BIOINFORMATICS

Vol. 15 no. 3 1999
Pages 194–202

Transformation distances: a family of dissimilarity measures based on movements of segments

Jean-Stephane Varré[1], Jean-Paul Delahaye[1] and Eric Rivals[2]

[1]Laboratoire d'Informatique Fondamentale de Lille (LIFL), UFR IEEA – Bât M3, 59655 Villeneuve d'Ascq Cedex, France and [2]Theoretische BioInformatik Deutsches Krebsforschungzentrum (DKFZ), Im Neuenheimer Feld 280, Heidelberg 69120, Germany

Received on October 6, 1998; revised on December 16, 1998; accepted on December 22, 1998

Abstract

Motivation: *Evolution acts in several ways on DNA: either by mutating a base, or by inserting, deleting or copying a segment of the sequence (Ruddle, 1997; Russell, 1994; Li and Grauer, 1991). Classical alignment methods deal with point mutations (Waterman, 1995), genome-level mutations are studied using genome rearrangement distances (Bafna and Pevzner, 1993, 1995; Kececioglu and Sankoff, 1994; Kececioglu and Ravi, 1995). The latter distances generally operate, not on the sequences, but on an ordered list of genes. To our knowledge, no measure of distance attempts to compare sequences using a general set of segment-based operations.*

Results: *Here we define a new family of distances, called* transformation distances, *which quantify the dissimilarity between two sequences in terms of segment-based events. We focus on the case where segment-copy, -reverse-copy and -insertion are allowed in our set of operations. Those events are weighted by their* description length, *but other sets of weights are possible when biological information is available. The transformation distance from sequence S to sequence T is then the* Minimum Description Length *among all possible scripts that build T knowing S with segment-based operations. The underlying idea is related to Kolmogorov complexity theory. We present an algorithm which, given two sequences S and T, computes exactly and efficiently the transformation distance from S to T. Unlike alignment methods, the method we propose does not necessarily respect the order of the residues within the compared sequences and is therefore able to account for duplications and translocations that cannot be properly described by sequence alignment. A biological application on Tnt1 tobacco retrotransposon is presented.*

Availability: *The algorithm and the graphical interface can be downloaded at http://www.lifl.fr/~varre/TD*

Contact: *{varre,delahaye}@lifl.fr, E.Rivals@dkfz-heidelberg.de*

Introduction

Evolution operates molecular alterations of two types: *point mutations*, namely insertion, deletion or substitution of single residues, and *segment-based modifications*: duplication, inversion, insertion, etc. of whole segments 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.

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 a whole gene or a protein, entire sequences are not compared, or only in case of high similarity. With such restrictions, one misses some biological meaningful information written in the molecules.

Certain alignment oriented methods take into account the existence of 'segment' of similarity. Morgenstern *et al.* (1996) proposed a segment-based alignment method which aligns pairs of direct segments having local similarities and excludes regions of low similarity from the alignment. Schöniger and Waterman (1992) found a dynamic programming algorithm to include in a classical alignment parts where the sequences are aligned in reverse order. These solutions are 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.

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 in another order. Several operations were considered (often one operation per distance): reversals (Hannenhalli and Pevzner, 1995), transpositions (Bafna and Pevzner, 1998), translocations (Hannenhalli, 1996), block interchanges (Christie, 1996) and the syntenic distance which is applied to (unordered) set of genes (Ferretti *et al.*, 1996). In most contexts, those methods do not apply directly to sequences, but to given lists of labels each one representing a gene. The costs are user parameters. Most of those distances are computationally expensive. In our framework, segments which are rearranged have to be discovered. Moreover, we propose to weight more precisely the operations.

We propose a new measure which evaluates segment-based dissimilarity between two sequences: the source *S* and the target *T*. This measure relates to the process of constructing the target sequence *T* with *segment operations*. The construction starts with the empty string and proceeds from left to right by adding segments, one segment per operation. A list of operations is called a *script*. Three *types of segment operations* are considered: the *copy* adds segments that are contained in the source sequence *S*, the *reverse copy* adds segments that are contained in *S* in reverse order, and the *insertion* adds segments that are not necessarily contained in *S*. For the sake of clarity, we call the insertion of a segment in general, a *type of operation*, while the insertion of the segment, say 'agtct', is an *operation* because it is completely specified. The measure depends on a parameter that is the *Minimum Factor Length*; it is the minimum length of the segments that can be copied or reverse copied. A positive weight is assigned to each operation and the weight of a script is the sum of the weights of its operations. Depending on the number of common segments between *S* and *T*, there exist several scripts that construct the target *T*. The *minimal scripts* are all scripts of minimum weight and the *transformation distance* (TD) is the weight of a minimal script. This defines a precise optimization problem which we solve in this work. We would like to emphasize that with another set of types of operations, one can use the same generic definition of transformation distance. Thus, the transformation distance generalizes in a family of measures of distance between sequences.

How are operations weighted? In our framework, unit cost is meaningless (see next section). From the biological point of view, a satisfactory probabilistic model does not exist that applies to segment operations on a sequence and would allow us to derive weights. Therefore we apply another idea and follow the principle of parsimony. The principle of parsimony is justified by the more general Minimum Description Length Principle (MDLP) (Li and Vitányi, 1997; Rissanen, 1989). We adopt the MDLP and weight each operation by its *description length*. A generic description scheme is associated with a type of operation. For the copy, the description requires a 2-bits code to distinguish the type of operation, an offset between the locations of the segment in *S* and in *T*, and the length of the segment. To a reverse copy corresponds the same description but with another 2-bits code for a reverse copy. For an insertion, one needs the 2-bits code for an insertion, the length of the segment and the sequence of segment. Some examples are given in the next section.

Description length is not an arbitrary way of weighting. In fact, it seems natural in the absence of specific knowledge: it represents the quantity of information necessary to describe an operation. This property has its underlying in the Algorithmic Information Theory (Kolmogorov, 1965). This theory suggests that the description length is the fairest weighting scheme that one could use a priori. The use of information theory is advocated by Yockey (1992). For an introduction to the AIT, we refer the reader to the book of Li and Vitányi (1997); an explanation of its use to 'weight events' in computer sequence analysis can be read in Rivals *et al.* (1996).

Additionally to this definition, we present a polynomial time algorithm to compute exactly the transformation distance according to the definition given above. For this, we consider a weighted graph of all possible scripts. We demonstrate that minimal scripts correspond to the shortest paths from a source node to a sink node, representing respectively the left- and right-end of the target sequence. Taking into account the relative weights of different segment-operations, we use properties of some non-optimal scripts which allow us to exclude them from the graph. The subsequent decrease in the size of the graph 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. These sequences are clearly not alignable and therefore only restricted comparisons are possible with alignment methods.

The transformation distances

In the introduction, we defined precisely the TD we use and wrote that it can be generalized to adopt other sets of types of operations. We detail this point here, and explain why the description length is a reasonable choice for the weights, although not the only possible one.

The set of types of segment operations

First, the set must contain an insertion. When constructing *T*, one may need to add a segment which does not occur in the source *S*. Clearly, the set must enclose a type of operation which enables this: it is the insertion. In addition to the insertion, most of the other operations can be included: those used in genome rearrangement distances, but also the du- or multiplication of a segment which could be in tandem or not, and the deletion.

Some remarks must be stated. First, the set we choose allows detecting most of the events mentioned above. For example, our 'copy' can displace a segment from the source in the target: special cases of this are transpositions and block interchanges. The difference is that in our framework, transpositions are not given a different weight than block interchanges. Nevertheless, transpositions and block interchanges can be detected. Second, the more types of operations that are considered, the higher the number of possible scripts: this will increase the complexity of the problem. Third, deletion is a very peculiar operation which requires that an already inserted segment could be modified in a second operation: this corresponds to a construction which is not left to right. This also requires a more complex algorithm.

Operations weights

We mentioned that unit costs are unsuitable in our framework. Indeed, a script containing a single insertion of the segment *T* itself always exists. With unit costs, this script would be minimal. Thus, when choosing a weighting function, one must let the insertion of a 'long' segment of *S* cost more than a copy of it, so that the TD is able to reveal sequence relationship. The description length fulfills this property because the segment of a copy is available in the source sequence *S*, while it must be encoded explicitly in an insertion, and also because the description length increases with the segment length.

The study of the construction of a target sequence is the object of the AIT in the case of general sequences. This theory suggests that for the measure we define here, description length is the fairest a priori weights one can choose. This is because the process of constructing a sequence using a set of operations has intrinsic properties, whatever the type of sequences. Because of these properties, the minimum description length measures the information content of an operation, and here operations are potential modifications of the genetic sequences. Detailed explanations are beyond the scope of this paper. In his book Yockey (1992) reviews the characteristics of information theory and concludes that using it should benefit sequence comparison.

The weighting function can be adapted according to biological knowledge. For instance, the weights can be multiplied by a given coefficient if one wants to favor copies instead of reverse copies. This does not change the definition, nor the algorithm.

Remarks and justification

Interpretation of the TD and of a script. The TD and the associated minimal scripts are clearly not an attempt to view evolution as a computer program. First in biological evolution, the source and the target sequences derived from a common ancestor sequence, say *A*. Our model is a simplification in that it considers a process linking directly the source to the target and which does not exist. Nevertheless, a minimal script suggests (1) what parts of the sequences are common to *S* and *T*, (2) which of those parts may have been rearranged in their relative order, and (3) possible operations for those rearrangements which could have happened between *A* and *S* or *A* and *T*.

Minimum Factor Length parameter. There is a limit of the length of the segment, below which inserting a segment costs more than copying it if it occurs in *S*. One may interpret this limit, saying that below, it is unclear whether the segment appeared by convergence or divergence. Therefore, all segments shorter than the *Minimum Factor Length* are systematically added using insertions.

A script as 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. The definition of the script is a restricted form of the definition of a *program* given by Kolmogorov in the AIT (Li and Vitányi, 1997). The conditional Kolmogorov complexity of a string *x* relatively to a string *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 approximates of the relative Kolmogorov complexity of *T* knowing *S*: as our 'machine' only allows three instructions (copy, reverse-copy and insertion) it is not universal. Therefore, the transformation distance does not consider all programs but only those limited to these 3 instructions. On the other hand, unlike the general algorithmic distance, the transformation distance is computable.

Properties. The computation of the transformation distance does not depend on 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: 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. However, we have made some experiments and it seems that, in practice, the triangular inequality is often satisfied.

Encoding scripts. To be comparable, descriptions have to be written in the same language. We use the binary language because efficient encoding procedures are known. As DNA is made up from $4(= 2^2)$ possible bases, each of them might be encoded over 2 (the exponent) bits. A *n*-bases long sequence is thus encoded over $2n$ bits. The number of bits required for representing an integer l is $\lceil \log_2(|l|+1) \rceil$. When the item can be either positive or negative, we add one bit for the sign. One needs a 2-bits code to encode each type of operation ($2 = \log_2(3)$).

Figure 1 gives an example of a script and its weights. The first line (insertion(UGUGCA)) has a weight of 2+12+3: $2 = \lceil \log_2(3) \rceil$ is the number of bits required to encode the type of the operation, $12 = 2 \times 6$ is the number of bits to encode the segment explicitly, $3 = \lceil \log_2(6+1) \rceil$ is the number of bits to encode the length of the segment. The fourth line (copy(27, 113, 8)) has a weight of 2+1+7+4: as for the insertion 2 is for encoding the type of the operation, $4 = \lceil \log_2(8+1) \rceil$ is for the length, $7 = \lceil \log_2((113 - 27)+1) \rceil$ is the number of bits required to encode the offset between the locations of the segment in *S* (113) and in *T* (27), 1 is for the sign of the offset because it can be negative (e.g. the copy of h).

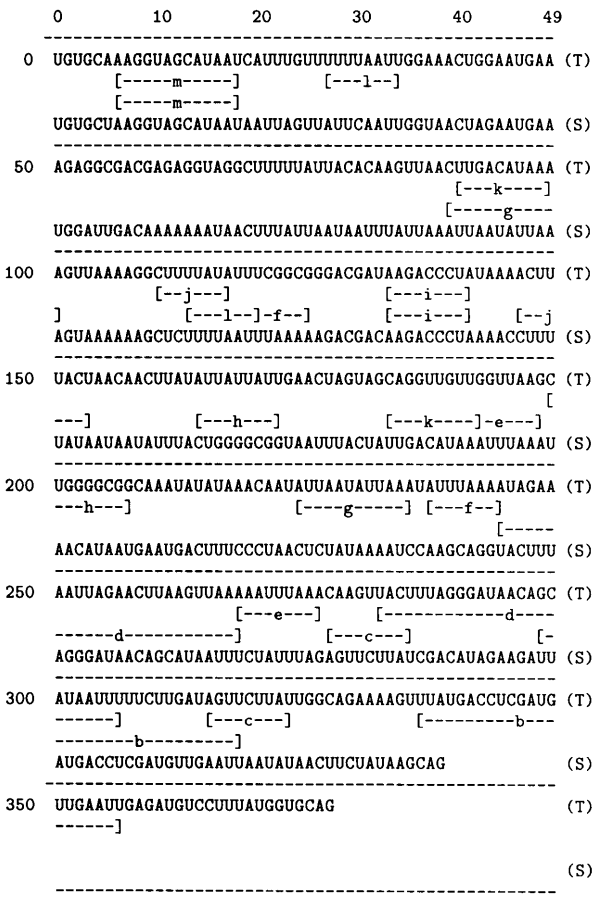
A script is defined by its copies. We remark that a script is entirely defined by specifying which copies or reverse-copies it contains. This means that insertions can be deduced from those information alone. Indeed, the insertions must provide the segments of *T* which are not brought by the copies, i.e. the complementary parts. In the algorithm, searching for a script is equivalent to a search for a combination of segment pairs that can be copied.

Algorithm

This section describes the algorithm we have designed to compute the transformation distance. We show that the minimal script corresponds to the shortest path from a source node to a sink node in a weighted directed graph we define below.

Factors

Let us call a *factor* a pair of segments, one segment from each sequence (source and target), such that the two segments are either identical or the first segment is the reverse complement of the second one. The set of all factors is denoted F. A factor is specified by the triplet (*p,q,l*): its starting positions in the target sequence (*p*), in the source sequence (*q*) and its length (*l*). We define a relation $<_0$ on the set of all factors such that



	Operation	Cost
	insertion(UGUGCA)	2+12+3
m	copy(6, 6, 13)	2+1+1+4
	insertion(CAUUUGUU)	2+16+4
l	copy(27, 113, 8)	2+1+7+4
	insertion(GGAAA...ACAAGUUAAC)	2+110+6
k	copy(90, 183, 10)	2+1+7+4
	insertion(AGUUAAAAGG)	2+20+4
j	copy(110, 146, 8)	2+1+6+4
	insertion(UU UCGGCGGGAC GAU)	2+30+4
i	copy(133, 133, 9)	2+1+1+4
	insertion(UAAAACUU...UUGGUUAAG)	2+114+6
h	copy(199, 164, 9)	2+1+6+4
	insertion(CA AAUAUAUAAA CAAU)	2+32+5
g	copy(224, 89, 12)	2+1+8+4
	insertion(U)	2+2+1
f	copy(237, 118, 8)	2+1+7+4
	insertion(UAGAA AAUUAGAACU UAAGUUAA)	2+46+5
e	copy(268, 190, 9)	2+1+7+4
	insertion(CAA GU)	2+10+3
d	copy(282, 244, 25)	2+1+6+5
	insertion(UUC UUGAU)	2+16+4
c	copy(315, 277, 9)	2+1+6+4
	insertion(UGGCAG AAAAGU)	2+24+4
b	copy(336, 298, 21)	2+1+6+5
	insertion(GAG AUGUCCUUUA UGGUGCAG)	2+42+5
		709

Fig. 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 weight (i.e. number of necessary bits for this operation).

for two factors $f = (p,q,l)$ and $g = (p',q',l')$, $f <_0 g$ holds if $p+l \leq p'$, i.e. if f strictly precedes g in the target sequence.

To search for all factors, we use the algorithm of Leung *et al.* which is able to discover exact but also point mutated repeats. To find factors, we apply this algorithm onto the text formed by the concatenation of *T, S* and the reverse-complement of *S*. We implemented the algorithm to allow substitutions inside factors without modifying the weight function, i.e. without adding a penalty. For simplicity, we describe it in the case of exact factors.

Script graph

As written above, a script is entirely specified by its copy operations (from now we do not distinguish copy and reverse copy and simply write 'copy'), i.e. when the set of its copied factors is known. Those factors in a script do not overlap in the target (see definition, condition (i)), this is a simple consequence of the construction process. Additionally, if many factors are concurrent for some segment of the target, i.e. if they are overlapping, at least one of them must be copied. In other words, an optimal script cannot 'forget' a factor that can be copied (condition (ii)). We introduce the definition of a *factor* script which is a script that fulfills those two properties.

Definition A *factor script FS* is a set of factors such that:
(i) for all $f,g \in FS$, $f <_0 g$ or $g <_0 f$
(ii) and for all f,g in FS such that $f <_0 g$ implies $\nexists\, h \in$ F–FS such that $f <_0 h <_0 g$.

We now define the script graph which is the basic structure for the computation of the transformation distance.

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

- $V =$ F $\cup\{A,Z\}$,
- $(f,g) \in E$ *iff*:
  (1) $f <_0 g$,
  (2) and $\nexists\, h \in$ F such that $f <_0 h <_0 g$,
- $(A,f) \in E$ *iff* $\nexists\, h \in$ F such that $h <_0 f$,
- $(f,Z) \in E$ *iff* $\nexists\, h \in$ F such that $f <_0 h$

The script graph is a directed graph where factors are the nodes. Two nodes joined by an edge represent successive copies in a script and their factors fulfill the $<_0$ relation. Each edge is oriented from the leftmost factor towards the rightmost one (we build the target sequence from left to right). Additionally, we define a source node *A* and a sink node *Z* which serve respectively as the beginning and end of any factor script. *A* can be viewed as the factor (0,0,0) and *Z* as the factor (|*T*|,|*S*|,0). A path from *A* to *Z* represents a selection of factors and as such defines a unique factor script.

In order to compute the description length of each script (path) we assign costs to edges and vertices. The cost of a vertex 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 on this path.

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

Sketch of the proof: each path of the graph is clearly a script because it combines copies and insertion operations such that the 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.

Time and space complexity

Computing the set of all factors with the Leung *et al.* (1991) algorithm is known to be efficient. Statistical studies have shown that its complexity increases almost linearly with the sequence lengths. Computing the shortest script is achieved in O(*Card*(*V*)+*Card*(*E*)) with the algorithm *Dag-Shortest-Path* presented in chapter 25.4 of Cormen *et al.* (1990). Let us denote *n* the length of longest sequence among the target sequence and the source sequence. There are at most n^3 segments between *S* and *T*, and therefore as much vertices in the script graph. As a complete graph over n^3 vertices has less than n^6 edges, the computation of the transformation distance requires less than O (n^6) units of time. This complexity is for worst cases and is a loose approximation. Nevertheless, in practice, the computation of the complete script graph is too inefficient to be applied on long sequences (more than 100 kb). It is the space requirement that prevents the computation.

Practicable algorithm

In practice, the size of the complete script graph grows dramatically for long sequences (of more than 100 kb), and particularly in the case of similar sequences because they share numerous factors. We studied the properties of copies operations that cannot belong to the minimal script. The corresponding factors can thus be removed from the graph. So without defining it here, we implement the *compact script graph* which encloses only 'interesting' copies and reverse copies. The above-mentioned properties and the definition of this compact graph cannot be detailed here for the sake of shortness. Moreover for implementation matter, only maximal factors, those that are not sub-factors of another factor, are included in the graph at run time. It is then possible to create their sub-factors only when necessary. Acceleration of

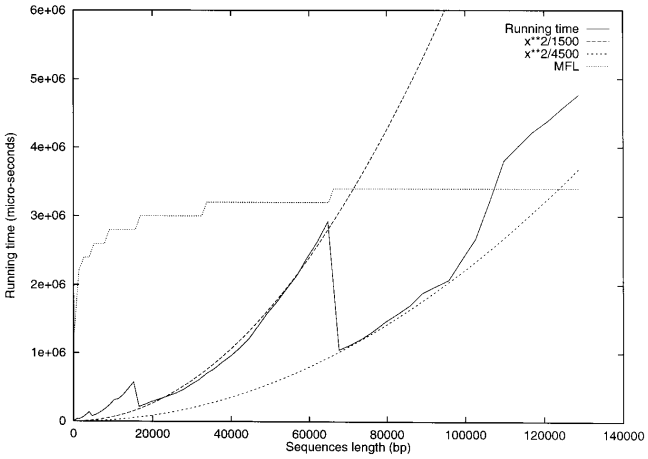


Fig. 2. Running times of the distance computation versus sequence length (on a PC Pentium 233 computer).

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 computing in little time the transformation distance on long sequences (more than 500 kb).

Figure 2 illustrates the variation of the running time in function of the sequence length. The source and target sequences are the tobacco's and rice's chloroplast genomes. To show the influence of the sequence length, we compute the TD for longer and longer prefixes of these sequences. For instance, the point of the curve with abscissa x = 80 000 corresponds to the running time for the prefixes of length 80 000 bps of the source and the target. 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 the sequences lengths (it is also plotted on the graphic). The vertical drops of the plain line curve denote 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. With the present implementation, the time requirement is short even for long sequences: around 5 s for 130 kb.

Implementation

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 visualize each comparison.

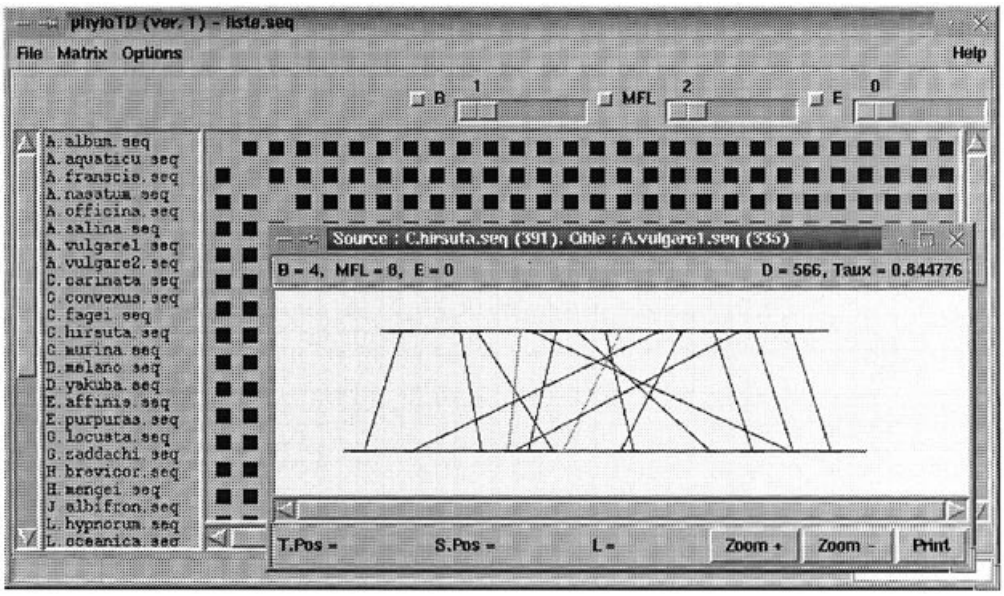


Fig. 3. Example of computation of the transformation distance for a set of sequences. One can display the actual distance and the corresponding minimal script between two sequences by clicking on the corresponding square in the matrix.

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 *HIV1* and *HIV2* are two clones of the HIV type 1 genome (accession numbers U34603 and U34604). *Bac. Subtilis 1* and *9* are parts of the bacillus subtilis complete genome (accession numbers Z99104 and Z99112). *C.E. 1* and *C.E. 2* are two clones of Caenorhabditis elegans (accession numbers Z98856 and Z92860). Running times have been computed on a PC Pentium 166MMX. A '–' indicates that the program exhausts the computer memory

		Number of factors		Running time (s)	
	Sequence length (kb)	SG	CSG	SG	CSG
Rice and tobacco	140	10 963	818	75	6.9
E. gracilis and *O. sinensis*	140	1683	209	18.4	16
HIV1 and HIV2	10	8578	111	–	7.6
Bac. Subtilis 1 and *Bac. Subtilis 9*	250	18 144	6	–	18
C.E. 1 and *C.E. 2*	630	20 172	5147	–	151

Results and discussion

We applied the TD to investigate the evolution of the TnT1 tobacco retrotransposon. The results illustrate the usefulness of the TD and its larger applicability compared to alignment methods: the TD allows finding 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 (Casacuberta *et al.*, 1995).

We computed all pairwise comparisons for the 140 sequences (the running time was less than 1 h). Figure 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 visualize some rearrangements events automatically, suggesting that the TD is a useful and complementary approach to classical alignment strategies.

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 distances as an alternative tool for the investigation of sequences relationships. Compared to alignments, our work shows that the concepts of the Algorithmic Information Theory may be useful to suggest 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.

The use of other weights and/or other sets of operations than those studied here yields variants of the TDs that may include biological knowledge. Thus, the general idea of our method is susceptible to various specifications to be explored.

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.

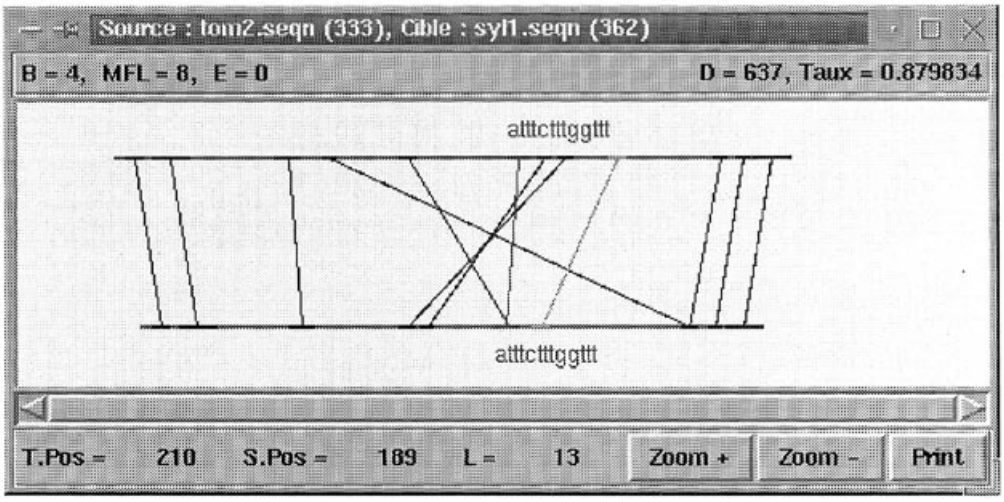


Fig. 4. Comparison of Tnt1 tobacco retrotransposon of tobacco tomentosiformis (as source) and tobacco sylvestris (as target).

Acknowledgments

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 Alice Michel (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 Anxolabéhére (Modélisation de la dynamique des populations d'éléments transposables, Dynamique du Génome et Evolution, Institut 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.

The authors also wish to thank the referees for their sharp and helpful comments.

References

Bafna,V. and Pevzner,P. (1993) Genome rearrangements and sorting by reversals. In *Proceedings of the 34th FOCS*, IEEE, pp. 148–157.

Bafna,V. and Pevzner,P. (1995) Sorting permutations by transpositions. In *6th Symposium on Discrete Algorithms SODA*, pp. 614–621.

Bafna,V. and Pevzner,P. (1998) Sorting by transpositions. *SIJDM: SIAM J. Discrete Math.*, **11.**

Casacuberta,J., Vernhettes,S. and Grandbastien,M.A. (1995) Sequence variability within the tobacco retrotransposon Tnt1 population. *EMBO J.*, **14,** 2670–2678.

Christie,D.A. (1996) Sorting permutations by block-interchanges. *Inf. Process. Lett.*, **60,** 165–169.

Cormen,T.H., Leiserson,C.E. and Rivest,R.L. (1990) *Introduction to Algorithms*. MIT Press.

Doolittle,R.F. (1981) Similar amino acid sequences: chance or common ancestry? *Science*, **214,** 149–159.

Ferretti,V., Nadeau,J.H. and Sankoff,D. (1996) Original synteny, combinatorial pattern matching. In: Hirschberg,D.S. and Myers,E.W. (eds), *7th Annual Symposium*, Lecture Notes in Computer Science, **1075,** 159–167. Springer, Berlin.

Grumbach,S. and Tahi,F. (1995) Compression et compréhension de séquences nucléotidiques. *Technique et Science Informatique*, **14,** 217–233.

Hannenhalli,S. and Pevzner,P. (1995) Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*. Las Vegas, Nevada, pp. 178–189.

Hannenhalli,S. (1996) Polynomial-time algorithm for computing translocation distance between genomes. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, Vol. 71.

Hillis,D.M., Moritz,C. and Mable,B.K. (1996) *Molecular Systematics.* Sinauer Associates Inc.

Kececioglu,J. and Sankoff,D. (1994) Efficient bounds for oriented chromosome inversion. In *5th Symposium on Combinatorial Pattern Matching*, pp. 307–325.

Kececioglu,J. and Ravi,R. (1995) Of mice and men: algorithms of evolutionary distances between genomes with translocations. In *6th Symposium on Discrete Algorithms SODA*, pp. 604–613.

Kolmogorov,A.N. (1965) Three approaches to the quantitative definition of information. *Probl. Inf. Transmiss.*, **1,** 1–7.

Leung,M.Y., Blaisdell,B.E., Burge,C. and Karlin,S. (1991) An efficient algorithm for identifying matches with errors in multiple long molecular sequences. *J. Molec. Biol.*, **221,** 1367–1378.

Li,W.H. and Graur,D. (1991) *Fundamentals of Molecular Evolution.* Sinauer Associates Inc.

Li,M. and Vitányi,P.M.B. (1997) *An Introduction to Kolmogorov Complexity and Its Applications.* 2nd Edn. Springer, New York.

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

Rissanen,J. (1989) *Stochastical Complexity and Statistical Inquiry.* World Scientific.

Rivals,E., Dauchet,M., Delahaye,J.P. and Delgrange,O. (1996) Compression and genetic sequences analysis. *Biochimie,* **78**.

Ruddle,F.H. (1997) Vertebrate genome evolution – the decade ahead. *Genomics*, **46,** 171–173.

Russel,R.B. (1994) Domain insertion. *Protein Eng.*, **7,** 1407–1410.

Sankoff,D. and Blanchette,M. (1998) Multiple genome rearrangements and breakpoint phylogeny. *J. Computat. Biol.*, **5,** 555–570.

Schöniger,M. and Waterman,M.S. (1992) A local algorithm for DNA sequence alignment with inversions. *Bull. Math. Biol.*, **54,** 521–536.

Varré,J.S., Delahaye,J.P. and Rivals,E. (1997) *The Transformation Distance.* Genome Informatics Workshop, Tokyo, Japan.

Waterman,M.S. (1995) *Introduction to Computational Biology.* Chapman and Hall.

Yockey,H.P. (1992) *Information Theory and Molecular Biology.* Cambridge University Press, Cambridge.