informatics (GIW ’97)*, december 12-13 1997, Tokyo, Japan.

9. Varré J-S., Delahaye J-P. and Rivals E.,
“Transformation distances: A family of dissimilarity measures based on movements of segments”
Bioinformatics, 1999, vol. 15 num. 3, pp. 194-202.