The Transformation Distance : A Dissimilarity Measure Based On Movements Of Segments

Jean-Stéphane VARRÉ [1] varre@lifl.fr
Jean-Paul DELAHAYE [1] delahaye@lifl.fr
Éric RIVALS [2] E.Rivals@dkfz-heidelberg.de

[1] Université des Sciences et Technologies de Lille
Laboratoire d'Informatique Fondamentale de Lille - URA CNRS 369
UFR IEEA - Bât M3
59655 Villeneuve d'Ascq Cedex, France
Tel: +33 3 20 43 69 02, Fax: +33 3 20 43 65 66

[2] Theoretische BioInformatik
Deutsches Krebsforschungzentrum (DKFZ)
Im Neuenheimer Feld 280
Heidelberg 69120, Germany

Abstract

Evolution acts in several ways on DNA : either by mutating a base, or inserting, deleting or copying a segment of the sequence [17, 18, **?**]. Classical alignment methods deal with point mutations [19], genome-level mutations are studied using genome rearrangement distances [1, 2, 8, 9]. Those distances are mostly evaluated by a number of transpositions of genes. Here we define a new distance, called *transformation distance*, which quantifies the dissimilarity between two sequences in term of segment-based events (without requiring a preliminary identification of genes). Those events are weighted by their *description length*. The transformation distance from S to T is the *Minimum Description Length* among all possible scripts that build the sequence T knowing the sequence S with segment-based operations. The underlying idea is related to Kolmogorov complexity theory. Herein, we focus on the case where segment-copy, -reverse-copy and -insertion operations are allowed. We present an algorithm which computes the transformation distance. A biological application on Tnt1 tobacco retrotransposon is presented.

Keywords : *sequence comparison, dissimilarity measure, MDLP, information theory.*

1 Introduction

Evolution operates molecular alterations of two types: *point mutations*, namely insertion, deletion or substitution of one residue, and *segment-based modifications*: duplication, inversion, insertion, *etc* of a segment of the sequence. Genome level mutations operate also on large pieces of DNA and can thus be included in segment-based modifications. To our knowledge, no measure attempts to quantify dissimilarity by assessing segment-based differences, and by describing the differences between two sequences with an edit-script containing such segment-based operations. Consequently, sequence comparison is usually performed on similar parts of the sequences, like structurally or functionally related domains of proteins. Even if they correspond to complete biological entities like whole gene or protein, entire sequences are not compared, or only in case of high similarity. With such restrictions, one misses some evolutionary meaningful information written in the molecules.

Morgenstern and al. proposed a segment-based alignment method [MDW96] which aligns pairs of segments having local similarities and excludes regions of low similarity from the alignment. Based on this pairwise algorithm, they also provide a heuristic approach to multiple alignment. Their solution is alignment oriented: the segments put into correspondence always respect the overall order of the positions in the sequences. In our approach, this order is disregarded. We do not want to restrict our attention to “alignable” segment similarities, but also consider a wider class of segment operations like: duplication, inversion, or translocation. We use a different definition of similarity and this leads to a different class of problems.

We propose a new distance which evaluates segment-based dissimilarity between two sequences. This measure relates to the process of constructing a target sequence T using segments from a source sequence S. We consider a set of segment based operations: the copy of a segment of S into T, a reverse- copy of a segment of S into T (the resulting segment in T is the reverse-complement of the one in S) and the insertion of a new segment. Constructing T is done by applying a list of such operations, which we name a *script*.

A characteristic of our definition is the way operations are weighted. As a target T can always be obtained from a single insertion of the segment T itself, it is useless to weight a script by its number of operations. From the biological point of view, it exists no satisfactory probabilistic model which applies to those operations on a sequence. A natural, theoretically well-founded solution comes from the Algorithmic Information Theory (AIT), which explains how fair a priori weights can be assigned in the absence of specific knowledge [11, 14]. Following the Minimum Description Length Principle (MDLP), we weight an operation by its description length and a script by the sum of its operations weights. The transformation distance is defined as the minimal weight among all possible scripts.

In the AIT, the description length is a measure of the information content of a sequence of characters. The set of operations (which can be chosen different from the one we use here), defines a model of sequence construction. In this model, the transformation distance (TD) may be interpreted as the most efficient way to design T using information from S. It approximates the relative information content of T knowing S. For a general model, the AIT defines similarly an “information distance”. The information content as a criterion for sequence analysis has already been used in many different contexts: formal machine learning, economy and complexity theory. In biology, several researches focus on compression of the genetic sequences in order to discover significant structures like: direct repeats [12, 13, 15], palindromes [6] and tandem repeats [16]. Although suggested by Grumbach and Tahi, this criterion nor the information distance has never been used to perform direct sequence comparisons ; this is what we propose here (for further explanations see [14]).

Additionally to this definition, we present a polynomial time algorithm to compute exactly the TD. For this, we consider the weighted graph of all possible scripts. We demonstrate that the optimal script corresponds to the shortest path from a source node to a sink, representing respectively the left- and right-end of the sequence. Taking into account the relative weights of different segment-copies, we show properties that avoid including some non-optimal scripts in the graph. The subsequent decrease in the graph's size results in a practicable algorithm. Moreover, we provide an efficient implementation together with a user-friendly interface, and make them available to the community through our web-site. In a study of the family of sequences of Tnt1 Tobacco retrotransposons, the TD reveals the presence of segments duplications and segments re-orderings in some parts of the sequences, which are therefore not alignable. This suggests the TD is a useful and complementary approach to classical alignment methods.

Section 2 defines the transformation distance, while section 3 describes the algorithm, and the last section discusses a biological application concerning the evolution of Tnt1 tobacco retrotransposons.

2 The Transformation Distance

In this section, we define the transformation distance. To compute the distance we search for a list of operations (a script) which describes the target sequence T when the source sequence S is given. The script represents a sequence of operations that build T.

The set of allowed operations to build T has to contain at least two types of operations: an operation which allows to create parts of T which are not shared with S, called the *insertion operation*; and an operation which allows to obtain parts of T which are shared with S, called the *copy operation*. In the following, we describe the transformation distance for the particular case where the set of operations contains the copy, reverse-copy and insertion operations. The reverse-copy operation is the copy of a segment from S which is the reverse-complement of a segment of T. The main difference between copies and insertion operations is the following. For both copies operations, one does not have to write extensively the segment's sequence; as the segment is taken from the S sequence, one just gives its location in S. For the insertion operation, the inserted sequence has to be given extensively. When S and T share numerous segments, the best description of T is short because it refers to those segments in its copies operations. The transformation distance between S and T is then small.

As in the framework of the edit-distance, to obtain a distance (i.e., a measure) from scripts, we have to assign a cost (a weight) to each operation. Here we use an information-theoretical perspective: *the cost of an operation is the length of its description*. The cost of a script is the sum of costs of its operations, and the distance is the minimum among all possible scripts costs. The transformation distance is therefore the Minimum Description Length of a script to build T knowing S.

The description of an operation is simply a set of items that defines any specific operation: for a copy operation it is the location of the segment in S, the location in T and its length, for an insertion operation, the complete segment's sequence, its length and the location in T. From the point of view of information theory, both copies operations are more economical than the insertion operation because the description of a copy will be shorter than the one of an insertion. Biologically, a copied segment may correspond to a well-conserved segment which stems from common ancestry or duplication. For simplification, scripts build the target sequence from left to right by adding segments one after the other[1] (this is termed "concatenating"). The operations are ordered in the script according to this. It follows that the target position is simply the ending position at the current stage of the construction; thus the target position is omitted in the description. To be comparable, descriptions have to be written in the same language. We use the binary language because efficient encoding procedures are known (see "Encoding of an item" further on).

Important remarks and justifications.

- *A script is entirely defined by precising which copies or reverse-copies it contains.* This means that insertions can be deduced from those informations alone. Indeed, the insertions must provide the segments of T which are not brought by the copies, i.e. the

[1] To avoid asymmetry, we centre the two sequences and we use relative position of the copied segment from S to T instead of absolute position in S and T for copies operations.

```
     0         10        20        30        40        50        60        70        80        90
    ----------------------------------------------------------------------------------------------------

  0  UGUGCAAAGGUAGCAUAAUCAUUUGUUUUUUAAUUGGAAACUGGAAUGAAAGAGGCGACGAGAGGUAGGCUUUUUAUUACACAAGUUAACUUGACAUAAA
           [-----m-----]        [---l--]                                                       [---k----]
           [-----m-----]                                                                      [-----g----
     UGUGCUAAGGUAGCAUAAUAAUUAGUUAUUCAAUUGGUAACUAGAAUGAAUGGAUUGACAAAAAAAUAACUUUAUUAAUAAUUUAUUAAAUUAAUAUUAA


100  AGUUAAAAGGCUUUUAUAUUUCGGCGGGACGAUAAGACCCUAUAAAACUUUACUAACAACUUAUAUUAUUAUUGAACUAGUAGCAGGUUGUUGGUUAAGC
               [--j---]               [---i---]                                                         [
     ]            [---l--]-f--]       [---i---]    [--j---]          [---h---]          [---k----]-e---]
     AGUAAAAAAGCUCUUUUAAUUUAAAAAGACGACAAGACCCUAAAACCUUUUAUAAUAAUAUUUACUGGGGCGGUAAUUUACUAUUGACAUAAAUUUAAAU


200  UGGGGCGGCAAAUAUAUAAACAAUAUUAAUAUUAAAUAUUUAAAAUAGAAAAUUAGAACUUAAGUUAAAAAUUUAAACAAGUUACUUUAGGGAUAACAGC
     ---h---]                [----g-----] [---f--]                       [---e---]     [-----------d----
                                                 [-----------d-----------]        [---c---]            [-
     AACAUAAUGAAUGACUUUCCCUAACUCUAUAAAAUCCAAGCAGGUACUUUAGGGAUAACAGCAUAAUUUCUAUUUAGAGUUCUUAUCGACAUAGAAGAUU


300  AUAAUUUUUCUUGAUAGUUCUUAUUGGCAGAAAAGUUUAUGACCUCGAUGUUGAAUUGAGAUGUCCUUUAUGGUGCAG
     ------]        [---c---]            [---------b---------]
     --------b---------]
     AUGACCUCGAUGUUGAAUUAAUAUAACUUCUAUAAGCAG

    ----------------------------------------------------------------------------------------------------
```

	Operation	Cost
	insertion(UGUGCA)	1+12+3
m	copy(6, 6, 13)	1+1+1+4
	insertion(CAUUUGUU)	1+16+4
l	copy(27, 113, 8)	1+1+7+4
	insertion(GGAAA CUGGAAUGAA AGAGGCGACG AGAGGUAGGC UUUUUAUUAC ACAAGUUAAC)	1+110+6
k	copy(90, 183, 10)	1+1+7+4
	insertion(AGUUAAAAGG)	1+20+4
j	copy(110, 146, 8)	1+1+6+4
	insertion(UU UCGGCGGGAC GAU)	1+30+4
i	copy(133, 133, 9)	1+1+1+4
	insertion(UAAAACUU UACUAACAAC UUAUAUUAUU AUUGAACUAG UAGCAGGUUG UUGGUUAAG)	1+114+6
h	copy(199, 164, 9)	1+1+6+4
	insertion(CA AAUAUAUAAA CAAU)	1+32+5
g	copy(224, 89, 12)	1+1+8+4
	insertion(U)	1+2+1
f	copy(237, 118, 8)	1+1+7+4
	insertion(UAGAA AAUUAGAACU UAAGUUAA)	1+46+5
e	copy(268, 190, 9)	1+1+7+4
	insertion(CAA GU)	1+10+3
d	copy(282, 244, 25)	1+1+6+5
	insertion(UUC UUGAU)	1+16+4
c	copy(315, 277, 9)	1+1+6+4
	insertion(UGGCAG AAAAGU)	1+24+4
b	copy(336, 298, 21)	1+1+6+5
	insertion(GAG AUGUCCUUUA UGGUGCAG)	1+42+5
		683

Figure 1: An example of the computation of the transformation distance onto two RNA sequences. The target and the source are displayed aligned on their first residue. Common segments are denoted by letters within brackets. The associated script is shown in the table below: one line per operation and with the corresponding cost (i.e., number of necessary bits for this operation).

complementary parts. In the algorithm, searching for a script is equivalent to search for a combination of common segments.

- *Encoding of an item of information.* Two types of information have to be encoded:
  - a segment's sequence: as DNA is built with $4(=2^2)$ possible bases, each of them might be encoded over 2 (the exponent) bits. A n-bases long sequence is thus encoded over $2n$ bits.
  - an integer: the number of bits required for representing an integer l is $\lceil \log_2(|l|+1) \rceil$. When the item can either be positive or negative, we add one bit for the sign.

- *Other measure of the script's length.* Measuring the length of the script by counting the number of operations is unsuitable. If we do so, the insertion of the target sequence would be a minimal script, even if most of the sequence can be obtained from S. In biology a long common segment between two sequences is an evidence of evolutionary divergence, and the longer the segment is the more unlikely it appeared by evolutionary convergence. Weighting an operation by its description length generalises this idea. The difference between the description length of a copy and an insertion for the same segment grows with the segment's length. There is a limit where inserting and copying have the same cost. Below this length, it is unclear whether the segment appeared by convergence or divergence. Therefore, all segments shorter than a parameter called *Minimum Factor Length*(MFL) are systematically inserted.

- *A script is a program.* In fact, "executing" the script builds the sequence T. The script is a program which outputs T when S is supplied as data. This is how the Kolmogorov complexity theory defines a program [11]. The conditional Kolmogorov complexity of string x relatively to y, denoted $K(x|y)$, is the length of a shortest binary program which, on a universal Turing machine, outputs x if y is furnished as an auxiliary input data. $K(x|y)$ measures the minimal amount of information required to generate x knowing y by any effective process. This defines the algorithmic information distance. The transformation distance is an approximation of the relative Kolmogorov complexity of T knowing S. It is an approximation because our "machine" is not universal, but only allows three instructions : copy, reverse-copy and insertion. But, unlike the general algorithmic distance, the transformation distance is computable.

- *The distance depends on the set of authorised operations.* To obtain a more general distance, one may include duplication, multiple copy, or other biologically sounded segment-based operations. Our definition is still valid. One needs to design a generic description for each operation involved. This includes the case where segments contain point mutations.

- *Properties of the transformation distance.* The computation of the transformation distance is independent from the way we read sequences (5' to 3' or 3' to 5'). This stems from the way we encode target positions. The transformation distance is not symmetrical ($d(S,T) \neq d(T,S)$). It is intrinsic to our definition as well as to the Kolmogorov complexity: the way of describing S from T is not necessarily the same that the one of describing T from S. When a symmetrical distance is required, one can use the following definition: $(d(S,T) + d(T,S))/2$. The transformation distance does not satisfy the triangular inequality, although in practice we have never found biological cases for which it does not hold.

- *Related works.* Other approaches, called genome rearrangement distances, quantify segment-based evolution through the minimal number of operations needed to transform a "chromosome", i.e., an ordered list of genes, into another "chromosome" (the same list of genes but with another order). Among possible operations, called *genome-level mutations* or *genome rearrangements*, are translocations, transpositions and reversals [1, 2, 8, 9]. In most contexts, those methods do not apply directly to sequences, but to given lists of labels each one representing a gene. The unit cost is assigned to any operation.

  In our framework, segments which are rearranged have to be discovered. Moreover, we propose to weight more precisely the operations, i.e., with their description length. In the Kolmogorov complexity theory, shortest programs are more probable to be the computational origin of a sequence. This is because programs are distributed according to the a priori Solomonoff-Levin distribution [11, 14]. As the statistics of duplications, insertions, or other mutational events are not known, it is better to weight operations with their description length than simply to count them as it is done in genome rearrangements distances.

3 Algorithm

This section describes the algorithm we designed to compute the transformation distance. We show that the minimum description length script (i.e., whose length equals the transformation distance) corresponds to the shortest path from a source node to a sink node in a graph we define below.

Factors. In the first step of the algorithm, we search for all segments shared by the source and the target: those are candidates for copy operations. An important constraint of the construction of T is that any two copies operations (reverse or normal) cannot overlap in the target and belong to the same script. Therefore, copied segments for possible scripts fulfil the non-overlap relation. Let us call a *factor* a segment longer than the minimum length and shared by the source and the target sequence. A factor (p, q, l) is specified by its starting positions in the target (p), in the source (q) sequences and its length (l). To search for all factors, we use the algorithm of Leung & al. [10] because of the ability to discover exact but also point mutated repeats. To find the factors, we apply it onto the text formed by the concatenation of S and T.

Script graph. With our definition of weights, a minimal script is one that best combines copies operations: it should not insert a factor which could be copied, (i.e., in some way it should enclose as much copies as possible) and its copied segments should fulfil the non-overlap relationship. From now on, we call a *script* one which satisfies those two conditions. We define a graph on the set of all factors, the *script graph*, which is a sub-graph of the non-overlap graph for all common segments. Moreover, the fact that a script builds T from left to right also influences the underlying relation of the graph. We define a partial order on factors, denoted by the *non overlap relation* $<_o$. Let two factors $f = (p, q, l)$ and $g = (p', q', l')$, $f <_o g$ iff f and g do not overlap in the target sequence and f appears before g in T (i.e. $p + l \leq p'$).

The *script graph* is a directed graph where factors are nodes. Each factor is bound by an edge to any of its possible successor in the script. In other words, two nodes joined by an edge represent successive copies in a script and their factors fulfill the $<_o$ relation. Each edge is oriented from the leftmost factor towards the rightmost one. Additionally, we define a source node A and a sink node Z which serve respectively as the beginning and end of any script. A path from A to Z represents a selection of factors and as such defines a unique script.

Definition *A and Z are respectively source and sink nodes of the graph. f and g are two factors of $\mathcal{F}$, the set of all factors. The* script graph *$G = (V, E)$ is defined by:*

- *$V = \mathcal{F} \cup \{A, Z\}$,*
- *$(f, g) \in E$ iff :*
  1. *$f <_o g$,*
  2. *and $\nexists h \in \mathcal{F}$ such that $f <_o h <_o g$,*
- *$(A, f) \in E$ iff $\nexists h \in \mathcal{F}$ such that $h <_o f$,*
- *$(f, Z) \in E$ iff $\nexists h \in \mathcal{F}$ such that $f <_o h$.*

In order to compute the description length of each script (path) we assign costs to edges and vertices. The cost of a vertice is the cost of the copy of the factor it represents. The cost of an edge is the cost of the insertion needed between the copied segments, i.e. the source and target nodes of this edge. The cost of a path is obtained by adding costs of all edges and vertices.

Proposition *The computation of the shortest path going from source node A to the sink node Z gives the minimum description length script.*

Each path of the graph is clearly a script because it combines copies and insertion operations such that points 1 and 2 of the definition are verified. Each possible script is represented by a path in the graph. In fact, all possible copies operations are included in the graph and the set of successors of a node f contains exactly all the nodes which can be reached from f. The cost of a path is the description length of the associated script: indeed, the cost of a path is the sum of the costs of the insertions operations (edges) and the copies operations (nodes) it contains.

Algorithm complexity. Computing the set of all factors with the Leung & al. algorithm is known to be efficient [10]. Statistical studies have shown that its complexity increases almost linearly with the sequences lengths. Computing the shortest script is achieved in $\mathcal{O}(Card(V) + Card(E))$ with the algorithm *Dag-Shortest-Path* presented in chapter 25.4 of [4]. Let us denote n the length of the target sequence T. There are at most n^2 factors in T and therefore as much vertices in the script graph. As a complete graph over n^2 vertices has less than n^4 edges, the computation of the transformation distance requires less than $\mathcal{O}(n^4)$ units of time.

Practicable algorithm. Although this complexity is for worst cases, in practice the computation of the complete script graph is too inefficient to be applied on long sequences (more than 100 kb). In most of cases, and particularly when sequences are very similar, the number of factors is so large that it prevents building the graph in main memory. To improve our algorithm, we studied properties which characterise non-optimal copies operations (i.e., which cannot belong to the optimal script). Those factors can thus be removed from the graph. Consequently, we define the *compact script graph* which encloses only "good" scripts. The above-mentioned properties and the definition of this compact graph cannot be detailed here for the sake of shortness. For implementation matter, only maximal factors (those that are not sub-factors of another factor) are included in the graph, their sub-factors being created when necessary. Acceleration of the computation time is illustrated hereunder. Table 1 reports the number of factors and the construction time for the script graph and the compact script graph. Only the compact script graph allows to compute in little time the transformation distance on long sequences (more than 500 kb).

		Number of factors		Running time	
	Seq. Length	SG	CSG	SG	CSG
Rice and Tobacco	140 kb	10963	818	75s	6.9 s.
E.gracilis and O.sinensis	140 kb	1683	209	18.4 s.	16 s.
HIV Type 1 and HIV Type 2	10 kb	8578	111	–	7.6 s.
Bac. Subtilis 1 and Bac. Subtilis 2	250 kb	18144	6	–	18 s.
C.E. 1 and C.E. 2	630 kb	20172	5147	–	151 s.

Table 1: Construction time and number of factors included in the graph for both the script graph (SG) and the compact script graph (CSG). For the *Rice*, *Tobacco*, *O.sinensis* and *E.gracilis*, the sequences used are the complete chloroplast genomes. Sequences *Bac. Subtilis 1* and *2* are parts of the bacillus subtilis complete genome. *C.E. 1* and *C.E. 2* are two clones of Caenorhabditis elegans. Running times have been computed on a PC Pentium 166MMX. A "–" indicates that the program exhausts the computer memory.

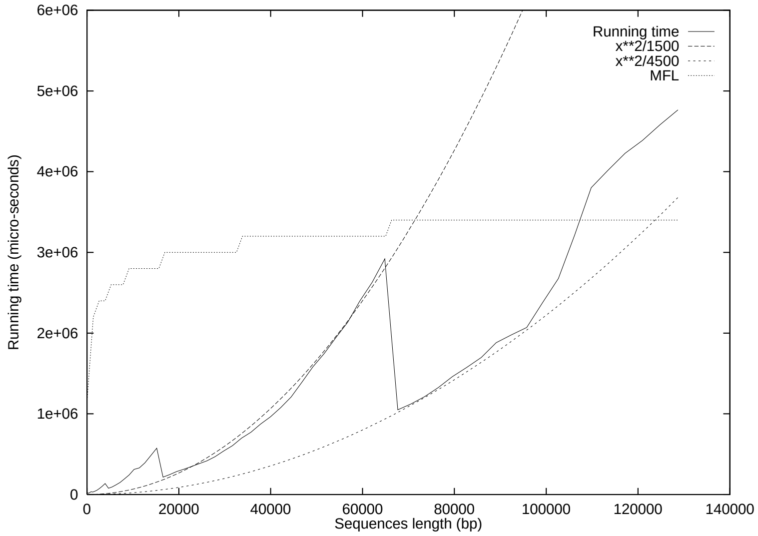

Figure 2: Running times of the distance computation versus sequence length (on a PC Pentium 233 computer)

Figure 2 illustrates the variation of the running time in function of the sequence length. The sequences are longer and longer prefixes of tobacco's and rice's chloroplast genomes (we observed the running time for different sequences lengths). The Minimal Factor Length parameter was set to $\lceil \log_2(|T|+1) \rceil$ where $|T|$ is the length of T and it varies with sequences lengths. The vertical drops of the plain line curve denotes an acceleration because the number of nodes decreased. They correspond to losses of factors when the minimal factor length reaches a new discrete value. In fact, they relate to drops in the MFL curve, whose function is $\lceil \log_2(|T|+1) \rceil$. With the present implementation, the time requirement is short even for long sequences: around 5 seconds for 130 kb.

Implementation details. The algorithm is implemented in C and available at http://www.lifl.fr/~varre/TD. As shown on figure 3, a user-friendly graphical interface (implemented with Tcl/Tk) allows to compute all against all comparisons for a set of sequences and to visualise each comparison.

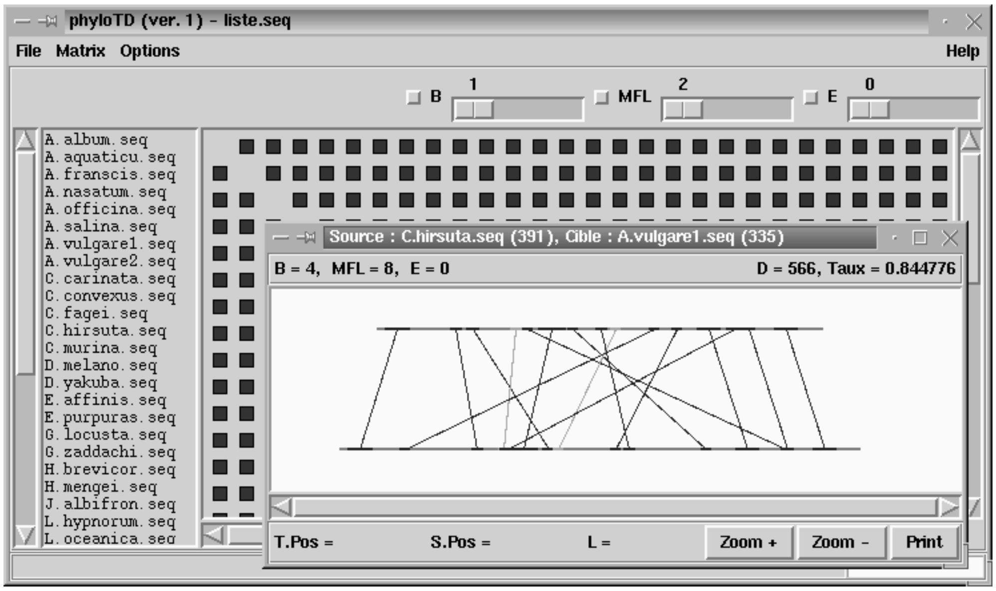

Figure 3: Example of computation of the transformation distance for a set of sequences. One can display the actual distance and the corresponding minimal script between two sequences by clicking on the corresponding square in the matrix.

4 Biological application

We applied the TD to investigate the evolution of the TnT1 tobacco retrotransposon. The results illustrates the usefulness of the TD and its larger applicability compared to alignment methods: the TD allows to find, between the compared sequences, segments duplications and re-orderings which may result from evolutionary events.

Families of Tnt1 tobacco retrotransposon. The problem considered here is the evolution of Tnt1 tobacco's retrotransposon, and specifically the evolution of its Long Terminal Repeat (LTR). The material is a set of 140 sequences of this LTR taken out of seven species of tobacco. The LTR feature is the following from 5' to 3': the RT box, the linker, then the U3 and R boxes. It has been suggested that the high mobility of Tnt1 may require rapid evolution [3].

We computed all pairwise comparisons for the 140 sequences (the running time was less than 1 hour). Fig. 4 shows the representative comparison of the retrotransposons of *Tobacco sylvestris* (top horizontal line) and *Tobacco tomentosiformis* (bottom horizontal line). The 5' and 3' ends feature each 3 parallel vertical bars that link the segments shared by both sequences. It shows that those regions corresponding respectively to the RT+linker and to the R box are well conserved, and thus also well aligned (this is observed on all comparisons). On the opposite in the middle part of the sequence, in the U3 region, one sees 6 vertical bars which intersect each other, suggesting that the order of the corresponding segments has changed during divergence of those sequences. Moreover, two vertical bars have the same end-point and then refer to the same segment in *T. tomentosiformis* sequence, while they point to different segments (i.e., end-points) in *T. sylvestris*: this segment may have been duplicated during evolution. Such segment relations observed in many pairwise comparisons simply prevent correct alignment. Those results are consistent with the analysis of Casacuberta et al. in which they suggested that the RT+linker and R boxes were conserved, while they suspected rearrangements inside the U3 box. However providing evidence for the latter was nearly impossible as it required studying by eye all pairwise alignments. The TD algorithm allowed us to detect and visualise some rearrangements events automatically, suggesting that the TD is a useful and complementary approach to classical alignment strategies.

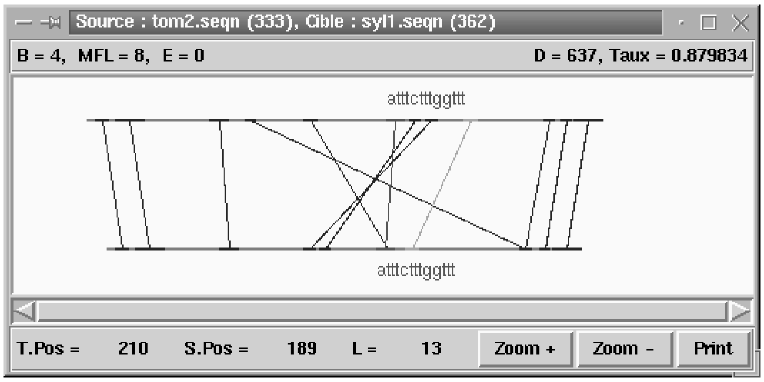

Figure 4: Comparison of Tnt1 tobacco retrotransposon of tobacco tomentosiformis (as source) and tobacco sylvestris (as target).

5 Conclusion

This work provides a new measure, the transformation distance, for comparing genetic sequences and an efficient algorithm to compute it. The application to Tobacco retrotransposons TnT1 points out types of sequence relationships which are undetectable with alignments. Indeed, it detects segments duplication and re-ordering which usually prevent correct alignments. This argues for the usefulness of the transformation distance as an alternative tool to investigate sequences relationships. Compared to alignments, Our work shows that the concepts of the Algorithmic Information Theory leads to practical approaches and effective algorithms in a biological context. Among others, wider applications of the transformation distance are: phylogenies, sequences clustering and analysis, and investigation of segment-based evolution.

We suggest that the transformation distance may be particularly appropriate to investigate the evolution of RNA sequences, where the palindromic segments may correspond to elements of the secondary structure. The TD favors conservation of such segments and would thus better account for the secondary structure. The study of the phylogeny of isopods based on mitochondrial RNA sequences is in progress.

Acknowledgements

We would like to thank Samantha Vernhettes, Helene Chiapello and Marie-Angèle Grandbastien (INRA Versailles, http://www.inra.fr/Versailles/BIOCEL) for Tnt1 retrotransposon data and very interesting discussions. The authors are also grateful to Didier Bouchon and its team (Génétique et Biologie des Populations de Crustacés, UMR CNRS 6556, http://wwwumr6556.univ-poitiers.fr) for their data and numerous useful comments, and to Dominique Anxolabéhère (Modélisation de la dynamique des populations d'éléments transposables, Dynamique du Génome et Evolution, Insititut Jacques Monod, Paris VII, http://www.ijm.jussieu.fr) for interesting discussions. E. R. thanks E. Bornberg and M.-L. Muiras (DKFZ) for their careful reading of the manuscript.

References

[1] V. Bafna and P. Pevzner, "Genome Rearrangements and Sorting by Reversals", Proc. 34th FOCS, IEEE, 148-157, 1993.

[2] Vincent Bafna and Pavel Pevzner, "Sorting Permutations by Transpositions", In 6th Symposium on Discrete Algorithms SODA,614–621,1995.

[3] Josep M.Casacuberta, Samantha Vernhettes and Marie-Angèle Grandbastien, "Sequence Variability within the Tobacco Retrotransposon Tnt1 Population", *EMBO Journal*, 14(11):2670-2678, 1995.

[4] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[5] R.F. Doolittle. Similar Amino Acid Sequences: Chance or Common Ancestry? *Science*, 214(4517):149–159, 1981.

[6] Stéphane Grumbach and Fariza Tahi. Compression et compréhension de séquences nucléotidiques. *Technique et science informatique*, 14(2):217–233, numéro spécial Informatique et Génomes 1995.

[7] David M.Hillis, Craig Moritz and Barbara K.Mable, *Molecular Systematics*, Sinauer Associates Inc., 1996.

[8] John Kececioglu and David Sankoff, "Efficient Bounds for Oriented Chromosome Inversion",In 5th Symposium on Combinatorial Pattern Matching,307–325, 1994.

[9] John Kececioglu and R. Ravi, "Of Mice and Men : Algorithms of Evolutionary Distances Between Genomes with Translocations",In 6th Symposium on Discrete Algorithms SODA, 604–613,1995.

[10] Ming-Ying Leung, B.Edwin Blaisdell, Chris Burge and Samuel Karlin, "An Efficient Algorithm for Identifying Matches with Errors in Multiple Long Molecular Sequences",*Journal of Molecular Biology*, (221):1367-1378, 1991.

[11] M.Li and P.M.B.Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, Spinger-Verlag, New-York, 2nd Edition, 1997.

[12] Aleksandar Milosavljevič and Jerzy Jurka. Discovering Simple DNA Sequences by the Algorithmic Significance Method. *CABIOS*, 9(4):407–411, 1993.

[13] Aleksandar Milosavljevič and Jerzy Jurka. Discovery by Minimal Length Encoding: a Case Study in Molecular Evolution. *Machine Learning*, 12:69–87, 1993.

[MDW96] B. Morgenstern, A. Dress and T. Werner, Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci. USA*, 93:12098–12103, 1996.

[14] Éric Rivals, Max Dauchet, Jean-Paul Delahaye and Olivier Delgrange, "Compression and Genetic Sequences Analysis", Biochimie vol. 78, 1996.

[15] É. Rivals, M. Dauchet, J-P. Delahaye, and O. Delgrange. Fast Discerning Repeats in DNA Sequences with a Compression Algorithm. In *Proceedings of 8th Workshop on Genome Informatics (GIW 97)*, pages 215–226. Universal Academy Press Inc., 1997.

[16] É. Rivals, O. Delgrange, J-P. Delahaye, M. Dauchet, M-O. Delorme, A. Hénaut, and E. Ollivier. Detection of significant patterns by compression algorithms: the case of Approximate Tandem Repeats in DNA sequences. *CABIOS*, 13(2):131–136, April 1997.

[17] Frank H. Ruddle. Vertebrate Genome Evolution - The Decade Ahead. *Genomics*, 46:171–173, 1997.

[18] Robert B. Russel. Domain Insertion. *Protein Engineering*, 7(12):1407–1410, 1994.

[19] Michael S. Waterman. *Introduction to Computational Biology.* Chapman and Hall, 1995.