Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science 335 (2005) 29–51

Theoretical Computer Science

www.elsevier.com/locate/tcs

Algorithms for pattern matching and discovery in RNA secondary structure

Giancarlo Mauri[a], Giulio Pavesi[b,*]

[a] *Department of Computer Science, Systems and Communication, University of Milan-Bicocca, Milan, Italy*
[b] *Department of Computer Science and Communication—D.I.Co., University of Milan, Via Comelico 39, 20135 Milan, Italy*

Abstract

Text-indexing structures provide significant advantages in the solution of many problems related to string analysis and comparison, and are nowadays widely used in the analysis of biological sequences. In this paper, we present some applications of affix trees to problems of exact and approximate pattern matching and discovery in RNA sequences. By allowing bidirectional search for symmetric patterns in the sequences, affix trees permit to discover and locate in the sequences patterns describing not only sequence regions, but also containing information about the secondary structure that a given region could form, with improvements in terms of theoretical and practical efficiency over the existing methods. The search can be either exact or approximate, where the approximation can be defined simultaneously both for the sequence and the structure of patterns. The approach presented in this paper could provide significant help in the analysis of RNA sequences, where the functional motifs often involve not only sequence, but also the structural constraints.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Pattern matching; Pattern discovery; Affix trees; RNA secondary structure

1. Introduction

Once considered to be a mere mediator of the genetic code, RNA is now recognized as a key player in a wide variety of cellular processes [5,9]. RNA (ribonucleic acid) is a

* Corresponding author.
E-mail address: pavesi@dico.unimi.it (G. Pavesi).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.12.015

linear molecule composed of four different nucleotides (bases), guanine, adenine, cytosine and uracil (that is similar to thymine, found in DNA). Thus, RNA sequences are usually represented by strings over the alphabet $\Sigma_{\mathrm{RNA}} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$. In living organisms, RNA is synthesized using one of the two DNA strands as a template. Once a RNA molecule has been copied from DNA, it starts to fold on itself forming a three-dimensional structure. Nucleotides composing the sequence bind to each other in different ways. The complementary nucleotides C–G and A–U form stable base pairs with each other through the creation of hydrogen bonds between donor and acceptor sites on the bases. These are called *Watson–Crick* base pairs, since they are the same bonds that hold the two strands of a DNA molecule together. In addition, some other base pairings are found in RNA structures, like the weaker G–U pair (*wobble pair*) where bases bind in a skewed fashion. All of these are called *canonical base pairs*. Other non-canonical pairs are sometimes also found, some of which are stable (like the G–A pair).

The *secondary structure* of a RNA sequence is usually represented by the list of the base pairs taking place between nucleotides, with the following constraints:

(1) no nucleotide takes part in more than one bond (base pair);
(2) base pairs never cross: if nucleotide in position i of the sequence is paired with nucleotide $j > i$, and nucleotide $k > i$ is paired with $l > k$, then either $i < j < k < l$ or $i < k < l < j$, but never $i < k < j < l$.

These properties allow us to represent the secondary structure in a very space efficient way. An example is the dot–bracket notation, where the secondary structure of a RNA sequence of n nucleotides is described by a n character string on the alphabet $\Sigma_S = \{\mathtt{(}, \mathtt{.}, \mathtt{)}\}$. A base pair between nucleotides i and j is denoted by an open bracket in position i and a close bracket in position $j > i$. Since base pairs cannot cross, it is always possible to determine for each open bracket which is the corresponding close one. Unpaired nucleotides are indicated by a dot. If the second constraint listed above is relaxed (base pairs can cross), then the structure is said to contain *pseudoknots*.

The various elements found in RNA secondary structure can be classified as follows:

(1) A *stack* (or *helix*) consists of nested, consecutive base pairs, (i, j), $(i+1, j-1), \ldots,$ $(i+k, j-k)$, with $i < j$. The length of the stack is $k+1$. Pair (i, j) is the *terminal* base pair of the stack.
(2) A *loop* is defined as all unpaired nucleotides immediately interior to a base pair.
(3) An *external nucleotide* is any unpaired nucleotide not contained in a loop. Consecutive external nucleotides are called *external elements*.

Each nucleotide involved in a base pair belongs to a unique stack. All unpaired nucleotides are either interior or exterior to a base pair. Interior unpaired nucleotides must belong to a loop, likewise exterior nucleotides must belong to an external element. Since each stack, loop and external element is unique the decomposition is also unique. These observations yield the following:

Lemma 1. *Any secondary structure S can be uniquely decomposed into external elements, loops and stacks.*

Moreover, loops can be characterized in different ways. Let $i+1, i+2, \ldots, i+k-1$ be a stretch of $k-1$ unpaired nucleotides forming a loop. Hence, bases i and $i+k$ must be

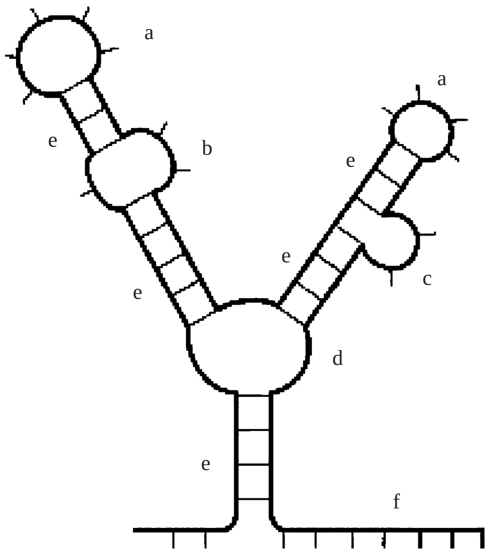

Fig. 1. Decomposition of RNA secondary structure in hairpin loops (*a*), internal loops (*b*), bulges (*c*), multi-loops (*d*), stacks (*e*), external elements (*f*). The elements *e–b–e–a* and *e–c–e–a* form two hairpins.

paired with some other nucleotide. We have different possibilities:

(1) nucleotide i is paired with $i+k$. Then, the loop is said to be a *hairpin* loop;

(2) nucleotide i is paired with some nucleotide $j > i$, and $i+k$ with l. Since base pairs cannot cross, we have that $l < j$. If $l = j-1$, then $i+1, \ldots, i+k-1$ are said to form a *bulge*; otherwise, if also all nucleotides $l+1, \ldots, j-1$ are unpaired, together with $i+1, \ldots, i+k-1$ they form an *internal loop*;

(3) nucleotide $i+k$ is paired with $l < i+k$. Then, i is paired with $j > l$. Analogously with the previous point, we have a bulge if $j = l+1$, or an internal loop if $j > l+1$ and the intervening nucleotides are unpaired;

(4) if none of the previous conditions is true, then nucleotides $i+1, \ldots, i+k-1$ form a *multiloop*.

An example is shown in Fig. 1. A series of stacks (possibly interrupted by internal loops and/or bulges) immediately followed by a hairpin loop is usually referred to as *stem*, and the whole structure is called *hairpin* or *stem–loop*. Returning to the dot–bracket alphabet, the structure shown in the figure is denoted as

```
..((((..(((((.(((.....)))..))))))..(((((((....)))..)))))..)))))..
```

1.1. Motifs in RNA secondary structure

Several experiments and observations have shown that local distinct structural elements in non-coding RNA molecules are strictly correlated with their function [9,38]. Thus, given one or more RNA sequences experimentally known or suspected to have a given biological function, finding similar structural elements in them could provide significant evidence

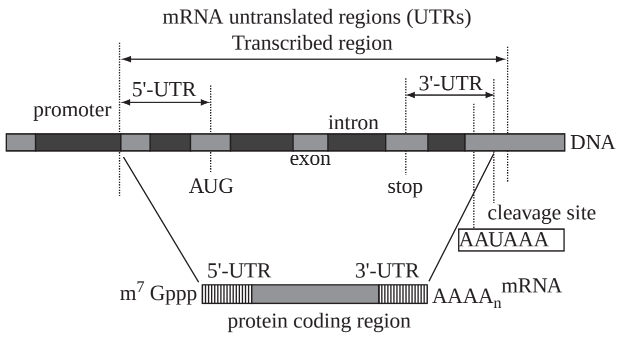

Fig. 2. Synthesis and structure of messenger RNA.

on which parts of the sequences, and their structure, is responsible for the function itself. Alternatively, the detection of a known motif in a newly determined sequence could shed some light on the function, or the mechanisms regulating the expression of the sequence itself. Also, one of the most important open problems is the detection in genomes of genes of non-coding RNA, like tRNA, ribosomal RNA, microRNA, srpRNA, RNAse P RNA, and many others [5]. Knowing in advance a functional motif conserved throughout different species in a given non-coding RNA molecule, together with a suitable pattern matching algorithm able to detect such a motif in a newly sequenced genome, could highlight some interesting regions that might contain the gene encoding for that type of non-coding RNA.

Moreover, many RNA motifs of biological interest can be described by secondary structure alone. A striking example is post-transcriptional regulation of gene expression [12]. After a messenger RNA (mRNA) molecule has been transcribed from a genome, with a complex mechanism influenced by the presence of enhancing or inhibitory signals in the genomic regions surrounding starting point of the corresponding gene, its actual translation into a protein undergoes further regulation, determined by the interaction of the mRNA with proteins and/or other RNA molecules. These interactions involve the presence in the mRNA of motifs presenting conservation both in structure and (more loosely) in sequence, that are recognized by regulatory proteins and/or other RNAs. These motifs are usually located in the so-called *untranslated regions* (UTRs), non-coding parts of the sequence located in the mRNA immediately before (5′UTR) or after (3′UTR) the part actually translated into a protein (see Fig. 2). Here are some examples.

1.1.1. Histone 3′UTR mRNA

Metazoan histone 3′UTR mRNAs, lacking a polyA tail, contain a highly conserved stem–loop structure with a six base stem and a four base loop. This stem–loop structure plays a different role in the nucleus and the cytoplasm. The histone 3′UTR hairpin structure is peculiar in that the bases of the stem are conserved, unlike most functional hairpin motifs, where conserved bases are found in single-stranded loop regions only, and its consensus is usually represented as in [42]:

```
((((((....))))))
GGYYYUHURHARRRCC
```

where Y = C,U (pyrimidines), R = A,G (purines) and H = not G.

1.1.2. *Iron responsive element*

The iron responsive element (IRE) is a hairpin structure conserved in the 5′UTR or in the 3′UTR of various mRNAs coding for proteins involved in cellular iron metabolism, that works as a sensor of the iron level of the cell. Two alternative IRE consensus structures have been found. Some IREs present an unpaired nucleotide (usually cytosine) on the stem, whereas in others the cytosine nucleotide and two additional bases seem to oppose one free 3′ nucleotide [14]:

```
(((((.(((((......))))))))))
NNNNNCNNNNNCAGWGHNNNNNNNNNN
(((((...(((((......))))).)))))
NNNNNNNCNNNNNCAGWGHNNNNNNNNNNN
```

where W = A,U. As we can see, there is no conservation in the nucleotides forming the stem, but only in those forming the loops. And, moreover, the motif can appear in two alternative structural forms.

1.1.3. *Selenocysteine insertion sequence*

Selenocysteine is the recently discovered 21st amino acid. The codon calling for this amino acid in mRNA is UGA, usually indicating the end of translation. In order to have UGA read as a selenocysteine codon, and bound by the corresponding tRNA, proteins containing this amino acid (usually called *selenoproteins*) present the Selenocysteine Insertion Sequence (SECIS) element, a conserved stem–loop motif located in the 3′UTR of the corresponding mRNAs. Two forms of the SECIS motif have been determined, respectively with long (12–14 nucleotides, SECIS I) and short (3–6 nucleotides) apical loops (SECIS II) [6]. SECIS I stems can be split in two parts, interspersed by an internal loop (see Fig. 3). The topmost helix contains non-canonical base pairings at the bottom, always with two G–A pairs. In the second form, nucleotides forming the apical loop bind to one another, and as a result the upper helix is interrupted by another internal loop. Other than the non-canonical G–A pairs, SECIS hairpins present two consecutive unpaired adenine nucleotides, either in the apical loop (I) or inside an internal loop (II). Experimental evidence supports the fact that the form II is found more often than the first. The non-canonical pairs are essential to mediate selenoprotein translation. Several algorithms and programs have been developed for the discovery of potential SECIS elements in mRNA, and genome-wide scans have led to the discovery of new selenoproteins [2,19,23]. All the methods relied on the presence of the non-canonical base pairs.

In biological sequences, sequence similarity usually implies structural and/or functional similarity. However, if we consider RNA secondary structure we notice that, very often, conserved motifs present similarity at the sequence level only in the unpaired elements (internal or hairpin loops), that are free to interact with other nucleotides and/or proteins.

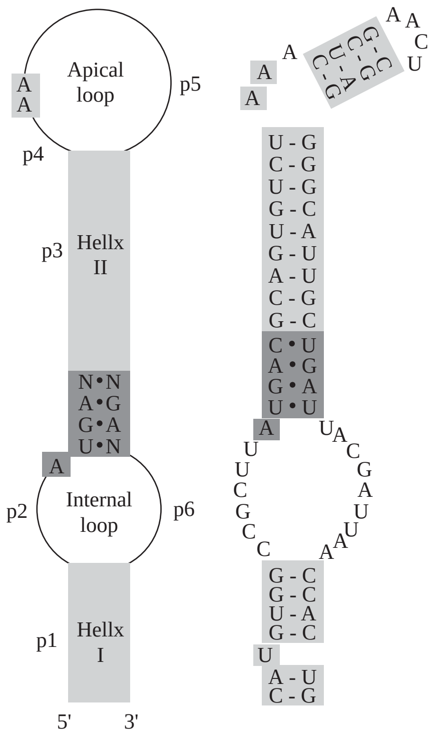

Fig. 3. Examples of SECIS I (left) and SECIS II (right) hairpin structures found in the mRNA of selenoproteins. (Figure taken from [2].)

Paired elements (i.e., along stacks) are instead much less conserved at sequence level, since what is required from them is only to bind to each other. Thus, looking for elements similar in sequence in a set of RNA sequences sharing the same biological function often yields unsatisfactory results, since regions loosely similar can fold anyway into similar structures playing the same biochemical role. Unfortunately, the secondary structure is almost never available.

It is therefore important, when developing algorithms for pattern matching and discovery in RNA sequences to consider not only sequence similarity, but also associate with the regions compared the (potential) secondary structure that they might form. For example, a genomic sequence can be scanned for coding regions or promoters by considering sequence information alone. Instead, sequence conservation in non-coding RNA is much looser, and almost undetectable by current methods (or detectable at the price of millions of false positives). A number of different secondary structure matching tools have been introduced so far for RNA [20,26,32]. However, to our knowledge, the one presented in this paper is the first algorithm for this problem based on an indexing structure. As we will see, this fact provides significant advantages in terms of theoretical and practical efficiency with respect to existing methods.

2. Affix trees

Let Σ be an alphabet, and Σ^* the set of strings over Σ. In our case, $\Sigma = \{\mathtt{A,C,G,T}\}$ (or $\Sigma = \{\mathtt{A,C,G,U}\}$). Given a string (sequence) $S = s_1 \dots s_n \in \Sigma^*$, we denote with S^R the *reverse* of S, that is, $S^R = s_n \dots s_1$. If $S = \alpha\beta\gamma$, with $\alpha, \beta, \gamma \in \Sigma^*$, then α is a prefix of S, and γ is a suffix of S. Text-indexing structures, like suffix trees, present significant advantages for the solution of problems related to string analysis and comparison, and are nowadays widely used for the representation of DNA and protein sequences. In this section we introduce the affix tree, a data structure that contains more information than the suffix tree, but can anyway be built in linear time and space. As we will see, this additional information can be exploited in the case of RNA.

Definition 1. An *affix tree* [37] for a non-empty string $S = s_1 \dots s_n$ is a directed acyclic graph [13] $\mathcal{A}(S) = (V, E)$ such that:
(1) one distinct node, with no incoming edges, is marked as *root*;
(2) $E = E_s \cup E_p$, and $E_s \cap E_p = \emptyset$. That is, the edges of $\mathcal{A}(S)$ are divided in two disjoint subsets, that we will call *suffix* and *prefix* edges;
(3) suffix edges are labeled with non-empty substrings of S;
(4) prefix edges are labeled with non-empty substrings of S^R;
(5) two edges of the same type leaving the same node cannot have labels starting with the same character;
(6) on each path starting from the root and following suffix edges only, the concatenation of the edge labels spells out a substring of S;
(7) on each path starting from the root and following prefix edges only, the concatenation of the edge labels spells out a substring of S^R;
(8) every substring of S is spelled out by a unique path starting from the root and following suffix edges only, while every substring of S^R is spelled by a unique path following prefix edges only.
(9) for each node p the tree, where α is the concatenation of the labels of suffix edges from the root to p, the path from the root to p following prefix edges spells out α^R.

The structure $\mathcal{A}(S)$ can be seen as composed by two sub-structures (V, E_p) and (V, E_s), that are trees that index the substrings of S^R and S, respectively. An example is shown in Fig. 4. As we have seen in the definition, if p is a node of the structure, and α is the concatenation of the labels of suffix edges from the root to p, the path from the root to p following prefix edges spells out α^R. There is a path on suffix (prefix) edges labeled α ending exactly at a node if substring α appears in S (or S^R, respectively) followed by at least two different characters, or if it is a suffix of S (S^R). In the example shown in Fig. 4, the node whose suffix label is ATA has been created because its label on prefix edges *ata* corresponds to a suffix of S^R.

Now, let p be a node of the structure, and α be the concatenation of the labels of suffix edges from the root to p. The path from the root to p following prefix edges will spell α^R. Let $p \to q$ be a prefix edge leaving p and entering node q, and $\gamma = \gamma_1 \dots \gamma_l$ be its label. Then, the suffix path from the root to q spells out substring $\gamma^R\alpha$ of S. This property is essential for the algorithms we will present in the following, since once a substring of S has been located

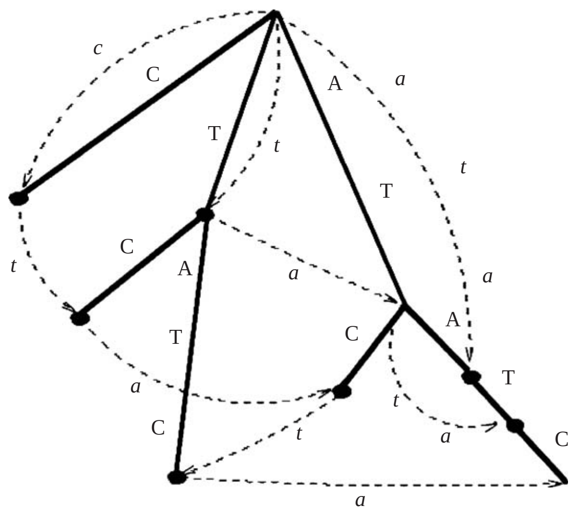

Fig. 4. Affix tree for string ATATC. Suffix edges are solid, while prefix edges are dotted. Prefix edges labels are in italic.

in $\mathcal{A}(S)$, we can expand it in both directions starting from the corresponding node in the structure. That is, we can add characters both to its right end (by following suffix edges) and to the left end, by following prefix edges, but always moving downward in the structure. If $|\gamma| > 1$, then $\gamma_1\alpha$ occurs in the string *always* preceded by substring $\gamma_l \dots \gamma_2$. If no prefix edge leaves node p, then substring α starts only at position 1 (is a prefix) of S. Conversely, if no suffix edge leaves node p, then α occurs only as a suffix of S. The same argument holds for the substrings of S^R, clearly by swapping suffix edges with prefix edges, and vice versa. From now on, we will call *string depth* of any path along the edges of the structure (either suffix or prefix) the length of the concatenation of the edge labels of the path.

The affix tree of a string can be built in linear time, and takes linear space [25]. An analogous structure can be built for a set of k strings, following for example the construction method described in [13] for suffix trees. Moreover, each node of the structure can be annotated with a k-bit string, where the ith bit is set iff the corresponding substring (spelled by suffix edges) occurs in the ith string of the set.

2.1. Pattern matching with affix trees

Text-indexing structures, like suffix trees and automata, usually permit to find the occurrences of a pattern in a string by matching the characters of the pattern from left-to-right (or, in some cases, from right-to-left), until all the characters have been matched, or a mismatch is encountered [13]. Instead, given the affix tree for a string S, we can match a pattern $P = p_1 p_2 \dots p_m$ with the sequence starting from any character of P. For example, let p_i be the middle character (we assume w.l.o.g. that m is odd): we will have to match the characters $p_{i+1} \dots p_m$ on suffix edges as usual (forward match), and those preceding p_i on *prefix* edges (backward match).

More in detail, we first locate p_i on the suffix edges leaving the root. To match p_{i-1} we

(1) Go to the node entered by the edge where we found p_i. Let c_s be the number of characters on the edge label we skip.
(2) Traverse the prefix edge whose label starts with character p_{i-1} (if it exists: otherwise, *P* does not appear in *S*). Move backward toward the root (on a suffix edge) by c_s characters. We will have to move along a single edge. Let $p_{i-1}\gamma$ be the label of the prefix edge we have traversed. We have two cases
    (a) $|\gamma| = 0$. The position reached corresponds to the endpoint of the path (read on suffix edges) corresponding to the occurrences in *S* of $p_{i-1}p_i$.
    (b) $|\gamma| > 0$. The position reached corresponds to substring $\gamma^R p_{i-1}p_i$. Since $p_{i-1}p_i$ and $\gamma^R p_{i-1}p_i$ occur exactly at the same positions in *S*, all paths starting from the endpoints of $p_{i-1}p_i$ and $\gamma^R p_{i-1}p_i$ will have the same suffix edge labels. The only difference is that in the next matches we will use the same prefix edge for the characters preceding p_{i-1} in the pattern, without traversing it but only matching characters, keeping the endpoints reached at the previous forward match, until all the characters of γ have been used. Then, we proceed as usual. In general, we will follow these steps for any prefix edge with label $a\gamma$ and $|\gamma| > 0$.

At this point, we match p_{i+1} on suffix edges from the endpoint reached at the previous step. Then, we continue with p_{i-2} as with p_{i-1}, and so on, alternating one forward match with one backward match, until all characters have been matched in both directions, or a mismatch is encountered. We chose to match the characters of *P* starting from the center and alternating the two directions. While for any pattern this is probably not the simplest and most efficient way, it is anyway the best choice when dealing with patterns describing RNA secondary structure, as we will see in the following.

3. Matching hairpins

The basic element of RNA secondary structure, often by itself responsible for a variety of biological functions, is the hairpin. If we observe the pattern associated with it, for example $H =$ `(((((....)))))` (possibly interspersed by internal loops and/or bulges), we can notice that it presents a peculiar symmetric layout, with the central part corresponding to the hairpin loop, flanked by the nucleotides that form the stem. Now, suppose that, given a RNA sequence $S = s_1 \dots s_n$, we want to find out whether it might form, somewhere, the hairpin structure *H*. The idea is that, if *H* can appear in the secondary structure of the sequence, then *S* will present at least one substring $s_{i+1} \dots s_{i+14}$ such that s_{i+1} can form a canonical base pair with s_{i+14}, nucleotide s_{i+2} with s_{i+13}, and so on.

Now, let $\mathcal{A}(S)$ be the affix tree for *S*. From the root, we follow every path on suffix edges only until we reach the string depth corresponding to the size of the hairpin loop (four in the example). Since we do not impose any sequence constraint, every substring of length four is a candidate for the formation of the hairpin loop. Then, from the endpoints we reached, we try to expand each path by adding one character to its right and one to its left, and, for each possible pair, we check whether it can form a canonical base pair. We discard the paths that do not satisfy this property. The surviving paths will spell out substrings of *S* that might fold into structure `(....)`. If the structure we want to find contains also internal loops and/or

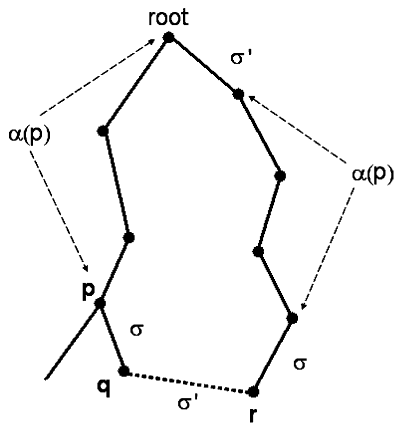

Fig. 5. Adding a base pair to substring $\alpha(p)$. Bases σ and σ' form a base pair. Node r spells substring $\sigma'\alpha(p)\sigma$. Suffix edges are solid, prefix edges are dotted.

bulges on the right-hand side, we just have to follow the suffix edges on each path for a number of characters matching their length, without any further check. The same holds for internal loops and bulges on the left-hand side, that will be matched against characters on prefix edges (see Fig. 5). Each path can be defined by a pair of pointers, one indicating the end of the path on suffix edges, and the other one indicating its end on prefix edges, that can be used when we have to traverse a prefix edge whose label is longer than one character.

More formally, let $P = p_1 p_2 \ldots p_m$ a pattern on the bracket alphabet Σ_S describing a stem–loop structure, and S a string on Σ_{RNA}. Let $\mathcal{A}(S)$ be the affix tree of S. Let p_i be the first character of P before the hairpin loop (an open bracket), and p_j the first character after it (a close bracket). Let $l = j - i - 1$. Each path on the structure is denoted by two pointers e_{p} and e_{s} for prefix and suffix edges, respectively. The steps of the algorithm can be summarized as follows:

Initialization: From the root of $\mathcal{A}(S)$, follow every possible path on suffix edges until string depth l is reached. Let E be the set of paths (each one denoted by two pointers) reached. Nucleotide p_i is the last nucleotide before the hairpin loop, p_j is the first one after the hairpin loop.

```
**While** $i > 0$ and $j < m$
(1) **While** $p_i =$ ( and $p_j =$ ) do
    (a) $E' = \emptyset$;
    (b) For each path $(e_{\mathrm{p}}^i, e_{\mathrm{s}}^i) \in E$:
        (i) add one character on prefix edges from the endpoint $e_{\mathrm{p}}^i$ (by moving by one character on the prefix edge, or by traversing it). Let $\hat{E}_P$ be the set of corresponding pointers;
        (ii) for each path $(\hat{e}_{\mathrm{p}}^i, \hat{e}_{\mathrm{s}}^i) \in \hat{E}_P$, move by one character from $\hat{e}_{\mathrm{s}}^i$ on those suffix edges that continue with a character that forms a canonical base pair with the one that was added to $e_{\mathrm{p}}^i$. Add the corresponding paths to $E'$.
    (c) $E = E'$;
    (d) $i = i - 1$; $j = j + 1$.
(2) $Left = 0$;
(3) **While** $p_i =$ . $Left = Left + 1$;
(4) **If** $Left > 0$
    • $E' = \emptyset$;
    • On each path $(e^i_{\mathrm{p}}, e^i_{\mathrm{s}}) \in E$ move by $Left$ characters on prefix edges starting from $e^i_{\mathrm{p}}$.
      For each endpoint $(\hat{e}^i_{\mathrm{p}}, \hat{e}^i_{\mathrm{s}})$ reached, $E' = E' \cup (\hat{e}^i_{\mathrm{p}}, \hat{e}^i_{\mathrm{s}})$;
    • $i = i - Left$;
    • $E = E'$;
(5) $Right = 0$;
(6) **While** $p_j =$ . $Right = Right + 1$;
(7) **If** $Right > 0$
    • $E' = \emptyset$;
    • On each path $(e^i_{\mathrm{p}}, e^i_{\mathrm{s}}) \in E$ move by $Right$ characters on suffix edges starting from
      $e^i_{\mathrm{s}}$. For each endpoint $\hat{e}^i_{\mathrm{s}}$ reached, $E' = E' \cup (\hat{e}^i_{\mathrm{p}}, \hat{e}^i_{\mathrm{s}})$;
    • $j = j + Right$;
    • $E = E'$;
```

The algorithm stops whenever the set of pointers E is empty. In this case, pattern P does not appear in S, that is, the structure it represents cannot be formed, according to base pairing rules, by the sequence, and the part of it that has been matched represents the largest sub-structure of P that has been found in S. Otherwise, at the end the set E will contain pointers to the substrings of S that might fold into the structure.

3.1. Validating candidate matches

Clearly, from a biological point of view, locating regions that match a structural pattern according to base pairing rules only is not enough to guarantee that they actually form the structure described by the pattern. The most widely used methods for the prediction of RNA secondary structure associate with each possible conformation an energy value [27,40]. The energy of the unfolded sequence is set at 0 kCal/mol. Every base pair brings a negative contribution to the energy, according to its stability. For example, the G–C pair has the lowest value, since it is stable and causes the formation of three hydrogen bonds between the two nucleotides. Pair A–U brings a slightly higher energy contribution, since one less hydrogen bond is formed, and so on. Unpaired nucleotides contained in the structure have a destabilizing effect, bringing a positive contribution to the energy, that changes according to which nucleotides are left unpaired, and the size of the unpaired elements (larger loops give a larger energy value). The space of all possible conformations is explored with a dynamic programming algorithm that takes $O(n^3)$ time for a sequence of length n, and the structure of minimal energy (if there exist a structure whose energy is lower than zero) is the one predicted. The problem is that often, in practice, the real structure of a RNA molecule does not correspond to the one of minimal energy, since RNA energy landscapes are often bumpy and rugged [3], and the actual structure is just a local minimum, instead of a global one.

However, a similar approach can be followed also in our case. That is, once the substrings that could fold into the structure described by the pattern have been located, we can also evaluate the energy of the structure associated with each one. Then, we can report the hits sorted according to their energy, in order to highlight the most probable ones, or report only those of negative energy, and so on. While energy by itself is not always reliable for the evaluation of the overall structure of a sequence, it is much more precise when it is applied to a single hairpin structure corresponding to a substring. In fact, our experiments revealed that functional hairpin motifs (not involved in more complex structures) in non-coding RNA like the ones shown in Section 1.1 have a negative energy value, usually proportional to the number of base pairs involved in the structure, as also demonstrated in [31]. Thus, considering only those substrings whose corresponding energy is lower than the unfolded one, permits to eliminate from the output a large number of false positives, usually with low risk of losing a true motif.

3.2. Complexity

Let S be a string over the alphabet Σ_{RNA} of length n. Given a pattern P of m characters on the dot–bracket alphabet, a naive approach to match P against S could be to consider each of the substrings of S of length m, and to check whether it fits the structural pattern P, with an overall $O(nm)$ time complexity.

In our method, instead, the construction of the affix tree takes $O(n)$ time. For matching pattern P in the structure in the worst case we have to follow every possible path leaving the root. When we are at string depth l, there are at most $p = \min\{4^l, n\}$ possible paths in the structure. However, only 6 pairs out of 16 can form a canonical base pair in RNA structure. Thus, when we add a base pair starting from all the possible 4^l paths we discard more than half of them. If string depth l was reached by adding b base pairs and u unpaired nucleotides, the number of paths is thus bounded by $p = \min\{(3/8 \cdot 4^2)^b \cdot 4^u, n\}$. Each expansion takes constant time on each path, and, if P appears in S, the algorithm performs at most $O(pm)$ expansions. The affix tree requires $O(n)$ nodes, and we need at most $2n$ pointers to perform the search. Thus, the overall space complexity of the algorithm is $O(n)$.

The advantages of the preliminary construction of the affix tree become evident when one has to match a large number of patterns against a sequence, and $n \gg m$, like a newly sequence bacterial genome scanned for known non-coding RNA motifs. In fact, while without the indexing structure the $O(nm)$ bound is tight, since we have to consider exactly $n - m + 1$ substrings of length m, with the affix tree we can expect the number of paths to be in practice much lower than p, also when $n > 4^l$. For example, if we assume that the input sequence has uniform nucleotide composition, we can expect, on the average, to discard more than one half of the possible paths each time we add a base pair.

4. Matching approximate descriptors

The method presented in the previous section can be easily extended in order to consider approximate matching of hairpin structures, that is, allowing differences in size and posi-

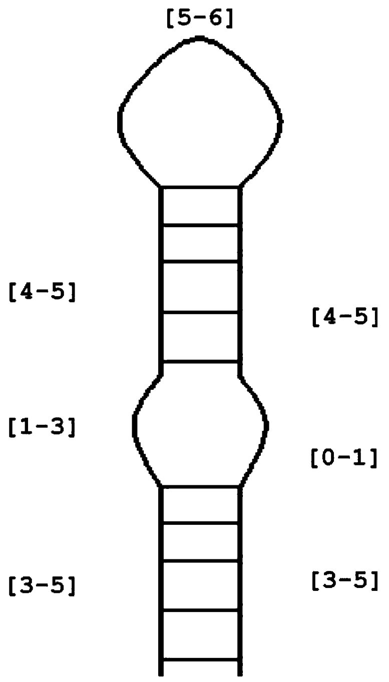

Fig. 6. Defining an approximate hairpin. The lower part of the stem can have from 3 to 5 base pairs, the internal loop can have from 1 to 3 unpaired nucleotides on the left-hand side and at most one on the right-hand side, and the hairpin loop is formed by 5 or 6 nucleotides.

tion in the various elements forming them. For example, we can define structural patterns like

```
([3-5].[1-3]([4-5].[5-6])[4-5].[0-1])[3-5]
```

(see Fig. 6). In this way, we describe an hairpin whose lower stem ranges from 3 to 5 base pairs, that has an internal loop ranging from 1 to 3 nucleotides on the left-hand side and from zero to one nucleotide on the right-hand side, 4 or 5 base pairs on the upper part of the stem, and finally an hairpin loop of five or six nucleotides. Note that range values of the paired elements on the left-hand side have to match those of the corresponding elements on the right-hand side. The idea is similar to the exact case: first, we locate in the affix tree, on suffix edges, all the substrings whose length corresponds to the range allowed for the hairpin loop. In the example, we consider all substrings of length five and six. Then, we proceed by adding base pairings as usual, but in this case we will consider as matching all those paths where we were able to add four or five base pairs. On each of these paths, we add one, two, or three unpaired nucleotides on the left-hand side (prefix edges), and zero or one nucleotide on the right-hand side (suffix edges). Finally, on the paths obtained, we add three, four, or five base pairs.

More in general, approximate patterns will have the form:

$$\texttt{([}s_1\texttt{-}S_1\texttt{].[}i_1\texttt{-}I_1\texttt{]} \ldots \texttt{([}s_p\texttt{-}S_p\texttt{].[h-H])[}s_p\texttt{-}S_p\texttt{]} \ldots \texttt{.[}i_{2p-2}\texttt{-}I_{2p-2}\texttt{])[}s_1\texttt{-}S_1\texttt{]}$$

describing a pattern composed of a stem divided in p parts, interspersed on each side by $p-1$ internal loops, and with an hairpin loop ranging from h to H nucleotides. For each element, we provide a minimum and maximum size value. In the initialization step we consider all the substrings of length ranging from h to H. Then, on each path, we try to add a minimum of s_p base pairs and a maximum of S_p, and so on. For example, a descriptor containing the two alternative forms of the IRE motif can be written as

```
([3-5] .[1-3] ([5-5] .[6-6] )[5-5] .[0-1] )[3-5]
```

4.1. Complexity

With respect to the exact case, in the approximate case paths of different length can correspond to occurrences of the same pattern. Moreover, different structures can be associated with the same substring, at least before the final energy check. For example, substring `CGGGCNNNNGUCUG` (where `N` stands for any nucleotide) could fold, according to canonical base pairing rules, into structures `.((((....).))))` or `(((((....)))))`. At the initialization step, we start with at most $(H-h+1)n$ paths. When we add the base pairs of the upper part of the stem, we have at most $(S_\mathrm{p}-s_\mathrm{p}+1)(H-h+1)n$ paths. When we add an internal loop of size ranging from i_j to I_j the number of paths is multiplied by (I_j-i_j+1). An upper bound on the number of paths b can be thus given by:

$$b \leqslant (H-h+1)\prod_{j=1}^{p/2}(S_j-s_j+1)\prod_{j=1}^{2p-2}(I_j-i_j+1)\cdot n.$$

Given $i = \max_j(I_j-i_j+1)$ and $s=\max_j(S_j-s_j+1)$ an estimate of the complexity is finally given by $\mathrm{O}((H-h)i^{2p-2}s^pn)$. In biological circumstances, usually $p \leqslant 3$, $(H-h) < 5$, the difference in size i between internal loops is lower than five (and their size is lower than ten), and the range of paired stacks s does not exceed three or four.

4.2. Adding sequence information

If something about the sequence composition of a motif, other than its structure, is known in advance, this information can be easily added to the algorithm, in practice making it faster. As we have seen in the examples, sequence conservation is often encountered in the unpaired regions of the motifs. Thus, while defining a pattern, we can extend the bracket alphabet with additional symbols, representing the four nucleotides or groups of them. The only difference is that now, instead of accepting any substring in the initialization step and any path in correspondence to internal loops, we check whether the characters defined are matched. For example, given one of the two forms of the IRE motif:

```
(((((.(((((......)))))))))))
NNNNNCNNNNNCAGWGHNNNNNNNNNN
```

in the initialization step we keep the substrings matching CAGWGH (CAGAGA, CAGUGA, CAGAGC, and so on). Then, after five base pairs have been added, we expand the surviving paths by looking for a C on prefix edges, discarding the others. The unpaired regions can be also described with a consensus and a maximum error allowed for their instances (according for example to Hamming or edit distances), or with a profile and a corresponding weight threshold.

5. Hairpin discovery with affix trees

In some cases, instead of knowing a functional motif in advance, we just suspect that a set of coregulated RNA sequences might contain a conserved motif, responsible for their function and/or regulation. The problem is that the motif, as well as its conservation, is not known in advance.

The problem of finding conserved secondary structure motifs in RNA sequences has been often coupled with the prediction of the structure itself. If the sequences are expected to fold into an overall similar structure, they can be aligned beforehand, and the alignment can be used to predict a consensus secondary structure [8,36,39,41]. A method to predict the same global structure for a set of sequences without aligning them is instead presented in [1].

If structural similarity is instead limited to some elements, like single hairpins, the problem becomes even more challenging. Sequence alignments can be anyway used also in this case, and common structural elements can be predicted only for the most conserved parts [43]. Alternatively, one can try to identify with statistical methods regions of the sequences that seem to be good candidates for the formation of a functional motif, such as unusual folding regions [22] (where the folding free energy of the region is significantly lower than that expected by chance) or well ordered folding regions [21], and try to predict a common fold only for these regions, even if a precise characterization of what can be considered an interesting region is far from immediate [35]. Another idea is to align the sequences and predict a common secondary structure simultaneously, either locally or globally, as in the FOLDALIGN and SLASH algorithms [10,11]. The main drawback of this kind of approach lies in its high time complexity, a priori exponential in the number of input sequences, that even if reduced to about $O(L^4N^4)$ for L sequences of N nucleotides with the introduction of some heuristics, limits its usage to short sequences. Finally, if one has some hint about the structure and/or the sequence of the motif, a pattern matching tool, like the one just presented, can be used, in order to locate regions in the sequences that can fold into a given structure [20,26,32], and the results can be post-processed in order to extract from them the most similar hits [7]. An approach based on evolutionary computation aimed at the optimization of the search parameters is presented in [18]. In any case, an ultimate tool, able to detect efficiently and reliably conserved secondary structure elements, such as simple stem–loop structures, has yet to be introduced. It is thus hardly a surprise the fact that this problem has been called “finding hairpins in a haystack” [4].

However, another precious feature of indexing structures is that they allow to implement recursively the exhaustive search for a given set of patterns. This property also holds for affix trees and RNA secondary structures. Suppose that we are given a set of k RNA sequences $\mathcal{S}$, and we want to find out whether the same (unknown) hairpin structure appears in every sequence, perhaps with some differences. The idea is to build the affix tree for the set of sequences, and implement a recursive search for the set of all possible hairpin structures.

In fact, the search, for example, of structures (((((....)))..)) and (((((....))))) will have to locate (((....))) first. Thus, the set of endpoints reached for this structure can be used as a starting point for both searches, and we do not have to start from the root again while searching for the second motif. Moreover, if we find out that no substring can fold into structure (((....))), then we know also that no substring will be able to fold into any structure having (((....))) as a core. In this way, the exhaustive search for all possible stem–loop folding patterns, including bulges and/or internal loops can be implemented in a much more efficient way. In order to determine in how many, and which, sequences a given pattern appears, we can employ the bitstring annotation describe in Section 2.

Given a set of RNA sequences S, the basic version of the discovery algorithm needs as input a maximum value for the loop size of the motifs (in biological circumstances, this usually is never larger than 20), a maximum number u of unpaired nucleotides that can form internal loops and bulges along the stem (both these parameters are optional, but providing an upper bound for them significantly accelerates the search, ten is a suitable value), and a quorum q representing the minimum number of sequences a motif has to appear in. The core of the algorithm can be summarized as follows:

```
**Initialization** (strings $\mathcal{S}$, int *max_loop_size*, quorum $q$):
(1) Build and annotate with bitstrings the affix tree for the strings of $\mathcal{S}$.
(2) For each loop size $l$, with $l \geqslant 3$ and $l \leqslant max\_loop\_size$:
    (a) Locate in the affix tree all the nodes corresponding to substrings of length $l$, and try
        to add a base pair to each one; let $P$ be the set of the surviving paths;
    (b) Call Expand($P$);

Procedure **Expand**(paths $P$):
(1) For each path in $P$ try to add a base pair; let $P'$ be the set of paths obtained.
(2) If at least $q$ bits are set in the bit string of $P'$, call Expand($P'$), otherwise Report($P$).
(3) If less than $u$ unpaired nucleotides have been added, for each path in $P$ add a single
    letter by following suffix edges; let $P'$ be the set of paths reached.
(4) Call Expand($P'$).
(5) If less than $u$ unpaired nucleotides have been added, and the current structure does not
    end with unpaired nucleotides, add a single letter to each path of $P$ by following prefix
    edges; let $P'$ be the set of paths reached.
(6) Call Expand($P'$).
(7) Return.
```

The additional condition in step (5) of procedure Expand ensures that no duplicate structures are generated, that is, avoids that the same structure is obtained more than once in different iterations by adding unpaired nucleotides in different order. As we have just presented it, the algorithm can detect regions candidate to form exactly the same structure, with an apical loop of size l and at most u unpaired nucleotides along the stem. Whenever a base pair cannot be added to a structure the algorithm reports it (Report(P)), as well as the substrings folding into it (possibly if the structure contains a minimum number of base pairs). Keeping the basic idea intact, we can also allow some differences in the occurrences of a motif, that is

- motifs with apical loops of different size in their occurrences (providing a range value Δl to the algorithm for the initialization step: the algorithm will start each iteration with loop sizes ranging from l to $l + \Delta l$);
- motifs presenting in their occurrences internal loops of different size (or bulges) located at the same positions along the stem;
- motifs presenting in their occurrences internal loops of different size (or bulges), located at different positions along the stem;
- motifs differing in their occurrences in any combination of the previous points;
- any stem–loop structure that appears in the sequences is considered as an instance of the same motif.

Once again, the algorithm looks for regions of the sequences that *might* form a given structure. The energy evaluation can however be applied also to this case. In other words, we expand each path that satisfies the base pairing rules, but, before reporting a motif, we check whether the energy associated with each substring is negative (or lower than a given threshold that depends on the substring size). If the constraint is satisfied in at least one occurrence for q sequences, we report the motif.

5.1. Sequence similarity

Sequence similarity can be included also in this algorithm, for example in the choice of the initial loop candidates, or during the expansions with unpaired nucleotides. However, it is not simple to establish a priori what degree of similarity require for unpaired elements. Thus, in our experiments, we took a different approach. First, we use the algorithm to locate structural motifs, without taking into account sequence information. Then, in case more than a single occurrence per sequence is reported by the algorithm, we post-process the output by looking for similarity in the unpaired nucleotides, in order to find out which candidate regions are more likely to correspond to a conserved structural motif. This can be done by employing a greedy approach similar to the pattern-discovery method `Consensus` [15,16]. In case also the structural similarity required to the algorithm is not very stringent, we can compare different candidate motif occurrences by considering paired and unpaired nucleotides simultaneously, employing an appropriate values for matches and mismatches between paired and unpaired nucleotides [30]. The difference is that, instead of aligning and comparing whole sequences, in this case the regions to compare, as well as the structure associated with them, have already been selected, with a few occurrences in each input sequence. In this way, it is also possible to group into separate motifs regions that on a structural basis were assigned to the same motif. For example, suppose that a set of sequences contains two distinct motifs, both having a four nucleotide loop. By considering only structural similarity, the algorithm will report their occurrences as belonging to the same motif. If sequence similarity in the loop can be used to discriminate one motif from the other, then the algorithm at the end of the post-processing step will be able to report two motifs instead of one. The same considerations apply also to structural similarity. Moreover, in case more than a single motif is reported, the score of the alignment of the respective occurrences can be used as a measure of significance to rank the motifs output according to their conservation.

5.2. Complexity

The affix tree for a set of k sequences takes $O(N)$ time to be built, and has $O(N)$ nodes, where N is the overall length of the strings. Annotating it with the bit strings requires additional $O(kN)$ time. If we do not consider sequence similarity in the choice of the initial loops, ranging in size from l to L we have at most N initial paths for each length, $O(\Delta L N)$ in all, where $\Delta L = (L - l + 1)$. At each expansion, either of a base pair or an unpaired base, we just have to follow the edges of the structure (constant time) and perform a $O(k)$ time OR of the bit strings. Since the structure contains at most $O(N)$ paths from the root for each loop length, each motif expansion takes $O(k\Delta Lp)$ time, where again p is the minimum between 4^l (the maximum string depth reached) and N. The number of iterations we have to perform clearly depends on the number of different structures we have to match on the sequences, and on the degree of structural similarity we allow. For example, if we require the motif to be without any internal loop or bulge, we have to perform b iterations, where b is the number of base pairs in the stem of the longest motif discovered, and we return to the exact case complexity of $O(b\Delta Lkp)$. If we allow a single internal loop of size range Δi, and require it to appear at the same position in every instance, we have $O(b)$ choices for its location along the stem, and thus $O(b^2\Delta i\Delta Lkp)$ iterations to perform. Analogous expressions can be derived for any type of structure and approximation.

5.3. Beyond hairpins

Clearly, RNA motifs are sometimes more complex than single hairpins. An example is the Y-shaped structure of the internal ribosomal entry site (IRES) [29], or, in general, multi-loops, as shown in Fig. 1. Moreover, if we also allow crossing base pairs, as in some definitions of RNA secondary structure, we also have *pseudoknots*. However, in both cases any complex RNA element can be decomposed into single hairpins, possibly with threshold values for the size of unpaired elements connecting them, and with additional base pairs closing the structure. The idea is thus to first locate in the sequence all the single hairpins composing the structure. Then, simple post processing can determine which hairpins are located within distance thresholds specified, and/or cross each other to form pseudoknots.

6. Experimental evaluation

The algorithms described in this article have been implemented on a standard Pentium III class computer with 256 Mb of RAM running the Linux operating system. The routines for folding and free energy calculation were taken from the `RNAlib` library (part of the

Vienna RNA package [17]), using Turner energy rules [27,40]. We report here the results obtained on some typical case studies, where the presence of a motif in a set of sequences, as well as its structure, has been verified experimentally.

6.1. IRE

We briefly described the structure and function of the IRE in the first section. Although quite simple and structurally well-conserved, this motif has proved itself to be quite elusive in the years, and it is a typical benchmark for RNA analysis methods [7,10,11,18]. In fact, if we compute the free energy associated with the structure, in most of the cases it is very close to zero (corresponding to the unfolded state). Thus, if we run a prediction algorithm on a sequence known experimentally to contain the motif, it often does not appear in the prediction, nor in most of the sub-optimal structures, since the energy contribution of the motif to the overall structure is significantly small [28]. In our test, we reproduced the experiment performed with SLASH [11], taking from the UTR data base [33] the 5′ UTR region of 20 ferritin mRNA sequences containing the IRE element (fourteen were included also in the original benchmark, plus one that had been discarded because too long for SLASH, plus five new ferritin sequences added to the data base in the meantime). Sequence length ranged from 100 to about 600 nucleotides. We ran the algorithm looking for motifs appearing in all the sequences with the same loop size, and an internal loop (of unknown and variable size, possibly composed of zero nucleotides, not necessarily located at the same position) on each side of the stem. Basically, these parameters described every short hairpin structure, with the sole additional constraint that the occurrences of the motif had to have the same (unknown) apical loop size. We successfully discovered the motif, without any false positive. Also, we did not need any post-processing of the results: that is, the only motif reported was the IRE, appearing once in each sequence, in either of the two forms, matching either form of the experimentally known structure. All in all, the execution of the program, including the preliminary construction of the affix tree, took a few (less than five) seconds. To make a comparison, the running time of SLASH on the same dataset was reported to be around 10–12 h [11]. As a further test, we re-ran the algorithm searching for motifs with different structures, allowing the loop size to vary from three to twenty nucleotides, and allowing up to ten unpaired nucleotides, regardless of their position in the sequences. The only constraint we kept was requiring an apical loop of the same size in all the occurrences. The algorithm took longer (less than 1 min), but once again the only conserved motif detected was the IRE, without false positives in its occurrences.

6.2. SRP RNA

Some proteins have to be translocated across a membrane to perform their biological function. These proteins are characterized by the presence of a specific N-terminal signal sequence. The Signal Recognition Particle (SRP) binds to ribosomes while they are translating the mRNA of a protein of this kind. Then, the SRP bound to the ribosome interacts with the SRP-receptor located on plasma membranes (prokaryotes) or the endoplasmic reticulum (eukaryotes), driving the protein and the ribosome to translocational membrane pores [24]. SRP RNA (4.5S in prokaryotes and 7S in eukaryotes) is an essential component

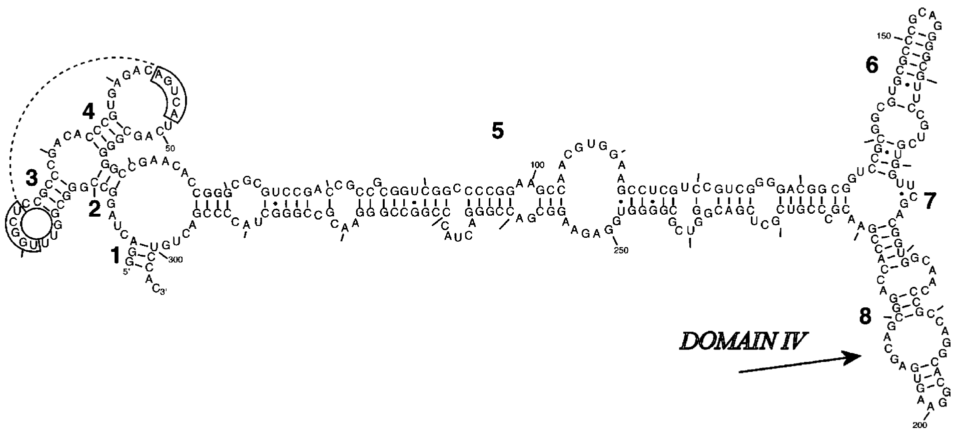

Fig. 7. The secondary structure of *Halobacterium alobium* SRP RNA [34].

```
>MET.JAN.
CCGCCAGGCCCGGAAGGGAGCAACGG
(((.....((((....))).....)))
>MET.VOL.
CCGCCAGGCCCGGAAGGGAGCAACGG
(((.....((((....))).....)))
>MET.FER.
GGT-CAGGCCCGGAAGGGAGCA-GCC
(((-....((((....)))....-)))
>MET.THE.
GGT-CAGGCCTGGAAAGGAGCA-GCC
(((-....((((....)))....-)))
>MET.ACE.
GTC-GAGGCCCGGAAGGGAGCA-GAC
(((-....((((....)))....-)))
>HAL.HAL.
CGC-CAGGCACGGAAGTGAGCA-GCG
(((-....((((....)))....-)))
>ARC.FUL.
GCC-CAGGCCCGGAAGGGAGCA-GGC
(((-....((((....)))....-)))
>PYR.ABY.
CGC-AAGGCCCGGAAGGGAGCA-GCG
(((-....((((....)))....-)))
```

Fig. 8. Alignment of the domain IV stem–loop structures of archaea SRP RNA output by the algorithm.

of the particle (see Fig. 7). Through phylogenetic comparison, SRP RNA has been divided into four structural domains (I–IV). A key part of SRP RNA is a stem–loop structure in domain IV (number 8 in the figure), that is the binding site for the protein component of the SRP. We retrieved from the SRP data base [34] all the SRP RNA sequences (without using the alignment provided for them). The set comprises 48 bacterial, 14 archaea, and

64 eukaryote sequences, ranging in size from 200 to 500 nucleotides. Then, we ran the algorithm on each separate superkingdom, with the same parameters (same hairpin size, internal loop of variable size) of the IRE test. In each case, the only motif reported had hairpin loop size four, with a few (at most around five) occurrences in each sequence. The post processing phase allowed to highlight the correct instance in each sequence, corresponding to the domain IV stem–loop structure (see Fig. 8). Each run of the algorithm (composed by the construction of the structure, pattern discovery, and post-processing) took less than 1 min.

7. Conclusions

We have presented algorithms for exact and approximate pattern matching and discovery in RNA sequences. All the algorithms permit to locate and discover motifs not only according to sequence information, but also considering the secondary structure formed by the sequences. The introduction of the affix tree permitted to obtain significant improvements, both in theory and in practice over existing methods. In fact, the implementation of the algorithms described on this paper and the tests performed on real biological instances like the ones briefly introduced at the beginning of this article have shown results comparable with the best existing methods, with a significant reduction on the time needed by the computations.

References

[1] D. Bouthinon, H. Soldano, A new method to predict the consensus secondary structure of a set of unaligned RNA sequences, Bioinformatics 15 (10) (1999) 785–798.

[2] S. Castellano, N. Morozova, M. Morey, M. Berry, F. Serras, M. Corominas, R. Guigó, In silico identification of novel selenoproteins in the *D. melanogaster* genome, EMBO Reports 21 (81) (2001) 697–702.

[3] S. Chen, K. Dill, RNA folding energy landscapes, Proc. Natl. Acad. Sci. 97 (2000) 646–651.

[4] T. Dandekar, M. Hentze, Finding the hairpin in the haystack: searching for RNA motifs, Trends Genet. 11 (1995) 45–50.

[5] S. Eddy, Computational genomics of noncoding RNA genes, Cell 109 (2002) 137–140.

[6] D. Fagegaltier, A. Lescure, E. Walczak, P. Carbon, A. Krol, Structural analysis of new local features in SECIS RNA hairpins, Nucleic Acids Res. 28 (14) (2000) 2679–2689.

[7] G. Fogel, W. Porto, D. Weekes, D. Fogel, R. Griffey, J. McNeil, E. Lesnik, D. Ecker, R. Sampath, Discovery of RNA structural elements using evolutionary computation, Nucleic Acids Res. 30 (23) (2002) 5310–5317.

[8] G. Fox, C. Woese, 5s RNA secondary structure, Nature 256 (1975) 505–507.

[9] R. Gesteland, T. Cech, J.F. Atkins (Eds.), The RNA World. Cold Spring Harbor Laboratory Press, New York, 1999.

[10] J. Gorodkin, L. Heyer, G. Stormo, Finding common sequence and structure motifs in a set of RNA sequences, Nucleic Acids Res. 25 (18) (1997) 3724–3732.

[11] J. Gorodkin, S. Stricklin, G. Stormo, Discovering common stem–loop motifs in unaligned RNA sequences, Nucleic Acids Res. 29 (10) (2001) 2135–2144.

[12] N. Gray, M. Wickens, Control of translation initiation in animals, Annu. Rev. Cell Dev. Biol. 14 (1998) 399–458.

[13] D. Gusfield, Algorithms on Strings Trees and Sequences: Computer Science and Computational Biology, Cambridge University Press, New York, 1997.

[14] M. Hentze, L. Kuhn, Molecular control of vertebrate iron metabolism: mRNA based regulatory circuits operated by iron, nitric oxide and oxidative stress, Proc. Natl. Acad. Sci. USA 93 (1996) 8175–8182.
[15] G. Hertz, G. Hartzell, G. Stormo, Identification of consensus patterns in unaligned DNA sequences known to be functionally related, Comput. Appl. Biosci. 6 (1990) 81–92.
[16] G. Hertz, G. Stormo, Identifying DNA and protein patterns with statistically significant alignment of multiple sequences, Bioinformatics 15 (1999) 563–577.
[17] I. Hofacker, W. Fontana, P. Stadler, S. Bonhoeffer, M. Tacker, P. Schuster, Fast folding and comparison of RNA secondary structures, Monatsh. Chem. 125 (1994) 167–188.
[18] Y.-J. Hu, Prediction of consensus structural motifs in a family of coregulated RNA sequences, Nucleic Acids Res. 30 (17) (2002) 3886–3893.
[19] G. Kryukov, V. Kryukov, N. Gladyshev, New mammalian selenocysteine containing proteins identified with an algorithm that searches for selenocysteine insertion sequence elements, J. Biol. Chem. 274 (48) (1999) 33888–33897.
[20] A. Laferriere, D. Gautheret, R. Cedergren, An RNA pattern matching program with enhanced performance and portability, Comput. Appl. Biosci. 10 (1994) 211–212.
[21] S.-Y. Le, J.-H. Chen, D. Konings, J. Maizel, Discovering well ordered folding patterns in nucleotide sequences, Bioinformatics 19 (3) (2003) 354–361.
[22] S.-Y. Le, W.-M. Liu, J. Maizel, A data mining approach to discover unusual folding regions in genome sequences, Knowledge Based Systems 15 (2002) 243–250.
[23] A. Lescure, D. Gautheret, P. Carbon, A. Krol, Novel selenoproteins identified in silico and in vivo by using a conserved RNA structural motif, J. Biol. Chem. 274 (53) (1999) 38147–38154.
[24] H. Lutcke, Signal recognition particle (SRP), a ubiquitous initiator of protein translocation, European J. Biochem. 228 (3) (1995) 531–550.
[25] M. Maass, Linear bidirectional on-line construction of affix trees, in: Proc. CPM 2000, Lecture Notes in Computer Science, Vol. 1848, Springer, Berlin, 2000, pp. 320–334.
[26] T. Macke, D. Ecker, R. Gutell, D. Gautheret, D. Case, R. Sampath, RNAmotif, an RNA secondary structure definition and search algorithm, Nucleic Acids Res. 29 (22) (2001) 4724–4735.
[27] D. Mathews, J. Sabina, M. Zucker, D. Turner, Expanded sequence dependence of thermodynamic parameters provides robust prediction of RNA secondary structure, J. Mol. Biol. 288 (1999) 911–940.
[28] G. Mauri, G. Pavesi, Pattern discovery in RNA secondary structure using affix trees, in: Proc. CPM 2003, Lecture Notes in Computer Science, Vol. 2676, Springer, Berlin, 2003, pp. 278–294.
[29] V. Pain, Initiation of protein synthesis in eukaryotic cells, European J. Biochem. 236 (1996) 747–771.
[30] G. Pavesi, Aligning RNA sequences and their secondary structures, Technical Report, University of Milano Bicocca, 2003.
[31] D. Pervouchine, J. Graber, S. Kasif, On the normalization of RNA equilibrium free energy to the length of the sequence, Nucleic Acids Res. 31 e49.
[32] G. Pesole, S. Liuni, M. D'Souza, Patsearch: a pattern matcher software that finds functional elements in nucleotide and protein sequences and assesses their statistical significance, Bioinformatics 16 (5) (2000) 439–450.
[33] G. Pesole, S. Liuni, G. Grillo, F. Licciulli, F. Mignone, C. Gissi, C. Saccone, UTRdb and UTRsite: specialized databases of sequences and functional elements of $5'$ and $3'$ untranslated regions of eukaryotic mRNAs, update 2002, Nucleic Acids Res. 30 (1) (2002) 335–340.
[34] M. Rosenblad, J. Gorodkin, B. Knudsen, C. Zwieb, T. Samuelsson, SRPDB: signal recognition particle database, Nucleic Acids Res. 31 (1) (2003) 363–364.
[35] E. Rivas, S. Eddy, Secondary structure alone is generally not statistically significant for the detection of noncoding RNA, Bioinformatics 16 (7) (2000) 583–605.
[36] W. Stephan, J. Parsch, J. Braverman, Comparative sequence analysis and patterns of covariation in RNA secondary structures, Genetics 154 (2) (2000) 909–921.
[37] J. Stoye, Affix trees. Technical Report 2000-04, University of Bielefeld, 2000.
[38] R. Simons, M. Grumberg Magnago (Eds.), RNA Structure and Function, Cold Spring Harbor Laboratory Press, New York, 1998.
[39] F. Tahi, M. Gouy, M. Regnier, Automatic RNA secondary structure prediction with a comparative approach, Comput. Chem. 26 (2002) 521–530.
[40] A. Walter, D. Turner, J. Kim, M. Lyttle, P. Muller, D. Mathews, M. Zuker, Coaxial stacking of helices enhances binding of oligoribonucleotides, Proc. Natl. Acad. Sci. 91 (1994) 9218–9222.

[41] E. Westhof, E. Auffinger, C. Gaspin, DNA and RNA structure prediction, in: M.J. Bishop, C.J. Rawlings (Eds.), DNA–Protein Sequence Analysis, Oxford, 1996, pp. 255–278.
[42] A. Williams, W. Marzluff, The sequence of the stem and flanking sequences at the 3′ end of histone mRNA are critical determinants for the binding of the stem–loop binding protein, Nucleic Acids Res. 23 (4) (1996) 654–662.
[43] C. Witwer, S. Rauscher, I. Hofacker, P. Stadler, Conserved RNA secondary structures in picornaviridae genomes, Nucleic Acids Res. 29 (2001) 5079–5089.