Flexible identification of structural objects in nucleic acid sequences: palindromes, mirror repeats, pseudoknots and triple helices

Marie-France Sagot * Alain Viari †

Abstract

This paper presents algorithms for flexibly identifying structural objects in nucleic acid sequences. These objects are palindromes, mirror repeats, pseudoknots and triple helices. We further explore here the idea of a model against which the words in a sequence are compared for finding these structural objects [17]. In the present case, models are words defined over the alphabet of nucleotides that have both direct and inverse occurrences in the sequence. Moreover, errors (substitutions, deletions and insertions) are allowed between a model and its inverse occurrences. Helix stems may therefore present bulges or interior loops, and mirror repeats need not be exact. Reasonably efficient performance comes from the fact that the parts composing the structures are kept separated until the end and that filtering for valid occurrences (occurrences that may form part of such a structure) can be done in $O(n)$ time where n is the length of the sequence. The time complexity for the searching phase (that is, before the structural parts are put together at the end) of the algorithms presented here (one for palindromes and mirror repeats, the other for pseudoknots and triple helices) is then $O(nk(e+1)(1+min\{(d_{max}-d_{min}+1+e)^p, k^e \mid \Sigma \mid^e\}))$ where n is the length of the sequence, d_{max} and d_{min} are, respectively, the maximal and minimal length of a hairpin loop, k is either the maximum length k_{max} of a model, is a fixed length or represents the maximum value of a range of lengths, e is the maximum number of errors allowed (substitutions, deletions and insertions), p is equal to 1 for palindromes and mirror repeats, 2 for triple helices and 3 for pseudoknots, and $\mid \Sigma \mid$ is the size of the alphabet of nucleotides.

keywords : nucleic acid sequence, nucleic structural object, palindrome, mirror repeat, pseudoknot, triple helix, approximate comparison, model, direct occurrence, (complementary) inverse occurrence.

1 Introduction

We present in this paper algorithms for flexibly identifying structural objects in nucleic acid sequences. These objects are palindromes, mirror repeats, pseudoknots and triple helices. The reason why these structural units have been chosen rather than other ones is that they are among the simplest ones present in DNA or RNA and that the algorithms for locating them are very similar.

Palindromes and mirror repeats are the simplest structures of the four, and represent the basic elements composing a helix in RNA (palindromes) or DNA (mirror repeats). The palindromes considered here are of course not lexical palindromes but biological ones and correspond to two segments of RNA that complement one another in inverse order [11]. The accepted complementary base pairs are the Watson-Crick A-T(U), C-G pairs, although the G-T(U) pair could also be considered. Mirror repeats are inverse exact (i.e. not complementary) repeats and may form part of triple helices in DNA [14]. The third part of the triple helix belongs to the other strand and links itself with both repeats. Palindromes represent secondary structures that are not always local as the distance between the two halves of the palindrome may be very big. Mirror repeats that may belong to a triple helix in DNA are less well known objects.

*Institut Gaspard Monge, Université de Marne-la-Vallée, 2, rue de la Butte Verte, 93160 - Noisy-le-Grand
†Atelier de BioInformatique, Universit de Paris VI, 12, rue Cuvier, 75005 - Paris.

Pseudoknots and triple pairs on the other hand form part of the tertiary structure of a nucleic acid sequence. A pseudoknot in a folded RNA molecule has a hairpin loop plus a single strand folded back to form base pairs with the bases in the loop [7] [8] [16]. A base triple is an approximately planar group of three bases involving at least one hydrogen bond joining each pair. A triple helix is a contiguous series of such base triples.

Algorithms for finding palindromes follow in general a naïve approach [23] or restrict themselves to exact comparisons [9] [12]. When only mismatches are allowed, the first method has a time complexity that is $O(nk(d_{max} - d_{min} + 1))$ where n is the length of the sequence, d_{max} and d_{min} are, respectively, the maximal and minimal length of a hairpin loop and k is the length of the helix stems composing the structure. This last value has to be established beforehand. The exact method has a complexity in $O(n \log k + n(d_{max} - d_{min} + 1))$ where k is, in this case, the maximum length k_{max} of the helices or is a fixed length. The first part of the formula corresponds to the search for the halves of potential helices, the second to the process of putting the halves together. Algorithms for locating pseudoknots on the other hand are usually thermodynamic simulation or grammar-based approaches, either general [5] [10] [21] [22] [25] [26] or specifically designed for finding pseudoknots [1] [4] [13]. When an analysis of the complexity of such searches is given, it usually appears to be high even though these methods are heuristics ($O(n^3)$ for [4]). Slightly better time complexities seem to be obtained by the approaches that do not allow for deletions and insertions. Indels are not believed to exist in pseudoknots, but this may simply be because pseudoknots with them have seldom been looked for. Finally, locating mirror repeats or triple helices seems to have been a far less studied problem.

Our main aim in this paper is to try to fill some of these gaps by proposing algorithms that are reasonably efficient and flexible in their definition of what constitutes structures such as these. The algorithms presented are not predictive in the sense that we cannot say which of the potentially structural objects found will actually be part of the final, global structure of the molecule, we just find <u>all</u> those that may do so. These algorithms, in particular the pseudoknots and triple helices finder, may be used independently but they are meant to form part of a more general program of global structure prediction or identification such as the one given in [2] or [3]. The first one is a database scanning tool based on a descriptive grammar of more complex structural objects. The second is a structure prediction algorithm.

This paper is a continuation of some of our previous ones [17] [18] [19] [20] in the sense that the problem of errors will be dealt with the help of an object called a model against which the comparisons are made. In this case, a model is either a word or a pair of words over the same alphabet as that of the sequences that corresponds exactly to one half of the structures we search for. This half is the second part of either one (palindromes, mirror repeats) or two (pseudoknots, triple helices) helix stems. It corresponds only approximately to the other half of the structures.

However, we present two innovations in relation to the previous work. The first one is the fact that a model may now be "read" in a sequence in both directions. We therefore introduce the idea of a direct and of an inverse occurrence. The second innovation is that we manipulate here representatives of equivalence classes of the occurrences of a model m instead of all occurrences as was done before. This allows us to obtain a better performance in theory as well as in practice. It also means we may find a solution space that is slightly larger than the one actually searched for, but the latter one may be efficiently recovered afterwards.

Importantly also, by not putting together the diverse parts of a structural object until the end, we are able, during the whole phase of the search for the structures, to depend not on the number of such objects there is in a sequence, but instead on the number of parts that compose them. In the case of exact palindromes for instance, an upper bound for their number is $O(n^2)$ where n is the length of the sequence, while the total number of the parts composing them is bounded over by $O(n)$.

Where errors are allowed, we are then able to propose algorithms that find all the parts composing the structural objects mentioned above in $O(nk(e+1)(1 + min\{(d_{max} - d_{min} + 1 + e)^p, k^e \mid \Sigma \mid^e\}))$ where n, d_{max} and d_{min} are as before, k is in this case either the maximum length k_{max} of a model, is a fixed length or represents the maximum value of a range of lengths, e is the maximum number of errors allowed (substitutions, deletions and insertions), p is equal to 1 for palindromes and mirror repeats, 2 for triple helices and 3 for pseudoknots, and $\mid \Sigma \mid$ is the size of the alphabet of nucleotides. Putting the parts together requires $O(N)$ time, where N is the number of possible structural objects and is majored by $O(n(d_{max} - d_{min} + 1)^p)$. This majoration is reached only for very degenerated sequences (as, for instance, a sequence composed of n As in the case of mirror repeats and for $d_{max} - d_{min} + 1 = n$). An $O(n \log k(e+1)(1 + min\{(d_{max} - d_{min} + 1 + e)^p, k^e \mid \Sigma \mid^e\}))$ version of the algorithm for the first searching phase is possible but may not be very interesting in practice when errors are permitted and so has not been implemented.

This paper is organized as follows. In section 2, we present the problem of finding palindromes and mirror repeats. Basic definitions are given, then we introduce the algorithm and a sketch of its complexity. In section 3, we show how to extend the algorithm to deal with the more complicated structures composing a pseudoknot or a triple helix. We end by a discussion on the limitations of the approach given here for treating nucleic structural problems of a more general nature and give some indications on how we propose to extend this work in the future.

2 Palindromes and Mirror Repeats

2.1 Basic Definitions

Let Σ be the alphabet of nucleotides, that is, $\Sigma = \{$A, C, G, T or U$\}$ and let a sequence s be an element of Σ^*. A word u of length k is an element of Σ^k for $k \geq 1$ and u is said to be a word in s if $s = xuy$ with $x, y \in \Sigma^*$.

Definition 2.1 *Given $u = s_i...s_j$ a word in s, we call i the start position of u in s and j its end position.*

Definition 2.2 *Given two words $u = u_1...u_k$ and $v = v_1...v_l$, we say that u is equal to v (denoted by $u = v$) if $k = l$ and $u_i = v_i$ for $1 \leq i \leq k$.*

Definition 2.3 *Given a word $u = u_1...u_k \in \Sigma^k$, we call inverse of u the word $\overline{u} = u_k...u_1$.*

Definition 2.4 *Given $u = s_i...s_j$ a word in s, we call j the start_inverse position and i the end_inverse position of its inverse $\overline{u}$ in s.*

Notation 2.1 *We denote by $\mathcal{M}_c$ the 4 × 2 matrix of the nucleotides complementary base pairs. We have:*

$\mathcal{M}_c =$

A	*T (U)*
C	*G*
G	*C*
T (U)	*A*

where (A, T) and (C,G) are the Watson-Crick base pairs.

As mentioned in the introduction, we could add to this table the (G, T or U) pair. Of course, this would increase the number of potential structures.

Definition 2.5 *Given a word $u = u_1...u_k \in \Sigma^k$ and a matrix $\mathcal{M}_c$ of complementary base pairs, we call complementary inverse of u the word $\overline{u_c} = \mathcal{M}_c(u_k)... \mathcal{M}_c(u_1)$.*

Observe that if the pair (G, T or U) is added to the definition of $\mathcal{M}_c$, a word could have more than one complementary inverse.

Definition 2.6 *Given non-negative integers e, d_{min} and d_{max}, we call $(u, v, e, d_{min}, d_{max})$ an approximate palindrome (resp. mirror repeat) in s if:*

- *u and v are words in s;*

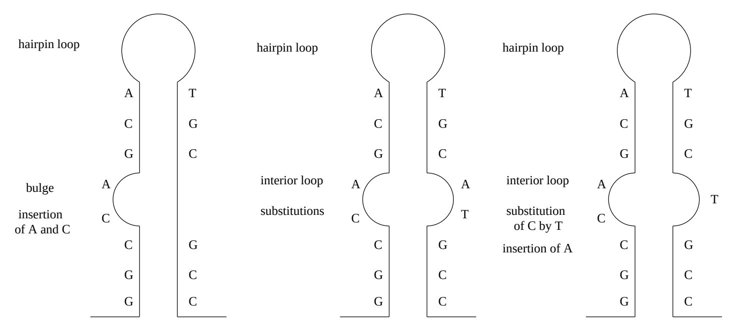

Figure 1: Mismatches and deletions/insertions corresponding to bulges and interior loops.

- *$dist_L(u, \overline{v_c})$ (resp. $dist_L(u, \overline{v})$) is no more than e where $dist_L(x, y)$ is the Levenshtein (or edit) distance between x and y (it is the minimum number of substitutions, deletions and insertions necessary to convert x into y);*
- *$d_{min} \le d \le d_{max}$ where d is the distance between the end position f_v of v and the start position i_u of u in s ($d = i_u - f_v + 1$).*

If $e = 0$, we have of course an exact palindrome (resp. mirror repeat).

Mismatches in a palindrome or repeat correspond to interior loops, and deletions and insertions to bulges. The segment of DNA or RNA sequence between the two halves of a palindrome or mirror repeat represent the hairpin loops (see figure 1).

Definition 2.7 *A model m of length k is an element of Σ^k.*

Definition 2.8 *Given a sequence s and non-negative integers e, d_{min} and d_{max}, we say that m is an approximate palindromic (mirror) model present in s and satisfying the constraints (e, d_{min}, d_{max}) if there exist words u and v in s that verify:*

- *$u = m$;*
- *$(u, v, e, d_{min}, d_{max})$ is an approximate palindrome (resp. mirror repeat) in s.*

From $dist_L(u, \overline{v_c}) \le e$ (resp. $dist_L(u, \overline{v}) \le e$), we deduce $dist_L(m, \overline{v_c}) \le e$ (resp. $dist_L(m, \overline{v}) \le e$).

We say that u is an (exact) direct occurrence of m in s and v an approximate complementary inverse (resp. inverse) occurrence of m.

Example 2.1 *See figure 2.*

Observe that the fact that direct occurrences are always exact and start after the complementary inverse (resp. inverse) occurrences with which they form a palindrome (resp. mirror repeat) does not add any further constraint on the structural objects that the algorithm is able to find than the ones introduced by e, d_{min} and d_{max}. It just allows for a simpler definition of the objects searched for and, subsidiarily, to an easier analysis of the algorithm's complexity.

Definition 2.9 *We note $OD(m)$ the set of all direct occurrences of a palindromic (resp. mirror) model m present in s and $OI(m)$ its set of complementary inverse (resp. inverse) occurrences.*

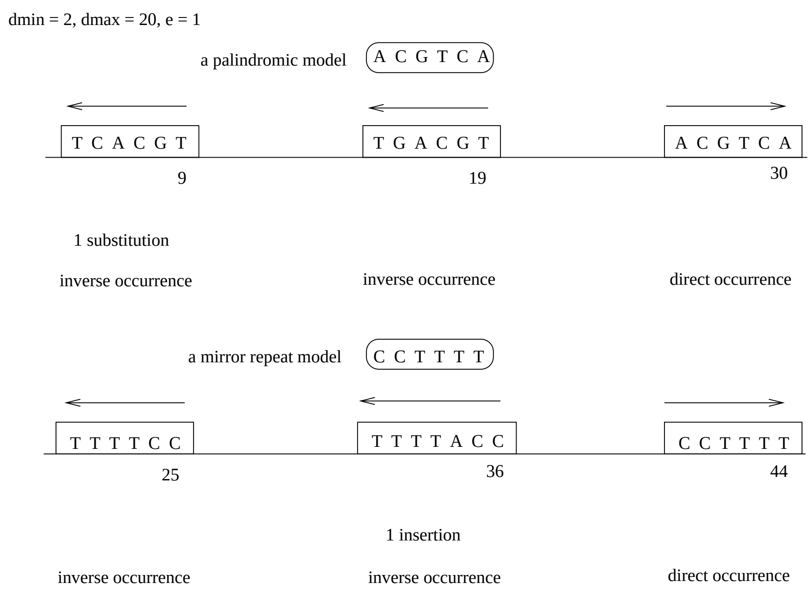

Figure 2: Examples of palindromic and mirror models.

With both palindromic and mirror models, we need not have $OD(m) \cap OI(m) = \emptyset$. For instance, mirror model $m =$ AAA is present in sequence $s =$ AAAAAAAAA four times with the constraints $(e = 0, d_{min} = 0, d_{max} = 0)$ (at positions 4, 5, 6 and 7) and the word AAA starting at position 4 is both a direct and an inverse occurrence of m. In the same way, palindromic model $m =$ AGCT which is twice present in $s =$ AGCTAGCTAGCT with the same constraints (at positions 5 and 10) has the word AGCT starting at position 5 as both a direct and a complementary inverse occurrence.

We may note already that, as announced in the introduction, the parts of the structural objects searched for are not kept "assembled" in the sets of occurrences of a model, but are instead kept separated until the end. In the exact case for instance ($e = 0$), keeping them assembled would require $O(n^2)$ space in the worst case while keeping them separated requires at most $2n$ space. Note however that, although the parts of a structure are kept apart, only valid ones are preserved. If a word u in s is such that $u = m$ for m a model, u will be stocked in $OD(m)$ only if there exists v in s such that $(u, v, e, d_{min}, d_{max})$ is a palindrome or a mirror repeat, and vice-versa. What is even more important to observe is that checking for this validity takes only $O(n)$ time as we show later on.

We can then now state the problem we wish to solve in the following way:

The Palindrome or Mirror Repeat Problem *Given a sequence s and non-negative integers e, d_{min} and d_{max}, the problem we propose to solve is to find all approximate palindromic models (resp. mirror models) present in s that satisfy the constraints (e, d_{min}, d_{max}).*

2.2 Algorithm

2.2.1 Main idea

The main idea of the algorithm is to progressively construct models having at least one direct occurrence and one approximate complementary inverse (resp. inverse) occurrence that form a palindrome (resp. mirror repeat) verifying the constraints (e, d_{min}, d_{max}). To simplify matters, we shall henceforward treat only the case of mirror repeats with the understanding that palindromes are dealt with in exactly the same way except for the introduction of a matrix of complementary bases.

The construction of models is done by increasing length in a way that is similar to what was done in [17] [18] [19] [20]. There are now two differences though. The first one is that, as we saw, a word that is an occurrence of a model is related not just to the model, but also to at least one other occurrence of the same model. The relation of an occurrence to a model is either an exact match or an upper bounded Levenshtein distance while the relation to another occurrence is positional (it is the distance between the end position of an occurrence and the start position of the other). A second difference in relation to the construction of models as done in our previous papers is that we introduce here the idea of an equivalence relation between inverse occurrences. Before we do that however, we formally introduce the rules of model construction.

2.2.2 Rules of Model Construction

We start by introducing a new definition and two notations.

Definition 2.10 *We say that a position i in a sequence s represents a possible occurrence u (resp. $\overline{u}$) of a model m if u (resp. $\overline{u}$) is an exact match of m (resp. is at a distance at most e from m). We note $POD(m)$ and $POI(m)$ the sets of possible occurrences (direct or inverse respectively) of m.*

The rules of construction are then given by the following lemmas. As usual, let s be a sequence of length n and m a model of length k.

For now an occurrence $s_i ... s_f$ (direct or inverse) of a model m is identified by:

- its end position f in s if it is a direct occurrence;
- its start and end positions (i, f) and its number of errors num_errors against the model otherwise (notice that we must have $num\_errors \leq e$ for any valid occurrence).

An example is given in figure 3.

Lemma 2.1 *$f \in POD(m' = m\alpha)$ with $\alpha \in \Sigma$ if, and only if:*

- $f - 1 \in OD(m)$
- $f \leq n$ *and* $s_f = \alpha$

Lemma 2.2 *$(i, f, num\_errors) \in POI(m' = m\alpha)$ with $\alpha \in \Sigma$ if, and only if, at least one of the following conditions is verified:*

(match)	$(i+1, f, num\_errors) \in OI(m)$, $i \geq 1$ *and* $s_i = \alpha$
(substitution)	$(i+1, f, num\_errors - 1) \in OI(m)$, $i \geq 1$ *and* $s_i \neq \alpha$
(deletion)	$(i, f, num\_errors - 1) \in OI(m)$
(insertion)	$\exists g > 0$ *(g is the minimal value for which this is true) such that* $(i+g+1, f, num\_errors - g - 1) \in OI(m)$, $i \geq 1$ *and* $s_i \neq \alpha$ *or* $(i+g+1, f, num\_errors - g) \in OI(m)$, $i \geq 1$ *and* $s_i = \alpha$

Lemma 2.3 *$f_d \in OD(m' = m\alpha)$ if, and only if, $f_d \in POD(m')$ and $\exists$ $(i, f_i, num\_errors) \in POI(m')$ such that $d_{min} \leq f_d - k - f_i \leq d_{max}$*

Conversely:

$(i, f_i, num\_errors) \in OI(m' = m\alpha)$ if, and only if, $(i, f_i, num\_errors) \in POI(m')$ and $\exists$ $f_d \in POD(m')$ such that $d_{min} \leq f_d - k - f_i \leq d_{max}$

An illustration is given in figure 3.

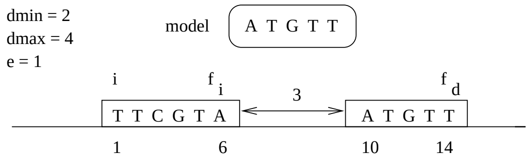

Figure 3: An illustration of lemma 2.3.

2.2.3 Equivalence Classes of Occurrences

The idea of an equivalence relation on the occurrences of a model will concern here only inverse occurrences because this relation is trivially true for direct ones.

Let us consider the following model m = ACCAGTG and let us suppose that $e = 2$ and that GTGAACGA is a word starting at position i and ending at position f in s. Its inverse is AG-CAAGTG. If having constructed model m up to length 4, that is we have constructed the model m' = ACCA, we identify the inverse occurrences by their end positions, that is, their start_inverse positions (see definition 2.4), we need to keep all those whose start_inverse positions are the same but which have different lengths because we do not know which ones will be extendable later on. If instead of that, we identify the inverse occurrences by their end_inverse positions, we need to keep only one occurrence per position even when several have the same end_inverse position. The one kept is the one having less errors against the model and the reason why preserving this one is enough is because if it cannot be extended, none of the other occurrences whose end_inverse position is the same may be extended either. Figure 4 gives an example of that. We may therefore establish the following equivalence relation E on the set of inverse occurrences of a model m. Given two words $u = u_i ... u_f$ and $v = v_k ... v_l$ in s that belong to $OI(m)$, we say that u is equivalent to v, noted $u \equiv v$, if, and only if, $i = k$. We identify an equivalence class of E by the common end_inverse position i of its elements and we represent it by the occurrence with end_inverse position i for which $dist_L(\overline{u}, m)$ is minimal among all $v \in OI(m)$ having same end_inverse position. We note $\overline{OI(m)}$ the quotient $OI(m)/\equiv$ and $\overline{POI(m)}$ the quotient $POI(m)/\equiv$.

Occurrences are therefore now identified by:

- their end positions f in s if they are direct occurrences;
- the end_inverse position i of any element of a class of inverse occurrences and the start_inverse position f and number of errors num_errors of the representative of the class against the model otherwise (once again, we have $num\_errors \leq e$ for valid occurrences).

An example is given in figure 5.

In this case, Lemmas 2.2 and 2.3 above become:

Lemma 2.4 $(i, f, num\_errors) \in \overline{POI(m' = m\alpha)}$ *with* $\alpha \in \Sigma$ *if, and only if:*

- *at least one of the following conditions is verified:*
  - **(match)** $(i+1, f, num\_errors) \in \overline{OI(m)}$, $i \geq 1$ *and* $s_i = \alpha$
  - **(substitution)** $(i+1, f, num\_errors - 1) \in \overline{OI(m)}$, $i \geq 1$ *and* $s_i \neq \alpha$
  - **(deletion)** $(i, f, num\_errors - 1) \in \overline{OI(m)}$
  - **(insertion)** $\exists g > 0$ *(g is the minimal value for which this is true) such that*
    $(i + g + 1, f, num\_errors - g - 1) \in OI(m)$, $i \geq 1$ *and* $s_i \neq \alpha$
    or
    $(i + g + 1, f, num\_errors - g) \in OI(m)$, $i \geq 1$ *and* $s_i = \alpha$

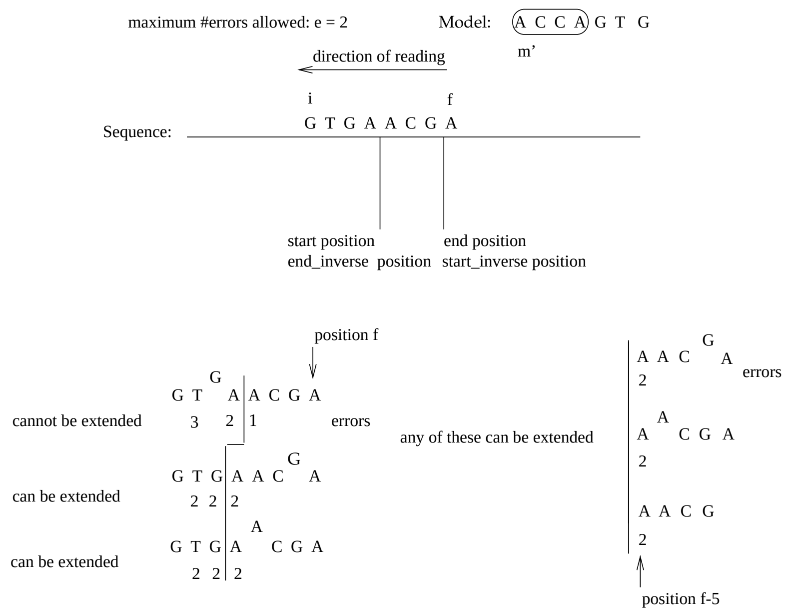

Figure 4: An example that shows why keeping end positions instead of start ones leads to no loss of occurrences.

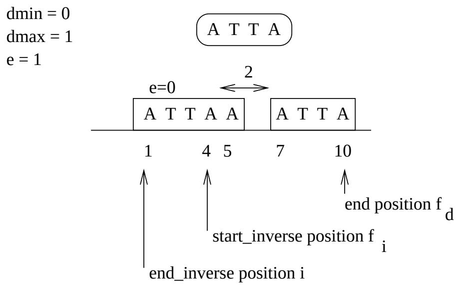

Figure 5: An example of why the representative of a class of inverse occurrences may not be the one that keeps verifying the constraint on the length of the hairpin loops.

- (i, f, num_errors) *is the occurrence with end_inverse position* i *whose Levenshtein distance from* m *is minimal*

Lemma 2.5 $f_d \in OD(m' = m\alpha)$ *if, and only if,* $f_d \in POD(m')$ *and* $\exists\ (i, f_i,\ num\_errors) \in \overline{POI(m')}$ *such that* $d_{min} - num\_errors \leq f_d - k - f_i \leq d_{max} + num\_errors$

Conversely:

$(i, f_i, num\_errors) \in \overline{OI(m' = m\alpha)}$ *if, and only if,* $(i, f_i, num\_errors) \in \overline{POI(m')}$ $\exists\ f_d \in POD(m')$ *such that* $d_{min} - num\_errors \leq f_d - k - f_i \leq d_{max} + num\_errors$

The need to consider a larger interval (its length is increased by $2 \times num\_errors$ for each occurrence) for the length of the hairpin loops comes from the fact that the representative chosen for a class of inverse occurrences may not be the one that keeps verifying the constraint on the length of these loops. An example is given in figure 5. Taking as representative the one presenting less errors against the model and enlarging the interval allows us to miss no valid occurrence. It may preserve a few that are not. These may be eliminated at the end of the algorithm as will be explained when we discuss the overall algorithm's complexity.

To simplify, we call from now on the double inequality $d_{min} - num\_errors \leq f_d - k - f_i \leq d_{max} + num\_errors$ the (d_{min}, d_{max}) constraint.

2.2.4 Filtering for Palindromes or Mirror Repeats

Applying the rules given by lemmas 2.1 and 2.4 is relatively straightforward and will not be further detailed here except to say that finding the representative (i, f, num_errors) of a class of inverse occurrences can be done in $O(e+1)$ time, or $O(1)$ time if bit vectors are employed to represent the $(e+1)$ occurrences of a same class. We shall now say a word about how we can apply the rule given by lemma 2.5 in an efficient way.

The point to notice is that the sets $OD(m)$ and $\overline{OI(m)}$ are kept permanently ordered (this follows naturally from the way models are constructed). The sets $POD(m')$ and $\overline{POI(m')}$ obtained by an application of lemmas 2.1 and 2.4 are therefore ordered too. Filtering them to eliminate direct or inverse occurrences that do no longer form part of a structure can then be done in time linear with the sum of the cardinality of both sets. An idea of the algorithm that performs such filtering is given in figure 6. One can compare this to the operation of emptying two bottles of water in synchrony. When the level of the water in one bottle stops at a certain point, it is because the level in the other bottle is going down, and vice versa. The total time needed to empty both bottles is then proportional to the quantity of water they contain. Such a simple operation is enough because we just need to flag those occurrences (direct or inverse) that form part of at least one mirror repeat or palindrome, we do not need to establish with what other occurrences it does so. The only delicate point arises when the level of water in the second bottle (corresponding to $POD(m')$) goes down a little too far. This happens when the last occurrence of $POD(m')$ is successful (i.e. it is flagged) while the current one is not (because $(f_d - k - f_i) > d_{max} + num\_errors$). In this case, one must temporarily stop emptying $POD(m')$ and go back to emptying $\overline{POI(m')}$. However, the next occurrence of $\overline{POI(m')}$ must be tested for validity not against the next or the current one in $POD(m')$ but against the last occurrence considered before the current one if it was successful or against the current one otherwise. The function RestoreLevel($POD(m')$) shown in figure 6 serves therefore the purpose of getting the level in $POD(m')$ back to the "right" place. It is not further described here. In terms of complexity, this means an occurrence of $POD(m')$ may be considered more than once during the filtering operation. However, for each element of $\overline{POI(m')}$ we do not get back more than two elements in $POD(m')$, the delay is therefore of at most 2 for each element of $\overline{POI(m')}$ and the final time complexity remains proportional to n.

2.3 Sketch of Complexity

We sketch here the complexity of the algorithm. The recursive form used for constructing the models leads to a formulation of its time complexity that is also recursive. The question we must therefore ask in order to find it is, how many operations do we have to do to construct the models of length $k+1$ from those of length k?

Let $POD(m' = m\alpha)$ and $\overline{POI(m' = m\alpha)}$ be the sets of direct and inverse occurrences of the model m' we are in the process of constructing. These sets are ordered by increasing position in s.

1: get the first element (i, f_i, num_errors) of $\overline{POI(m')}$
2: do
3: get the next element f_d of $POD(m')$
4: if the corresponding occurrences satisfy the (d_{min}, d_{max}) constraint
5: flag them as valid occurrences
6: until $(f_d - k - f_i > d_{max} + num\_errors$ or no element f_d is available$)$
7: RestoreLevel$(POD(m'))$
8: get the next element (i, f_i, num_errors) of $\overline{POI(m')}$
9: if none is available
10: stop
11: else goto **2**

The sets $\overline{OI(m')}$ and $OD(m')$ are then the elements of $\overline{POI(m')}$ and $POD(m')$ flagged as valid occurrences.

Figure 6: Filtering to keep only the parts that can form a structure.

Since a palindromic or mirror model m of length k must have at least one direct occurrence, there can be at most $O(n)$ models of length k. Conversely, these $O(n)$ models have at most a total of $O(n)$ direct occurrences between them. Inverse occurrences on the other hand may belong to more than one model. Given one such occurrence, the number of models it may belong to is bounded over by $min\{d_{max} - d_{min} + 1 + e, k^e \mid \Sigma \mid^e\}$. The first term $(d_{max} - d_{min} + 1 + e)$ comes from the fact that a model must have an exact direct occurrence "close" enough to the inverse one. The second term $(k^e \mid \Sigma \mid^e)$ represents the models in the neighboorhood [15] of the inverse occurrence. The time needed to build the models of length $k+1$ from those of length k is therefore proportional to:

$$(\underbrace{n}_{1} + \underbrace{n\, min\{d_{max} - d_{min} + 1 + e, k^e \mid \Sigma \mid^e\}}_{2}\underbrace{(e+1)}_{3}) \times \underbrace{3}_{4}$$

where

- 1 is the total number of direct occurrences of all models of length k
- 2 is the total number of inverse occurrences of all models of length k
- 3 is the number of possible ways of extending an inverse occurrence (a direct occurrence can be extended in only one way)
- 4 comes from the fact that occurrences have to be extended (one time the first term of the product) and then to be filtered to determine which ones can be part of a structure (2 times that term)

The total construction takes then $O(nk(e+1)(1 + min\{d_{max} - d_{min} + 1 + e, k^e \mid \Sigma \mid^e\}))$ time. Since a single model is being constructed at each instant, the space requirement is $O(n)$. We can make the length of a model double at each step instead of incrementing it by just one unit at a time. This leads to an $O(n \log k(e+1)(1 + min\{d_{max} - d_{min} + 1 + e, k^e \mid \Sigma \mid^e\}))$ algorithm. In practice though, since k is a small value in general, such an approach does not allow us to gain much time when errors are allowed. When errors are forbidden, it results in an $O(n \log k)$ algorithm for the search phase.

As mentioned in section 2.2.3, the solution space obtained by applying the rule given in lemma 2.5 may be slightly bigger than the one corresponding to the solution of the problem as stated in section 2.1. Filtering that space means considering each element of $\overline{OI(m')}$, that is, each representative of a class of indirect occurrences (i, f_i, num_errors) for each model m and verifying if $d_{min} \leq f_d - k - f_i \leq d_{max}$. If that is not the case, a pattern matching algorithm such as the one given in [24] allows us to recover in $O((k+e)e)$ time all f_i' such that $d_{min} \leq f_d - k - f_i' \leq d_{max}$ and $\overline{m}$ occurs in s with at most num_errors starting at position i and ending at position f_i'. The whole operation takes time proportional to $n(k+e)e\, min\{d_{max} - d_{min} + 1 + e, k^e \mid \Sigma \mid^e\}$ which does not increase the algorithm's overall theoretical complexity when $e^2 \leq k$.

Finally, printing the resulting structures takes $O(N)$ time where N is the number of palindromes or mirror repeats that satisfy the constraints. This number is bounded over by $O(n(d_{max} - d_{min} + 1))$. Where $d_{max} = n$ and $d_{min} = 0$, this gives an $O(n^2)$ majoration.

2.4 Discussion

We wish to call attention to two points here. The first one is the fact that the time complexity of the algorithm in practice becomes independent of the length L of the hairpin loops for values of L above a certain threshold L_0. This is not true of the naïve approaches such as Waterman's [23] for instance. This is shown experimentally in the curves given in figure 7(a), (b) and (c). The sequence used for the test is a randomly generated one of length $n = 25000$, the x-axis plots the lengths of the hairpin loops (varying between 1000 and 25000 with a step of 1000) and the y-axis the execution time (in seconds). The structures searched for were palindromes, with stem lengths (i.e. models lengths) of 5 or more nucleotides. This experiment was run on a Silicon Graphics Indigo WorkStation (R4000). Figures 8(a), (b) and (c) show also that the time complexity of the search phase of the algorithm is in practice linear with the length of the sequence (each sequence is a randomly generated one of the corresponding length, stem lengths are superior or equal to 5 and the lengths of hairpin loops may vary between 0 and 600). In both figures (7 and 8), plot (a) corresponds to 0 error, (b) to 1 substitution at most authorized, and (c) to 1 error at most (substitution, deletion or insertion).

The second point we wish to call attention to is the fact that allowing for errors in the heads and feet of helices is not very meaningful in general. It is however easy to add a constraint into the algorithm forbidding errors at a given distance p from either end of an helix without increasing its time and space complexity. For errors at the head, we just need to disallow them while $\mid m \mid < p$ where m is the model being constructed. Coloring the complementary inverse (resp. inverse) occurrences that have an error in the last p symbols allows us to determine in $O(1)$ time which ones should be discarded in the final printing phase.

3 PseudoKnots and Triple Helices

3.1 Basic Definitions

Definition 3.1 *Given non-negative integers e, $d[i]_{min}$, $d[i]_{max}$ for $0 \leq i \leq 3$, we call $((u_1, v_1), (u_2, v_2), e, d[i]_{min}, d[i]_{max})$ an approximate pseudoknot in s if:*

- $(u_1, v_1, e, d[0]_{min}, d[0]_{max})$ *and* $(u_2, v_2, e, d[0]_{min}, d[0]_{max})$ *are approximate palindromes in s*
- $d[i]_{min} \leq d[i] \leq d[i]_{max}$ *where $d[i]$ is the distance between:*
  for i=1 *the end position of v_1 and the start position of v_2*
  for i=2 *the end position of u_1 and the start position of u_2*
  for i=3 *the end position of v_2 and the start position of u_1*

Triple helices are elements $((u_1, v_1), (u_2, v_2), e, d[i]_{min}, d[i]_{max})$ for which i takes only the values 0, 1 and 2 but which in addition verify that the start positions of v_2 and u_1 are the same.

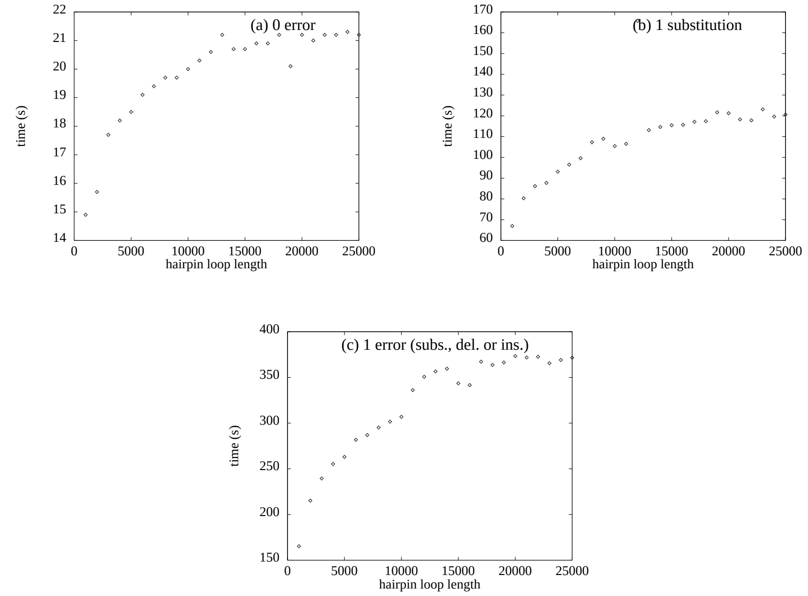

Figure 7: Execution time of the algorithm (palindrome and mirror repeat finder) versus length of hairpin loop for 0 error, 1 substitution at most authorized, and 1 error (substitution, deletion or insertion).

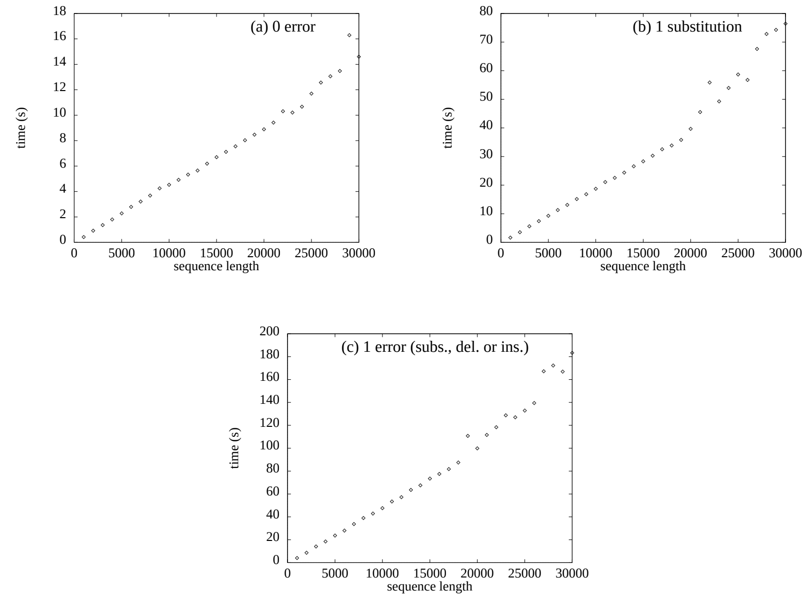

Figure 8: Execution time of the algorithm (palindrome and mirror repeat finder) versus length of sequence for 0 error, 1 substitution at most authorized, and 1 error (substitution, deletion or insertion).

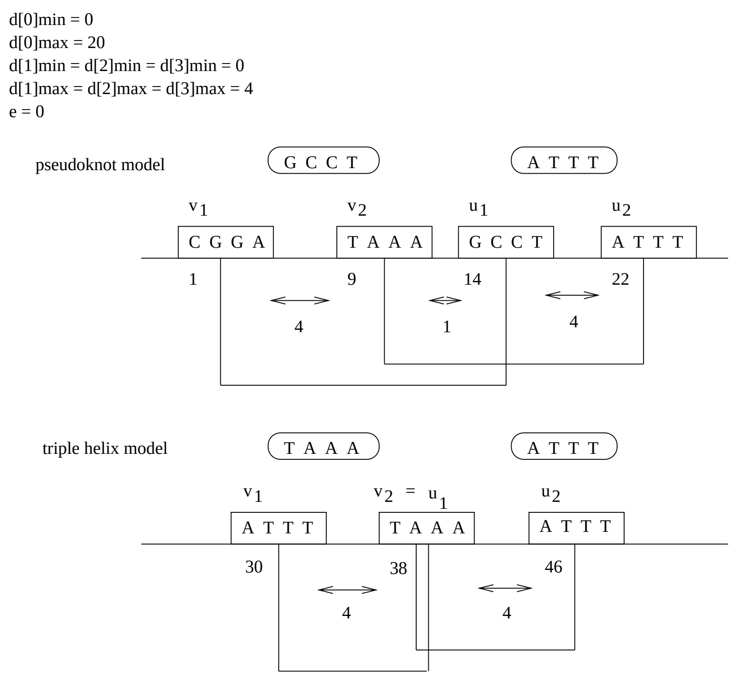

Figure 9: Examples of pseudoknot and triple helix models.

In practice, we set $d[0]_{min} = 0$ and $d[0]_{max} = \infty$ and verify only the $(d[i]_{min}, d[i]_{max})$ constraints for $1 \leq i \leq 3$.

Example 3.1 *See figure 9.*

When working with pseudoknots or triple helices, models are no longer words as with palindromes or mirror repeats but pairs of words.

Definition 3.2 *A model (m_1, m_2) is an element of $\Sigma^{k_1} \times \Sigma^{k_2}$.*

We shall for now consider only models for which $k_1 = k_2$. If any one of the two may be further extended, this can be done at the end, when printing the structural objects. Note that in the case of triple helices, models m_1 and m_2 are not completely independent.

Definition 3.3 *Given a sequence s and non-negative integers e, $d[i]_{min}$ and $d[i]_{max}$ for $0 \leq i \leq 3$ (resp. for $0 \leq i \leq 2$), we say that (m_1, m_2) is an approximate pseudoknot (resp. triple helix) model present in s and satisfying the constraints (e, $d[i]_{min}$, $d[i]_{max}$) if there exist pairs of words (u_1, v_1) and (u_2, v_2) in s that verify:*

- *$u_1 = m_1$ and $u_2 = m_2$;*
- *$((u_1, v_1), (u_2, v_2), e, d[i]_{min}, d[i]_{max})$ is an approximate pseudoknot (resp. triple helix) in s.*

Definition 3.4 *We note $OD_1(m_1)$ (resp. $OD_2(m_2)$) the set of all direct occurrences of the first (resp. second) palindrome in the pseudoknot or triple helix model (m_1, m_2) present in s and $OI_1(m_1)$ (resp. $OI_2(m_2)$) the sets of complementary inverse occurrences.*

$POD_1(m_1)$, $POD_2(m_2)$, $POI_1(m_1)$, $POI_2(m_2)$ are the corresponding sets of possible occurrences.

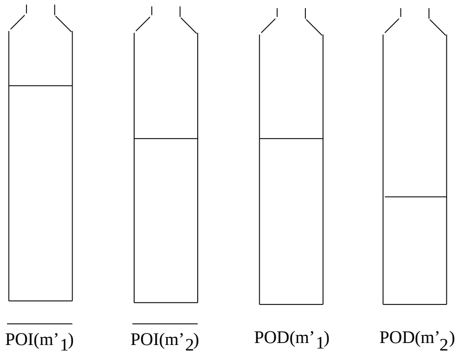

Figure 10: Filtering operation regarded as emptying the contents of four bottles in synchrony.

3.2 Constructing the Models and Filtering for Pseudoknots and Triple Helices

Lemmas 2.1 and 2.4 remain valid for the initial stages of the construction of the models (m_1, m_2) (construction of the palindromic models m_1 and m_2) but lemma 2.5 is now different. We give it here for pseudoknots. In the case of triple helices, the last pair of inequalities is not verified and the equality of the start positions of v_2 and u_1 is checked only at the end of the algorithm.

Lemma 3.1 *$f1_d \in OD_1(m'_1 = m_1\alpha)$ if, and only if, $f1_d \in POD_1(m'_1)$ and [$\exists\ f2_d \in POD_1(m'_2 = m_2\alpha)$, $(i1, f1_i, num\_errors1) \in \overline{POI_1(m'_1)}$ and $(i2, f2_i, num\_errors2) \in \overline{POI_2(m'_2)}$] such that:*

- $d[1]_{min} - num\_errors1 \leq i2 - f1_i - 1 \leq d[1]_{max} + num\_errors1$
- $d[2]_{min} \leq f2_d - f1_d - 1 \leq d[2]_{max}$
- $d[3]_{min} - num\_errors2 \leq f1_d - k - f2_i \leq d[3]_{max} + num\_errors2$

Similar conditions hold for $f2_d$, $(i1, f1_i, num\_errors1)$ and $(i2, f2_i, num\_errors2)$ and are not given here.

Example 3.2 *See again figure 9.*

To simplify, we call from now on the double inequalities of the preceding lemma for $1 \leq i \leq 3$ the $(d[i]_{min}, d[i]_{max})$ constraints.

The algorithm that filters for pseudoknots (or triple helices) by verifying the rules given in the preceding lemma is now slightly more complex than the previous one. An idea of it is given in figure 11. It uses the function RestoreLevel(SetofOccurrences) described previously (see section 2.2.4). Once again, we can compare this to the operation of emptying this time four bottles of water in synchrony. When the level in three out of the four bottles stops at a given point, it is because the level in the fourth one is going down. At each moment, one bottle is filling out while the water in the other ones is staying still. Furthermore, if one considers the level of water in a bottle to be directly related to the position of the occurrences in the sequence, then the bottles must be emptied in a given order and the level of water in one bottle must never be higher than the level in the bottles that precede it in the ordering or lower than the level in the bottles that follow it (see figure 10). This equilibrium must be maintained all the time while the bottles are being emptied so as not to lose any valid occurrence. Emptying the four bottles takes then time proportional to the total quantity of water they initially contained.

3.3 Complexity

The overall complexity of the algorithm for constructing pseudoknot or triple helix models may a multiplicative factor of $(d_{max} - d_{min} + 1 + e)^{p-1}$ bigger than that for palindromic or mirror models given in section 2.3 (with p equal to 2 for triple helices and 3 for pseudo-knots). This comes from the fact that, although there might be more occurrences, whether of pseudoknots or triple helices, these occurrences are kept separated as before and filtering for the structures remains therefore linear in the total number of such occurrences there is at any given step in the algorithm. Eliminating the false occurrences at the end (see section 2.3) takes the same time. Putting the parts together for printing, that is, actually printing the occurrences as structures, may now also be a more complex task. The time needed is still $O(N)$ where N is the number of pseudoknots or triple helices there is in a sequence, but N is now bounded over by $O(n.(d_{max} - d_{min} + 1)^p)$ where $d_{max} - d_{min} = max(d[i]_{max} - d[i]_{min})$ for $1 \leq i \leq 3$. This is of course a worst case situation that is observed only in very degenerated sequences. In the exact case, it may be reached if, for instance, $k << n$, $d_{max} - d_{min} + 1 = n$ and $s = \mathrm{A}^{\frac{n}{4}}\mathrm{C}^{\frac{n}{4}}\mathrm{T}^{\frac{n}{4}}\mathrm{G}^{\frac{n}{4}}$. Finally, extending the length of one of the two palindromes when maximal pseudoknots or triple helices are searched for takes $O(N.(k_{max} - k'_{max}))$ where k_{max} is the maximal length of the longest palindromes found in a structure and k'_{max} the maximal length of the smallest. We may of course have $k_{max} = k'_{max}$ (none of the palindromes can be extended).

3.4 Discussion

We give in figures 12(a), (b) and (c) plots showing that the time complexity of the search phase of the algorithm for identifying pseudoknots and triple helices is in practice linear in the length of the sequence as was the case for the palindromes and mirror repeats finder. The experiment was run on the same machine as previously, with the following parameters :

- length of models searched for ranging between 4 and 20 ;
- $d[1]_{min} = 0$ for $1 < i < 3$;
- $d[1]_{max} = 20$ for $1 < i < 3$.

The sequences were randomly generated, the x-axis plots their lengths (varying between 1000 and 20000 with a step of 1000) and the y-axis the execution time (in seconds). Plot (a) corresponds to 0 error, (b) to 1 substitution at most authorized, and (c) to 1 error at most (substitution, deletion or insertion).

4 Perspectives

As has been shown in [2], procedural algorithms are not ideally suited for looking for structures more complex than the ones we have dealt with here. The pseudoknot and triple helix finders are an example of that. In the procedural approach, the rules governing such structures have to be written directly into the algorithms's code. This soon becomes an awkward business for more elaborated structures. However, it is an approach that is quite justified for palindromes which form the basic units of such more complex structures, as well as for mirror repeats, pseudoknots and triple helices that are either considered important objects in themselves, or thought to be frequent in DNA or RNA molecules. Looking for these simpler structural objects in a quick yet flexible way becomes then necessary and the algorithms we propose here try to fill this need.

Another idea concerning the structure of nucleic acids that is absent in this paper is the fact that molecules having a same function often have the same structure. It is therefore interesting when trying to determine the structure of an RNA for instance, to proceed by analogy with other molecules having same structure. However, this requires again having a global vision of such structures. It also requires that we know how to define the similarity between two, or more, RNA structures. This is a difficult problem, and one that does not seem to have been dealt with in a completely satisfying way. This is what we propose to do in our future work.

Let $POD(m'_j = m_j\alpha)$ and $\overline{POI(m'_j = m_j\alpha)}$ for $j = 1, 2$ be the sets of direct and inverse occurrences of the model (m'_1, m'_2) we are in the process of constructing. As before, these sets are ordered by increasing position in s.

1: get the first element $(i1, f1_i, num\_errors1)$ of $\overline{POI(m'_1)}$
2: do
3:     get the next element $(i2, f2_i, num\_errors2)$ of $\overline{POI(m'_2)}$
4:     do
5:         get the next element $f1_d$ of $POD(m'_1)$
6:         do
7:             get the next element $f2_d$ of $POD(m'_2)$
8:             if the corresponding occurrences satisfy the (d_{min}, d_{max}) constraint
9:                 flag them as valid occurrences
10:         until $(f2_d - f1_d - 1 > d[2]_{max}$ or no element $f2_d$ is available)
11:         RestoreLevel($POD(m'_2)$)
12:         get the next element $f1_d$ of $POD(m'_1)$
13:         if none is available
14:             RestoreLevel($POD(m'_1)$)
15:             goto **2**
16:         else goto **6**
17:     until $(f1_d - k - f2_i > d[3]_{max} + num\_errors2$ or no element $f1_d$ is available)
18:     RestoreLevel($POD(m'_1)$)
19:     get the next element $(i_2, f2_i, num\_errors2)$ of $\overline{POI(m'_2)}$
20:     if none is available
21:         RestoreLevel($\overline{POI(m'_2)}$)
22:         goto **1** replacing “first” by “next”
23:     else goto **4**
24: until $((i_2 - f1_i - 1) > d[1]_{max} + num\_errors1)$
25: RestoreLevel($\overline{POI(m'_2)}$)
26: get the next element $(i_1, f1_i, num\_errors1)$ of $\overline{POI(m'_1)}$
27: if none is available
28:     stop
29: else goto **2**

The sets $\overline{OI(m'_j)}$ and $OD(m'_j)$ for $j = 1, 2$ are then the elements of $\overline{POI(m'_j)}$ and $POD(m'_j)$ flagged as valid occurrences.

Figure 11: Filtering to keep only the parts that can form a pseudoknot.

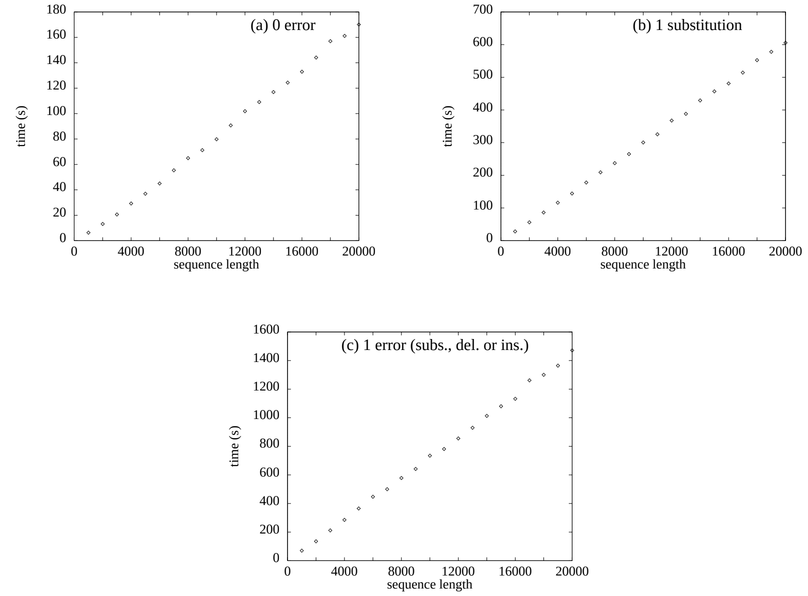

Figure 12: Execution time of the algorithm (pseudoknot and triple helix finder) versus length of sequence for 0 error, 1 substitution at most authorized, and 1 error (substitution, deletion or insertion).

Finally, the idea of "reading" a sequence in both directions, and of allowing for indirect as well as direct occurrences of a model, is interesting also when one is looking for sequence motifs instead of structural objects as in this paper. In such cases, one may be confronted with the problem of having to identify a motif common to a set of sequences read from a database without knowing whether the sequence annotated in the database corresponds to the strand that contains the signal searched for. This problem has been illustrated by Waterman [6] in the case of the promotor signals implicated in *Escherichia coli* transcription process.

Acknowledgements

The authors would like to thank Maxime Crochemore for a careful reading of the manuscript and for his advice and E. W. Myers for an exchange of ideas that do not bear directly on this work but greatly influenced it.
The first author would also like to express her gratitude to Giuseppina Rindone from the Institut Gaspard Monge for having encouraged her to finish this work.
Both wish to thank the referees for correcting some mistakes in the definitions.
Finally, we wish to thank the Institut Pasteur (Project PowerGene) for partial financing of this work, in particular Antoine Danchin for his support of it.

References

[1] J. P. Abrahams, M. v. d. Berg, E. v. Batenburg, and C. Pleij. Prediction of RNA secondary structure, including pseudoknotting, by computer simulation. *Comput. Appli. Biosci.*, 8:243–248, 1992.

[2] B. Billoud, M. Kontic, and A. Viari. Palingol: a declarative programming language to describe nucleic acids' secondary structures and to scan sequence databases. *Nucleic Acids Res.*, 24:1395–1403, 1996.

[3] D. Bouthinon, H. Soldano, and B. Billoud. Apprentissage d'un concept commun à un ensemble d'objets dont la description est hypothétique : application à la découverte de structures secondaires d'ARN. In *11èmes Journées Françaises d'Apprentissage*, 1996.

[4] M. Brown and C. Wilson. RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search. manuscript - University of California, Santa Cruz, Oct. 1995, 1995.

[5] J.-H. Chen, S.-Y. Le, and J. V. Maizel. A procedure for RNA pseudoknot prediction. *Comput. Appli. Biosci.*, 8:243–248, 1992.

[6] D. J. Galas, M. Eggert, and M. S. Waterman. Rigorous pattern-recognition methods for DNA sequences. Analysis of promoter sequences from E*scherichia coli*. *J. Mol. Biol.*, 186:117–128, 1985.

[7] I. Tinoco Jr., P. W. Davis, C. C. Hardin, J. D. Puglisi, G. T. Walker, and J. Wyatt. RNA structures from A to Z. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume LII, pages 135–146. Cold Spring Harbor Laboratory, 1987.

[8] N. A. Kolchanov, I. I. Titov, I. E. Vlassova, and V. V. Vlassov. Chemical and computer probing of RNA structure. In W. E. Cohn and K. Moldave, editors, *Progress in Nucleic Acid Research and Molecular Biology*, pages 131–196. Academic Press, 1996.

[9] M. Kontic. Palingol. Langage pour la description et la recherche de structures secondaires dans les séquences nucléotidiques, 1993. DEA d'Intelligence Artificielle, Université de Paris Nord.

[10] F. Lefebvre. An optimized parsing algorithm well suited for RNA folding. In *Proceedings First International Conference on Intelligent Systems for Molecular Biology*, Cambridge, England, 1995.

[11] B. Lewin. *Genes V.* Oxford University Press, 1994.

[12] H. M. Martinez. An efficient method for finding repeats in molecular sequences. *Nucleic Acids Res.*, 11:4629–4634, 1983.

[13] H. M. Martinez. Detecting pseudoknots and other local base-pairing structures in RNA sequences. 183:306–317, 1990.

[14] S. M. Murkin, V. I. Lyamichev, K. N. Druhlyak, V. N. Dobrynin, S. A. Filipov, and M. D. Frank-Kamenetskii. DNA H form requires a homopurine-homopyrimidine mirror repeat. *Nature*, 330:495–497, 1987.

[15] E. W. Myers. A sublinear algorithm for approximate keyword searching. *Algorithmica*, 12:345–374, 1994.

[16] C. W. A. Pleij and L. Bosch. RNA pseudoknots: structure, detection, and prediction. 180:289–303, 1989.

[17] M.-F. Sagot, V. Escalier, A. Viari, and H. Soldano. Searching for repeated words in a text allowing for mismatches and gaps. pages 87–100, Viñas del Mar, Chili, 1995. Second South American Workshop on String Processing.

[18] M.-F. Sagot and A. Viari. A double combinatorial approach to discovering patterns in biological sequences. In D. Hirschberg and G. Myers, editors, *Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 186–208. Springer-Verlag, 1996.

[19] M.-F. Sagot, A. Viari, and H. Soldano. A distance-based block searching algorithm. pages 322–331, Cambridge, England, 1995. Third International Symposium on Intelligent Systems for Molecular Biology.

[20] M.-F. Sagot, A. Viari, and H. Soldano. Multiple comparison: a peptide matching approach. In Z. Galil and E. Ukkonen, editors, *Combinatorial Pattern Matching*, volume 937 of *Lecture Notes in Computer Science*, pages 366–385. Springer-Verlag, 1995. to appear in *Theoret. Comput. Sci.*

[21] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjolander, R. C. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Res.*, 22:5112–5120, 1994.

[22] D. Searls. The linguistics of DNA. *American Scientist*, 80:579–591, 1992.

[23] M. S. Waterman. Consensus methods for folding single-stranded nucleic acids. In M. S. Waterman, editor, *Mathematical Methods for DNA Sequences*, pages 185–224. CRC Press, 1989.

[24] S. Wu, U. Manber, and E. W. Myers. An O(NP) sequence comparison algorithm. *Inf. Proc. Letters*, 35:317–323, 1990.

[25] M. Zuker and D. Sankoff. RNA secondary structures and their prediction. *Bull. Math. Biol.*, 46:591–621, 1984.

[26] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, 9:133–148, 1981.