RAPPORT DE RECHERCHE

L R I

ALIGNMENT OF RNA SECONDARY STRUCTURES USING A FULL SET OF OPERATIONS

HERRBACH C / DENISE A / DULICQ S / TOUZET H

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud – LRI

06/2006

Rapport de Recherche N° 1451

CNRS – Université de Paris Sud
Centre d'Orsay
LABORATOIRE DE RECHERCHE EN INFORMATIQUE
Bâtiment 490
91405 ORSAY Cedex (France)

Alignment of RNA secondary structures using a full set of operations

Claire Herrbach[1,2], Alain Denise[1], Serge Dulucq[2], and Hélène Touzet[3]

[1] LRI Orsay, UMR CNRS 8623,
bât. 490, Université Paris-Sud 11 , F91405 Orsay cedex, France,
[2] LaBRI, UMR CNRS 5800,
Université Bordeaux I F33405 Talence cedex, France.
Claire.Herrbach@lri.fr
[3] LIFL, UMR CNRS 8022,
Université des Sciences et Technologies de Lille
bât. M3, F59655 Villeneuve d'Ascq cedex, France.

Research Report n° 1451 - June 2006

Abstract. We present a new alignment algorithm for pairwise comparison of RNA secondary structures represented as arc-annotated sequences. This allows to perform biologically relevant operations on RNA structures (as pairing or unpairing nucleotides) in a natural and realistic way. We describe three variants of the algorithm: global alignment, local alignment, and motif searching. All the variants run in $O(n^3)$ space and $O(n^4)$ time in the worst case, whereas the general edit distance is known to be NP-complete [2]. Incidentally, we improve the time complexity of the tree local alignment algorithm of Höchsmann *et al.* [9] by a factor n. Additionally, our experiments strongly suggest an average-case complexity in $\theta(n^2)$ for our algorithms. Finally, experiments on real RNA structures are reported.

Résumé. Nous présentons un nouvel algorithme d'alignement pour la comparaison deux à deux de structures secondaires d'ARN représentées par des séquences arc-annotées. Cet algorithme permet d'exécuter des opérations pertinentes d'un point de vue biologique sur des structures d'ARN (par exemple créer ou détruire un appariement de nucléotides) d'une manière naturelle et réaliste. Nous décrivons trois variantes de cet algorithme : l'alignement global, l'alignement local, et la recherche d'un motif. Toutes ces variantes s'exécutent en $O(n^3)$ en espace et $O(n^4)$ en temps dans le pire des cas, alors que le problème de distance d'édition est NP-complet [2]. Par ailleurs, nous améliorons la complexité en temps de l'algorithme d'alignement local d'arbres de Höchsmann *et al.* [9] d'un facteur n. De plus, nos expérimentations suggèrent fortement une complexité moyenne en temps de nos algorithmes en $\theta(n^2)$. Enfin, nous présentons en annexe des résultats d'expériences sur des structures d'ARN réelles.

1 Introduction

A RNA molecule is a linear polymer in which the nucleotides are linked together by means of phosphodiester bonds. Although generally an RNA molecule has only a single polynucleotide chain, it folds on itself to form spatial structures. Structural features of RNAs are important in the molecular mechanism involving their functions. The presumption, of course, is that to a preserved function there corresponds a preserved molecular conformation. Therefore the ability to compare RNA molecules is particularly useful. Since two different sequences can produce similar structures, comparison must take into account not only the sequence but also the structure of the molecule. Here we focus on the secondary structure, which is composed of the sequence together with the set of bonds connecting paired bases. Furthermore, we assume a model where there is no so-called *pseudoknot* in the secondary structure. Roughly, this means that for the secondary structure, the bonds are non-crossing.

The main objective of the paper is to present a new polynomial algorithm for comparing secondary structures. It involves realistic basic edit operations on RNA structures, such as pairing or unpairing bases, or altering an arc. Additionally, and contrary to current algorithms, the score of our edit operations are mutually independent. In Section 2, we first recall the main approaches used for comparing RNA structures. In Section 3, we present a new global alignment algorithm and we study its complexity. We notably prove that its worts-case time complexity is in $O(n^4)$, and experimental evidence strongly suggests that it runs in $\theta(n^2)$ in average. In Section 4 we develop some variants for local alignment and motif searching, whose complexity is the same as the global alignment. In section 5 we give some applications and we compare experimental results of our algorithm. Finally, we outline some perspectives in Section 6.

2 RNA structure comparison

There are two well acknowledged models for representing RNA structures: ordered rooted trees and arc-annotated sequences. We present both formalisms in further details.

From a historical perspective, RNA secondary structures were first encoded by labeled ordered rooted trees [19,16]. Figure 1 gives such an example. The tree representation relies on two main edit operations:

- Substitution: The node's label is renamed (mismatch), or possibly not (match).
- Insertion / Deletion: When a node is deleted, all its children become the children of its parent. Insertion is the symmetrical operation of deletion.

Each operation is given a score. Given two trees, the *edition* problem consists in finding the best scoring sequence of operations that changes one tree into the other one. The first efficient edition algorithm for ordered rooted trees is due to Zhang and Shasha [17]. It runs in $O(n^4)$ worst-case complexity, and in $O(n^3)$ average-case complexity [5], for two trees of size n. Some authors have given variants of the algorithm which improve the worst-case complexity [12,6]. Jiang *et al.* also introduced the *tree alignment* which is based on a common super-tree. The tree alignment algorithm runs in $O(n^4)$ worst-case complexity [11].

From a biological point of view, the evolutionary operations that are associated to the tree representation are not realistic. Indeed, there are some basic modifications in RNA structures that cannot be directly translated into a tree operation. For example, when comparing two RNA structures, it often happens that two nucleotides are paired in one structure and get unpaired in the other one. A likely explanation is that one of the two nucleotides has been mutated, so that they can be paired in the first structure but not in the second one. In the corresponding tree, no single operation can represent this simple evoluationary event: this should be done by deleting the node that corresponds to the pair, then inserting two new nucleotides. Figure 2 gives such an example. These observations lead to consider a richer model with more expressive edit operations on RNA structures: arc-annotated sequences, introduced initially in [7] and further

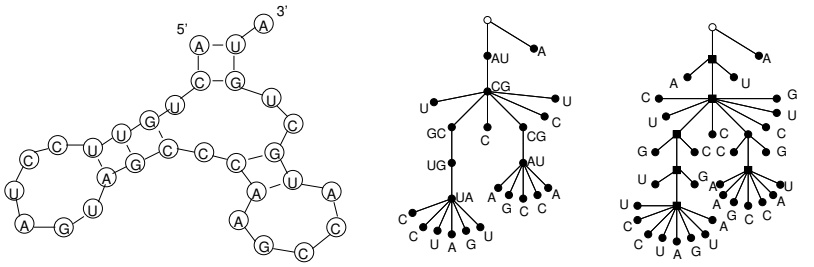


Fig. 1. A secondary structure without pseudoknot and its associated tree according to the classical representation (middle), and according to [9] (right).

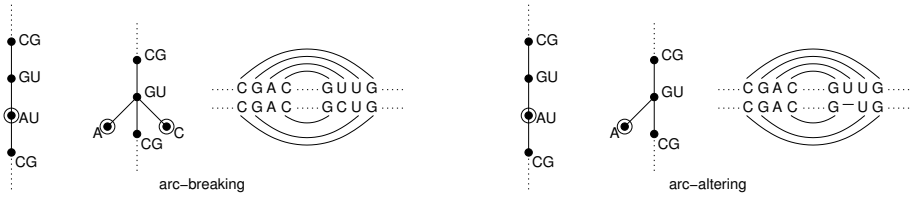


Fig. 2. Arc-breaking and arc-altering and the corresponding changes in a tree. Each of these operations corresponds up to 3 tree-editing operations

studied in [10,18]. The sequence is represented as is, and an arc is drawn between the two bases of any base-pair. Following [10], we consider the following edit operations.

- base-match, base-mismatch, arc-match, arc-mismatch which corresponds to the substitution of a node in the tree representation,
- base-deletion, base-insertion, arc-removing, which corresponds to the deletion or insertion of a node in the tree representation,
- arc-breaking: the arc is deleted but not its extremities,
- arc-altering: the arc is deleted with one of its extremities.

Arc-breaking and arc-altering are two new operations that have no simple counterpart in tree editing. Figure 2 gives an example of arc-breaking and arc-altering operations on trees and on arc-annotated sequences. The set of RNA secondary structures without pseudoknots exactly corresponds to the set of *nested* arc-annotated sequences, that is arc-annotated sequences without crossing arcs. The algorithmic complexity of the edition problem of two arc-annotated sequences has been studied in its full generality by Jiang *et al.* in [10]. They stated that the problem is NP-complete as soon as one allows crossing arcs in at least one of the arc-annotated sequences. Later, Blin *et al.* in [2] proved that the problem is NP-complete even if both arc-annotated sequences are nested. To circumvent this difficulty, several authors have recently investigated alternative solutions. Guignon *et al.* give a polynomial algorithm for simple non-branching stems, and then combine couples of matching stems into a tree [8]. Hochsmann *et al.* use a clever tree representation of RNA [9], which is implemented in the RNAForester software. Each pair of nucleotides is encoded by three nodes: an inner one which represents the arc and two leaves which represent the nucleotides (see Figure 1). Thus, the arc-breaking operation consists in deleting the inner node, and the arc-altering operation consists in deleting it and one of its children. Unfortunately, this approach has two drawbacks. First, it increases the size of the tree, by adding supplementary nodes. Secondly, the encoding involves some constraints in terms of relations between the edit operations. For example, the score of an arc-removing must be equal to the score of an arc-breaking plus the score of two base deletions. This can lead to unexpected edit scripts, such as depicted in Figure 3.

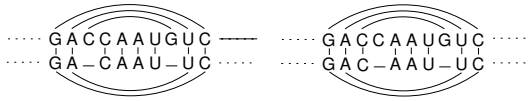


Fig. 3. The first alignment corresponds to the application of a single edit operation: one arc-removing. The second alignment results from three distinct evolutionary operations: one arc-breaking and two base deletions. For tree-derived models, such as RNAforester, those two alignments have the same score, whereas the first alignment is more relevant.

In this paper, we present a new way to compare arc-annotated sequences including arc-breaking and arc-altering operations that can apply to any kind of nested arc-annotated sequences. The idea is that alignment can be seen as a common super-structure that is obtained by adding bases and arcs in the two initial arc-annotated sequences. Formally, the problem is defined as follows:
Input: two nested arc-annotated sequences S_1 and S_2.
Question : finding the nested arc-annotated sequence S_3 of optimal score such that both S_1 and S_2 can be deduced from S_3 by a series of deletion and renaming operations. The score of S_3 is defined as the sum of the scores of edit operations necessary to transform S_3 into S_2 and the scores of edit operations necessary to transform S_3 into S_1.

For plain sequences (without arcs), this definition coincides with the usual alignment on strings. It can be seen as a generalization of the tree alignment of Jiang [11], with more flexibility in the edit operations. It should also be noticed that it encompasses the model of RNAforester: Each valid alignment provided by RNAforester is a valid alignment in this new scheme, since RNAforester is based on tree alignment. The difference is that arc-breaking and arc-altering operations are now defined as independent operations that do not result from combination of initial operations on nodes. Consequently, there is no restriction on the values of the scores of edit operations, as mentioned previously.

We shall show in the next section that the alignment problem for nested arc-annotated sequences can be solved in polynomial time. Furthermore, we achieve the same asymptotic complexity as for the tree alignment. This is a valuable advance compared to the NP-completeness of edit distance problem for the same edit operations. The table below summarizes the algorithmic complexities of comparing trees and nested arc-annotated sequences.

	Ordered rooted trees	Nested arc-annotated sequences
Edition	$O(n^3 \log n)$ [12]	NP-complete [2]
Alignment	$O(n^4)$ [11]	$O(n^4)$ **(this paper)**

Compared to the general edit distance, the alignment model for arc-annotated sequence yields the same restrictions as the tree alignment compared to the tree edit distance. Insertion operations on bases and on arcs should be performed before any deletion operation. In practice, when considering RNA structures, it amounts to prohibit internal rearrangements in the core of the structure when these rearrangements are not accompanied by subsequent changes in the surrounding stem loops. However, this kind of mutational event seems very unlikely. Figure 4 shows an example where edit distance and alignment give rise to two different mappings.

3 Global Alignment algorithm

3.1 Operation scheme

To compare RNA structures, we need a score system, or alternatively a distance, which measures the similarity (or the difference) between the structures. These two versions of the problem –score and distance– are equivalent. In the following, we will tackle it in terms of score. Ideally the score should fit with the evolutionary distance between the two molecules. The nearest the molecules are in an evolutionary point of view, the greatest the score must be.

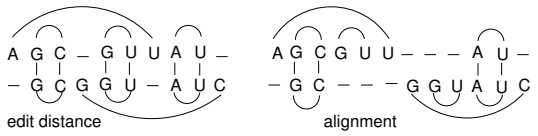


Fig. 4. Alignment versus edit distance. The picture on the left corresponds to an optimal mapping for the general edit distance, whereas the picture on the right corresponds to an optimal mapping for alignment. In the latter case, the substructure G-U is not recognized as a conserved substructure and has to be duplicated.

As mentioned before, we consider the following edit operations: base-match and base-mismatch, whose score is w_s, base-deletion and base-insertion, whose score is w_d, arc-match and arc-mismatch, whose score is w_m, arc-removing, whose score is w_r, arc-breaking, whose score is w_b, and arc-altering, whose score is w_a. For the sake of readability, the last three operations are slightly different from the usual ones defined in [10]. We consider that any operation on an arc concerns its incident bases too, and the operations and their associated scores are defined in such a way that any nucleotide is subject to one operation at most. Thus for example, in our scheme the arc-breaking operation may correspond up to three operations in the usual model: an arc-breaking and up to two base substitutions. And the score of an arc-breaking will be equal to the sum of the scores of the corresponding operations in the usual model. On the other hand, any operation of the usual model can be performed using one of our operations. Hence our model is exactly equivalent to the usual one: any sequence of operations can be translated from one model to the other, whatever the scoring scheme is.

3.2 Notation

Although the operations are defined on arc-annotated sequences, we choose, for the sake of readability, to write the algorithms in term of trees, using the usual representation detailed in Section 2. We write $\alpha(f)$ a tree which is composed of a root α and a subforest f. If the subforest in empty, the root is an isolated node denoted by b (as it represents an unpaired base). A forest is defined recursively by concatenating a tree and a forest: $\alpha(f) \circ t$ or $b \circ t$, where t is a forest and $\alpha(f)$ and b are trees as defined above.

3.3 Global alignment

Our algorithm is based on dynamic programming. In the following, we state the recurrences which enable to compute the alignment score of two trees or two subforests, as soon as the alignment scores of their respective subcomponents are known. Essentially, the general principle is the same as in [11], but several particular cases are needed for the arc-breaking and arc-altering operations: Isolated nodes within a subforest play a particular role because they can be involved in either of these operations.

$$
\begin{aligned}
&Align(\alpha(f), \alpha'(f')) \ = \\
&\max \begin{cases}
w_m(\alpha, \alpha') + Align(f, f') & \text{arc-(mis)match} \\
w_r(\alpha') + \max\{Align(u, f') + Align(v, \varepsilon) | u \circ v = \alpha(f)\} & \text{arc-removing} \\
w_r(\alpha) + \max\{Align(f, u') + Align(\varepsilon, v') |\ u' \circ v' = \alpha'(f')\} & \text{arc-removing}
\end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&Align(\alpha(f) \circ t, \alpha'(f') \circ t') \ = \\
&\max \begin{cases}
Align(\alpha(f), \alpha'(f')) + Align(t, t') & \text{arc-(mis)match} \\
w_r(\alpha') + \max\{Align(u, f') + Align(v, t') | u \circ v = \alpha(f) \circ t\} & \text{arc-removing} \\
w_r(\alpha) + \max\{Align(f, u') + Align(t, v') |\ u' \circ v' = \alpha'(f') \circ t'\} & \text{arc-removing} \\
w_a(\alpha, b') + \max\{Align(f, u') + Align(t, v') |\ u' \circ b' \circ v' = \alpha'(f') \circ t'\} & \text{arc-altering} \\
w_a(\alpha', b) + \max\{Align(u, f') + Align(v, t') |\ u \circ b \circ v = \alpha(f) \circ t\} & \text{arc-altering}
\end{cases}
\end{aligned}
$$

$$Align(b \circ t, \alpha'(f') \circ t') \;=$$
$$\max \begin{cases} w_d(b) + Align(t, \alpha'(f') \circ t') & \text{base-deletion} \\ w_r(\alpha') + \max\{Align(u, f') + Align(v, t') | u \circ v = b \circ t\} & \text{arc-removing} \\ w_b(\alpha', b, b_2) + \max\{Align(u, f') + Align(v, t') | u \circ b_2 \circ v = t\} & \text{arc-breaking} \\ w_a(\alpha', b) + \max\{Align(u, f') + Align(v, t') |\ u \circ v = t\} & \text{arc-altering} \\ w_a(\alpha', b_2) + \max\{Align(u, f') + Align(v, t') |\ u \circ b_2 \circ v = b \circ t\} & \text{arc-altering} \end{cases}$$

$$Align(\alpha(f) \circ t, b' \circ t') \;=$$
$$\max \begin{cases} w_d(b') + Align(\alpha(f) \circ t, t') & \text{base-deletion} \\ w_r(\alpha) + \max\{Align(f, u') + Align(t, v') | u' \circ v' = b' \circ t'\} & \text{arc-removing} \\ w_b(\alpha, b', b'_2) + \max\{Align(f, u') + Align(t, v') | u' \circ b'_2 \circ v' = t'\} & \text{arc-breaking} \\ w_a(\alpha, b') + \max\{Align(f, u') + Align(t, v') |\ u' \circ v' = t'\} & \text{arc-altering} \\ w_a(\alpha, b'_2) + \max\{Align(f, u') + Align(t, v') |\ u' \circ b'_2 \circ v' = b' \circ t'\} & \text{arc-altering} \end{cases}$$

$$Align(b \circ t, b' \circ t') \;=$$
$$\max \begin{cases} w_d(b) + Align(t, b' \circ t') & \text{base-deletion} \\ w_d(b') + Align(b \circ t, t') & \text{base-deletion} \\ w_s(b, b') + Align(t, t') & \text{base-(mis)match} \end{cases}$$

3.4 Trace

Let A and B be two trees. During the calculation of their alignment score, we fill a score matrix in which the cells are indexed by the pairs of subforests of A and B which are involved in the algorithm. At each step of the computation, we add in the matrix the score of the current comparison and a link towards the precedent score or the two precedent scores (depending on the chosen operation) which yield the current score. The score for aligning A and B is given by $Align(A, B)$. The alignment is given by a path in the matrix (the *trace*), that follows a set of those links between a score and his precedent score(s), beginning at $Align(A, B)$. Contrary to sequence alignment, where the trace is linear, here it is similar to a tree, where all the branches meet in $Align(\varepsilon, \varepsilon)$.

3.5 Worst-case complexity

Let us state some definitions. The *degree* of a node v, denoted d_v, is its number of children. The *width* of a forest f, denoted $w(f)$, is the number of trees it contains. A *closed subforest* is a subforest containing consecutive sibling trees, *i.e.* trees whose root nodes are consecutive brothers. A *complete subforest* is a forest containing all the subtrees which have the same parent. A *suffix subforest* is a closed subforest which contains the rightmost tree of a complete subforest. Given a tree T, we write $\mathrm{Closed}(T)$, $\mathrm{Suffix}(T)$, $\mathrm{Subtree}(T)$, respectively, for the set of closed subforests, the set of suffix subforests, the set of subtrees whose parent is the root of T. The two following lemmas are useful for the following.

Lemma 1. *Let A and B be two trees having, respectively, n_A and n_B nodes, ℓ_A and ℓ_B leaves. The pairs of forests appearing in the dynamic programming decomposition are exactly those of* $(\mathrm{Suffix}(A) \cup \mathrm{Subtree}(A)) \times \mathrm{Closed}(B)$ *plus those of* $\mathrm{Closed}(A) \times (\mathrm{Suffix}(B) \cup \mathrm{Subtree}(B))$.

This proves by induction on the sizes of A and B. For short, let us write $\mathcal{S}_T = \mathrm{Suffix}(T) \cup \mathrm{Subtree}(T)$ and $\mathcal{C}_T = \mathrm{Closed}(T)$.

Lemma 2. *For any tree T, we have*

$$|\mathcal{S}_T| \;=\; n_T + \ell_T - 1\,, \qquad |\mathcal{C}_T| \;=\; \textstyle\sum_{v \in T} \binom{d_v+1}{2}\,,$$
$$\textstyle\sum_{f \in \mathcal{S}_T} w(f) \;=\; \ell_T + \sum_{v \in T} \binom{d_v+1}{2}\,, \quad \sum_{f \in \mathcal{C}_T} w(f) \;=\; \sum_{v \in T} \binom{d_v+2}{3}\,.$$

The following proposition estimates the number of operations needed to compare two trees.

Proposition 1. *Let A and B be two trees having, respectively, n_A and n_B nodes, ℓ_A and ℓ_B leaves. The number of operations necessary to compute $Align(f,g)$ is a $\theta(N(n_A,n_B,\ell_A,\ell_B))$, where* $N(n_A,n_B,\ell_A,\ell_B) =$

$$\left(\sum_{v\in B}\binom{d_v+1}{2}\right)\left(\ell_A+\sum_{v\in A}\binom{d_v+1}{2}\right)+(n_A+\ell_A-1)\sum_{v\in B}\binom{d_v+2}{3}$$
$$+(n_B+\ell_B-1)\sum_{v\in A}\binom{d_v+2}{3}+\left(\sum_{v\in A}\binom{d_v+1}{2}\right)\left(\ell_B+\sum_{v\in B}\binom{d_v+1}{2}\right).$$

Sketch of proof. For each pair of subforests $(f,g)\in A\times B$, the number of operations neeeded to compute $Align(f,g)$ is a $\theta(w(f)+w(g))$. From Lemma 1, the number of operations needed to compute $Align(A,B)$ is $\theta(\sum_{(f,g)\in\mathcal{S}_A\times\mathcal{C}_B}(w(f)+w(g))+\sum_{(f,g)\in\mathcal{C}_A\times\mathcal{S}_B}(w(f)+w(g)))$, which is $\theta(|\mathcal{C}_B|\sum_{f\in\mathcal{S}_A}w(f)+|\mathcal{S}_A|\sum_{g\in\mathcal{C}_B}w(g)+|\mathcal{S}_B|\sum_{f\in\mathcal{C}_A}w(f)+|\mathcal{C}_A|\sum_{g\in\mathcal{S}_B}w(g))$. Applying Lemma 2 gives the result. □

Now we can state the worst-case complexity of the algorithm.

Theorem 1. *Let A and B be two trees having, respectively, n_A and n_B nodes. Let d_A and d_B be, respectively, the maximum degree of A and of B. Then the number of scores to be computed is $O(n_An_B(d_A+d_B)$ and the number of operations needed to compute them is $Align(f,g)$ is $O(n_An_B(d_A+d_B)^2)$.*

Sketch of proof. From Lemma 2 we get

$$|\mathcal{S}_T|\leq 2n_T\,,\qquad |\mathcal{C}_T|\leq\tfrac{n_T(d_T+1)}{2}\,,$$
$$\textstyle\sum_{f\in\mathcal{S}_T}w(f)\leq 2n_Td_T\,,\quad \sum_{f\in\mathcal{C}_T}w(f)\leq\frac{n_Td_T(d_T+1)}{2}\,.$$

Putting this in Lemma 1 and in Proposition 1 gives the result. □

This means that aligning trees with bounded degree, which is a reasonable assumption when working with RNA structures, yields an overall quadratic time complexity. The above corollary follows:

Corollary 1. *The space complexity of aligning two trees with n nodes is $O(n^3)$; the time complexity is $O(n^4)$.*

Thus the worst-case complexity is of the same order as the usual tree alignment algorithm [11].

Additionally, as announced is Section 4.1, we reduce by a factor n the time complexity of the local tree alignment algorithm given in [9], though we use a larger set of operations. Of course, reducing our set of operations as in [9] would give the same $O(n^4)$ time complexity.

3.6 Average-case complexity

We experimentally estimated the average complexity of the algorithm by randomly generating large trees and using Proposition 1. Thanks to the GenRGenS software [15], 1000 trees of each size n = 50, 150, 200, 250, ..., 2000 were generated uniformly and randomly, giving 500 pairs of random trees for each size. Then the number of operations needed by the algorithm was computed for each pair, according to Proposition 1, and its mean value was computed within each of the size 41 different sizes (including size 0). Results are given in the graph of Figure 5. We carried out two interpolation methods on these data: polynomial interpolation and least squares (with the Maple function CurveFitting[Interactive]). We made the hypothesis that the complexity would be between $O(n^2)$ and $O(n^4)$, and would possibly contain a $(\log n)^k$ factor. Our results strongly suggest that the average complexity is in $\theta(n^2)$. Indeed, the far best fit is got with $f(n) = 22.09717440n^2 - 67.224600n$, computed by polynomial interpolation on 3 experimental values. The maximum relative error between the values of this function and the 48 remaining experimental values is less than 6.10^-3. Intuitively, this result seems natural since the average degree of a node in a random tree is less than 2. Indeed, the number of trees of size $n+1$ is the Catalan number $C_n=\frac{1}{n+1}\binom{2n}{n}$ and the number of trees of size $n+1$ having k leaves is the Narayana number $N(n,k)=\frac{1}{n}\binom{n}{k}\binom{n}{k-1}$. The average number of leaves in a random tree is $\sum_k N(n,k)/C_n=\frac{n+1}{2}$. It remains $\frac{n+1}{2}$ inner nodes in average for n edges, hence the result.

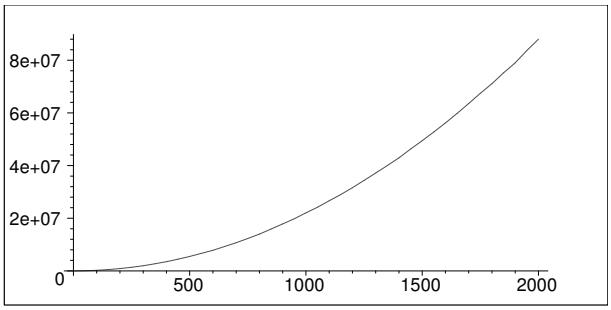


Fig. 5. Experimental average complexity on pairs of random trees according to their size.

4 Local alignment and motif searching

4.1 Local alignment

Local alignment aims at identifying conserved subregions between two RNA structures. Beyond this informal definition, it is subject to various interpretations depending on the shape of conserved substructures that should be highlighten. The most general definition is to search for any pair of *prefix subforests*. Prefix subforests are obtained as concatenation of connected subgraphs of closed subforests. In this context, an algorithm for local alignment can be derived from the main algorithm of Section 3 simply by retaining only positive or zero scores in the dynamic programming decomposition, in the same spirit as the Smith and Waterman local alignment on strings. Each case of the algorithm should be updated with extra value 0 in order to discard negative scores, like below:

$$Align(\alpha(f), \alpha'(f')) \ =$$
$$\max \begin{cases} 0 & \\ w_m(\alpha, \alpha') + Align(f, f') & \text{arc-(mis)match} \\ w_r(\alpha') + \max\{Align(u, f') + Align(v, \varepsilon) | u \circ v = \alpha(f)\} & \text{arc-removing} \\ w_r(\alpha) + \max\{Align(f, u') + Align(\varepsilon, v') |\ u' \circ v' = \alpha'(f')\} & \text{arc-removing} \end{cases}$$

An alternative approach for local similarity focuses on *complete substructures*. The local alignment between two trees is then given by the pair of closed subforests of maximal score. This definition, used in [9], is more appropriate to identify common motifs in large structures. From an algorithmic point of view, we already noticed that the global alignment algorithm does not visit all pairs of closed subforests (Lemma 1). So the recursive relationships should be modified to deal with local alignment. For that purpose, one can explicitely include all pairs of closed subforests in the computational process, such as in [9]. This solution gives rise to an increase of the complexity: the number of recursive calls is in $O(n^4)$ and the associated time complexity is in $O(n^5)$. We propose an alternative method that solves the local alignment problem without increasing the asymptotic time complexity. The principle is that any pair (f, g) of $\text{Closed}(A) \times \text{Closed}(B)$ may be obtained by truncating a pair of subforests of $\text{Closed}(A) \times \text{Suffix}(B)$. We introduce a supplementary table L and supplementary associate rules. For each pair (f, g) in $\text{Closed}(A) \times \text{Suffix}(B)$, $L(f, g)$ gives the optimal score between f and a prefix of g.

$$L(f, \varepsilon) = Align(f, \varepsilon)$$
$$L(\varepsilon, g) = 0$$

$$L(\alpha(f) \circ t, \alpha'(f') \circ t') \ =$$
$$\max \begin{cases} Align(\alpha(f), \alpha'(f')) + L(t, t') & \text{arc-(mis)match} \\ w_r(\alpha') + \max\{Align(u, f') + L(v, t') | u \circ v = \alpha(f) \circ t\} & \text{arc-removing} \\ w_r(\alpha) + \max\{Align(f, u') + L(t, v') |\ u' \circ v' = \alpha'(f') \circ t'\} & \text{arc-removing} \\ w_a(\alpha, b') + \max\{Align(f, u') + L(t, v') |\ u' \circ b' \circ v' = \alpha'(f') \circ t'\} & \text{arc-altering} \\ w_a(\alpha', b) + \max\{Align(u, f') + L(v, t') |\ u \circ b \circ v = \alpha(f) \circ t\} & \text{arc-altering} \end{cases}$$

$$L(b \circ t, \alpha'(f') \circ t') \;=$$
$$\max \begin{cases} w_d(b) + L(t, \alpha'(f') \circ t') & \text{base-deletion} \\ w_r(\alpha') + \max\{Align(u, f') + L(v, t') | u \circ v = b \circ t\} & \text{arc-removing} \\ w_b(\alpha', b, b_2) + \max\{Align(u, f') + L(v, t') | u \circ b_2 \circ v = t\} & \text{arc-breaking} \\ w_a(\alpha', b) + \max\{Align(u, f') + L(v, t') |\ u \circ v = t\} & \text{arc-altering} \\ w_a(\alpha', b_2) + \max\{Align(u, f') + L(v, t') |\ u \circ b_2 \circ v = b \circ t\} & \text{arc-altering} \end{cases}$$

$$L(\alpha(f) \circ t, b' \circ t') \;=$$
$$\max \begin{cases} w_d(b') + L(\alpha(f) \circ t, t') & \text{base-deletion} \\ w_r(\alpha) + \max\{Align(f, u') + L(t, v') | u' \circ v' = b' \circ t'\} & \text{arc-removing} \\ w_b(\alpha, b', b'_2) + \max\{Align(f, u') + L(t, v') | u' \circ b'_2 \circ v' = t'\} & \text{arc-breaking} \\ w_a(\alpha, b') + \max\{Align(f, u') + L(t, v') |\ u' \circ v' = t'\} & \text{arc-altering} \\ w_a(\alpha, b'_2) + \max\{Align(f, u') + L(t, v') |\ u' \circ b'_2 \circ v' = b' \circ t'\} & \text{arc-altering} \end{cases}$$

$$L(b \circ t, b' \circ t') \;=$$
$$\max \begin{cases} w_d(b) + L(t, b' \circ t') & \text{base-deletion} \\ w_d(b') + L(b \circ t, t') & \text{base-deletion} \\ w_s(b, b') + L(t, t') & \text{base-(mis)match} \end{cases}$$

Clearly, the time complexity remains in $O(n^4)$, as for the global alignment. As announced before, we reduce by a factor n the time complexity of the local tree alignment algorithm given in [9], though we use a larger set of operations. Of course, reducing our set of operations as in [9] would give the same $O(n^4)$ time complexity. The trace can be found in the same manner as for global alignment, yet beginning at the maximum score in the alignment score matrix.

4.2 Motif searching

The motif searching problem consists in aligning a whole small structure locally on a large structure. Like for the local alignment problem, we propose two methods for the motif searching problem. The first one searches for a pair of prefix subforests, while the second one searches for a pair of closed subforests. The first method is derived from the algorithm of global alignment, only by changing one case of the initialization: $M(\varepsilon, g) = 0$. For the other cases, $M(f, g) = Align(f, g)$. The alignment is the trace from $\max\{M(A, g) : g \in B\}$ to the first $M(\varepsilon, g)$ met. The second method is the local alignment detailed above, and the alignment is the same trace as the first method. So, for any pair (f, g) of Closed$(A) \times$ Closed(B), $M(f, g) = L(f, g)$. Obviously, the worst-case complexity of these two methods is the same as the other variants.

5 Experimental results

Our algorithms have been implemented in a prototype software. We have ran it on several RNases P, tRNAs and ribosomal RNAs, respectively taken from the RNase P database ([3]), the Genomic tRNA Database ([14]) and the Gutell Lab Comparative RNA Web Site ([4]). The alignments were compared with those given by RNAForester with the same parameter values. The resulting alignments are equivalent. We have produced some other alignements with different parameter values. Those results are in the appendix. On the other hand, we have compared the execution time of our prototype with RNAForester, on some RNA's of various sizes. Some representative results are given, in the table below (Figure 6).

6 Conclusion and Perspectives

We have presented an algorithm which ouputs the optimal (global, local or motif searching) alignment of two RNA secondary structures without pseudoknots, using a set of biologically relevant edit operations and mutually independent scores. The theoretical complexity results, as well as our first experiments on real data, show that the method performs well compared to the current ones.

	RNAforester	Our prototype
Two tRNAs *E. coli* alanine and leucine	0.061s	0.035s
Two RNase P *C. trachomatis* and *H. cutirubrum*	5.533s	2.966s
Two Group I introns *A. griffini* and *C. sorokiniana*	40.872s	21.758s
Two 16S RNAs *B. subtilis* and *E. coli*	30.997s	23.101s

Fig. 6. Comparison of execution time of our prototype with RNAforester

We plan to carry out a wider campaign of experiments. An important feature of our approach is that the scores of the operations can be set totally independently from one another. We are currently investigating the design of biologically relevant RNA scoring matrices (in a way similar to [13]) which should benefit from this property. Finally, we would like to compare our approach with the *multiscale* comparison algorithms developed by Allali and Sagot in [1]. It could be worth combining the two approaches, since they seem to be complementary.

Acknowledgement. We are grateful to Yann Ponty for his kind help. This work was partially supported by the projects "Navgraphe" of the "ACI Masses de Données", and "AReNa" of the "ACI IMPBio". Claire Herrbach is supported by a grant from the Conseil Régional d'Aquitaine.

References

1. J. Allali and M.-F. Sagot, "A new distance for high level RNA secondary structure comparison." *Transactions on Computational Biology and Bioinformatics*, vol. 2, no. 1, pp. 3–14, 2005.
2. G. Blin, G. Fertin, I. Rusu, and C. Sinoquet, "RNA sequences and the EDIT(NESTED,NESTED) problem," IRIN, Tech. Rep. RR-IRIN-03.07, 2003.
3. J. Brown, "The Ribonuclease P database," *Nucl. Acids Res.*, vol. 27, no. 1, p. 314, 1999.
4. J. Cannone, S. Subramanian, M. Schnare, J. Collett, L. D'Souza, Y. Du, B. Feng, N. Lin, L. Madabusi, K. Muller, N. Pande, Z. Shang, N. Yu, and R. Gutell, "The comparative rna web (crw) site: an online database of comparative sequence and structure information for ribosomal, intron, and other rnas," *Bioinformatics*, 2002.
5. S. Dulucq and L. Tichit, "RNA secondary structure comparison: exact analysis of the Zhang-Shasha tree edit algorithm." *Theor. Comput. Sci.*, vol. 306, no. 1-3, pp. 471–484, 2003.
6. S. Dulucq and H. Touzet, "Analysis of tree edit distance algorithms." in *Proc. 14th Annual Symposium on Combinatorial Pattern Matching (CPM 2003)*, 2003, pp. 83–95.
7. P. Evans, "Algorithms and complexity for annotated sequences analysis," Ph.D. dissertation, University of Victoria, 1999.
8. V. Guignon, C. Chauve, and S. Hamel, "An edit distance between RNA stem-loops," in *String Processing and Information Retrieval (SPIRE 2005)*, vol. 3772. Lecture Notes in Computer Science, 2005, pp. 335–347.
9. M. Höchsmann, T. Töller, R. Giegerich, and S. Kurtz, "Local similarity in RNA secondary structures." in *Proc. 2nd IEEE Computer Society Bioinformatics Conference (CSB 2003), 11-14 August 2003, Stanford, CA, USA*, 2003, pp. 159–168.
10. T. Jiang, G.-H. Lin, B. Ma, and K. Zhang, "A general edit distance between RNA structures." *Journal of Computational Biology*, vol. 9, no. 2, pp. 371–388, 2002.
11. T. Jiang, L. Wang, and K. Zhang, "Alignment of trees - an alternative to tree edit," *Theoretical Computer Science*, vol. 143, pp. 137–148, 1995.
12. P. N. Klein, "Computing the edit-distance between unrooted ordered trees." in *Proc. 6th Annual European Symposium on Algorithms (ESA '98)*, 1998, pp. 91–102.
13. R. J. Klein and S. R. Eddy, "Rsearch: Finding homologs of single structures rna sequences," *BMC Bioinformatics*, vol. 4, 2003.
14. T. M. Lowe and S. R. Eddy, "tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence," *Nucl. Acids Res.*, vol. 25, pp. 955–964, 1997.

15. Y. Ponty, M. Termier, and A. Denise, "Genrgens: software for generating random genomic sequences and structures," *Bioinformatics*, 2006, to appear.
16. B. A. Shapiro, "An algorithm for comparing multiple RNA secondary structures." *Computer Applications in the Biosciences*, vol. 4, no. 3, pp. 387–393, 1988.
17. K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems." *SIAM J. Comput.*, vol. 18, no. 6, pp. 1245–1262, 1989.
18. K. Zhang, L. Wang, and B. Ma, "Computing similarity between RNA structures." in *CPM*, ser. Lecture Notes in Computer Science, vol. 1645, 1999, pp. 281–293.
19. M. Zuker and D. Sankoff, "RNA secondary structures and their prediction," *Bull. Math. Biol.*, vol. 46, pp. 591–621, 1984.

A Alignments given by RNAforester and by our prototype

We have compared some tRNAs (appendix B), 5S RNAs (appendix C), RNases P (appendix D) and Group I introns (appendix E) with RNAforester, and with our prototype. For each of those RNA families, we have chosen a couple of secondary structures, with which we have run RNAforester with its default parameters, our prototype with scoring parameters of RNAforester, and our prototype with different scoring parameters. We present below three alignments for each family, their global optimal score and their execution time. The stars in alignments indicate arc-breaking and arc-altering operations.

B tRNAs

```
Alanine tRNA of E. coli, Leucine tRNA of E. coli, by RNAforester

Scoring type: similarity
Scoring parameters:
pm:   10
pd:   -5
bm:   1
br:   0
bd:   -10

Global optimal score: 111

Execution time: 0.061s

(((((((..((((........)-))).(((((.......))))).--.---.-.--(((((.......))))))))))))....
(((((((..(((...........))).(((((.......))))).(((....))))..(((((.......))))))))))))....
ggggcuauagcucagcugggag-agcgcuugcauggcaugcaagag--g---u-c--agcgguucgaucccgcuuagcuccacca
gccgaaguggcgaaaucgguagacgcaguugauucaaaaucaaccguagaaauacgugccgguucgaguccggccuucggcacca
            *        *                       * *    * *

------------------------------------------------------------------------------------------------

Alanine tRNA of E. coli, Leucine tRNA of E. coli, by our prototype, with scoring parameters of
RNAforester

Scoring parameters:
wd(b):  -10
ws(b,b'):  1 if b=b'
 0 if not
wr(p):  -25
wm(p,p'):  10
wb(p1-p2,b,b'):  -5+ws(p1,b)+ws(p2,b)
wa(p1-p2,b):  -5+wd(p1)+ws(p2,b) if p1 is deleted
 -5+wd(p2)+ws(p1,b) if p2 is deleted


Global optimal score: 111

Execution time: 0.035s

(((((((..((((........)-))).(((((.......))))).---...--.-.--(((((.......))))))))))))....
(((((((..(((...........))).(((((.......))))).(((....))))..(((((.......))))))))))))....
ggggcuauagcucagcugggag-agcgcuugcauggcaugcaag---agg--u-c--agcgguucgaucccgcuuagcuccacca
gccgaaguggcgaaaucgguagacgcaguugauucaaaaucaaccguagaaauacgugccgguucgaguccggccuucggcacca
            *        *                       * *    * *
```

Alanine tRNA of E. coli, Leucine tRNA of E. coli, by our prototype, with different scoring parameters

Scoring parameters:
wd(b): -8
ws(b,b'): 12 if b=b'
8 if not
wr(p): -16
wm(p,p'): 20 if p=p'
18 if one base is conserved and the other one is substituted
16 if the two bases are substituted
wb(p,b,b'): 5 if p=b-b'
1 if p=b-b" or if p=b"-b'
-3 if not
wa(p,b): -0.5 if p=b'-b and the left base of p is deleted
-0.5 if p=b-b' and the right base of p is deleted
-1.5 if not

Global optimal score: 629

Execution time: 0.035s

```
(((((((..((((........)-))).(((((.......))))).---....-----(((((.......))))))))))))....
(((((((..(((..........))).(((((.......))))).(((....))))..(((((.......))))))))))))....
ggggcuauagcucagcugggag-agcgcuugcauggcaugcaaga---gguc-----agcgguucgaucccgcuuagcuccacca
gccgaaguggcgaaaucgguagacgcaguugauucaaaaucaaccguagaaauacgugccgguucgaguccggccuucggcacca
            *        *
```

C 5S RNAs

```
Bacillus subtilis, Deinococcus Radiodurans, by RNAforester

Scoring type: similarity
Scoring parameters:
pm:   10
pd:   -5
bm:   1
br:   0
bd:   -10

Global optimal score: 335

Execution time: 0.121s

..(((((((-..-...(((((((....(((((((.............))))..)))...))))))).)).(((((((..(((((((...-)))))))
..((((((((((.....(((((((....(((((((.............))))..)))...))))))).)).(((((((..(((((((....)))))))
ugcuuggugg-cg-auagcgaagaggucacacccguucccauaccgaacacggaaguuaagcucuucagcgccgaugguagucggggguuu-cccccugu
acacccccgugcccauagcacuguggaaccaccccacccccaugccgaacugggucgugaaacacagcagcgccaaugauacucggaccgcagggucccgg
          *

..)))))))..-.-)))))))).-
..)))))))...)))))))))..
gagaguaggac-g-ccgccaagc-
aaaagucggucagcgcgggggguuu
           *

-------------------------------------------------------------------------------------------------------

Bacillus subtilis, Deinococcus Radiodurans, by our prototype, with scoring parameters of RNAforester

Scoring parameters:
wd(b):  -10
ws(b,b'):  1 if b=b'
 0 if not
wr(p):  -25
wm(p,p'):  10
wb(p1-p2,b,b'):  -5+ws(p1,b)+ws(p2,b)
wa(p1-p2,b):  -5+wd(p1)+ws(p2,b) if p1 is deleted
 -5+wd(p2)+ws(p1,b) if p2 is deleted


Global optimal score: 335

Execution time: 0.066s

.-.(((((((((..-...(((((((....(((((((.............))))..)))...))))))).)).(((((((..(((((((...-)))))))
..((((((((((.....(((((((....(((((((.............))))..)))...))))))).)).(((((((..(((((((....)))))))
U-GCUUGGUGGCG-AUAGCGAAGAGGUCACACCCGUUCCCAUACCGAACACGGAAGUUAAGCUCUUCAGCGCCGAUGGUAGUCGGGGGUUU-CCCCCUGU
ACACCCCCGUGCCCAUAGCACUGUGGAACCACCCCACCCCAUGCCGAACUGGGUCGUGAAACACAGCAGCGCCAAUGAUACUCGGACCGCAGGGUCCCGG
  *        *

..)))))))..-.)))))))).--
..)))))))...)))))))))..
GAGAGUAGGAC-GCCGCCAAGC--
AAAAGUCGGUCAGCGCGGGGGUUU
           *        *
-------------------------------------------------------------------------------------------------------
```

Bacillus subtilis, Deinococcus Radiodurans, by our prototype, with different scoring parameters

Scoring parameters:
wd(b): -8
ws(b,b'): 12 if b=b'
8 if not
wr(p): -16
wm(p,p'): 20 if p=p'
18 if one base is conserved and the other one is substituted
16 if the two bases are substituted
wb(p,b,b'): 5 if p=b-b'
1 if p=b-b" or if p=b"-b'
-3 if not
wa(p,b): -0.5 if p=b'-b and the left base of p is deleted
-0.5 if p=b-b' and the right base of p is deleted
-1.5 if not

Global optimal score: 1078

Execution time: 0.066s

```
..--(((((((((.....(((((((((....(((((((.............))))..)))...)))))))..)).(((((((..(((((((((...-)))))))))
..((((((((((.....(((((((((....(((((((.............))))..)))...)))))))..)).(((((((..(((((((((....)))))))))
UG--CUUGGUGGCGAUAGCGAAGAGGUCACACCCGUUCCCAUACCGAACACGGAAGUUAAGCUCUUCAGCGCCGAUGGUAGUCGGGGGUUU-CCCCCUGU
ACACCCCCGUGCCCAUAGCACUGUGGAACCACCCCACCCCAUGCCGAACUGGGUCGUGAAACACAGCAGCGCCAAUGAUACUCGGACCGCAGGGUCCCGG


..)))))))...)))))))))--.-
..)))))))...))))))))))..
GAGAGUAGGACGCCGCCAAG--C-
AAAAGUCGGUCAGCGCGGGGGUUU
```

D RNases P

```
Alcaligenes eutrophus, Streptomyces bikiniensis, by RNAforester

Scoring type: similarity
Scoring parameters:
pm:   10
pd:   -5
bm:   1
br:   0
bd:   -10

Global optimal score: 122

Execution time: 6.047s

((((((((((((((((((.((.(((((((((....)))))))))..))..............((((........(((((.((((.....)))).(((((....)
((((((((((((((((((.((.(((((((...--.)))))))..))..............((((........((((((((((.....)))))))))(((((....)
aaagcaggccaggcaaccgcugccugcaccgcaaggugcaggggggaggaaaguccggacuccacagggcaggguguuggcuaacagccauccacggcaac
cgagccgggcgggcggccgcgugggggucuuc--ggaccucccgaggaacguccgggcuccacagagcagggugguggcuaacggccacccggggugac
                              *    *                                        *             *

))).)((...((..............((((.(((((((((....)))))))))..)-)))-.------------------------------...----.---.
))).)((...((..............((-((((((((((....))))))))))..))).......((((((.......))))))..(((((((((....))))
gugcggaauagggccacagagacgagucuugccgccggguucgcccggcggga-agg-g------------------------------uga----a---a
ccgcgggacagugccacagaaaacagacc-gccggggaccucgguccucgguaagggugaaacggugguguaagagaccaccagcgccugaggcgacuca
                              **                    **                                  *  ***      ***

--.-..--.....)))))))))))))...((((..-..---.----------------.-((((((...-(((((.-....-))))))))))).....)))
))))........)))))))))))))...((((....(((((((((....)))))))))..((((((....((((((.....))))))))))).....)))
--c-gc--gguaaccuccaccuggagcaaucccaa-au---a---------------g-gcaggcgau-gaagcg-gccc-gcugagucugcggguaggg
ggcggcuagguaaaccccacucggagcaaggucaagaggggacaccccggugucccugcgcggauguucgagggcugcucgcccgaguccgcggguagac
  *                                  *   *          *   *                 *     *

).....(((((((((....)))))))).........)))))))..((.((((((((((....)))))))))..)).................).)))))))
).....(((((((((....)))))))).........)))))))..-((-(((..((((....))))-.)))-))--.............).)))))))
agcuggagccggcugguaacagccggccuagaggaaugguugucacgcaccguuugccgcaaggcgggcggggcgcacagaauccggcuuaucggccugc
cgcacgaggccggcggcaacgccggcccuagauggauggccgucg-cc-ccgacgaccgcgagguc-ccgg-gg--acagaacccggcguacagcccgac
                                                 **           **
)))....
)))....
uuugcuu
ucgucug

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

Alcaligenes eutrophus, Streptomyces bikiniensis, by our prototype, with scoring parameters of RNAforester

Scoring parameters:
wd(b): -10
ws(b,b'): 1 if b=b'
0 if not
wr(p): -25
wm(p,p'): 10
wb(p1-p2,b,b'): -5+ws(p1,b)+ws(p2,b)
wa(p1-p2,b): -5+wd(p1)+ws(p2,b) if p1 is deleted
-5+wd(p2)+ws(p1,b) if p2 is deleted

Global optimal score: 122

Execution time: 3.364s

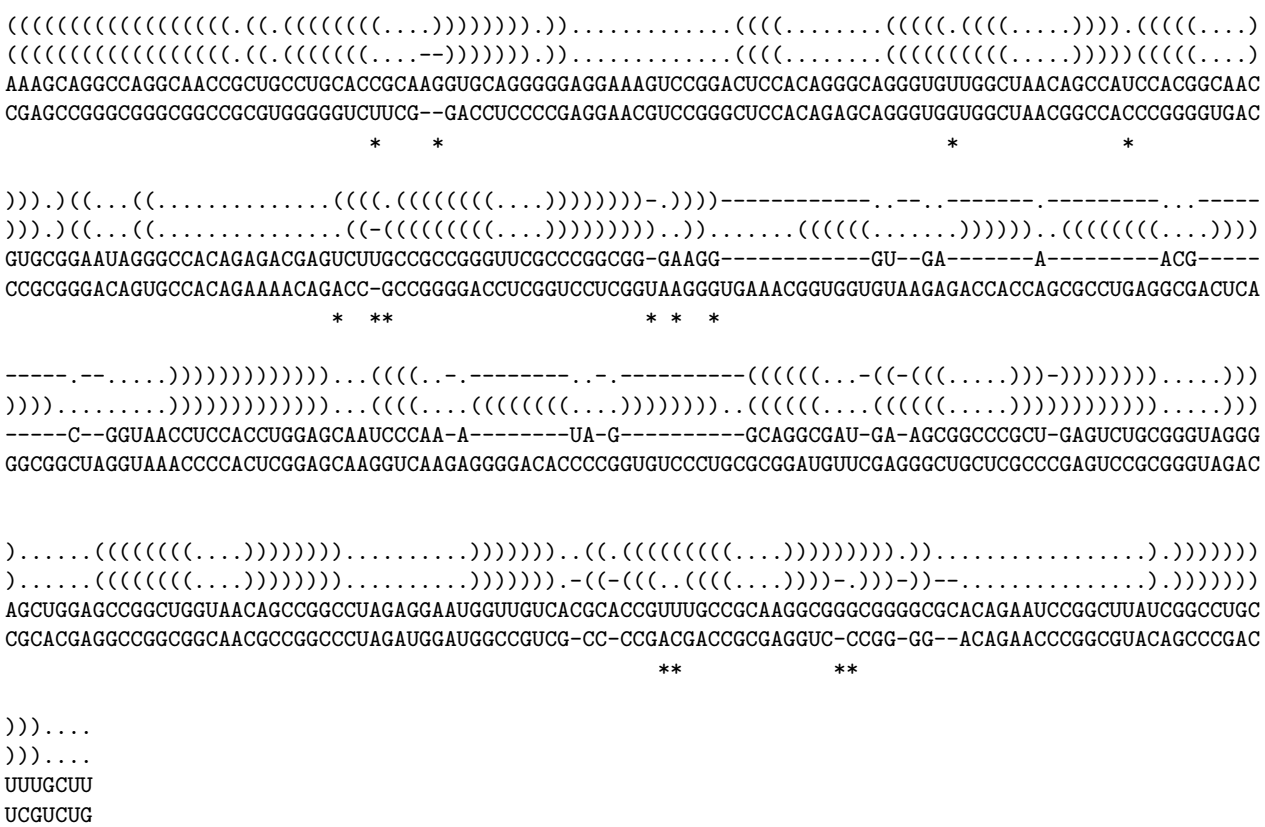
```
((((((((((((((((((.((.(((((((((....)))))))))..)).............((((........(((((.((((.....)))).(((((....)
((((((((((((((((((.((.(((((((...--.)))))))..)).............((((........((((((((((.....)))))))))(((((....)
AAAGCAGGCCAGGCAACCGCUGCCUGCACCGCAAGGUGCAGGGGGAGGAAAGUCCGGACUCCACAGGGCAGGGUGUUGGCUAACAGCCAUCCACGGCAAC
CGAGCCGGGCGGGCGGCCGCGUGGGGGUCUUC--GGACCUCCCCGAGGAACGUCCGGGCUCCACAGAGCAGGGUGGUGGCUAACGGCCACCCGGGGUGAC
                              *    *                                        *            *

))).)((...((..............((((.(((((((((....)))))))))..)-)))----------...-..---.-------..-.----.-.--.-.
))).)((...((..............((-((((((((((....))))))))))..))).......((((((.......))))))..(((((((((....))))
GUGCGGAAUAGGGCCACAGAGACGAGUCUUGCCGCCGGGUUCGCCCGGCGGGA-AGG----------GUG-AA---A-------CG-C----G-G--U-A
CCGCGGGACAGUGCCACAGAAAACAGACC-GCCGGGGACCUCGGUCCUCGGUAAGGGUGAAACGGUGGUGUAAGAGACCACCAGCGCCUGAGGCGACUCA
                              **                  **             **       **      ** ** *      * *

-------.-----)))))))))))))...((((..------...--------.-------((((((...--(((((.....)))))-))))))).....)))
))))........)))))))))))))...((((....(((((((((....)))))))))..((((((....((((((.....))))))))))))).....)))
-------A-----CCUCCACCUGGAGCAAUCCCAA------AUA-------G-------GCAGGCGAU--GAAGCGGCCCGCUGA-GUCUGCGGGUAGGG
GGCGGCUAGGUAAACCCCACUCGGAGCAAGGUCAAGAGGGGACACCCCGGUGUCCCUGCGCGGAUGUUCGAGGGCUGCUCGCCCGAGUCCGCGGGUAGAC
* **                                    ***      ***

)......(((((((((....)))))))).........)))))))..((.((((((((((....)))))))))).))................).)))))))
)......(((((((((....)))))))).........)))))))..-((-(((..((((....))))-.)))-))--.............).)))))))
AGCUGGAGCCGGCUGGUAACAGCCGGCCUAGAGGAAUGGUUGUCACGCACCGUUUGCCGCAAGGCGGGCGGGGCGCACAGAAUCCGGCUUAUCGGCCUGC
CGCACGAGGCCGGCGGCAACGCCGGCCCUAGAUGGAUGGCCGUCG-CC-CCGACGACCGCGAGGUC-CCGG-GG--ACAGAACCCGGCGUACAGCCCGAC
                                                  **           **

)))....
)))....
UUUGCUU
UCGUCUG
```

Alcaligenes eutrophus, Streptomyces bikiniensis, by our prototype, with different scoring parameters

Scoring parameters:
wd(b): -8
ws(b,b'): 12 if b=b'
8 if not
wr(p): -16
wm(p,p'): 20 if p=p'
18 if one base is conserved and the other one is substituted
16 if the two bases are substituted
wb(p,b,b'): 5 if p=b-b'
1 if p=b-b" or if p=b"-b'
-3 if not
wa(p,b): -0.5 if p=b'-b and the left base of p is deleted
-0.5 if p=b-b' and the right base of p is deleted
-1.5 if not

Global optimal score: 2641

Execution time: 3.364s



E Group I introns

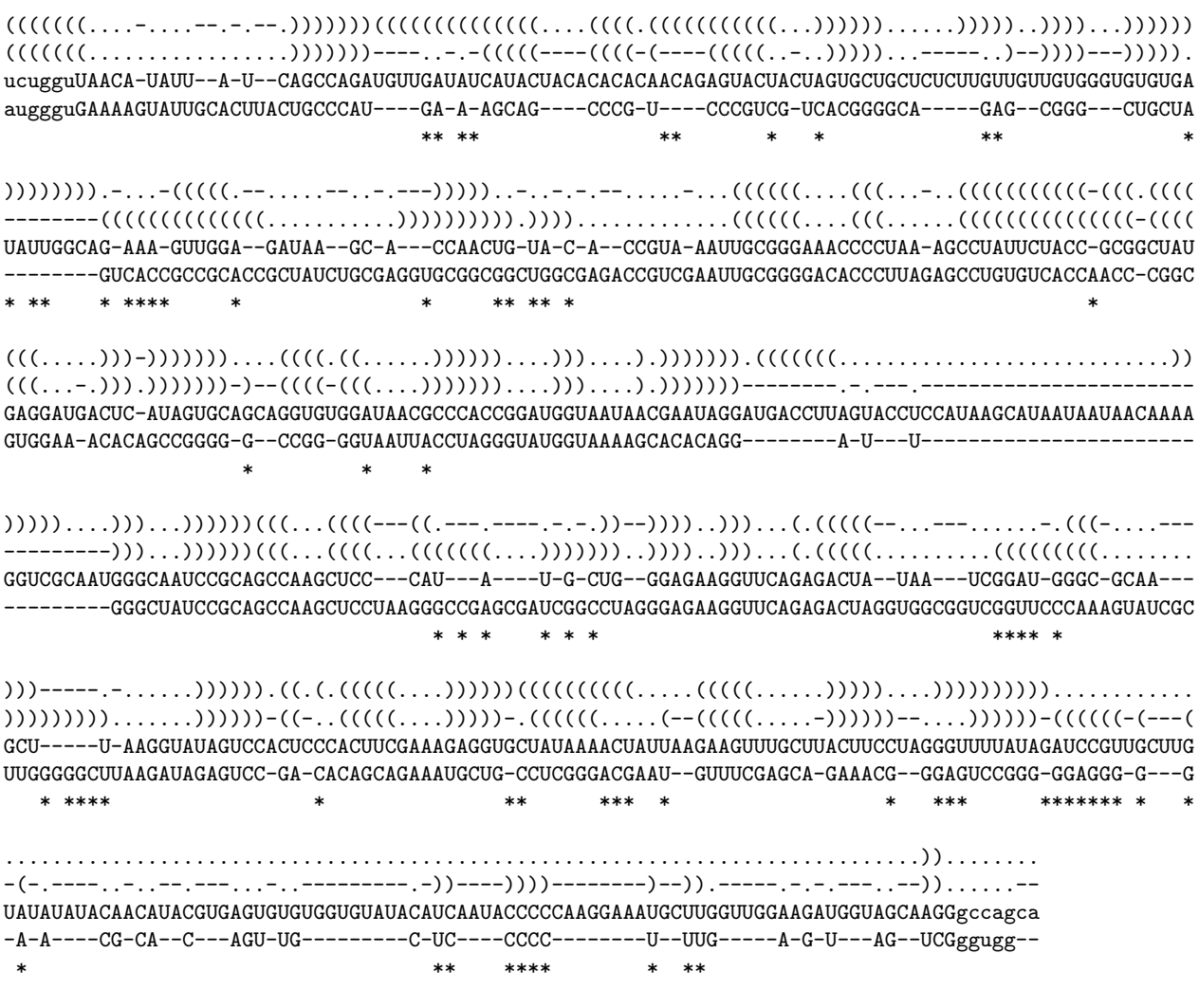
```
Acanthamoeba griffini, Chlorella sorokiniana, by RNAforester

Scoring type: similarity
Scoring parameters:
pm:   10
pd:   -5
bm:   1
br:   0
bd:   -10

Global optimal score: -1196

Execution time: 40.872s

(((((((....-....--.-.--.)))))))((((((((((((((....((((.((((((((((...))))))......))))))..))))...))))))
(((((((.................)))))))-...((((((((((-------(-((((--------------.-.-..---))))--).-....))))))
ucugguuaaca-uauu--a-u--cagccagauguugauaucauacuacacacacaacagaguacuacuagugcugcucucuuguuguuguggguguguga
augggugaaaaguauugcacuuacugcccau-gaaagcagcccgu-------c-ccgu--------------c-g-uc---acgg--gg-cagagcgggc
                              ****             * *       *           *              * *

))))))))....(((((--.--.....--..-.-----)))))-.-.-.........-...((((((....(((...-..((((((((((-(((.((((
))))---.((((((((((((((.....)))))))))).))))..............((((((....(((......((((((((((((((-((((
uauuggcagaaaguugg--a--gauaa--gc-a-----ccaac-u-g-uacaccgua-aauugcgggaaaccccuaa-agccuauucuacc-gcggcuau
ugcu---agucaccgccgcaccgcuaucugcgaggugcggcggcuggcgagaccgucgaauugcggggacacccuuagagccugugucaccaacc-cggc
    ********       *                        ****                                        *

(((.....)))-)))))))....((((.((......))))))....)))....).)))))))..(((((((.............................))
(((...-.))).)))))))-)--((((-(((....)))))))....)))....).)))))))..----------------------------------
gaggaugacuc-auagugcagcagguguggauaacgcccaccggaugguaauaacgaauaggaugaccuuaguaccuccauaagcauaauaauaacaaaa
guggaa-acacagccgggg-g--ccgg-gguaauuaccuaggguaugguaaaagcacacaggau----------------------------------
                     *         *    *                         *

)))))....)))...)))))))(((...((((---((.---.----.--..))--))))..)))...(.(((((--...---.......(((--....---
--------.)))...)))))))(((...((((...(((((((....)))))))..))))..)))...(.(((((..........(((((((((........
ggucgcaaugggcaauccgcagccaagcucc---cau---a----u--gcug--ggagaaggguucagagacua--uaa---ucggaugggc--gcaa---
--------ugggcuauccgcagccaagcuccuaagggccgagcgaucggccuagggagaaggguucagagacuagguggcggucgguucccaaaguaucgc
    *                              **  *    *  **                                 *****

-)))-----.......))))))..((.(.(((((....))))))((((((((((.....(((((......)))))....))))))))))............
))))))))).......))))))-((..-(((((....)))))-.((((((.....(--(((((.....-))))))--....))))))-----(--(--((
-gcu-----uaagguauaguccacucccacuucgaaagaggugcuauaaaacuauuaagaaguuugcuuacuucccuagggguuuuauagauccguugcuug
uuggggcuuaagauagagucc-gaca-cagcagaaaugcug-ccucgggacgaau--guuucgagca-gaaacg--ggaguccggg-----g--g--ag
    *****                *              **      ***  *                 *   ***      *    *  *  **

.....................................................................))........
-(((((-...-..--.---...-..---------.-))----))))--------)--)).-----.-.-.---..--)).....--
uauauauacaacauacgugaguguguggguguauacaucaauacccccaaggaaaugcuugguuggaagaugguagcaagggccagca
-gggga-acg-ca--c---agu-ug---------c-uc----cccc--------u--uug-----a-g-u---ag--ucgggugg--
 *****                           **    ****        *  **

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

Acanthamoeba griffini, Chlorella sorokiniana, by our prototype, with scoring parameters of RNAforester

Scoring parameters:
wd(b): -10
ws(b,b'): 1 if b=b'
0 if not
wr(p): -25
wm(p,p'): 10
wb(p1-p2,b,b'): -5+ws(p1,b)+ws(p2,b)
wa(p1-p2,b): -5+wd(p1)+ws(p2,b) if p1 is deleted
-5+wd(p2)+ws(p1,b) if p2 is deleted

Global optimal score: -1196

Execution time: 21.758s



Acanthamoeba griffini, Chlorella sorokiniana, by our prototype, with different scoring parameters

Scoring parameters:
wd(b): -8
ws(b,b'): 12 if b=b'
8 if not
wr(p): -16
wm(p,p'): 20 if p=p'
18 if one base is conserved and the other one is substituted
16 if the two bases are substituted
wb(p,b,b'): 5 if p=b-b'
1 if p=b-b" or if p=b"-b'
-3 if not
wa(p,b): -0.5 if p=b'-b and the left base of p is deleted
-0.5 if p=b-b' and the right base of p is deleted
-1.5 if not

Global optimal score: 1896

Execution time: 21.758s