Lifting Prediction to Alignment of RNA Pseudoknots

Mathias Möhl⋆, Sebastian Will⋆, and Rolf Backofen⋆⋆

[1] Programming Systems Lab, Saarland University, Saarbrücken, Germany, mmohl@ps.uni-sb.de
[2] Bioinformatics, Institute of Computer Science, Albert-Ludwigs-Universität, Freiburg, Germany,{will,backofen}@informatik.uni-freiburg.de

Abstract. Prediction and alignment of RNA pseudoknot structures are NP-hard. Nevertheless, several efficient prediction algorithms by dynamic programming have been proposed for restricted classes of pseudoknots. We present a general scheme that yields an efficient alignment algorithm for arbitrary such classes. Moreover, we show that such an alignment algorithm benefits from the class restriction in the same way as the corresponding structure prediction algorithm does. We look at five of these classes in greater detail. The time and space complexity of the alignment algorithm is increased by only a linear factor over the respective prediction algorithm. For four of the classes, no efficient alignment algorithms were known. For the fifth, most general class, we improve the previously best complexity of $O(n^5m^5)$ time to $O(nm^6)$, where n and m denote sequence lengths. Finally, we apply our fastest algorithm with $O(nm^4)$ time and $O(nm^2)$ space to comparative de-novo pseudoknot prediction.

1 Introduction

In the last years, it has become clear that RNA molecules play very important roles in a cell, among them regulatory and catalytic ones [1], much beyond acting only as a messenger. A recent computational screen for structural RNA [2] has revealed that there are more than 30.000 putative non-coding RNAs (ncRNAs). For the functional annotation, classification and further investigation of these RNAs, it is essential to infer the secondary structure of these ncRNA, and to use this structure information for comparative analysis.

Nearly all major approaches for the computational analysis of ncRNAs have been restricted to nested secondary structure, neglecting pseudoknots. However, pseudoknots are far from being a rare event. As stated in [3], the pseudoknot motif is "among the most prevalent RNA structures". Using exactly clustered stochastic simulations, Xayaphoummine et al. [4] have estimated that up to 30% of the base pairs in G+C-rich sequences are forming pseudoknots. For *E.coli*, it was estimated that $15.5\% \pm 6.5\%$ of the base pairs are included in pseudoknots.

⋆ These authors contributed equally.
⋆⋆ Corresponding author

There are three major problems concerning the analysis of pseudoknotted RNAs, which depend on each other. First, there are only few known pseudoknots. Second, the prediction of pseudoknots is computationally very expensive. The full problem is known to be NP-hard [5], efficient algorithms exist only for restricted classes of pseudoknots. And third, the reliability of the existing prediction programs is not very good. The main reason for the low quality of the prediction programs is that there are too few known structures where the programs can be trained on, an approach that was successfully applied to nested RNA structures (see e.g. CONTRAfold [6]). Vice versa, the low prediction quality and the high computational costs hinder the research on pseudoknot structures.

The most promising way out of this dilemma is to use comparative approaches for predicting pseudoknotted secondary structures. Since the structure is more conserved than the sequence of RNA, this requires an alignment of both sequence *and* structure (called sequence-structure alignment in the following). This has been a very successful approach for nested secondary structures, where a complete variety of practical tools (e.g. LocARNA [7], MARNA [8], FOLDALIGN [9, 10], Dynalign [11, 12] to name some) exists. However, there are only few approaches for sequence-structure alignment for pseudoknotted approaches (lara [13]). The research on this topic is far behind the computational analysis of pseudoknot structure prediction, where several approaches in varying complexity for restricted classes have been introduced [5, 14–19]. All these algorithms use the properties of the restricted class in a dynamic programming approach to efficiently solve the prediction problem.

In this paper, we consider the problem of sequence-structure alignment with known pseudoknot structures. This is the necessary basis for comparative analysis of pseudoknots. We introduce a general scheme that generates a pseudoknot alignment algorithms for a restricted class of pseudoknots, given the corresponding prediction approach. Our general scheme can be applied to pseudoknot classes for which a dynamic programming approach exists. Basically, we use the decomposition strategy of the structure prediction, and apply this strategy to solve the associated alignment problem for known structures efficiently. The additional complexity of the alignment algorithm compared to the corresponding prediction problem is only linear in the length of the sequence, both in space and time, which is surprisingly small. For comparison, in the case of the only known pseudoknot alignment methods for a restricted class of pseudoknots, namely for the class handled by Rivas&Eddy's $O(n^6)$ algorithm [14], the corresponding alignment problem was solved by Evans [20] with a time complexity of $O(n^{10})$ and space complexity of $O(n^8)$. Compared to that, our approach requires only $O(n^7)$ time and $O(n^5)$ space on this class, although it supports a more general scoring scheme. Furthermore, the run time of the algorithms generated by our general scheme scales with the complexity of the aligned structures. In particular, the worst case complexity applies only to the actual pseudoknots, whereas simpler parts of the structures are aligned much faster. A central technical insight of the work is how pseudoknot alignment can benefit from a variety of structural restrictions in the same way as structure prediction does. We discuss five classes of pseudoknot structures in detail, namely [5, 14–19]. For four of these classes, no exact alignment algorithms existed up to now.

Then, we used the fastest of our introduced alignment approaches that handles the structure class described by Reeder&Giegerich [19]. The resulting algorithm has a time complexity of $O(n^5)$, compared to $O(n^4)$ for the prediction. We tested this approach on known pseudoknot classes from the Rfam-databases [21] and compared it with an implementation of the nested-nested sequence-structure alignment method of [22] in the tool MARNA [8]. We found that using the known pseudoknot structures improves the quality of alignments, especially for low pairwise sequence identity. Furthermore, we set up a first prototype pipeline for combining alignment and prediction of pseudoknot structures. Such a pipeline is the basis for automatic pseudoknot annotation of large amounts of candidate ncRNA as predicted by ncRNA screens (e.g. provided by the RNAz-tool [23]), where the secondary structure is unknown. We applied our prototype pipeline to a data set of 50 putative ncRNAs from a current *Ciona intestinalis* screen [24]. This revealed that using a comparative approach increases the reliability of pseudoknot structure annotation. Whereas the Reeder&Giegerich prediction method pknotsRG would find pseudoknots in all of the 50 examples, only 14 of them show sufficiently many compensatory base pair mutations after using alignment in addition to the pseudoknot structure prediction. Such compensatory base pair mutations enlarges the reliability of the prediction since they give a strong evidence for an evolutionary conservation of the structure. Compensatory base pair mutations are commonly used as a verification for real biologically relevant structures. This prototypical pipeline is only an initial step and some work remains to be done to setup an integrated pipeline. However, the prototype application shows that such an integrated pipeline is feasible. With the work presented here, we have done the first but essential step to set up such an approach.

Related work Evans proposed a fixed parameter tractable algorithm that computes the longest arc preserving common subsequence of pseudoknots [25] and an algorithm to compute the maximum common ordered substructure of two RNA structures [20]. As mentioned above, the latter guarantees polynomial run time, but the polynomial is significantly larger than in our approach. In our own previous work, we developed a fixed parameter tractable algorithm to align arbitrary pseudoknots [26]. Despite addressing the same task, this algorithm differs much from our current approach and the run time depends on other properties of the input structures. Finally, there exists a heuristic approach based on integer linear programming that is usually fast in practice, but does not guarantee optimal solutions [13].

Roadmap After giving basic concepts (Sec. 2), we present a general framework for decomposing RNA structures that can be instantiated with the methods used by various pseudoknot prediction algorithms (Sec. 3). Then we describe a general method to construct a corresponding sequence structure alignment algorithm for each of these instances (Sec. 4) and have a detailed look at the different instances (Sec. 5). Finally, we present experimental results obtained with our implementation (Sec. 6) and give some concluding remarks (Sec. 7).

2 Preliminaries

Sequences and fragments An *arc-annotated sequence* is a pair (S,P), where S is a string over the set of bases $\{A,U,C,G\}$ and P is a set of arcs (l,r) with $1 \leq l < r \leq |S|$ representing bonds between bases, such that each base is adjacent to at most one arc. We denote the i-th symbol of S by $S[i]$. For an arc $p=(l,r)$, we denote its left end l and right end r by p^{L} and p^{R}, respectively. An arc $p \in P$ is *crossing* if there is an arc $p' \in P$ such that $p^{\mathrm{L}} < p'^{\mathrm{L}} < p^{\mathrm{R}} < p'^{\mathrm{R}}$ or $p'^{\mathrm{L}} < p^{\mathrm{L}} < p'^{\mathrm{R}} < p^{\mathrm{R}}$; then p and p' form a *pseudoknot*. An arc-annotated sequence is *crossing* if it contains crossing arcs, otherwise it is *nested*.

An *alignment* A of two arc-annotated sequences (S_a,P_a) and (S_b,P_b) is a set $A_1 \cup A_2$, where $A_1 \subseteq [1..|S_a|] \times [1..|S_b|]$ is a set of *match edges* such that for all $(i,j),(i',j') \in A_1$ holds 1.) $i > i'$ implies $j > j'$ and 2.) $i = i'$ if and only if $j = j'$ and A_2 is the set of *gap edges* $\{\,(x,-) \mid x \in [1..|S_a|] \wedge \nexists y.(x,y) \in A_1\,\} \cup \{\,(-,y) \mid y \in [1..|S_b|] \wedge \nexists x.(x,y) \in A_1\,\}$. A base that is adjacent to a gap edge is called *aligned to a gap.* Two bases $S_a[i]$, $S_b[j]$ are matched by A if $(i,j) \in A$ and two arcs $p_a \in P_a$, $p_b \in P_b$ are matched if $(p_a^{\mathrm{L}},p_b^{\mathrm{L}}) \in A$ and $(p_a^{\mathrm{R}},p_b^{\mathrm{R}}) \in A$. Each alignment has an associated cost based on an edit distance with two classes of operations. The operations where first introduced by Jiang et al. [22] and are illustrated in Fig. 1. Base operations (mismatch and insertion/deletion) work solely on positions that are not incident to an arc. *Base mismatch* replaces a base with another base and has associated cost w_m. A *base insertion/deletion* removes or adds one base and costs w_d. The second class consists of operations that involve at least one position that is incident to an arc. An *arc mismatch* replaces one or both of the bases incident to an arc. It costs $\frac{w_{am}}{2}$ if one base is replaced or w_{am} if both are replaced. An *arc breaking* removes one arc and leaves the incident bases unchanged. The associated cost is w_b. *Arc removing* deletes one arc and both incident bases and costs w_r. Finally, *arc altering* removes one of the two bases that are incident to an arc and costs $w_a = \frac{w_b}{2} + \frac{w_r}{2}$(cf. [22]).

Define for two arc annotated sequences (S_a,P_a) and (S_b,P_b) and $k \in \{a,b\}$

$$\begin{aligned}
&\chi(i,j) := \text{ if } S_a[i] \neq S_b[j] \text{ then } 1 \text{ else } 0\\
&\psi_k(i) := \text{ if } \exists p \in P_k.p^{\mathrm{L}} = i \text{ or } p^{\mathrm{R}} = i \text{ then } 1 \text{ else } 0\\
&\mathrm{gap}_k(i) := w_d + \psi_k(i)(\frac{w_r}{2} - w_d)\\
&\mathrm{basematch}(i,j) := \chi(i,j)w_m + (\psi_1(i) + \psi_2(j))\frac{w_b}{2}.
\end{aligned}$$

$\mathrm{gap}_k(i)$ denotes the cost to align base $S_k[i]$ to a gap, $\mathrm{basematch}(i,j)$ the cost to align $S_a[i]$ to $S_b[j]$ under the assumption that their possibly adjacent arcs are not matched.

3 Decomposing Sequences

In general, dynamic programming (DP) algorithms for RNA alignment, structure prediction or similar tasks, rely on a recursive decomposition of the RNA sequence into subsequences or combinations of subsequences (which we will call

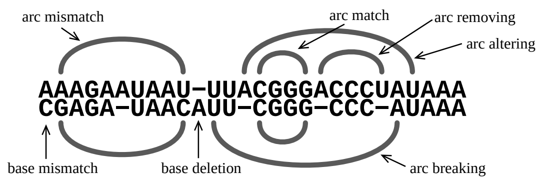

Fig. 1. Edit operations (cf. [22])

fragments in the following). For example, the standard nested secondary structure prediction [27, 28] decomposes a subsequence into two consecutive subsequences, whereas the Rivas and Eddy algorithm [14] for predicting a restricted class of pseudoknots uses fragments that consist of two unconnected subsequences. Thus, the type of decompositions considered by each algorithm has major impact on both complexity and the class of structures handled by the algorithm.

We will develop a general view on decomposition strategies for RNA structures with pseudoknots. As a first insight, we point out that the central difference between the various DP based structure prediction algorithms is their choice of one such strategy. This choice determines the characteristic trade-off between the class of handled pseudoknots and the resulting complexity of the algorithm. As our main contribution, we will introduce a general framework for sequence-structure alignment based on an arbitrary decomposition strategy. Compared to the structure prediction algorithm for this decomposition strategy, the alignment algorithm will have only linear time and space overhead. Thus the scheme provides the first sequence-structure alignment methods for many pseudoknot classes and covers the classes of all known DP based pseudoknot prediction algorithms.

A *fragment* F of an arc annotated sequence (S, P) is a k-tuple of *intervals* $([l_1, r_1], \dots, [l_k, r_k])$ with $1 \leq l_1 \leq r_1 + 1 \leq \cdots \leq l_k \leq r_k + 1 \leq |S|$. Note that this definition allows *empty intervals* $[i+1, i]$. The ranges between the intervals, i.e $[r_1+1, l_2-1], \dots, [r_{k-1}+1, l_k-1]$, are called *gaps of* F. We call k the *degree* of F and $l_1, r_1, \dots, l_k, r_k$ its *boundaries*. The *set of positions covered by* F (denoted with $\hat{F}$), is defined as the union of the intervals contained in F. The i-th interval $[l_i, r_i]$ of F is denoted with $F[i]$ and with $F[i]^L$ and $F[i]^R$ we denote its *left and right boundary* l_i *and* r_i, respectively. For better readability, we abbreviate intervals of the form $[i, i]$ as $\langle i \rangle$.

F is called *arc-complete*, iff $l \in \hat{F} \Leftrightarrow r \in \hat{F}$ for each $(l, r) \in P$. F is called *atomic* if F covers either exactly the two ends of an arc of P or a single position not adjacent to an arc. As an example, in Fig 2a, the boxed fragments have all a degree of 2, F_a^2 and F_b^2 are atomic and F_b as well as F_b^1 are not arc-complete.

Let F, F^1 and F^2 be fragments of the same sequence. The pair (F^1, F^2) is a *split* of F iff $\hat{F} = \hat{F^1} \uplus \hat{F^2}$.[3] We call F^1 and F^2 the *children* and F the *parent*

[3] For simplicity, we introduce only binary splits. However, the introduced concepts are raised to n-ary splits straightforwardly.

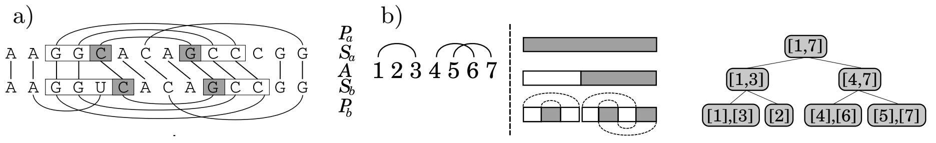

Fig. 2. a) Alignment with some boxed fragments F_a,F_b that are split into their white and gray parts F_a^1, F_b^1 (white boxes) and F_a^2, F_b^2 (gray boxes), respectively. Gap edges of A are not shown. b) A structure and two ways to visualize a parse tree thereof. Note that in the parse tree, each leaf is atomic.

of the split. The split is called *arc-preserving*, if F, F^1 and F^2 are arc complete. In Fig. 2a, the split (F_a^1, F_a^2) of F_a is arc-preserving – the split (F_b^1, F_b^2) of F_b is not arc-preserving due to the leftmost arc of P_b.

A *parse tree of a sequence* (S, P) is a binary tree where each node is an arc-complete fragment of (S, P) such that (a) the root is $([1, |S|])$, (b) each inner node is a fragment F and has two children F_1 and F_2, such that (F_1, F_2) is an arc-preserving split of F, (c) each leaf is an atomic fragment. Fig. 2b shows two ways to visualize a parse tree.

A parse tree represents one fixed recursive decomposition of a sequence. The basic idea of our alignment algorithm scheme is to handle the sequences asymmetrically. The algorithm recursion follows a single fixed parse tree for the first sequence. At each split of the parse tree, it considers *all* compatible splits of the second sequence. In contrast to the splits of the parse tree, these compatible splits don't have to be arc-preserving. Formally, two splits are compatible, if they have the same split type which is defined as follows.

The *basic type of a split* (F^1, F^2) *of a fragment* F is defined by the following construction. The interval $[\min(\hat{F}), \max(\hat{F})]$ decomposes into the intervals of F^1, the intervals of F^2 and gaps of F. If we order these from left to right and replace the intervals of F^1 by 1, the ones of F^2 by 2 and the remaining ones by G (for gap), we obtain a string T over $\{1, 2, G\}$ that we call the *basic type* of the split. Every split has exactly one basic type.

The type can be further refined by annotating constraints. In particular, we introduce the *size constraint* that restricts an interval to have at most size one in each instance of the type. It is indicated by marking the respective symbols 1 or 2 in the type with $'$. Size constraints will be used to describe the common case of splits that split off an atomic fragment. A type containing constraints is called a *constrained type.* Note that each size constraint reduces the number of splits of this type by one order of magnitude, because it reduces the degrees of freedom by one.

If (F^1, F^2) is of type T, we call it a T*-split.* As an example, in Fig. 2a the splits of F_a and F_b are of basic type $12G21$ and of constrained type $12'G2'1$.

The complexity of the alignment algorithm depends on the number of children and parent instances of the considered split types. On the number of parents, because for a given fragment in one sequence, we consider all fragments in the other sequence having the same type. On the number of children, because it determines the number of ways an aligned fragment can be split in sub-alignments.

These numbers depend on the length m of the second sequence and are defined as the *number of children* and the *number of parents*, respectively:

$$\#_C^m(T) = |\{\, (F^1, F^2) \mid (F^1, F^2) \text{ is a T-split of some } F \text{ and } \hat{F} \subseteq [1, m] \,\}|$$
$$\#_P^m(T) = |\{\, F \mid \exists (F^1, F^2) \text{ that is a T-split of } F \text{ and } \hat{F} \subseteq [1, m] \,\}|$$

Lemma 1. *For some sequence with length m and a split type T, let the degree of the parent and the two children be k_p, k_1 and k_2, respectively. Furthermore, let c denote the degrees of freedom that are reduced by the constraints of T and $c' \leq c$ denote the corresponding reduction for the parent instances. Then $\#_C^m(T) \in O(m^{k_p+k_1+k_2-c})$ and $\#_P^m(T) \in O(m^{2k_p-c'})$.*

Due to space limitations, the proofs of all lemmata are available only online at `http://www.bioinf.uni-freiburg.de/Supplements/pkalign-recomb09`.

4 The Alignment Algorithm Scheme

4.1 The Variant for Basic Types

The algorithm takes two arc-annotated sequences (S_a, P_a), (S_b, P_b) and a parse tree for (S_a, P_a) as input[4]. For each fragment in the parse tree, the algorithm recursively computes alignments to all fragments of (S_b, P_b) that have the same basic type. In order to present the precise recursions, we need the following formal notion for alignments of fragments.

The *restriction of an alignment A to fragments F_a, F_b* is defined as $A|_{F_a \times F_b} := \{\, (i, j) \in A \mid i \in \hat{F_a} \cup \{-\}, j \in \hat{F_b} \cup \{-\} \,\}$. A *aligns* two fragments F_a and F_b of the same degree k, short $\text{align}_A(F_a, F_b)$, if and only if for all $(a_1, a_2) \in A$ and for all $i \in 1 \ldots k$ it holds that $a_1 = -$ or $a_2 = -$ or $a_1 \in F_a[i] \Leftrightarrow a_2 \in F_b[i]$. Note that for a given alignment A, a fragment of one sequence can be aligned to several fragments of the other; consider e.g. Fig. 2a, where A aligns F_a^1 to $F_b^1 = ([3, 5], [11, 12])$ and also to $([3, 4][11, 12])$.

The cost of A can be computed recursively as $\text{cost}(A)$ with

$$\begin{aligned}
\text{cost}(\{(i, -)\} \uplus A') &= \text{gap}_a(i) + \text{cost}(A') \\
\text{cost}(\{(-, j)\} \uplus A') &= \text{gap}_b(j) + \text{cost}(A') \\
\text{cost}(\{(l_1, l_2), (r_1, r_2)\} \uplus A') &= (\chi(l_1, l_2) + \chi(r_1, r_2)) \frac{w_{am}}{2} + \text{cost}(A') \\
&\quad \text{if } (l_1, r_1) \in P_a, (l_2, r_2) \in P_b \\
\text{cost}(\{(i, j)\} \uplus A') &= \text{basematch}(i, j) + \text{cost}(A') \\
&\quad \text{if third case is not applicable.}
\end{aligned}$$

This computation relies only on the property of the scoring scheme that all costs except for matching an arc are local to a single base. If A aligns two fragments

[4] Such a parse tree can be constructed using standard parsing techniques. Furthermore, the structure prediction algorithms discussed later implicitly construct parse trees that can also be reused for this purpose.

F_a and F_b, the cost is computed analogously as $C_A(F_a, F_b) := \mathrm{cost}(A|_{F_a \times F_b})$. The optimal cost to align two fragments is defined as

$$C\big(F_a, F_b\big) := \min_{A \text{ with } \mathrm{align}_A(F_a,F_b)} \{C_A(F_a, F_b)\}\,.$$

The optimal cost to align the entire sequences is $C\big(F_a, F_b\big)$ for $F_a = ([1, |S_a|])$ and $F_b = ([1, |S_b|])$. It can be computed by recursively applying the next lemma, where F_a^1 and F_a^2 are chosen according to the parse tree.

Lemma 2 (Split lemma). *Let F_a and F_b be fragments of (S_a, P_a) and (S_b, P_b), respectively. Let (F_a^1, F_a^2) be an arc-preserving split of F_a of basic type T. Then*

$$C\big(F_a, F_b\big) = \min_{T\text{-}split\ (F_b^1,F_b^2)\ of\ F_b} \Big\{C\big(F_a^1, F_b^1\big) + C\big(F_a^2, F_b^2\big)\Big\} \qquad (1)$$

Note that if the split of F_b is not arc-preserving, the respective arcs are broken or removed, since there is no arc of F_a that they can be matched to. The cost for beaking or removing the two ends of the arcs is contained in $C\big(F_a^1, F_b^1\big)$ and $C\big(F_a^2, F_b^2\big)$, respectively. The evaluation of the recursion is done efficiently by dynamic programming, i.e. all intermediate values $C\big(F_a, F_b\big)$ are tabulated, such that each instance is computed only once. The recursive case, shown in Fig. 3a, is directly given by Eq. (1). At the leafs of the parse tree, the base cases, shown in Fig. 3b, are applied. The actual alignment can be constructed using the usual back-trace techniques.

a) Recursive case:

$$C\big(F_a, F_b\big) = \min_{T\text{-split } (F_b^1,F_b^2) \text{ of } F_b} \Big\{C\big(F_a^1, F_b^1\big) + C\big(F_a^2, F_b^2\big)\Big\}\,,$$

where the parse tree splits F_a into (F_a^1, F_a^2) by a split of basic type T

b) Base cases:

$$C\big(\langle i\rangle, [l,r]\big) = \min \begin{cases} C\big(\langle i\rangle, [l+1, r]\big) + \mathrm{gap}_2(l) & \text{if } l \le r \\ C\big(\langle i\rangle, [l, r-1]\big) + \mathrm{gap}_2(r) & \text{if } l \le r \\ \mathrm{basematch}(i, l) & \text{if } l = r \\ \mathrm{gap}_1(i) & \text{if } l > r \end{cases}$$

$$C\big(F_a {=} (\langle p^{\mathrm{L}}\rangle, \langle p^{\mathrm{R}}\rangle),\ ([l_1, r_1], [l_2, r_2])\big) =$$

$$\min \begin{cases} C\big(\langle p^{\mathrm{L}}\rangle, [l_1, r_1]\big) + C\big(\langle p^{\mathrm{R}}\rangle, [l_2, r_2]\big) & \\ C\big(F_a, ([l_1+1, r_1], [l_2, r_2])\big) + \mathrm{gap}_2(l_1) & \text{if } l_1 \le r_1 \\ C\big(F_a, ([l_1, r_1-1], [l_2, r_2])\big) + \mathrm{gap}_2(r_1) & \text{if } l_1 \le r_1 \\ C\big(F_a, ([l_1, r_1], [l_2+1, r_2])\big) + \mathrm{gap}_2(l_2) & \text{if } l_2 \le r_2 \\ C\big(F_a, ([l_1, r_1], [l_2, r_2-1])\big) + \mathrm{gap}_2(r_2) & \text{if } l_2 \le r_2 \\ (\chi(p^{\mathrm{L}}, l_1) + \chi(p^{\mathrm{R}}, l_2))\,\frac{w_{am}}{2} & \text{if } (l_1, l_2) = (r_1, r_2) \in P_b \end{cases}$$

Fig. 3. a) Recursive case for basic split type and b) base cases of the algorithm.

Complexity Let n and m be the length of the two sequences, respectively. First note that the parse tree has only $O(n)$ nodes, since each split introduces at least one new boundary, of which there exist only $O(n)$ many. Let T_p and T_c be types of splits in the parse tree, where $\#_P^m(T_p)$ and $\#_C^m(T_c)$ are maximal among the occurring split types, respectively. For a node F_a with split T_p the algorithm materializes the costs $C\big(F_a, F_b\big)$ for $\#_P^m(T_p)$ fragments F_b. Assuming the worst case for each node, this results in a space complexity of $O(n \cdot \#_P^m(T_p))$. The time complexity is dominated by the computation at T_c-splits. There, according to Lem. 2, the algorithm minimizes over $\#_C^m(T_c)$ terms; each is computed in $O(1)$. This results in a worst case time complexity of $O(n \cdot \#_C^m(T_c))$. $\#_P^m(T_p)$ and $\#_C^m(T_c)$ are asymptotically bounded due to Lemma 1. For the case of non-constrained, basic types we instantiate to $O(nm^{2k})$ space and $O(nm^{3k})$ time complexity, where k is the maximal degree among the splits in the parse tree.

4.2 An Optimized Variant for Constrained Types

By the preceding complexity analysis, the time and space complexity directly depend on $\#_P^m(T)$ and $\#_C^m(T)$ for the basic types. Lemma 1 shows how constraints in types reduce these numbers, and thus bear the potential to reduce the complexity. However, we cannot simply use constraint types instead of basic types in the recursion of Fig. 3a. Let's assume that F_a^1 is atomic and T is constrained correspondingly; furthermore, T_u denotes the unconstrained, basic split type corresponding to T. In the optimal alignment, F_a^1 is not necessarily aligned to an atomic F_b^1. However, we know (by Lem. 2) that for any T_u-split (F_b^1, F_b^2) of F_b, at most one of the bases of F_b^1 per interval of F_b^1 is matched to F_a^1 and the others are aligned to gaps. Using this observation, we can still split off a fragment F_b^1 of F_b satisfying the constraint type T *after* 'eating away' the gaped bases, which we do by introducing 'shrink'-cases.

The following lemma directly leads to the optimized recursion equation as given in Fig. 4.

Lemma 3 (Split lemma for constrained types). *Let F_a and F_b be fragments of (S_a, P_a) and (S_b, P_b), respectively. Let (F_a^1, F_a^2) be an arc-preserving T-split of F_a, where T contains size constraints for at most one of the fragments and at least one boundary of each interval of the constrained fragment coincides with a boundary of F_a.*[5] *Let A be an optimal alignment of F_a and F_b. Then there is a T-split (F_b^1, F_b^2) of F_b such that either the constrained fragment of the split is matched to one or two gaps by A and the remaining fragment is aligned to F_a or there is a T-split (F_b^1, F_b^2) such that A aligns F_a^1 to F_b^1 and F_a^2 to F_b^2.*

In extension of Lem. 2, Lem. 3 allows size constraints in the split type T. According to the lemma, the recursion of the optimized algorithm shown in Fig. 4, introduces additional shrink cases. These cover the minimization cases where one cannot split as in Lem. 2 by the (now possibly constrained) split type T.

[5] This condition can be generalized to constraints on more than one fragment as long as no two adjacent symbols are constrained. This is mostly relevant for n-ary splits.

$$C\big(F_a, F_b\big) = \min_{T\text{-split } (F_b^1, F_b^2) \text{ of } F_b} \min \begin{cases} C\big(F_a^1, F_b^1\big) + C\big(F_a^2, F_b^2\big) & \\ C\big(F_a, F_b^2\big) + C\big(-, F_b^1\big) & \text{if } T \text{ contains some 1'} \\ C\big(F_a, F_b^1\big) + C\big(-, F_b^2\big) & \text{if } T \text{ contains some 2'} \end{cases}$$

Fig. 4. Optimized recursive case. This applies to the general case of a constrained type T satisfying the conditions in Lem. 3. $C\big(-, F_b^i\big)$ denotes the cost of deleting F_b^i.

Table 1. Pseudoknot classes and complexity of their prediction and alignment.

class		R&E	A&U	L&P	D&P	R&G
prediction	time	$O(m^6)$	$O(m^4)$	$O(m^5)$	$O(m^5)$	$O(m^4)$
	space	$O(m^4)$	$O(m^3)$	$O(m^3)$	$O(m^4)$	$O(m^2)$
alignment (literature)	time	$O(n^5m^5)$	-	-	-	-
	space	$O(n^4m^4)$	-	-	-	-
alignment (new scheme)	time	$O(nm^6)$	$O(nm^4)$	$O(nm^5)$	$O(nm^5)$	$O(nm^4)$
	space	$O(nm^4)$	$O(nm^3)$	$O(nm^3)$	$O(nm^4)$	$O(nm^2)$

5 Instances of the Algorithm Scheme

In this section, we focus on the behaviour of our general algorithm scheme for different restricted classes of pseudoknots. We analyse the classes of pseudoknots produced by different structure prediction algorithms [5, 14–19] and show that the alignment can benefit of the structural restrictions in exactly the same way as the prediction. In particular we show for each of the prediction algorithms how to construct a corresponding alignment algorithm with only a linear increase in complexity (see Table 1). Following Condon *et al.* [29], we name the classes of structures according to the authors of the respective prediction algorithms: R&E (Rivas and Eddy [14]), A&U (Akutsu [16] and Uemura [15]), L&P (Lyngsø and Pedersen [5]), D&P (Dirks and Pierce [18] and R&G (Reeder and Giegerich [19]). Also note that on nested structures the algorithm behaves like an algorithm by Jiang *et al.* [22].

R&E structures The prediction algorithm by Rivas and Eddy [14] requires $O(n^6)$ time and $O(n^4)$ space. It is restricted to structures for which parse trees exist where each fragment has a degree of at most 2. Our algorithm aligns structures from this class in $O(nm^6)$ time and $O(nm^4)$ space. Compared to that, the best alignment algorithm for this class known so far (by Evans [20]) requires $O(n^5m^5)$ time and $O(n^4m^4)$ space.[6]

A&U structures The algorithms of Akutsu [16], Uemura [15] and Deogun et al. [17] predict structures with simple pseudoknots in $O(n^4)$ time and $O(n^3)$ space. For the structures of this class, there always exist parse trees, where the splits are limited to the constrained types

$$12 \;\; 121 \;\; 12'G2'1 \;\; 1G2'12' \;\; 12'G1 \;\; 1G2'1 \;\; 1G12' \;\; 12'G2'.$$

[6] Evan's algorithm computes the longest arc-preserving common subsequence, which can be considered as a special case of our edit distance measure.

By Lem. 1, there exist only $O(m^4)$ splits (F_b^1, F_b^2) of these types (i.e. $\#_C^m(T) \in O(m^4)$ for all but the first and last of the above types). Hence our algorithm runs in $O(nm^4)$ time on these structures. Due to Lem. 1, the space complexity is at most $O(nm^4)$ too; this can be improved to $O(nm^3)$, because for the allowed splits all unconstrained children fragments of degree 2, which dominate the space complexity, have the same leftmost boundary as their parent. In the computation of the costs $C\big(F_a, F_b\big)$, we can thus group all such fragments F_b with the same leftmost boundary. For each of these groups only $O(m^3)$ fragments F_b exist and the space is reused for each group. A rigorous argument for the improvement is given by the next lemma, which is proved by simultaneous induction over the parse tree.

Lemma 4 (Improved space complexity for A&U). *Let F_a be a node of a parse tree for an A&U structure, where F_a has degree k and n' descendants. Then, $O(n'm^3)$ space suffices to compute 1.) for $k = 1$ or atomic F_a of degree 2, all $O(m^2)$ costs $C\big(F_a, F_b\big)$ and 2.) for $k = 2$, all $O(m^3)$ costs $C\big(F_a, F_b\big)$, where all F_b share a fix leftmost boundary.*

This space improvement follows a general principle applicable in dynamic programming that makes use of invariants on the most complex items and groups their computation. This invariant is conveniently reflected in our representation by split types.

L&P structures Lyngsø and Pedersen [5] predict certain pseudoknots in $O(n^5)$ time and $O(n^3)$ space. Their class of pseudoknots is restricted such that there must exist a parse, where the split of the root has basic type 12121 and such that the two fragments of this split are both nested. This implies that they can be further decomposed with splits of constrained types

12'G2'1G1	1G21G1	12G1G1	2G1G12	(for fragments with degree 3)
2'1G12'	21G1	1G12	1G2	(for fragments with degree 2)
2'12'	12			(for fragments with degree 1)

The degree 3 splits decompose only fragments that start with the first sequence position and end with the last sequence position; it suffices to consider only splits of F_b that share this property. This means, we see here a further example of a type constraint. This *maximality constraint* reduces the degrees of freedom for this type by 2. Indicating the constraints, we refine the first four types to $\downarrow 12'G2'1G1\downarrow$ $\downarrow 1G21G1\downarrow$ $\downarrow 12G1G1\downarrow$ $\downarrow 2G1G12\downarrow$.

To see that no other splits for fragments of degree 3 are required, note that with the first three split types, the fragment can be decomposed until its first two intervals are no more connected by arcs and once this is the case, a split of the fourth type can be applied.[7]

[7] Lyngsø and Pedersen give another intuition based on the idea of considering cyclic sequences. This is reflected by the split types in the way that each split for fragments of degree 3 is analogous to the split for fragments of degree 2 below it, if the part before the first gap is cyclically moved to the end of the type

The complexity analysis with Lem. 1 leads for each of these split types to at most $O(m^5)$ instances, which implies $O(nm^5)$ time complexity. For example, for split type $T = \downarrow 12'G2'1G1\downarrow$, we have $k_p = k_1 = 3$, $k_2 = 2$, $c = 4$ and hence $\#^m_C(T) \in O(m^{3+3+2-4}) = O(m^4)$ and for $T' = \downarrow 2G1G12\downarrow$ we have $k_p = 3$, $k_1 = k_2 = 2$, $c = 2$ and hence $\#^m_C(T) \in O(m^{3+2+2-2}) = O(m^5)$.

The space complexity can be reduced from $O(nm^4)$ to $O(nm^3)$ by the same general principle that we observed for the A&U structures. That is, first we can distinguish *simple fragments* that have degree 1 or are constrained and *complex fragments* that are unconstrained and have degree 2. Then, we identify an invariant for the occurring complex fragments and use it for space reduction by grouping. In the L&P split types, the second fragment is always simple. Hence, for each split there are only $O(m^2)$ values to store for the second fragments.

The first fragment of each split type has either degree less than 2 or it contains the pattern $1G1$, where G is the last gap in the split type. This implies that the computation of each fragment of degree at least 2, recursively relies only on fragments that have the size of the last gap in common. The costs for the simple fragments can thus always be computed by grouping the complex fragments by this gap size and reusing the memory.

D&P structures Dirks and Pierce [18] developed an algorithm to compute the partition function for RNA pseudoknots, which can also be modified to predict the MFE structure. The algorithm takes $O(n^5)$ time and $O(n^4)$ space. The corresponding parse trees of the predicted structures are limited to fragments of degree at most two and the splits are of types

$$12 \quad 1212 \quad 21G1 \quad 12G1 \quad 1G21 \quad 1G12 \quad 1'2G21'.$$

Again, according to Lem. 1 there exist at most $O(m^5)$ splits (F_b^1, F_b^2) for each of these types and hence our alignment algorithm requires $O(nm^5)$ time and $O(nm^4)$ space on these structures.

R&G structures The efficiency of the Reeder and Giegerich [19] structure prediction algorithm ($O(n^4)$ time, $O(n^2)$ space) is due to the restriction to canonical pseudoknots. A stem of base pairs is called *canonical* if it cannot be extended by another valid base pair. The canonical stem containing a given base pair is thus uniquely determined. In R&G structures, pseudoknots are formed only by two crossing canonical stems. Reeder and Giegerich structures can be decomposed by split types

$$2'1 \quad 12' \quad 12 \quad 1'21' \quad E_1 = 1^c23^c41^c53^c \quad E_2 = 12'G2'1,$$

where we introduce another type constraint in E_1 (denoted by $\cdot^c$) claiming that fragments 1 and 3 each form a canonical stem.[8]

The type E_1 represents a split into 5 independent parts, where fragment 1 and 3 form the canonical pseudoknot. To simplify the presentation, so far we limited splits to contain only two fragments, but the concept generalizes to more

[8] Note that our split types (with the exception of E_2) correspond to the grammar rules given in [19]: $S \rightarrow . \mid . \, S \mid S. \mid SS \mid (S) \mid [^k S\{^l S]^k S\}^l$.

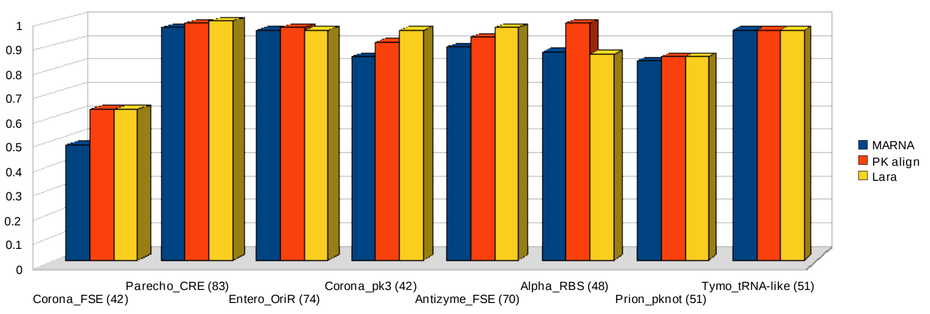

Fig. 5. Comparison MARNA vs. PKalign vs. lara on 8 pseudoknot families. The numbers in brackets give the sequence identity.

complex splits without any difficulty. E_2 is used to further decompose the stems corresponding to the first and third fragment of E_1.

All types have at most $O(m^4)$ instances, resulting in $O(nm^4)$ time complexity. The type E_1 deserves special attention, because its unconstrained variant has $O(m^8)$ many instances. However due to the constraints this is reduced to $O(m^2)$, since a split of constrained type E_1 is already determined by fixing start and end position of the parent fragment. Because the RNA structures are fix, the start position of the fragment is the start position of a uniquely determined base pair in stem 1 and this uniquely determines the first stem. The second stem is determined analogously by the end position.

In the same way as the structure prediction algorithm finds the best canonical structure for a sequence, the alignment algorithm finds the best canonical alignment. Here, canonical means that a stem of a pseudoknot can only be aligned to another maximally extended stem.

The space complexity is reduced as follows. For all split types except E_1 and E_2 applies again that they only contain degree 1 or atomic fragments. In E_1, from all fragments of degree 2 only the $O(n^2)$ canonical instances are considered. In E_2, the second fragment is size constrained and the first fragment shares its first and last boundary with the splitted fragment. Hence, the $O(m^4)$ instances of the first fragment can be grouped into groups of $O(m^2)$ elements that have a common first and last boundary. This reduces the space complexity to $O(nm^2)$.

6 Results

6.1 Accuracy of Pseudoknot Alignment

We use a benchmark set of 8 RNA families of Rfam [30] that are annotated with pseudoknots. Albeit in total Rfam contains 16 such families, we restricted the test set to RNAs with length of at most 125. From each family, we selected the pair of members with the lowest sequence identity, in order to maximally challenge

a)
```
                [[-[[[[[....((.....-)).-((((((((((..]]]]]-]]....))))))))))
X90572.1     6  cu-uguacagaaugguaag-cac-guguaguaggagguaca-agcaacccuauugcau 59
                [[-[[[[[....((.....-)).-((((((((((..]]]]]-]]....))))))))))
X66718.1     6  cu-uguacagaaugguaag-cac-guguaaugggagguaca-agcaaccccauugcau 59
                [[[[[[-[.(((...-..)))...((((((((((.-]-]]]]]]-...))))))))))
AF058944.1   6  cucuau-cagauugg-augucuugcugcuauaaua-g-auagag-aagguuauagcag 58

cons. str.      [[-[[[-[................((((((((((.-]-]]]-]]....))))))))))
```
b)
```
                [[[[[[[..((((]]]]]]].(((((((...)))))))......))))--
ci_658349   52  ucucagggugaaaucugagacggaaacgauucguuuccuauauauuuc-- 99
                [[[[[[.(((((((]]]]]].(((((((...))))))).....)))))))
cs_658349   55  ucucaguuuaauaccugggacggaaacgauucguuuccucuauguauuaa 104
                [[[[[[[.(((((]]]]]]].((-(((.....)))-)).....-)))))-
od_658349   53  ucucagugugacagcugagaccg-uccuacuggga-cgucuau-uguca- 98

cons. str.      [[[[[[[..((((]]]]]]].((-(((.....)))-))......))))..
```

Fig. 6. a) Correctly predicted pseudoknot in the Rfam family Corona_pk3 and its alignment. b) Predicted pseudoknot of potential ncRNA.

the algortihms. We used the consensus structure (projected to the respective sequence) as structure input, since Rfam does not contain a separate structure for each sequence.

We compared our tool PKalign to MARNA [31] and lara [13]. While lara represents the only other pseudoknot alignment method available so far, the comparison to MARNA allows us to evaluate the benefit of taking pseudoknots into account: MARNA is based on the exact same scoring scheme as PKalign (and we used the same parameter values), but since it is unable to handle pseudoknots, we had to resolve the crossing by removing some base pairs. The accuracy to reproduce the Rfam alignment is measured by the COMPALIGN score. The results are shown in Fig. 5.

The comparison to MARNA shows that taking the pseudoknots into account in general improves the accuracy. The accuracy of PKalign and lara is comparable, the minor differences between their results seem to be caused by differences in the scoring scheme, different choices if several optimal alignments exist and by the fact that PKalign does not yet support affine gap costs.

6.2 Detecting Conserved Pseudoknots

The reliability of pseudoknot de-novo prediction is still very low. Common prediction programs, tend to predict pseudoknots even in pseudoknot free RNA and do not allow to distinguish safely between true pseudoknots and false positives. This behavior could already be observed for pknotsRG [19] in the following small study.

Therefore, it is desirable to increase the specificity of predictions by requiring confirmation due to homologous RNAs. This approach provides much stronger evidence by observing compensatory mutations in conserved crossing base pairs that are predicted in several homologous RNAs. Such a procedure is useful for reliably annotating pseudoknots in unknown RNA, e.g. from genome wide screens for non-coding RNA.

As a preliminary approach towards comparative pseudoknot identification, we suggest a pipeline for detecting potential pseudoknots that starts with a set of homologous RNAs, and performs the following steps: 1.) for each sequence predict locally optimal and suboptimal pseudoknots of the R&G class (using pknotsRG [19] in local mode). 2.) determine candidate pseudoknots that occur at similar positions in k of our sequences (here, $k = 3$). 3.) using our approach, align the k-tuples of pseudoknots pairwise all-against-all; this information is used to construct a multiple alignment by T-Coffee [32]. 4.) analyze the alignment for conserved, crossing compensatory mutations.

First, we tested our approach on Rfam data using the same set of 8 families as above. For each family, we randomly selected six sequences for our analysis. We found pseudoknot candidates with crossing compensatory mutations in all of these families. For four families, we could reproduce triplet alignments of the known pseudoknots that showed crossing compensatory mutations for three of the families; an example is given in Fig. 6a. The figure depicts an alignment of the pseudoknotted sub-sequences with start and end position. For each sub-sequence we show the structure predicted by pknotsRG. The last line gives the consensus structure and highlights base pairs of the pseudoknot which are confirmed by crossing compensatory mutations.

The procedure was then applied to the 50 unannotated ncRNA candidates predicted by an RNAz screen of *Ciona intestinalis* [24]. In this screen, the *C. intestinalis* genome was compared to *C. savignyi* and *O. dioica*, thus per candidate we get three sequences from the three organisms that are analyzed by the above pipeline. In total, we predicted pseudoknot candidates for only 14 of the 50 RNAs; in contrast, pknotsRG predicts pseudoknots in all of the ncRNAs. Fig. 6b shows one prediction by this experiment.

7 Conclusions

We presented a general algorithm scheme for pairwise alignment of pseudoknots. This scheme yields an efficient alignment algorithm for arbitrary classes of pseudoknots that can be predicted efficiently by dynamic programming. Moreover, we showed that such an alignment algorithm benefits from restrictions to certain structure classes in the same way as structure prediction algorithms do. This theoretically interesting result actually yields a series of new alignment algorithms for specific pseudoknot classes; for earlier pseudoknot alignment algorithms, it improves time and space complexity.

Our short study for increasing the reliability of pseudoknot prediction by accounting for comparative information is probably the first biologically meaningful application of pseudoknot alignment to biological data and demonstrates the new possibilities due to our method. It points directly to the appealing idea of automatic pseudoknot annotation in unknown, potential ncRNA from genome-wide screens.

Acknowledgements. We thank Kristin Reiche for providing data of the Ciona RNAz screen and our Master student Jörg Bruder for implementing the parsing component of PKalign. M. Möhl is funded by the German Research Foundation.

References

1. Couzin, J.: Breakthrough of the year. Small RNAs make big splash. Science **298**(5602) (2002) 2296–7
2. Washietl, S., Hofacker, I.L., Lukasser, M., Huttenhofer, A., Stadler, P.F.: Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome. Nat Biotechnol **23**(11) (2005) 1383–90
3. Staple, D.W., Butcher, S.E.: Pseudoknots: RNA structures with diverse functions. PLoS Biol **3**(6) (2005) e213
4. Xayaphoummine, A., Bucher, T., Thalmann, F., Isambert, H.: Prediction and statistics of pseudoknots in RNA structures using exactly clustered stochastic simulations. Proc. Natl. Acad. Sci. USA **100**(26) (2003) 15310–5
5. Lyngso, R.B., Pedersen, C.N.S.: Pseudoknots in RNA secondary structures. In: Proc. of the Fourth Annual International Conferences on Compututational Molecular Biology (RECOMB00), ACM Press (2000) BRICS Report Series RS-00-1.
6. Do, C.B., Woods, D.A., Batzoglou, S.: CONTRAfold: RNA secondary structure prediction without physics-based models. Bioinformatics **22**(14) (2006) e90–8
7. Will, S., Reiche, K., Hofacker, I.L., Stadler, P.F., Backofen, R.: Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering. PLOS Computational Biology **3**(4) (2007) e65
8. Siebert, S., Backofen, R.: MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. Bioinformatics **21**(16) (2005) 3352–9
9. Gorodkin, J., Heyer, L., Stormo, G.: Finding the most significant common sequence and structure motifs in a set of RNA sequences. Nucleic Acids Res **25**(18) (1997) 3724–32
10. Havgaard, J.H., Torarinsson, E., Gorodkin, J.: Fast pairwise structural RNA alignments by pruning of the dynamical programming matrix. PLoS Comput Biol **3**(10) (2007) 1896–908
11. Mathews, D.H., Turner, D.H.: Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. Journal of Molecular Biology **317**(2) (2002) 191–203
12. Harmanci, A.O., Sharma, G., Mathews, D.H.: Efficient pairwise RNA structure prediction using probabilistic alignment constraints in Dynalign. BMC Bioinformatics **8** (2007) 130
13. Bauer, M., Klau, G.W., Reinert, K.: Accurate multiple sequence-structure alignment of RNA sequences using combinatorial optimization. BMC Bioinformatics **8** (2007) 271
14. Rivas, E., Eddy, S.R.: A dynamic programming algorithm for RNA structure prediction including pseudoknots. Journal of Molecular Biology **285**(5) (1999) 2053–68
15. Uemura, Y., Hasegawa, A., Kobayashi, S., Yokomori, T.: Tree adjoining grammars for RNA structure prediction. Theoretical Computer Science **210** (1999) 277 – 303 Paper as Print Copy.

16. Akutsu, T.: Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. Discrete Applied Mathematics **104** (2000) 45–62
17. Deogun, J.S., Donis, R., Komina, O., Ma, F.: RNA secondary structure prediction with simple pseudoknots. In: APBC '04: Proceedings of the second conference on Asia-Pacific bioinformatics, Darlinghurst, Australia, Australia, Australian Computer Society, Inc. (2004) 239–246
18. Dirks, R.M., Pierce, N.A.: A partition function algorithm for nucleic acid secondary structure including pseudoknots. J Comput Chem **24**(13) (2003) 1664–77
19. Reeder, J., Giegerich, R.: Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. BMC Bioinformatics **5** (2004) 104
20. Evans, P.A.: Finding common RNA pseudoknot structures in polynomial time. In: Combinatorial Pattern Matching (CPM 2006). Volume 4009/2006 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2006) 223–232
21. Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S.R., Bateman, A.: Rfam: annotating non-coding RNAs in complete genomes. Nucleic Acids Research **33 Database Issue** (2005) D121–4
22. Jiang, T., Lin, G., Ma, B., Zhang, K.: A general edit distance between RNA structures. Journal of Computational Biology **9**(2) (2002) 371–88
23. Washietl, S., Hofacker, I.L., Stadler, P.F.: Fast and reliable prediction of noncoding RNAs. Proc. Natl. Acad. Sci. USA **102**(7) (2005) 2454–9
24. Missal, K., Rose, D., Stadler, P.F.: Non-coding RNAs in Ciona intestinalis. Bioinformatics **21 Suppl 2** (2005) ii77–ii78
25. Evans, P.A.: Finding common subsequences with arcs and pseudoknots. In: CPM '99: Proceedings of the 10th Annual Symposium on Combinatorial Pattern Matching, London, UK, Springer-Verlag (1999) 270–280
26. Möhl, M., Will, S., Backofen, R.: Fixed parameter tractable alignment of RNA structures including arbitrary pseudoknots. In: Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching (CPM 2008). LNCS, Springer-Verlag (2008) 69–81
27. Zuker, M., Stiegler, P.: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic Acids Research **9**(1) (1981) 133–48
28. Hofacker, I.L., Fontana, W., Stadler, P.F., Bonhoeffer, S., Tacker, M., Schuster, P.: Fast folding and comparison of RNA secondary structures. Monatshefte Chemie **125** (1994) 167–188
29. Condon, A., Davy, B., Rastegari, B., Zhao, S., Tarrant, F.: Classifying RNA pseudoknotted structures. Theoretical Computer Science **320**(1) (2004) 35–50
30. Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., Eddy, S.R.: Rfam: an RNA family database. Nucleic Acids Research **31**(1) (2003) 439–41
31. Siebert, S., Backofen, R.: Methods for multiple alignment and consensus structure prediction of RNAs implemented in MARNA. Methods Mol Biol **395** (2007) 489–502
32. Notredame, C., Higgins, D.G., Heringa, J.: T-Coffee: A novel method for fast and accurate multiple sequence alignment. Journal of Molecular Biology **302**(1) (2000) 205–17