

Structural Alignment of Pseudoknotted RNA

*BUHM HAN,¹ *BANU DOST,¹ VINEET BAFNA,¹ and SHAOJIE ZHANG²

ABSTRACT

In this paper, we address the problem of discovering novel non-coding RNA (ncRNA) using primary sequence, and secondary structure conservation, focusing on ncRNA families with pseudoknotted structures. Our main technical result is an efficient algorithm for computing an optimum structural alignment of an RNA sequence against a genomic substring. This algorithm has two applications. First, by scanning a genome, we can identify novel (homologous) pseudoknotted ncRNA, and second, we can infer the secondary structure of the target aligned sequence. We test an implementation of our algorithm (PAL) and show that it has near-perfect behavior for predicting the structure of many known pseudoknots. Additionally, it can detect the true homologs with high sensitivity and specificity in controlled tests. We also use PAL to search entire viral genome and mouse genome for novel homologs of some viral and eukaryotic pseudoknots, respectively. In each case, we have found strong support for novel homologs.

Key words: dynamic programming, pseudoknotted RNA, RNA alignment, RNA secondary structure.

1. INTRODUCTION

RIBONUCLEIC ACID (RNA) is the third, and (until recently) most underrated of the trio of molecules that govern most cellular processes; the other two are proteins and DNA. While much cellular RNA carries a message encoding an amino-acid sequence, other, “non-coding” (nc) RNA participates directly in performing essential functions. Recent and unanticipated discoveries of novel ncRNA families (Argaman et al., 2001; Novina and Sharp, 2004; Storz, 2002; Vitreschak et al., 2003; Winkler and Breaker, 2003) point to the possibility of a “Modern RNA world” in which RNA molecules are as abundant and diverse as protein molecules (Eddy, 2001). The analog of the computational gene-finding problem—“given genomic DNA, identify all substrings that encode ncRNA”—is increasingly relevant, and relatively unexplored.

While potentially abundant, RNA signals are weaker than proteins, making them harder to identify computationally. Possibly the strongest clue is from secondary structure. Being single-stranded, the base-pairs stabilize by forming hydrogen bonds, leading to a characteristic secondary and tertiary structure. With a few exceptions, the base-pairs are *non-crossing* and form a tree-like structure. This recursive structure

¹Department of Computer Science and Engineering, University of California, San Diego, La Jolla, California.

²School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida.

*These authors contributed equally in this work.

is the basis for efficient algorithms to predict RNA structure (Jaeger et al., 1989; Zuker and Sankoff, 1984). With this extensive work in structure prediction, it is natural to expect that novel ncRNA could be discovered simply by looking for genomic sub-strings that fold into low-energy structures. Unfortunately, that idea does not work. Rivas and Eddy (2000) showed that random DNA (usually with high guanine-cytosine (GC) content) can also “fold” into low-energy configurations, making it unlikely for a purely *de novo* approach to be successful. Therefore, a comparative approach is employed, often typified by the question: “Given a query RNA with known structure, and a genome, identify all genomic sub-strings that match the query sequence and structure.” The query itself can be either a single molecule or a model (covariance model/stochastic context-free grammar) of an RNA structure. This approach has been quite successful, and single queries as well as covariance based models are routinely used to annotate genomes with ncRNA (Griffiths-Jones et al., 2005; Klein and Eddy, 2003). Central to these approaches is an algorithm for computing a local alignment between a query structure and a DNA string. The search itself is simply a scan of the genome to obtain all high scoring local alignments.

Here we pose a related question: *Given a query RNA with known structure, allowing for pseudoknots, and a genome, identify all genomic sub-strings that match the query sequence and structure.* Without being precise, pseudoknots are base-pairs that violate the non-crossing rule (Fig. 1). While not as common as other sub-structures (e.g., bulges, loops), they are often critically important to function. Pseudoknotted RNAs are known to be active as ribozymes (Rastogi et al., 1996) and self-splicing introns (Adams et al., 2004), and to participate in telomerase activity (Theimer et al., 2005). They have also been shown to alter gene expression by inducing ribosomal frame-shifting in many viruses (Nixon et al., 2002). However, understanding the extent and importance of these molecules is partially handicapped by the difficulty of identifying them (computationally). The algorithm presented here will facilitate identification.

In order to compute a local structural alignment, we must start with a formal definition of a pseudoknot in Section 2. Many definitions of pseudoknots have been postulated (Akutsu, 2000; Dirks and Pierce,

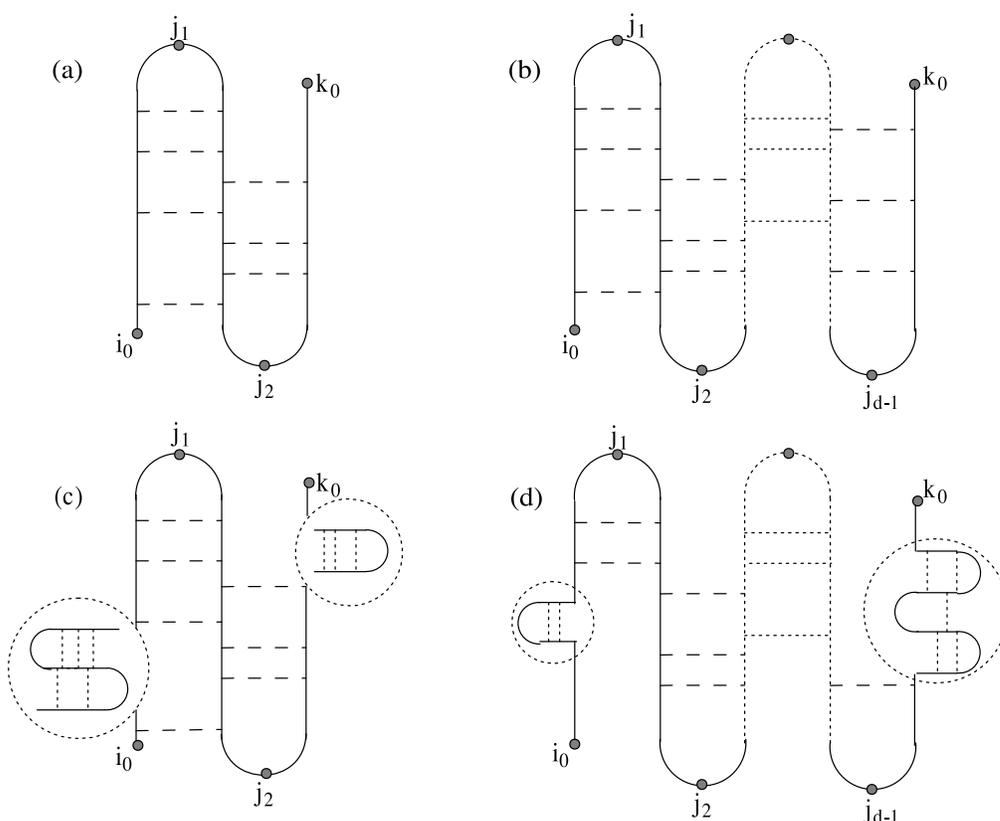


FIG. 1. (a) Simple pseudoknot. (b) Standard pseudoknot of degree d . (c) Recursive simple pseudoknot. (d) Recursive standard pseudoknot of degree d .

2003; Evans, 1964; Jiang et al., 2002; Rivas and Eddy, 1999), and recent research investigates the power of these definitions in describing real pseudoknots (Condon et al., 2004). We start here with Akutsu's (2000) formalism (*simple* pseudoknots), which has a clean recursive structure and encompasses a majority of the known cases (Condon et al., 2004; Rastegari and Condon, 2005). We also present algorithms that extend this class of allowed pseudoknots (standard pseudoknots). Section 3 describes the chaining procedure, which is key to the alignment algorithm that follows in Section 4. However, the simple pseudoknots usually do not occur independently, but are embedded in regular RNA structures. In Section 5, we extend the algorithm to handle these cases. Other extensions are considered in Section 7. It has been brought to our attention that a recent publication (Matsui et al., 2005) considers the identical problem using the formation of tree adjoining grammars to model pseudoknots. The pseudoknots considered by them are a restricted version of our simple pseudoknots. Furthermore, our alignment combines sequence and structural similarity.

The local alignments can be used in two ways. First, they can be used to infer the structure of the aligned substring that is conserved with the query. We show in Section 8.1 that, in a majority of the cases, this leads to a perfect prediction of secondary (pseudoknotted) structure. Next, they can be used to predict novel ncRNA in genomic sequences. While our algorithms are computationally intensive, they can be used in combination with database filtering approaches to search large genomic regions. In Section 8.2, we validate our approach on real sequences embedded in random sequence. Finally, in Section 9, we identify (putative) novel pseudoknotted ncRNA in a search of viral and eukaryotic genomes.

2. DEFINITIONS AND PRELIMINARY INFORMATION

Let $A = a_1, \dots, a_n$ be an RNA sequence. The secondary structure is represented simply as the set of base-pairs

$$M = \{(i, j) | 1 \leq i < j \leq n, (a_i, a_j) \text{ is a base pair}\}$$

Also, let $M_{i_0, k_0} \subseteq M$ be defined by $M_{i_0, k_0} = \{(i, j) \in M | i_0 \leq i < j \leq k_0\}$. The secondary structure, in the absence of crossing or interweaving base-pairs, is called *regular* and has the following recursive definition.

Definition 1. *An RNA secondary structure M_{i_0, k_0} is regular if and only if $M_{i_0, k_0} = \phi$ or $\exists (i, j) \in M_{i_0, k_0}$ such that*

- $M_{i_0, k_0} = M_{i_0, i-1} \cup M_{i+1, j-1} \cup M_{j+1, k_0} \cup \{(i, j)\}$ (no base-pairs cross the partitions).
- Each of $M_{i_0, i-1}$, $M_{i+1, j-1}$, M_{j+1, k_0} is regular.

Next, we can define the class of allowed pseudoknots (Akutsu, 2000).

Definition 2. *M_{i_0, k_0} is a simple-pseudoknot (Fig. 1a) if and only if M_{i_0, k_0} is regular or $\exists j_1, j_2 \in \mathbb{N}$ ($i_0 \leq j_1 < j_2 \leq k_0$) such that the resulting partition, $D_1 = [i_0, j_1 - 1]$, $D_2 = [j_1, j_2 - 1]$, $D_3 = [j_2, k_0]$, satisfies the following:*

- $M_{i_0, k_0} = (S_L \cup S_R)$, where $S_L = \{(i, j) \in M_{i_0, k_0} | i \in D_1, j \in D_2\}$ and $S_R = \{(i, j) \in M_{i_0, k_0} | i \in D_2, j \in D_3\}$.
- S_L and S_R are regular.

Definition 3. *M_{i_0, k_0} is a standard-pseudoknot with degree d ($d \geq 3$; Fig. 1b) if and only if M_{i_0, k_0} is regular or $\exists j_1, \dots, j_{d-1} \in \mathbb{N}$ ($i_0 \leq j_1 < \dots < j_{d-1} \leq k_0$) which divide $[i_0, k_0]$ into d parts, $D_1 = [i_0, j_1 - 1]$, $D_2 = [j_1, j_2 - 1]$, \dots , $D_d = [j_{d-1}, k_0]$, and satisfy the following:*

- $M_{i_0, k_0} = \bigcup_{l=1}^{d-1} S_l$, where $S_l = \{(i, j) \in M_{i_0, k_0} | i \in D_l, j \in D_{l+1}\}$ for all $1 \leq l < d$.
- S_l is regular for all $1 \leq l < d$.

Note that a simple-pseudoknot is a standard-pseudoknot of degree 3.

Definition 4. M_{i_0,k_0} is recursive-standard-pseudoknot with degree d ($d \geq 3$; see Fig. 1d) if and only if M_{i_0,k_0} is a standard pseudoknot of degree d or $\exists i_1, k_1, \dots, i_t, k_t \in \mathbb{N}$ ($i_0 \leq i_1 < k_1 < i_2 < k_2 < \dots < i_t < k_t \leq k_0, t \geq 1$), which satisfy the following:

- $(M_{i_0,k_0} - \bigcup_{l=1}^t M_{i_l,k_l})$ is a standard pseudoknot of degree $\leq d$.
- M_{i_l,k_l} ($1 \leq l \leq t$) is a recursive standard pseudoknot of degree $\leq d$.

A recursive-simple-pseudoknot is a recursive-standard-pseudoknot of degree 3 (Fig. 1c). While we can devise algorithms to align recursive-standard-pseudoknots, they are computationally expensive, and most known families have a simpler structure. Therefore, we will limit our description and tests to a simpler structure (with a single level of recursion), defined as follows:

Definition 5. M_{i_0,k_0} is embedded-simple-pseudoknot if and only if $\exists i_1, k_1, \dots, i_t, k_t \in \mathbb{N}$ ($i_0 \leq i_1 < k_1 < i_2 < k_2 < \dots < i_t < k_t \leq k_0, t \geq 1$), which satisfy the following:

- $(M_{i_0,k_0} - \bigcup_{l=1}^t M_{i_l,k_l})$ is regular.
- M_{i_l,k_l} ($1 \leq l \leq t$) is a simple-pseudoknot.

2.1. Structural alignment preliminaries

For alignment purposes, we do not distinguish between RNA and DNA, as every substring in the genome might encode an RNA string. Let $q[1 \dots m]$ and $t[1 \dots n]$ be two RNA strings over the alphabet $\Sigma = \{A, C, G, U\}$ where q has a known structure M . An alignment of q and t is defined by a 2-rowed matrix A , in which row 1 (respectively, 2) contains q (respectively, t) interspersed with spaces, and for all columns j , $A[1, j] \neq -'$ or $A[2, j] \neq -'$. For $r \in \{1, 2\}$, define $\iota_r[i] = i - |\{l < i \text{ s.t. } A[r, l] = -'\}|$. In other words, if $A[1, i] \neq -'$, it contains the symbol $q[\iota_1[i]]$. The score of alignment A is given by

$$\sum_j \gamma(A[1, j], A[2, j]) + \sum_{i, j \text{ s.t. } (\iota_1[i], \iota_1[j]) \in M} \delta(\iota_1[i], \iota_1[j], \iota_2[i], \iota_2[j])$$

The function γ scores for sequence similarity, whereas δ scores for conservation of structure. While this formulation encodes a linear gap penalty, we note here that alignments of RNA molecules may contain large gaps, particularly in the loop regions, and we implement affine penalties for gaps. Naturally, we wish to compute alignments with the maximum score.

The key ideas are as follows: First, note that regular and pseudoknotted structures have a recursive formulation. Therefore, the problem of structurally aligning an RNA structure against a subsequence, can be decomposed into the problems of (recursively) aligning its sub-structures against the appropriate subsequences, and combining the results. For regular structures, the structure is tree-like, and the recursion follows the nodes of the tree. For simple-pseudoknots, the structure is more complex, and will be described in Section 4. The structure for embedded-simple-pseudoknots is simply a combination of the two (see Section 7).

However, it is not sufficient to consider structural elements alone, as we wish to score for sequence conservation as well. The recursive structure described only contains a subset of the nucleotides that participate in structure. Therefore, we employ a second trick of introducing spurious structural elements (base-pairs) to M . The augmented structure M' must have the following properties:

- Each nucleotide i appears in M' .
- $|M'| = O(m)$, so that the size of the structure does not increase too much.
- The recursive structure of M is maintained.

Pseudoknots and regular structures have very different recursive structure, and require different augmentation procedures. In Section 3, we present *chaining*, a novel augmentation procedure for simple pseudoknots. An augmentation for regular structures, *binarization* was presented in Bafna et al. (1995), and is implicit in the covariance models used to align regular RNA (Durbin et al., 1998). Here, we extend binarization to include chaining for embedded-simple-pseudoknots (Fig. 5). These augmentations are used in the alignment algorithms for simple-pseudoknots (Section 4), embedded-simple-pseudoknots (Section 5), and standard-pseudoknots (Section 7).

3. CHAINING

Before describing the chaining procedure, we revisit the problem of aligning a simple pseudoknot to a genomic sub-string. Unlike regular structures, we cannot partition the genome into contiguous substrings, because of interweaving base pairs. Thus, we need a new substructure for simple pseudoknot structures. We start by defining a total ordering among the base pairs of a simple pseudoknot. Recall (Definition 3) that a simple-pseudoknot structure M_{i_0, k_0} can be divided into three parts: $D_1 = [i_0, j_0 - 1]$, $D_2 = [j_0, j'_0 - 1]$, $D_3 = [j'_0, k_0]$ (Fig. 2a). For each base pair $(i, j) \in M$, exactly one of i and j is in D_2 part. We define an ordering of the base pairs in M by sorting the coordinate in D_2 . Formally, define $D_2(i, j)$ for all $(i, j) \in M$ as follows: $D_2(i, j) = i$ if $(i, j) \in S_R$, and $D_2(i, j) = j$ otherwise. For each $(i, j), (i', j') \in M$,

$$(i, j) \geq_p (i', j') \text{ iff } D_2(i, j) \geq D_2(i', j').$$

As distinct base-pairs do not share any coordinates, \geq_p defines a total ordering on the actual base-pairs, and can be used to define a partial order on substructures that we can recurse on. Define a *subpseudoknot* $\mathcal{P}(i, j, k)$ as the union of two subintervals $\mathcal{P}(i, j, k) = [i_0, i] \cup [j, k]$ (Fig. 2b). Denote the triple (i, j, k) as the *frontier* for $\mathcal{P}(i, j, k)$. Note that i_0 is implicit from the context. Suppose that we are aligning frontier (i', j', k') of the query against frontier (i, j, k) of the target, with the score represented by $B[i, j, k, i', j', k']$. A naive algorithm would need to consider $O(m^3 n^3)$ pairs of frontiers. We improve this as follows: consider the special case of $(i', j') \in M$ where $(i', j') \in S_L$. The following recursion gives the score for B :

Theorem 1.

$$B[i, j, k, i', j', k'] = \max\{MATCH, INSERT, DELETE\}$$

$$\begin{aligned} MATCH &= B[i - 1, j + 1, k, i' - 1, j' + 1, k'] + \delta(q[i'], q[j'], t[i], t[j]) \\ &\quad + \gamma(q[i'], t[i]) + \gamma(q[j'], t[j]), \end{aligned}$$

$$DELETE = \max \begin{cases} B[i - 1, j, k, i' - 1, j' + 1, k'] + \gamma(q[i'], t[i]) + \gamma(q[j'], '-'), \\ B[i, j + 1, k, i' - 1, j' + 1, k'] + \gamma(q[i'], '-') + \gamma(q[j'], t[j]), \\ B[i, j, k, i' - 1, j' + 1, k'] + \gamma(q[i'], '-') + \gamma(q[j'], '-') \end{cases}$$

$$INSERT = \max \begin{cases} B[i - 1, j, k, i', j', k'] + \gamma('-', t[i]), \\ B[i, j + 1, k, i', j', k'] + \gamma('-', t[j]), \\ B[i, j, k - 1, i', j', k'] + \gamma('-', t[k]) \end{cases}$$

Note that in every sub-case of MATCH and DELETE, we move from the query frontier (i', j', k') to the frontier $(i' - 1, j' + 1, k)$, because if either i' or j' is not used, we cannot score for the pair (i', j') . In the INSERT case, we stay at the frontier (i', j', k') . The situation is symmetric when $(j', k') \in S_R \subseteq M$, but is not defined when $(i', j') \notin M \wedge (j', k') \notin M$. The key idea for the chaining procedure is that we can define a unique frontier to move to in all cases, and still ensure that each nucleotide is touched by at least one frontier. By starting with a fixed frontier, and always moving to a fixed child, we only have $O(m)$ frontiers to consider.

From Definition 2, there exist indices j_1, j_2 that divide the simple pseudoknot structure into D_1 , D_2 , and D_3 . We choose $(j_1 - 1, j_1, k_0)$ as the *root* frontier. Note that $\mathcal{P}(j_1 - 1, j_1, k_0)$ represents the entire simple-pseudoknot (Fig. 3a). We maintain the invariant that if (i, j, k) is a frontier and j participates in a base-pair, then the base-pair must be “below” or within the frontier. In other words, if $(i', j) \in S_L$, then $i' \leq i$. Likewise, if $(j, k') \in S_R$, then $k' \leq k$. For a frontier (i, j, k) , we have different cases: for example, if $(i', j) \in S_L$, we add spurious base pairs $(i, j), (i - 1, j), \dots, (i', j)$. These base pairs define an ordered

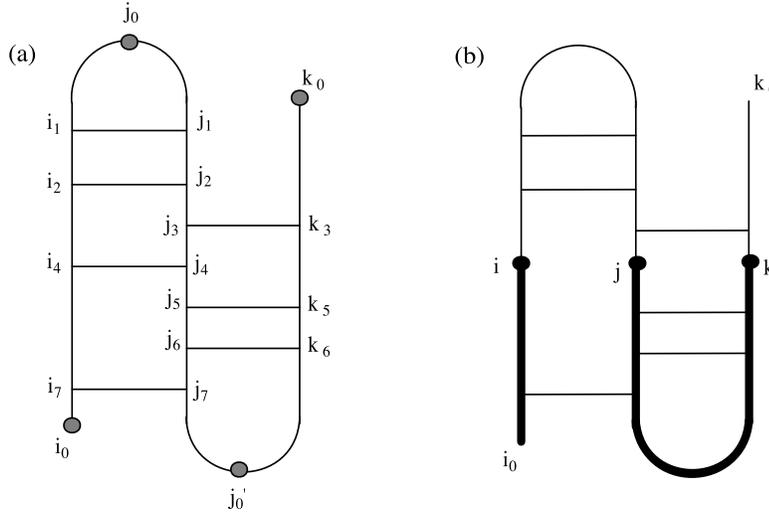


FIG. 2. (a) Base pairs in a simple pseudoknot are ordered according to the index of the endpoint along $[j_0, j'_0]$. Therefore, $(i_1, j_1) > (i_2, j_2) > (j_3, k_3) > (i_4, j_4) > (j_5, k_5) > (j_6, k_6) > (i_7, j_7)$. (b) Subpseudoknot structure.

```

CHAINING( $i, j, k$ )
1  if  $i = i_0 - 1$  and  $j > k$ 
2  then return NIL
3  if  $(i, j) \in S$ 
4  then  $v = \text{CHAINING}(i - 1, j + 1, k)$ ;
5  return CREATE $\text{NODE}(i, j, \text{solid}, \text{move}(1, 1, 0), v)$ 
6  if  $(j, k) \in S$ 
7  then  $v = \text{CHAINING}(i, j + 1, k - 1)$ ;
8  return CREATE $\text{NODE}(j, k, \text{solid}, \text{move}(0, 1, 1), v)$ 
9  if  $j \in V_L$ 
10 then  $v = \text{CHAINING}(i - 1, j, k)$ ;
11 return CREATE $\text{NODE}(i, j, \text{empty}, \text{move}(1, 0, 0), v)$ 
12 if  $j \in V_R$ 
13 then  $v = \text{CHAINING}(i, j, k - 1)$ ;
14 return CREATE $\text{NODE}(j, k, \text{empty}, \text{move}(0, 0, 1), v)$ 
15 if  $i \in V_L$ 
16 then  $v = \text{CHAINING}(i, j + 1, k)$ ;
17 return CREATE $\text{NODE}(i, j, \text{empty}, \text{move}(0, 1, 0), v)$ 
18 if  $k \in V_R$ 
19 then  $v = \text{CHAINING}(i, j + 1, k)$ ;
20 return CREATE $\text{NODE}(j, k, \text{empty}, \text{move}(0, 1, 0), v)$ 
21 if  $i > i_0$ 
22 then  $v = \text{CHAINING}(i - 1, j, k)$ ;
23 return CREATE $\text{NODE}(i, j, \text{empty}, \text{move}(1, 0, 0), v)$ 
24 if  $i = i_0$ 
25 then  $v = \text{CHAINING}(i - 1, j + 1, k)$ ;
26 return CREATE $\text{NODE}(i, j, \text{empty}, \text{move}(1, 1, 0), v)$ 
27 if  $i = i_0 - 1$ 
28 then  $v = \text{CHAINING}(i, j + 1, k)$ ;
29 return CREATE $\text{NODE}(j, k, \text{empty}, \text{move}(0, 1, 0), v)$ 
    
```

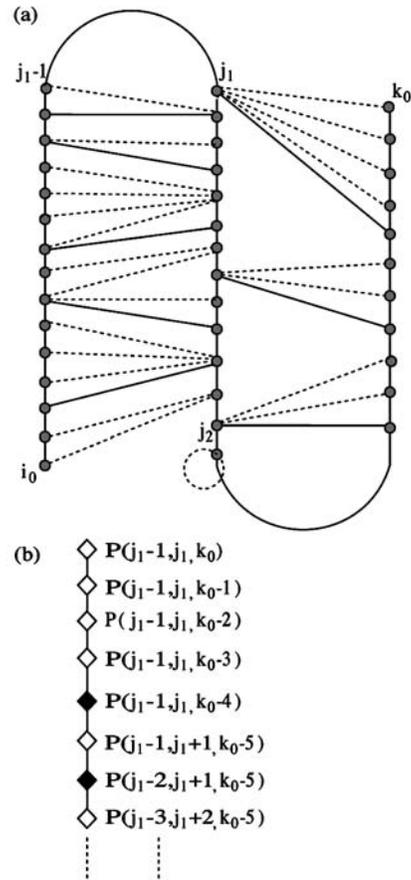


FIG. 3. The chaining procedure on a simple pseudoknot structure M_{i_0, k_0} . (a) Solid base pairs are the actual base pairs, dotted ones are the spurious base pairs. (b) Chain structure representing the simple pseudoknot structure M_{i_0, k_0} . Solid nodes represents a sub-pseudoknot with frontier (i, j, k) where (i, j) or (j, k) is an actual base pair. Empty nodes represents a sub-pseudoknot with frontier (i, j, k) where neither (i, j) nor (j, k) is an actual base pair.

set of frontiers $(i, j, k) \geq (i-1, j, k) \geq \dots, (i', j, k) \geq (i'-1, j+1, k)$. Likewise, if $(j, k') \in S_R$, we add spurious base-pairs $(j, k), (j, k-1), \dots, (j, k')$, which define the frontiers $(i, j, k) \geq \dots \geq (i, j+1, k'-1)$. The chaining algorithm, with a complete listing of cases is described in Figure 3. The output of chaining is a directed path of “frontiers.” The number of nucleotides in a frontier (i, j, k) is given by the expression $((i-i_0+1)+(k-j+1)) \leq m$. Further, this number decreases by at least 1 for each adjacent frontier. Thus the number of nodes in the chain is $O(m)$. We still need to consider $O(n^3)$ target frontiers in aligning, for a complexity of $O(mn^3)$.

4. ALIGNMENT ALGORITHM FOR SIMPLE-PSEUDOKNOTS

Figure 4a describes the algorithm ALIGN-SP for aligning a simple-pseudoknot to a DNA substring. Its input is a chain of query sub-pseudoknots, which is aligned to all sub-pseudoknots $\mathcal{P}(i, j, k)$ of the target sequence $t[1 \dots n]$. Let M_L (respectively, M_R) be the set of solid nodes representing subpseudoknots $\mathcal{P}(i, j, k)$ where $(i, j) \in S_L$ (respectively, $(j, k) \in S_R$). Let M_S be set of the nodes representing subpseudoknots $\mathcal{P}(i, j, k)$ where neither $(i, j) \notin S_L$, and $(j, k) \notin S_R$.

As an example, suppose we are aligning sub-pseudoknot $\mathcal{P}(i, j, k)$ in t to the subchain rooted at v . Let $B[i, j, k, v]$ be the score of the optimal alignment. First, we have cases involving insertion of target nucleotides: $t[i], t[j]$, and $t[k]$, as described by the recurrence in Figure 4a (line 15). Next, we have the cases corresponding to match or deletion of v . We consider the case $v \in M_L$ corresponding to the subpseudoknot $\mathcal{P}(l_v, m_v, r_v)$ in q . The following cases can occur

1. $(t[i], t[j])$ is a pair in t corresponding to the pair $(q[l_v], q[m_v])$ in q .
2. $q[l_v]$ is substituted with $t[i]$ and $q[m_v]$ is deleted.
3. $q[m_v]$ is substituted with $t[j]$ and $q[l_v]$ is deleted.
4. $q[l_v]$ and $q[m_v]$ are both deleted.

The corresponding recurrences are shown on Line 6 of the procedure in Figure 4. The other cases are handled in an analogous fashion and are described in Figure 4a.

5. ALIGNMENT ALGORITHM FOR EMBEDDED-SIMPLE-PSEUDOKNOTS

We consider now the special case of aligning recursive-simple-pseudoknots in which simple-pseudoknots are embedded in a regular structure. This is by far the most common occurrence of pseudoknots. While it is relatively easy to extend our algorithms to handle the full generality of recursive-pseudoknots, the complexity increase makes the algorithms untractable for real problems. Thus, this special case offers a compromise between generality and practicality.

The first step in the procedure is to *binarize* the query RNA, so that every nucleotide is in a base-pair, and can be represented by a binary tree of size $O(m)$ (Bafna et al., 1995). The main difference is that we invoke the chaining procedure whenever a simple-pseudoknot is encountered. Thus, in the binary tree, the simple pseudoknot substructure appears as a chain rooted at a *pseudo-node* (Fig. 5).

After the binary tree structure M' of query sequence q is created, target sequence t is aligned to this tree. The following procedure ALIGN shown in Figure 6 aligns a given subsequence $(t[i \dots j])$ in target sequence to a subtree of M' . The scores of optimal alignments are stored in matrix A . The entry $A[i, j, v]$ keeps the optimal alignment of the subproblem of aligning a subsequence $(t[i], t[j])$ to the subtree rooted at the node v —in other words, to the subinterval $(q[l_v], q[r_v])$ of the query sequence.

6. COMPLEXITY

In Align-SP, lines 3–15 run in $O(n^3)$ time to align all subpseudoknots in target to a node. Those lines are executed for each subinterval (i_0, k_0) in target and for each node in the query tree. Then, time complexity of procedure Align-SP becomes $O(mn^5)$. However, we do not need to compute $O(n^3)$ scores for each subinterval (i_0, k_0) . Since k_0 does not appear in the recurrences of Align-SP procedure and $B[i, j, k]$ does

(a)

```

ALIGN-SP( $M', t[1..n]$ )
1 //  $M'$  is the chain representing the simple pseudoknot region to be aligned in query  $q$ 
2 for all intervals  $(i_0, k_0)$  in  $t[1..n]$ 
3 do for all  $(i, j, k), i_0 \leq i < j \leq k \leq k_0$ 
4 do for all nodes  $v \in M'$ 
5 do if  $v \in M_L$ 
6 then  $B[i, j, k, v] = \max \begin{cases} B[i-1, j+1, k, \text{child}(v)] + \delta(q[l_v], q[m_v], t[i], t[j]) \\ \quad + \gamma(q[l_v], t[i]) + \gamma(q[m_v], t[j]), \\ B[i-1, j, k, \text{child}(v)] + \gamma(q[l_v], t[i]) + \gamma(q[m_v], '-'), \\ B[i, j+1, k, \text{child}(v)] + \gamma(q[l_v], '-') + \gamma(q[m_v], t[j]), \\ B[i, j, k, \text{child}(v)] + \gamma(q[l_v], '-') + \gamma(q[m_v], '-') \end{cases}$ 
7 if  $v \in M_R$ 
8 then  $B[i, j, k, v] = \max \begin{cases} B[i, j+1, k-1, \text{child}(v)] + \delta(q[m_v], q[r_v], t[j], t[k]) \\ \quad + \gamma(q[m_v], t[j]) + \gamma(q[r_v], t[k]), \\ B[i, j, k-1, \text{child}(v)] + \gamma(q[m_v], '-') + \gamma(q[r_v], t[k]), \\ B[i, j+1, k, \text{child}(v)] + \gamma(q[m_v], t[j]) + \gamma(q[r_v], '-'), \\ B[i, j, k, \text{child}(v)] + \gamma(q[m_v], '-') + \gamma(q[r_v], '-') \end{cases}$ 
9 if  $v \in M_S$  and  $\text{move}(v) = (1, 0, 0)$ 
10 then  $B[i, j, k, v] = \max \begin{cases} B[i-1, j, k, \text{child}(v)] + \gamma(q[l_v], t[i]), \\ B[i, j, k, \text{child}(v)] + \gamma(q[l_v], '-') \end{cases}$ 
11 if  $v \in M_S$  and  $\text{move}(v) = (0, 0, 1)$ 
12 then  $B[i, j, k, v] = \max \begin{cases} B[i, j, k-1, \text{child}(v)] + \gamma(q[r_v], t[k]), \\ B[i, j, k, \text{child}(v)] + \gamma(q[r_v], '-') \end{cases}$ 
13 if  $v \in M_S$  and  $\text{move}(v) = (0, 1, 0)$ 
14 then  $B[i, j, k, v] = \max \begin{cases} B[i, j+1, k, \text{child}(v)] + \gamma(q[m_v], t[k]), \\ B[i, j, k, \text{child}(v)] + \gamma(q[m_v], '-') \end{cases}$ 
15  $B[i, j, k, v] = \max \begin{cases} B[i, j, k, v] \\ B[i-1, j, k, v] + \gamma('-', t[i]), \\ B[i, j+1, k, v] + \gamma('-', t[j]), \\ B[i, j, k-1, v] + \gamma('-', t[k]) \end{cases}$ 
17  $B_{SP}[i_0, k_0, i_{SP}, k_{SP}] = \max_{j=i+1, k=k_0} \{B(i, j, k, \text{ROOT}(M'))\}$ 

```

(b)

```

IMPROVED ALIGN-SP()
1 for all  $v \in M'$ 
2 do for  $i_0 = 1$  to  $n-1$ 
3 do for  $i = i_0 - 1$  to  $n-1$ 
4 do for  $j = n+1$  downto  $i+1$ 
5 do for  $k = j-1$  to  $n$ 
6 do Compute  $B[i, j, k, v]$ 

```

FIG. 4. (a) Align-SP procedure for alignment of a simple pseudoknot structure to a target sequence $t[1..n]$. (b) Improved Align-SP procedure.

not depend on $B[i', j', k']$ such that $k' > k$, $B[i, j, k]$ does not depend on k_0 . Thus, it is enough to compute $O(n^3)$ scores for each i_0 as shown in Figure 4b. Then, total running time of Align-SP is $O(mn^4)$.

In Align procedure in Figure 6, we first call Binarization-SP procedure which runs in $O(m)$ time. We also call Align-SP procedure whenever we encounter a pseudonode in the binary tree formed. Let m_p be the length of the pseudoknot regions in $q[1..m]$, m_1 and m_2 be the number of the nodes with one child and two children in the binary tree of q representing the regions with regular structure. Then, the total running time of Align procedure will be $O(m_p n^4 + m_1 n^2 + m_2 n^3)$. It is useful to note that very often, $m_p, m_2 \in o(m)$, and so the true complexity is better than the worst case complexity. Also, in computing

```

BINARIZE-SP( $i, j$ )
1  if ( $i, j$ ) is a simple pseudoknot structure
2  then return CHAINING( $i, j, pseudo - node, Nil$ );
3  if  $i = j$ 
4  then return CREATENODE( $i, j, empty, Nil$ );
5  if ( $i, j$ )  $\in M$ 
6  then  $v =$  BINARIZE-SP( $i + 1, j - 1$ );
7  return CREATENODE( $i, j, solid, v$ );
8  if ( $k, j$ )  $\in M$  for some  $i < k < j$ 
9  then
10      $vl =$  BINARIZE-SP( $i, k - 1$ );
11      $vr =$  BINARIZE-SP( $k, j$ );
12     (A empty node with 2 children,  $vl$  and  $vr$ .)
13     return CREATENODE( $i, j, empty, vl, vr$ );
14 if  $i < j$ 
15 then  $v =$  BINARIZE-SP( $i, j - 1$ );
16 return CREATENODE( $i, j, empty, v$ );

```

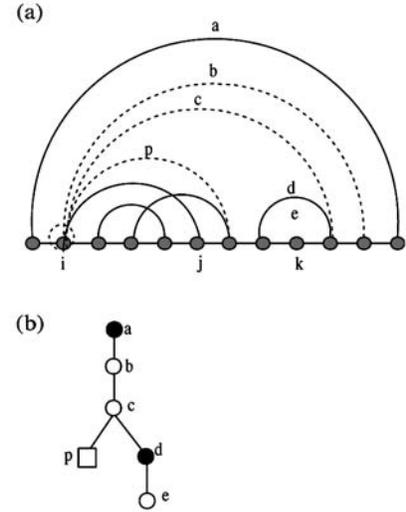


FIG. 5. Binarization procedure revised for embedded-simple-pseudoknots and an illustration. **(a)** An embedded-simple-pseudoknot with spurious base pairs added. **(b)** Resulting binary tree. Solid nodes correspond to actual base pairs while empty (circular) nodes correspond to spurious base pairs. \square represents a pseudonode, and subtree rooted at a pseudonode is formed by Chaining procedure.

```

ALIGN( $q[1..m], t[1..n]$ )
1   $M' =$  BINARIZE-SP( $Q$ )
2  for all intervals ( $i, j$ ) in  $t$  and all nodes  $v$  in  $M$ 
3  do if  $v$  is NIL
4  then  $A[i, j, NIL] = \sum_{l=i}^j \gamma(t[l], '-')$ 
5  if  $v$  is a pseudo node
6  then  $A[i, j, v] =$  return ALIGN-SP( $i, j, v$ )
7  if  $v \in M$ 
8  then  $A[i, j, v] = \max \begin{cases} A[i + 1, j - 1, \text{child}(v)] + \delta(t[i], t[j], q[lv], q[rw]) \\ A[i, j - 1, v] + \gamma('-', t[j]) \\ A[i + 1, j, v] + \gamma('-', t[i]) \\ A[i + 1, j, \text{child}(v)] + \gamma(q[lv], t[i]) + \gamma(q[rw], '-') \\ A[i, j - 1, \text{child}(v)] + \gamma(q[lv], '-') + \gamma(q[rw], t[j]) \\ A[i, j, v] + \gamma(q[lv], '-') + \gamma(q[rw], '-') \end{cases}$ 
9  if  $v \in M' - M$  and  $v$  has one child
10 then  $A[i, j, v] = \max \begin{cases} A[i, j - 1, \text{child}(v)] + \gamma(q[rw], t[j]) \\ A[i, j, \text{child}(v)] + \gamma(q[rw], '-') \\ A[i, j - 1, v] + \gamma('-', t[j]) \\ A[i + 1, j, v] + \gamma('-', t[i]) \end{cases}$ 
11 if  $v \in M' - M$  and  $v$  has two children
12 then  $A[i, j, v] = \max_{i \leq k \leq j} \{A[i, k - 1, \text{left\_child}(v)] + A[k, j, \text{right\_child}(v)]\}$ 

```

FIG. 6. Alignment algorithm for aligning an embedded-simple-pseudoknot $q[1..m]$ to a target sequence $t[1..n]$.

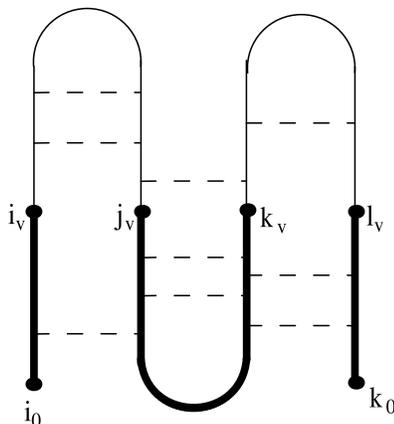


FIG. 7. Substructure in a standard pseudoknot structure.

good alignments, we can often bound the gap-lengths. To take advantage of this, we employ a banding procedure. A discussion of scoring matrices and gap penalties is deferred to another paper.

7. ALIGNMENT ALGORITHM FOR STANDARD-PSEUDOKNOTS

It is possible to extend the algorithm for aligning a simple pseudoknot to an alignment algorithm for a standard pseudoknot with degree $d > 3$. In this section, we present an extension of our algorithm for standard pseudoknot structures with degree 4, and achieve the following result:

Theorem 2. *The optimal alignment for a standard pseudoknot with degree 4 can be computed in $O(mn^4)$ time which is identical to the degree 3 case (simple pseudoknots). In general, standard pseudoknots of degree $2k - 1$ and $2k$ can be aligned in $O(mn^{2k})$ time.*

To handle this kind of pseudoknot, we firstly have to modify our substructure and chaining procedure accordingly. In a standard pseudoknot structure, the substructure will be the union of three subintervals as shown in Figure 7, and each will be identified by a quadruple (i_v, j_v, k_v, l_v) . We denote the substructure identified with quadruple (i_v, j_v, k_v, l_v) with $\mathcal{P}(i_v, j_v, k_v, l_v)$.

Chaining procedure again creates a chain of nodes where each corresponds to a substructure. There will be three different solid nodes corresponding to a quadruple (i_v, j_v, k_v, l_v) . Let M_L , M_M , and M_R be the sets of nodes in which $(q[i_v], q[j_v])$, $(q[j_v], q[k_v])$, and $(q[k_v], q[l_v])$ is a pair, respectively, and let M_S be the set of empty nodes.

Align-SPd4 procedure, as shown in Figure 8a, is very similar to the one for simple pseudoknot structure which is also a standard pseudoknot but with degree 3. Now, we need to keep four indices i , j , k , and l along the target sequence. In the recursion, we slide the indices i , j , k , and l down along the regions D_1 , D_2 , D_3 , and D_4 , respectively. To align a node $v \in M_L$ representing substructure $\mathcal{P}(i_v, j_v, k_v, l_v)$ in query to substructure $\mathcal{P}'(i, j, k, l)$ in target, it is enough to consider the same seven cases we did for simple pseudoknot structure. However, we have to extend the algorithm to also account for the solid nodes in M_M and M_R . Cases for the empty nodes are also handled the same as in alignment procedure for simple pseudoknot structures.

Similar to what we did for simple pseudoknots, we can control the indices i , j , k , and l as shown in Figure 8 to improve running time. It turns out that aligning a standard pseudoknot with degree 4 has the same complexity with degree 3, i.e., $O(mn^4)$ where m is the length of the standard pseudoknot structure, because we limit the subinterval by only i_0 and the complexity of the for loop for a fixed i_0 is $O(n^4)$. Since the order of nodes for the tree of a standard pseudoknot of length m is $O(m)$, total complexity is $O(mn^4)$.

(a)

```

ALIGN-SPD4( $M'$  in  $M$  of  $q[1\dots m], t[1\dots n]$ )
1 //  $M'$  is the chain representing the standard pseudoknot region to be aligned in query  $q$ 
2 for all intervals  $(i_0, k_0)$  in  $t[1\dots n]$ 
3 do for all  $(i, j, k, l), i_0 \leq i \leq j < k \leq l \leq k_0$ , all nodes  $v \in M'$ 
4 do
5   if  $v \in M_L$ 
6     then  $B[i, j, k, l, v] = \max \left\{ \begin{array}{l} B[i+1, j-1, k, l, \text{child}(v)] + \delta(q[i_v], q[j_v], t[i], t[j]) \\ \quad + \gamma(q[i_v], t[i]) + \gamma(q[j_v], t[j]), \\ B[i+1, j, k, l, \text{child}(v)] + \gamma(q[i_v], t[i]) + \gamma(q[j_v]', -'), \\ B[i, j-1, k, l, \text{child}(v)] + \gamma(q[i_v]', -') + \gamma(q[j_v], t[j]), \\ B[i, j, k, l, \text{child}(v)] + \gamma(q[i_v]', -') + \gamma(q[j_v]', -') \end{array} \right.$ 
7   if  $v \in M_M$ 
8     then  $B[i, j, k, l, v] = \max \left\{ \begin{array}{l} B[i, j-1, k+1, l, \text{child}(v)] + \delta(q[j_v], q[k_v], t[j], t[k]) \\ \quad + \gamma(q[j_v], t[j]) + \gamma(q[k_v], t[k]), \\ B[i, j, k+1, l, \text{child}(v)] + \gamma(q[j_v]', -') + \gamma(q[k_v], t[k]), \\ B[i, j-1, k, l, \text{child}(v)] + \gamma(q[j_v], t[j]) + \gamma(q[k_v]', -'), \\ B[i, j, k, l, \text{child}(v)] + \gamma(q[j_v]', -') + \gamma(q[k_v]', -') \end{array} \right.$ 
9   if  $v \in M_R$ 
10    then  $B[i, j, k, l, v] = \max \left\{ \begin{array}{l} B[i, j, k+1, l-1, \text{child}(v)] + \delta(q[k_v], q[l_v], t[k], t[l]) \\ \quad + \gamma(q[k_v], t[k]) + \gamma(q[l_v], t[l]), \\ B[i, j, k, l-1, \text{child}(v)] + \gamma(q[k_v]', -') + \gamma(q[l_v], t[l]), \\ B[i, j, k+1, l, \text{child}(v)] + \gamma(q[k_v], t[k]) + \gamma(q[l_v]', -'), \\ B[i, j, k, l, \text{child}(v)] + \gamma(q[k_v]', -') + \gamma(q[l_v]', -') \end{array} \right.$ 
11  if  $v \in M_S$  and  $\text{move}(v) = (1, 0, 0, 0)$ 
12    then  $B[i, j, k, l, v] = \max \left\{ \begin{array}{l} B[i+1, j, k, l, \text{child}(v)] + \gamma(q[i_v], t[i]), \\ B[i, j, k, l, \text{child}(v)] + \gamma(q[i_v]', -') \end{array} \right.$ 
13  if  $v \in M_S$  and  $\text{move}(v) = (0, 1, 0, 0)$ 
14    then  $B[i, j, k, l, v] = \max \left\{ \begin{array}{l} B[i, j-1, k, l, \text{child}(v)] + \gamma(q[j_v], t[j]), \\ B[i, j, k, l, \text{child}(v)] + \gamma(q[j_v]', -') \end{array} \right.$ 
15  if  $v \in M_S$  and  $\text{move}(v) = (0, 0, 1, 0)$ 
16    then  $B[i, j, k, l, v] = \max \left\{ \begin{array}{l} B[i, j, k+1, l, \text{child}(v)] + \gamma(q[k_v], t[k]), \\ B[i, j, k, l, \text{child}(v)] + \gamma(q[k_v]', -') \end{array} \right.$ 
17  if  $v \in M_S$  and  $\text{move}(v) = (0, 0, 0, 1)$ 
18    then  $B[i, j, k, l, v] = \max \left\{ \begin{array}{l} B[i, j, k, l-1, \text{child}(v)] + \gamma(q[l_v], t[l]), \\ B[i, j, k, l, \text{child}(v)] + \gamma(q[l_v]', -') \end{array} \right.$ 
19   $B[i, j, k, l, v] = \max \left\{ \begin{array}{l} B[i, j, k, l, v] \\ B[i+1, j, k, l, v] + \gamma(' -', t[i]), \\ B[i, j-1, k, l, v] + \gamma(' -', t[j]), \\ B[i, j, k+1, l, v] + \gamma(' -', t[k]), \\ B[i, j, k+1, l-1, v] + \gamma(' -', t[l]) \end{array} \right.$ 
20   $B_{SP}[i_0, k_0, i_{SP}, k_{SP}] = \max_{i=i_0, k=j+1, l=k_0} \{B(i, j, k, l, \text{ROOT}(M'))\}$ 

```

(b)

```

IMPROVED ALIGN-SPD4()
1 for all  $v \in S'$ 
2 do for  $i_0 = 1$  to  $n-1$ 
3   do for  $j = i_0$  to  $n-2$ 
4     do for  $i = j$  downto 1
5       do for  $k = n$  downto  $j+1$ 
6         do for  $l = k$  to  $n$ 
7           do Compute  $B[i, j, k, v]$ 

```

FIG. 8. (a) Alignment procedure (Align-SPd4) for a standard pseudoknot with degree $d = 4$. (b) Improved Align-SPd4 procedure.

8. RESULTS

A C++ implementation of the algorithm given for simple pseudoknots (PAL) is done. PAL takes an RNA query and target sequence, and returns all high scoring structural local alignments in the target sequence. All tests were performed on a PC computer (3.4 Ghz, 1 GB RAM) unless otherwise stated. The structure of the target sub-sequence is inferred from the alignment. In order to assess the performance of PAL, we tested six RNA families from the Rfam database: UPSK, Antizyme, Parecho CRE, Corona-FSE, Corona-pk3, and IFN-gamma. Each of these families has an embedded-simple-pseudoknot structure. General information about these families is given in Table 1.

8.1. Predicting structure with PAL

To test structural inference, we select a pair of members from a family as the query and target. PAL is used to align the query to the target. The inferred structure of the target is compared against the annotated structure in the Rfam database. We evaluate the predicted structure by computing TP (true positives), FP (false positives), and FN (false negatives), defined as follows: TP is the number of base pairs in inferred target structure that are correct, FP is the number of base pairs in the inferred structure that are not in the true structure, and FN is number of base pairs in the true structure that are not inferred. We define $Specificity = TP / (TP + FP)$ and $Sensitivity = TP / (TP + FN)$. Good performance is indicated by both being close to 1. Table 2 summarizes the result of testing each pair in the six families. As the results show, PAL is a strong predictor of structure, with mean sensitivity and specificity of 0.95. We also investigated the few cases in which the prediction was away from the mean. In most of those cases, the target had stem loops that were longer than the query. As they were not aligned to the query structure, they were not inferred. In practice, we would augment the inferred structure by a local extension of stem loops in both directions. A second source of errors was incorrect annotation in Rfam. Other than these two scenarios, the structure inference was essentially correct.

There is a second caveat in these results that is not apparent. Many (but not all) of the sequences have high sequence similarity, which might be making the alignment task easier. We believe that this is because a sequence search tool such as BLAST is used to fish out candidates, which are then manually aligned and experimentally validated. We will show in the following sections that our tool can pick out candidates that BLAST cannot find, and also align them structurally. Also, in the cases where there isn't high sequence similarity, the structure inference was just as good.

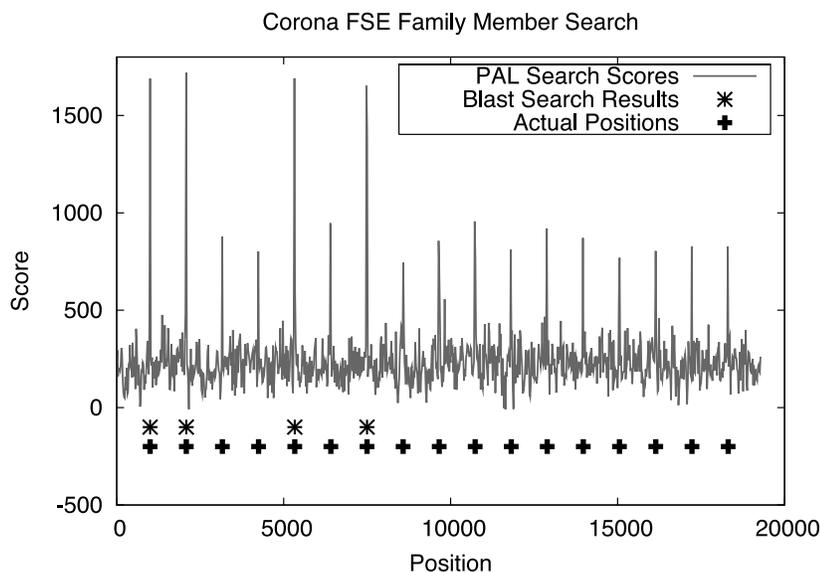


FIG. 9. The score plot for Corona-FSE homologue search using PAL as a pseudoknot RNA search tool. * Actual positions of the members. + Members located by Blastn.

TABLE 3. *P*-VALUE OF THE STRUCTURAL HOMOLOG SEARCH TEST, AS THE PROPORTION OF RANDOM TESTS WHICH EXCEEDS THE SCORE AMONG 1 MILLION RANDOM TESTS

<i>GC content</i>	<i>Description</i>	<i>Score</i>	<i>No. of tests exceeding the score</i>	<i>p-value (unadjusted)</i>	<i>p-value (adjusted)</i>
25%	Lowest TP	769	3	3.0×10^{-6}	2.9×10^{-3}
	Highest FP	557	2912	2.9×10^{-3}	1.0
50%	Lowest TP	769	1	1.0×10^{-6}	9.6×10^{-4}
	Highest FP	557	633	6.3×10^{-4}	6.0×10^{-1}
75%	Lowest TP	769	0	1.0×10^{-6}	9.6×10^{-4}
	Highest FP	557	360	3.6×10^{-4}	3.4×10^{-1}

Adjusted *p*-value is Bonferroni corrected value.

8.2. Searching for structural homologs

In this test, we use one of the members of an RNA family as a query, and look for its homolog in a large random sequence, with the other members inserted. Figure 9 shows the results for the Corona-FSE family, in which 17 members were embedded in a 19-kb random sequence. The windowed scores are shown by solid lines. The actual positions of the remaining 17 members are denoted by “+.” We note that the true hits are easily the highest scoring regions along the sequence and that all true positives score higher than all the false hits. The lowest scoring TP has a score of 769, and the highest scoring FP has a score of 557.

We performed a random test to assess significance of these result. We generated a bulk (10^6) of random sequences of equal size with the query member of the RNA family and determined the *p*-value of a score as the proportion of the sequences which exceeds the score. However, since we performed many (965) tests shifting windows in the previous test on 19-kb sequence, there is a high chance that we assessed the score by chance; we should correct multiple hypotheses. We use Bonferroni correction method to adjust the *p*-value to a correct value. Note that Bonferroni correction is conservative here because windowed tests on nearby regions are correlated, and the score we use to assess *p*-value is not always the best score chosen among all tests. (Bonferroni correction assumes that we select the best score among independent results.) We performed three experiments assuming different GC contents containment (25%, 50%, 75%) in random sequences (Table 3), and in all cases, the lowest TP shows significant *p*-value even after being adjusted by conservative Bonferroni correction.

In contrast, Blastn (E-value 10, Word-size 7) is able to locate only four of the members. These results also show the significance of the secondary structure for searching homologue in addition to the primary structure. We repeat the same experiment for RNA families, UPSK, Antizyme, Parecho, Corona-FSE, Corona-pk3, and IFN-gamma. In all cases, PAL locates all members as the topmost hits (Table 4). We agree that BLAST is not as appropriate a tool for comparison as tools such as RSEARCH, and our own tool FastR can search for structural homologs of RNA (Klein and Eddy, 2003; Zhang et al., 2005). However, these other tools cannot align pseudoknotted RNA, and the search must be followed up with a correct alignment to determine homologs. Also, the complexity of these methods often force use of Blast

TABLE 4. COMPARISON OF PAL AGAINST BLAST ON PSEUDOKNOTTED RNA FAMILIES

<i>RNA family</i>	<i>No. found</i>	
	<i>BLAST</i>	<i>PAL</i>
UPSK	3	3
Antizyme	12	12
Parecho CRE	4	4
Corona-FSE	4	17
Corona-pk3	5	13
IFN-gamma	4	4

to determine initial candidates. In the next section, we show that our tool used in conjunction with RNA filters can efficiently search large genomes.

9. SEARCHING GENOMES FOR PSEUDOKNOTS

While PAL is accurate in fishing for structural homologs, it is computationally intensive, making genome-scale searches intractable. However, there has been much recent research (including our own work) on *computational filters* for RNA, which quickly eliminate much of the database, while retaining the true homologs (Weinberg and Ruzzo, 2004; Zhang et al., 2005). We used PAL in conjunction with sequence-based filters (Zhang et al., 2006) to search genomes for the three most interesting families.

The Corona-FSE family (RF00507) is a conserved pseudoknot in Coronaviruses which can promote ribosomal frameshifting (Baranov et al., 2005). We searched the entire viral genome (79 Mb) for homologs of this family in 33.8 CPU hours on 1.6-GHz AMD Opteron Grid, and identified 11 novel members of the sub-family. Like other known members, these are found in coronaviruses, murine hepatitis virus, and Avian flu viruses. Only two of the 11 were similar enough in sequence to be identified by BLAST. The alignments can be retrieved from www.cse.ucsd.edu/~bdost/RF00507.htm. A similar result was obtained for Corona-pk3. This family has a conserved ~55-nt pseudoknot structure which has been shown to be necessary for viral genome replication (Williams et al., 1999). We identified 20 novel members of this family with significant scores (see www.cse.ucsd.edu/~bdost/RF00165.htm). Only one of the 20 was similar enough in sequence to be identified by BLAST.

The interferon-gamma family is an interesting example of a pseudoknot that is found in the 5'UTR of the interferon-gamma gene. It regulates translation of the downstream gene by binding to the kinase PKR, a known regulator of IFN-gamma translation (Ben-Asouli et al., 2002). After its discovery in 2002, the pseudoknot was found to be conserved in many mammals. Its presence in rodents was speculated, but the homolog was not located. We searched in mouse and rat genomic DNA, and in the complete gene of gerbil. In all three species, we clearly identified the homologs as the top-scoring alignment. The alignment of human and mouse pseudoknots are shown in Figure 10. The conserved location in the two species, just upstream of the start codon, and conservation of key elements validates the hit. We are working with collaborators on experimental validation and on location of more members of this family.

In conclusion, we demonstrate that the algorithm for aligning pseudoknots, implemented as PAL, represents a viable tool for searching for novel homologs and for structural inference. We hope that our tool will help increase the impact and influence of pseudoknotted RNA in cellular function. PAL and supplemental data are available upon request.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation (grant NSF-DBI:0516440 to S.Z. and V.B.).

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

- Adams, P.L., Stahley, M.R., Kosek, A.B., et al. 2004. Crystal structure of a self-splicing group I intron with both exons. *Nature* 430, 45–50.
- Akutsu, T. 2000. Dynamic programming algorithm for RNA secondary structure prediction with pseudoknots. *Disc. Appl. Math.* 104, 45–62.
- Argaman, L., Hershberg, R., Vogel, J., et al. 2001. Novel small RNA-encoding genes in the intergenic regions of *Escherichia coli*. *Curr. Biol.* 11, 941–950.

- Bafna, V., Muthukrishnan, S., and Ravi, R. 1995. Computing similarity between RNA strings. *Combin. Pattern Matching* 937, 1–14.
- Baranov, P.V., Henderson, C.M., Anderson, C.B., et al. 2005. Programmed ribosomal frameshifting in decoding the SARS-CoV genome. *Virology* 332, 498–510.
- Ben-Asouli, Y., Banai, Y., Pel-Or, Y., et al. 2002. Human interferon-gamma mRNA autoregulates its translation through a pseudoknot that activates the interferon-inducible protein kinase PKR. *Cell* 108, 221–232.
- Condon, A., Davy, B., Rastegari, B., et al. 2004. Classifying RNA pseudoknotted structures. *Theoret. Comput. Sci.* 320, 35–50.
- Dirks, R.M., and Pierce, N.A. 2003. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *J. Comput. Chem.* 24, 1664–1677.
- Durbin, R., Eddy, S., Krogh, A., et al. 1998. *Biological Sequence Analysis*, Cambridge University Press, New York.
- Eddy, S. 2001. Non-coding RNA genes and the modern RNA world. *Nat. Rev. Genet.* 2, 919–929.
- Evans, P. 1964. Algorithms and complexity for annotated sequence analysis [Ph.D. dissertation]. University of Victoria, Victoria, BC, Canada.
- Griffiths-Jones, S., Moxon, S., Marshall, M., et al. 2005. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res.* 33, 121–124.
- Jaeger, J., Turner, D., and Zuker, M. 1989. Improved prediction of secondary structures for RNA. *Proc. Natl. Acad. Sci. USA* 86, 7706–7710.
- Jiang, T., Lin, G., Ma, B., et al. 2002. A general edit distance between RNA structures. *J. Comput. Biol.* 9, 371–388.
- Klein, R., and Eddy, S. 2003. Rsearch: finding homologs of single structured RNA sequences. *BMC Bioinformatics* 4, 44.
- Matsui, H., Sato, K., and Sakakibara, Y. 2005. Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures. *Bioinformatics* 21, 2611–2617.
- Nixon, P.L., Rangan, A., Kim, Y.G., et al. 2002. Solution structure of a luteoviral P1-P2 frameshifting mRNA pseudoknot. *J. Mol. Biol.* 322, 621–633.
- Novina, C.D., and Sharp, P.A. 2004. The RNAi revolution. *Nature* 430, 161–164.
- Rastegari, B., and Condon, A. 2005. Linear time algorithm for parsing RNA secondary structure. *5th Workshop Algorithms Bioinform. (WABI)*.
- Rastogi, T., Beattie, T.L., Olive, J.E., et al. 1996. A long-range pseudoknot is required for activity of the Neurospora VS ribozyme. *EMBO J.* 15, 2820–2825.
- Rivas, E., and Eddy, S. 1999. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.* 285, 2053–2068.
- Rivas, E., and Eddy, S. 2000. Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics* 16, 583–605.
- Storz, G. 2002. An expanding universe of noncoding RNAs. *Science* 296, 1260–1263.
- Theimer, C.A., Blois, C.A., and Feigon, J. 2005. Structure of the human telomerase RNA pseudoknot reveals conserved tertiary interactions essential for function. *Mol. Cell* 17, 671–682.
- Vitreschak, A., Rodionov, D., Mironov, A., et al. 2003. Riboswitches: the oldest mechanism for the regulation of gene expression? *Trends Genet.* 20, 44–50.
- Weinberg, Z., and Ruzzo, W.L. 2004. Faster genome annotation of non-coding RNA families without loss of accuracy. *Proc. RECOMB 2004*.
- Williams, G.D., Chang, R.Y., and Brian, D.A. 1999. A phylogenetically conserved hairpin-type 3' untranslated region pseudoknot functions in coronavirus RNA replication. *J. Virol.* 73, 8349–8355.
- Winkler, W.C., and Breaker, R.R. 2003. Genetic control by metabolite-binding riboswitches. *ChemBiochem.* 4, 1024–1032.
- Zhang, S., Hass, B., Eskin, E., et al. 2005. Searching genomes for non-coding RNA using fastr. *IEEE Trans. Comput. Biol. Bioinform.* 2, 366–379.
- Zhang, S., Borovok, I., Aharonowitz, Y., et al. 2006. A sequence-based filtering method for ncRNA identification and its application to searching for riboswitch elements. *Bioinformatics* 22, e557–e565.
- Zuker, M., and Sankoff, D. 1984. RNA secondary structures and their prediction. *Bull. Math. Biol.* 46, 591–621.

Address reprint requests to:

Dr. Shaojie Zhang
School of Electronic Engineering and Computer Science
University of Central Florida
4000 Central Florida Blvd.
Orlando, FL 32816

E-mail: shzhang@cs.ucf.edu