Fast Detection of Common Sequence Structure Patterns in RNAs

Rolf Backofen and Sven Siebert

Friedrich-Schiller-Universität Jena
Institute of Computer Science
Department of Bioinformatics
Ernst-Abbe Platz 2
07743 Jena, Germany

Abstract

We developed a dynamic programming approach for computing common exact sequential and structural patterns between two RNAs, given their sequences *and* their secondary structures. An RNA consists of a sequence of nucleotides and a secondary structure defined via bonds linking together complementary nucleotides. It is known that secondary structures are more preserved than sequences in the evolution of RNAs.

We are able to compute all patterns between two RNAs in time $O(nm)$ and space $O(nm)$, where n and m are the lengths of the RNAs. Our method is useful for describing and detecting local motifs. It is especially suitable for finding similar regions of large RNAs that do not share global similarities. An implementation is available in C++ and can be obtained by contacting one of the authors.

Key words: RNA pattern matching, sequence/structure alignments, RNA local motifs

1 Introduction

RNAs are polymers consisting of the four nucleotides A,C,G and U which are linked together by their phosphodiester bonds. Bases (which are part of the nucleotides) form hydrogen bonds within the same molecule leading to structure formation. In recent years, RNA molecules gained increasing interest since

Email address: {backofen,siebert}@inf.uni-jena.de (Rolf Backofen and Sven Siebert).

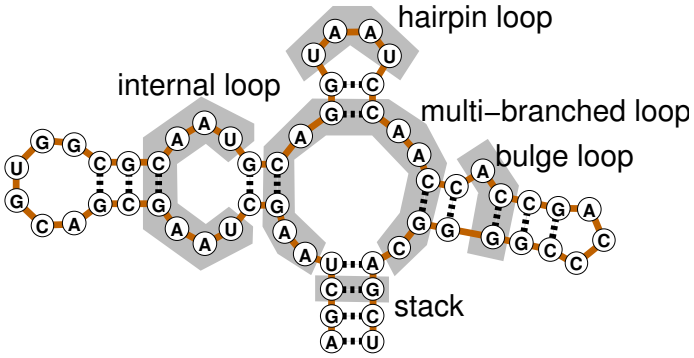

Figure 1. Structure elements of an RNA secondary structure.

a huge variety of functions associated with them was found. Consequently, research on small RNAs has been elected as the scientific breakthrough of the year 2002 by the readers of the science magazine [6].

One major challenge is to find common patterns in RNAs since they suggest functional similarities of these molecules. For this purpose, one has to investigate not only sequential features, but also structural features for the following reasons. First, a major fraction of the function of an RNA-molecule is determined by its (secondary) structure. The structure in combination with the sequence of a molecule dictates its function. And second, it is known that RNA-structure is often more conserved than the sequence.

Most approaches for finding RNA sequence/structure motifs are based on (locally) aligning two RNAs of lengths n. They use dynamic programming methods with a high complexity between $O(n^4)$ and $O(n^6)$ (see e.g. [10], [15] and [2]). Hence, these approaches are only suited for RNAs of moderate sizes. For that reason, we want to use a general approach that is inspired by a common approach in sequence alignment, which uses sequence segments that can be aligned without gaps. In the multiple sequence alignment method DIALIGN [17], these segments are generated in a first step for all pairs of input sequences. In a second step, these segments are used (and extended) to build a single multiple alignment.

So far, there exists no analogous method for finding common sequential and structural segments in two RNAs that can be aligned without using gaps. To solve this problem, we first give a formal definition of exact patterns in Sections 3 and 4. Then, we present a dynamic programming approach for finding this kind of patterns in the following sections. We can list all exact patterns between two RNAs in time $O(nm)$ and space $O(nm)$, where n and m are the lengths of the RNAs.

We have several applications for our method in mind. First, it can be directly used to find pairs of related pieces of RNAs in large genomes. This is neither possible via sequence alignment since structural information is ig-

nored, nor via advanced RNA sequence/structure alignment methods due to their high complexity. Fig B.1 in the appendix shows the largest local exact pattern found in two sequentially different microRNAs from related species. This pattern covers the major part of the corresponding microRNA contained in the sequence. MicroRNAs are a newly discovered large class of regulatory genes that do not encode proteins (see e.g. [1] for a review). Second, our method can be used as a filter for the more advanced but time consuming RNA sequence/structure alignment methods (e.g. [10,15,2]). The basic idea is to generate a set of position pairs that are promising starting candidates for the more complex sequence/structure alignment methods. Here, one would not search for a single exact pattern but for several non-overlapping patterns that together cover larger parts of the RNAs. Fig. B.2 in the appendix shows the four largest patterns for two rev response elements (RRE) in HIV. Filtering has already been used successfully in the FASTR system [4]. They used a purely structural filtering method, whereas we consider both sequential and structural information. Finally, one could think of using our approach to build the analogue of the DIALIGN system for RNAs.

The dynamic programming method is based on the notion of nested RNA structures, where for any two base-pairs (i_1, i_2) and (j_1, j_2) either $i_1 < i_2 < j_1 < j_2$ or $i_1 < j_1 < j_2 < i_2$. Thus, it is possible to describe secondary structures not only as base-pair interactions but at a higher level of structure elements known as hairpin loops, right bulges, left bulges, internal loops or multi-branched loops (see Figure 1). These structure elements have their own stacking order, which are defined by their depth of nucleotide positions concerning the number of base-pairs to the start/end of an RNA. The computation of the RNA patterns is then performed on these structure elements in an inside to outside manner.

A naive attempt is to consider all combinations of positions i in the first RNA and positions j in the second RNA and to extend these starting patterns by looking at neighboring nucleotides sharing the same sequential and structural properties. If these properties are fulfilled then the nucleotides are taken into the pattern. At a first glance, this idea may work, but the crucial point are the loops. Consider e.g. the case shown in Figure 2. Suppose the algorithm starts at position 1 in the first RNA and position 1 in the second RNA and is

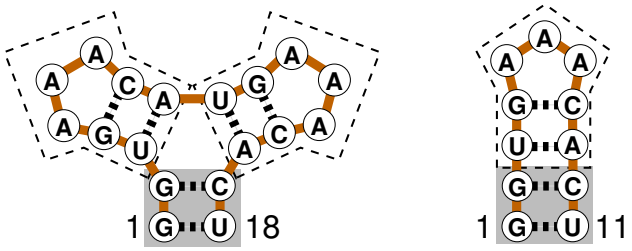

Figure 2. Alternative matching

working towards the multiple loop in the first RNA. The lower stem has been successfully matched. But now there is no clear decision to match the upper part of the stem-loop of the second RNA either to the left side or to the right side of the multiple loop. This decision depends on how a common pattern is defined, of course, and how to reach a maximally extended pattern. Therefore, the only solution is to make some pre-computations of sequential and structural components of RNAs. Finally, we end up with a dynamic programming approach that compares inner parts of RNAs first, stores the results in different matrices and builds up the solutions successively. Note that it is also a mistake to compute common sequential parts first and then to recompose these parts by their structural properties. This problem is obviously a computationally intractable problem since we need to consider all combinations of subsets of sequence parts.

Related Work

Related work on pattern analysis of RNAs has been done mostly for RNAs with nested secondary structures. That is, if RNAs are represented as graphs preserving a linear vertex order, then the base-pairing edges which are drawn in the upper half plane do not cross.

Wang et al. [20] published an algorithm for finding a largest approximately common substructure between two trees. This inexact pattern matching algorithm is also suitable for RNA secondary structures. Höchsmann et al. [13] gives a method for finding local patterns in a tree-representation of RNAs. A survey of methods for computing the similarity between RNAs with and without secondary structures until 1995 is given by Bafna et al. [3]. Gramm et al. [9] formulated the arc-preserving problem: given two nested RNAs S_1 and S_2 with lengths n and m $(n \geq m)$, respectively, does S_2 occur in S_1 such that S_2 can be obtained by deleting bases from S_1 with the property that the arcs are preserved? This problem can not be seen as biologically motivated because the structure of S_2 would be found split in S_1. It has been shown by Jiang et al. [14,15] that finding the longest common arc-preserving subsequence for arc-annotated sequences (LAPCS), where at least one of them has crossing arc structure, is MAXSNP-hard. Exact pattern matching on RNAs has been done by Gendron et al. [8]. They propose a backtracking algorithm, which is similar to an algorithm from Ullman [18] that solves the subgraph isomorphism problem from graph theory with a complexity of $(O(n^3))$. It aims at finding recurrent patterns in a single RNA.

Plan of the Paper

The paper is organized as follows: In section 2, we introduce definitions and notations of RNAs. In section 3, we define matchings between two RNAs such that they can be described by matching and matched paths. In Section 4, bond preserving matching is proposed which is used for the dynamic programming matrices. These matrices and their recursion equations are given in section 5. Finally, the pseudo code is given in section 6.

2 Definitions and Notations

An *RNA* is a tuple (S, P), where S is a string of length n over the alphabet $\Sigma = \{A, C, G, U\}$. We denote $S(i)$ as the base at position i. P is a set of base-pairs (i, i'), $1 \leq i < i' \leq n$, such that $S(i)$ and $S(i')$ are complementary bases. Here, we refer to Watson-Crick base-pairs A—U and C—G, as well as the non-standard base-pair G—U. In the following, we write $i \overset{P}{\text{---}} i'$ instead of $(i, i') \in P$ meaning that the two bases $S(i)$ and $S(i')$ are linked together by a bond in the structure P.

Definition 2.1 (Nested Structure) *A set P of base-pairs is called* nested *if for any two base-pairs $i \overset{P}{\text{---}} i'$ and $j \overset{P}{\text{---}} j'$ either $i < i' < j < j'$ (*independent*) or $i < j < j' < i'$ (*nested*)*

A non-nested secondary structure is called a pseudoknot. The structure prediction problem for pseudoknots is known to be NP-complete [16]. Therefore, most of the secondary structure prediction programs restrict themselves to nested structures. For the rest of the paper, we will assume that secondary structures are nested. This allows us to partially order the bases of an RNA.

Definition 2.2 (Stacking Order) *Let (S, P) be an RNA. The* stacking order *of a base S(i) (abbreviated. as* $\mathrm{stord}_P(i)$*) is the number of bonds $k \overset{P}{\text{---}} l$ with $k < i < l$, plus one.*

Thus, we are able to partition a secondary structure into structure elements with the same stacking order. We call them loops (see Figure 1 for various types of loops). For calculating the sequence/structure alignment, we have to look at neighboring bases belonging to the same loop. This is achieved by a function right (left) of an RNA (S, P), with

$$\mathrm{right}_P(i) = \begin{cases} j & \text{if } (i,j) \in P \\ i+1 & \text{otherwise.} \end{cases}$$

See Figure 3 for an example. $\text{left}_P(i)$ is defined analogously. The function $\text{right}_P^k(i)$ (resp. $\text{left}_P^k(i)$) is a short term for applying the right function (resp. left) k-times to i. We define $\text{lbd}_P(i)$ (resp. $\text{rbd}_P(i)$) to be true if there is a bond $i \overset{P}{\text{---}} i'$ (resp. $i' \overset{P}{\text{---}} i$), false otherwise. We define $\text{bd}_P(i)$ to be $\text{rbd}_P(i) \vee \text{lbd}_P(i)$. For an RNA (S, P), the *loop which is enclosed by a bond* $i \overset{P}{\text{---}} i'$ (written $\text{loop}(i \overset{P}{\text{---}} i')$) is the set of positions $\{r \mid i < r < i' \wedge \exists k : r = \text{right}_P^k(i)\}$.

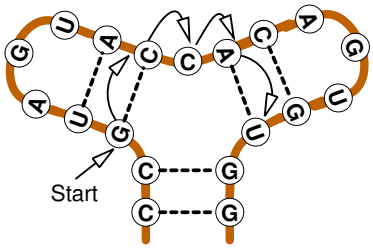

Figure 3. The $\text{right}_P(\cdot)$ function is applied to a base belonging to a loop. The chain of successive $\text{right}_P(\cdot)$ applications traverses this loop.

3 Matchings

In this section, we will define formally what we understand under "common sequential and structural segments in two RNAs that can be aligned without gaps". Suppose we are given two RNAs (S_1, P_1) and (S_2, P_2). The sets $V_1 = \{i \mid 1 \leq i \leq |S_1|\}$ and $V_2 = \{j \mid 1 \leq j \leq |S_2|\}$ contain the positions of both RNAs. In the following, we will define matchings between two RNAs that implicitly work on two graphs with nodes V_1 and V_2. The edges in these implicit graphs are the backbone and bond connections of the two RNAs. I.e., two nodes $(i, i') \in V_1$ form an edge if either $i' = i \pm 1$, $i \overset{P_1}{\text{---}} i'$ or $i' \overset{P1}{\text{---}} i$. An exact pattern (without gaps) corresponds then to a matching (with some properties) that maps a connected component of the first implicit graph to a connected component of the second one.

Definition 3.1 (Matching) *A* matching M *between two RNAs* (S_1, P_1) *and* (S_2, P_2) *is a set of pairs* $M = \{(i, j) \mid i \in V_1 \wedge j \in V_2\}$ *which describes a partial bijection from* V_1 *to* V_2 *and satisfies the following conditions:*

(1) ***structure cond.:*** $\forall (i, j) \in M$, $\text{rbd}_{P_1}(i) \Leftrightarrow \text{rbd}_{P_2}(j) \wedge \text{lbd}_{P_1}(i) \Leftrightarrow \text{lbd}_{P_2}(j)$
(2) ***base cond.:*** $\forall (i, j) \in M$, $\neg\text{bd}_{P_1}(i) \Rightarrow S_1(i) = S_2(j)$

Note that the structure condition implies $\neg\text{bd}_{P_1}(i) \Rightarrow \neg\text{bd}_{P_2}(j)$ for any $(i, j) \in M$ since $\forall k \in \{1, 2\} \forall i$, $\text{bd}_{P_k}(i) \Leftrightarrow \text{rbd}_{P_k}(i) \vee \text{lbd}_{P_1}(i)$. The structure condition guarantees that specific biological entities (namely stems, which consist of a sequence of stacked bonds) are handled in a correct way (see Figure 4).

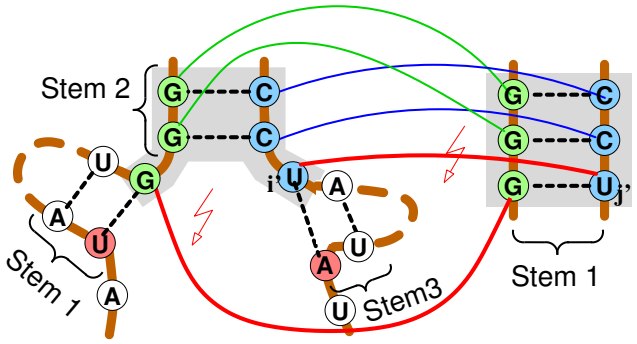

Figure 4. Matching excluded by the structure condition. The red lines indicate matches that do not satisfy the structure condition. They would map bases from stems 1 and 3 of the first structure to the same stem 1 of the second structure, which is biologically unreasonable.

The base condition is applied to unbound (or single) bases. The reason is that we want to treat basepairs as units. Furthermore, we want to have different treatments of basepairs depending on our application. In some applications, we are not interested in preserving the type of basepairs in matchings. Thus, we could match a A—U bond onto a G—C. In this case, we don't need to extend the definition of matchings further. Fig. B.2 in the appendix shows patterns that can be detected using this definition of matchings.

If, on the other hand, we want to preserve the bond type, then we need to add the following condition.

Definition 3.2 (Matching (Cont.))

(3) ***bond cond.:*** *for each* $\{(i,j),(i',j')\} \subseteq M$ *with* $i \overset{P_1}{\text{—}} i'$ *and* $j \overset{P_2}{\text{—}} j'$ *we have* $S_1(i) = S_2(j) \wedge S_1(i') = S_2(j')$

The range of positions matched in the first RNA is defined by

$$\mathrm{ran}_1(M) = \{i \mid \exists j : (i,j) \in M\},$$

and $\mathrm{ran}_2(M)$ is defined analogously. Given an element $i \in \mathrm{ran}_1(M)$, we denote by $M(i)$ the uniquely determined element j with $(i,j) \in M$. Similarly, given an element $j \in \mathrm{ran}_2(M)$, we denote by $M^{-1}(j)$ the uniquely determined element i with $(i,j) \in M$.

Note that so far, we have not added the condition that a matching must preserve the order of the bases (i.e., $i < i'$ and $(i,j),(i',j') \in M$ implies $j < j'$). We will show later in Lemma 3.8 that this is a consequence of our definition of connected matchings. As we will discuss in this and the next section, it is not sufficient that the ranges of a matching in both RNAs are connected. Instead, some of the connections (namely bonds) themselves have to be preserved.

When considering an RNA as a graph, we have two different kinds of edges, namely backbone connections and bonds. To define paths through this graph, we define the *transition type* of an edge connecting two positions i and i' to be 1, 2, 3 or 4, depending on whether $i' = i + 1$, $i' = i - 1$, $i \overset{P}{\text{---}} i'$, or $i' \overset{P}{\text{---}} i$. A *path* in an RNA is a sequence of positions $i_1 \dots i_k$, such that the bases $S(i_l)$ and $S(i_{l+1})$ for $l = 1, \dots, k-1$ are connected by the backbone or by bonds in the RNA.

Definition 3.3 (Matching/Matched Path) *Let (S_1, P_1) and (S_2, P_2) be two RNAs and M a matching between them. An* M-matching path *is a list of pairs $(i_1, j_1) \dots (i_k, j_k) \in M$ such that*

(1) $i_1 \dots i_k$ is a path in (S_1, P_1);
(2) $j_1 \dots j_k$ is a path in (S_2, P_2); and
(3) for each $1 \leq l < k$ the transition types of (i_l, i_{l+1}) and (j_l, j_{l+1}) are equal.

A matching is connected *if there is an M-matching path between any two pairs in M. A path in only one of the RNAs consisting of only matched bases is called an* M-matched path.

We define an *M-matching path in structure 1 (resp. structure 2)* to be the corresponding restriction of an M-matching path to the elements in V_1 (resp. V_2). If the structure is clear from the context, we omit "in structure i".

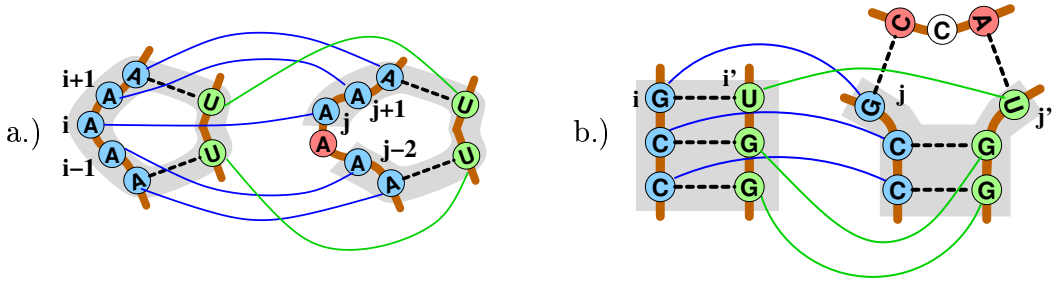

Figure 5. Matchings with unpreserved bonds (backbone and secondary). a.) i is matched to j, but the backbone bond between $i-1$ and i is not preserved. b.) i, i' is matched with j, j', respectively, but the bond $i \overset{P_1}{\text{---}} i'$ is not preserved. The matching is indicated by blue and green nodes. In both cases, the corresponding bases in the second structure are connected with nodes (in red) that are not part of the matching.

Note the difference between *matching* paths and *matched* paths. A matched path is a path occurring in one structure, but there must not be necessarily a corresponding path in the other structure. Furthermore, the restriction of *matching* paths to some structure clearly produces a *matched* path. But the contrary is not true. To clarify this, consider the simplest matched paths, which are edges (backbone connections or bonds) between matched bases. By definition, they are matched paths, but there might not be a matching path associated with them. This happens for bases which mark the "ends" of the

matching. The two cases for backbone edges and bond edges are shown in Figure 5.

Before we proceed, we want to show some properties required for the calculation of maximally extended matchings. They will be calculated by a dynamic programming method, using an ordering combined from the stacking order and backbone order. From the definitions of matchings, it is not yet clear whether they respect stacking and backbone ordering.

Proposition 3.4 *Let* $(i_1, j_1) \dots (i_k, j_k) \in M$ *be a matching path. Then the path preserves the relative stacking order, i.e. for all* $1 \leq r \leq k$ *we have* $\mathrm{stord}_{P_1}(i_1) - \mathrm{stord}_{P_2}(j_1) = \mathrm{stord}_{P_1}(i_r) - \mathrm{stord}_{P_2}(j_r)$.

Proof. By induction. Consider i_{r+1}, j_{r+1}.

$i_{r+1} = i_r + 1 \wedge \neg \mathrm{bd}_{P_1}(i_r)$. Then $\mathrm{stord}_{P_1}(i_r) = \mathrm{stord}_{P_2}(i_{r+1})$. Furthermore, we have $j_{r+1} = j_r + 1 \wedge \neg \mathrm{bd}_{P_1}(j_r)$, which immediately implies $\mathrm{stord}_{P_1}(j_r) = \mathrm{stord}_{P_2}(j_{r+1})$.

$i_{r+1} = i_r - 1 \wedge \neg \mathrm{bd}_{P_1}(i_{r+1})$ Same as the last case.

$i_r \overset{P_1}{\text{———}} i_{r+1}$ **(resp.** $i_{r+1} \overset{P_1}{\text{———}} i_r$**)** . Then $\mathrm{stord}_{P_1}(i_r) = \mathrm{stord}_{P_2}(i_{r+1})$. Since we consider a matching path, the transition of j_r to j_{r+1} must be of the same matching type. Hence, $j_r \overset{P_1}{\text{———}} j_{r+1}$ (resp. $j_{r+1} \overset{P_1}{\text{———}} j_r$), which implies immediately $\mathrm{stord}_{P_1}(j_r) = \mathrm{stord}_{P_2}(j_{r+1})$.

$i_{r+1} = i_r + 1 \wedge \mathrm{lbd}_{P_1}(i_r)$**:** Then we know $\mathrm{stord}_{P_2}(i_{r+1}) = \mathrm{stord}_{P_1}(i_r) + 1$. Furthermore, we have $j_{r+1} = j_r + 1 \wedge \mathrm{lbd}_{P_1}(j_r)$ and therefore $\mathrm{stord}_{P_2}(i_{r+1}) = \mathrm{stord}_{P_1}(i_r) + 1$.

The other cases $(i_{r+1} = i_r + 1 \wedge \mathrm{rbd}_{P_1}(i_r))$, $(i_{r+1} = i_r - 1 \wedge \mathrm{lbd}_{P_1}(i_{r+1}))$ and $(i_{r+1} = i_r - 1 \wedge \mathrm{rbd}_{P_1}(i_{r+1}))$ are analogous to the last case. □

Since we are considering only connected matchings in the following, it immediately follows that matchings preserve the stacking order. For the backbone order, we need one additional technical construction.

Definition 3.5 *For a matching* M*, we define the* sub-matching $M^{\leq r}$ of stacking order $\leq r$ *to be the subset of* M *containing only pairs* (i, j) *with* $\mathrm{stord}_{P_1}(i) \leq r$*. Analogously, we define* $M^{\geq r}$ *and* M^r*.*

Note that we considered in the definition only the stacking order in the first structure. We will later show that the stacking order is preserved by connected matchings. Furthermore, note that by definition, $M^{\leq 0}$ is the empty set.

In the following, we will show that $M^{\leq r}$ is connected. In principle, it suffices for that purpose to show that the minimal matching path that visits both endpoints of a bond consists of the bond itself.

Proposition 3.6 *Let $i \overset{P_1}{\text{---}} i'$ be two elements that are matched. If there is a matching path using only positions k with $i \leq k \leq i'$, then $M(i) \overset{P_2}{\text{---}} M(i')$.*

Proof. By the structure property, we know that $M(i) \overset{P_2}{\text{---}} j$ and $j' \overset{P_2}{\text{---}} M(i')$ for some j, j'. Since $\text{stord}_{P_1}(i) = \text{stord}_{P_1}(i')$, we know by Proposition 3.4 that $\text{stord}_{P_1}(M(i)) = \text{stord}_{P_1}(M(i'))$. By the nestedness, we get $M(i) < j < j' < M(i')$. If $j = M(i')$ and $j' = M(i)$, then the claim holds.

So suppose that $j \neq M(i')$, which implies $j' \neq M(i)$. I.e., we have the situation as depicted in Figure 5b. Consider a matching path that uses only bases between k with $i \leq k \leq i'$ in P_1. Then the corresponding path in P_1 will contain exactly two bases with the same stacking order as $\text{stord}_{P_1}(i)$, namely i itself and i'. On the other hand, the corresponding path in P_2 will contain at least 4 bases with the same stacking order than $\text{stord}_{P_1}(M(i))$, namely $M(i), j, j'$ and $M(i')$, which is a contradiction to Proposition 3.4. □

Lemma 3.7 *Let M be a connected matching, and let $M^{\leq r}$ be a sub-matching of order $\leq r$ with $r \geq 1$. Then $M^{\leq r}$ is connected.*

Proof. Via induction on the stacking order. For the base case, there is nothing to prove since $M^{\leq 0} = \emptyset$. For the induction step, consider the set of added elements $\text{ran}_1(M^{r+1}) = \text{ran}_1(M^{\leq r+1}) \backslash \text{ran}_1(M^{\leq r})$ in the first RNA. By induction hypotheses we know that $M^{\leq r}$ is connected.

Let $i < i'$ be any two elements in $\text{ran}_1(M^{r+1})$, and let $i = i_1 \dots i_m = i'$ be the shortest M-matching path between i and i' in P_1. We will show that every position in the connecting path in $\{i_1 \dots i_m\}$ has stacking order $\leq r+1$, which implies $\{i_1 \dots i_m\} \subseteq \text{ran}_1(M^{\leq r+1})$. By Proposition 3.4 it then follows that $M^{\leq r+1}$ is connected.

So suppose that there is a i_{l+1} with stacking order $\geq r+2$. Let i_{l+1} be minimal with this property. We have the following cases according to the bond type of the connection between i_{l+1} and i_l used in the shortest M-matching path:

$i_l \overset{P_1}{\text{---}} i_{l+1}$**:** By our assumption, $i_l \in \text{ran}_1(M^{r+1})$, which implies $\text{stord}_{P_1}(i_l) \leq r+1$. Since i_{l+1} must have the same stacking order as i_l by definition, we get immediately $i_{l+1} \in \text{ran}_1(M^{r+1})$, which is a contradiction.

$i_{l+1} \overset{P_1}{\text{---}} i_l$ Analogous to the last case.

$i_{l+1} = i_l + 1$**:** Since we have supposed that $i_{l+1} \notin \text{ran}_1(M^{r+1})$, we get immediately that $\text{stord}_{P_1}(i_l) = r+1$ and $\text{stord}_{P_1}(i_{l+1}) = r+2$. This is only possible if there is a bond $i_l \overset{P_1}{\text{---}} k$ with $i_l \leq i_{l+1} \leq k$. Since $\text{stord}_{P_1}(i_m) \leq r+1$, we know that any path from i_{l+1} to i' must go through i_l or k. In the first case, we get an immediate contradiction to the assumption that we have used the shortest matching path. In the second case, we know that k must be contained in $\text{ran}_1(M)$ and therefore in $\text{ran}_1(M^{r+1})$.

Hence, $i_l \overset{P_1}{\text{———}} k$ is a matched path. Furthermore, we can apply Proposition 3.6, from which we get that $i_l \overset{P_1}{\text{———}} k$ is also a matching path. But this is a contradiction to the assumption of a shortest matching path.

$i_{l+1} = i_l - 1$**:** Analogously to the last case.

□

Now we are able to show that the backbone order is preserved as well.

Lemma 3.8 (Backbone Order) *Let M be a connected matching, and (i, i'), $(j, j') \in M$. Then $i < i'$ if and only if $j < j'$.*

Proof. Via induction on $M^{\leq r}$. So let $M^{\leq r}$ be any sub-matching, and let the claim hold for any $(k, k'), (l, l') \in M^{\leq r}$. Consider any $(i, M(i)), (i', M(i')) \in M^{r+1}$. Since M is connected, so is $M^{\leq r+1}$ by Lemma 3.7. Hence, there must be a shortest matching path from i to i' in $\mathrm{ran}_1(M^{\leq r+1})$. We will show the claim only for the case $i \in \mathrm{ran}_1(M^{r+1})$, $i' \in \mathrm{ran}_1(M^{\leq r})$ and $i < i'$. The other cases are similar.

Let $k < i < l$ be the bond with largest stacking order enclosing i (which must exist). Then k, l have stacking order r. Since any element in $]k..l[$ has stacking order $\geq r + 1$ and $i' \in M^{\leq r}$ with $i < i'$, we know that $k < l \leq i'$.

Now any $M^{\leq r+1}$-matching path between i, i' must go either via k or l. In the first case, we know that $M(i) < M(l)$ since the shortest connecting path can use only transitions that increase the positions in both structures. By induction hypotheses we know that $M(l) \leq M(i')$, from which the claim follows immediately.

In the second case, there must be a bond $k' \leq k < l \leq l'$ that is matched, since any path between i and i' using k must go through such a bond. Let $i = i_1 \dots i_s = k$ be the matching sub-path connecting i and k. Since we have assumed the shortest path, we know by Proposition 3.6 that $i_1 \dots i_{s-1}$ have all stacking order $r + 1$. Then by Proposition 3.4, all $M(i_1) \dots M(i_{s-1})$ have the same stacking order $t + 1$, where t is the stacking order of $M(k)$. Since $M(k)$ must satisfy $\mathrm{lbd}_{P_2}(k)$, has stacking order t and is connected with i_1 via elements having stacking order $t + 1$, we get immediately $M(k) < M(i_1) = M(i) < g$, where $M(k) \overset{P_2}{\text{———}} g$. This implies

$$M(k') \leq M(k) < M(i) < g \leq M(l') \leq M(i')$$

by induction hypotheses and the nestedness of structures. □

4 Bond Preserving Matching

As Figure 5b indicates, a matched bond $i \overset{P_1}{\text{---}} i'$ which does not correspond to a matching path occurs if we have a stem in the first structure that is matched to a multiple loop in the second structure (or vice versa). This does not make sense in a biological interpretation, since it is very unlikely that this pattern could have been generated by evolution. For that reason, we are interested in matchings that preserve bonds.

Definition 4.1 (Bond-Preserving Matching) *A connected matching M is said to be bond-preserving if every matched bond in P_1 or P_2 is also a matching path, i.e. if $\{(i,j),(i',j')\} \subseteq M$ and $i \overset{P_1}{\text{---}} i'$, then $j \overset{P_2}{\text{---}} j'$, and vice versa.*

In the following, we will consider bond-preserving matchings. We say that a connected, bond-preserving matching M is *maximally extended*, if there is no M' such that $M \subsetneq M'$. We are interested in finding all (non-overlapping) maximally extended matchings. The following proposition allows us to decompose the problem of finding a maximally extended matching into subproblems of finding maximally extended loop matchings.

Proposition 4.2 *Let M be a connected matching with* $\text{ran}_1(M) \cup \text{loop}(i \overset{P_1}{\text{---}} i') \neq \emptyset$. *Then M restricted to* $\text{loop}(i \overset{P_1}{\text{---}} i') \cup \{i, i'\}$ *is a connected matching.*

Proof. Let $r = \text{stord}_{P_1}(i) + 1$ be the stackorder of the elements in $\text{loop}(i \overset{P_1}{\text{---}} i')$. We have already shown in Lemma 3.7 that $M^{\leq r}$ is a connected matching if M is. Now let $j < j'$ be two elements in $\text{loop}(i \overset{P_1}{\text{---}} i')$. Now either there is directly a matching path between j and j' using only $\text{loop}(i \overset{P_1}{\text{---}} i')$, or the matching path $M^{\leq r}$ has to use both bond ends i and i' to connect j with j'. In this case, $i - i'$ is a matching path due to the bond-preserving property, and there is a matching path using only elements from $\text{loop}(i \overset{P_1}{\text{---}} i') \cup \{i, i'\}$. □

Dynamic Programming Matrices

We want to find all non-overlapping, maximally extended, bond-preserving matchings. For overlapping matchings, we choose the one with maximal size. If there are overlapping matchings of the same size, then only one is selected.

We use a dynamic programming approach by filling a matrix $M(r,s)$, with the following interpretation. Since our matchings preserve backbone as well as

stacking order, we define an order $\prec$ on elements as follows:

$$i \prec j \equiv \begin{cases} i < j & \text{if } \mathrm{stord}_{P_1}(i) = \mathrm{stord}_{P_1}(j) \\ \mathrm{stord}_{P_1}(i) < \mathrm{stord}_{P_1}(j) & \text{otherwise} \end{cases}$$

For pairs (r,s) and (k,l) we define $(r,s) \prec (k,l)$ if and only if $r \prec k$. Then

$$M(r,s) = \max \left\{ |M| \;\middle|\; \begin{array}{l} M \text{ is a maximally extended matching} \\ \text{with } (r,s) \in M \text{ and there is no} \\ (r',s') \in M \text{ with } (r',s') \prec (r,s) \end{array} \right\}$$

contains the size of a maximal matching. For simplicity, we assume the maximum value over an empty set to be 0. Note that the size is stored only for the left-most, bottom-most pair (r,s) in M. For calculating $M(r,s)$, we will additionally need auxiliary matrices $M^{r-^{end}}$, M^{bb} and M^{rb}, which are defined as follows.

Definition 4.3 (Auxiliary Matrices) *Let $R_1 = (S_1, P_1)$ and $R_2 = (S_2, P_2)$ be two RNAs. Let r (resp. s) be an element of* $\mathrm{loop}(i \overset{P_1}{\text{---}} i')$ *(resp.* $\mathrm{loop}(j \overset{P_2}{\text{---}} j')$*). Then $M^{loop}_{\preceq}(r,s)$ is the size of the maximal matching within the loops that contain (r,s), and is extended to the right or above (r,s), i.e.*

$$M^{loop}_{\preceq}(r,s) = \max \left\{ |M| \;\middle|\; \begin{array}{l} M \subseteq [i..i'] \times [j..j'] \text{ is a matching with } (r,s) \in M \\ \text{such that } M \text{ restricted to } \mathrm{loop}(i \overset{P_1}{\text{---}} i') \text{ is connected} \\ \text{and } \forall (r',s') \in M \backslash \{(i,i'),(j,j')\} : (r,s) \preceq (r',s') \end{array} \right\}$$

In addition, we define for every i, j such that $i \overset{P_1}{\text{---}} i'$ and $j \overset{P_2}{\text{---}} j'$ the matrix element $M^{bb}(i,j)$ to be the maximal matching that matches the bonds $i \overset{P_1}{\text{---}} i'$ and $j \overset{P_2}{\text{---}} j'$, i.e.

$$M^{bb}(i,j) = \max \left\{ |M| \;\middle|\; \begin{array}{l} M \subseteq [i..i'] \times [j..j'] \text{ is a} \\ \text{connected matching with} \\ (i,j) \in M \text{ and } (i',j') \in M \end{array} \right\}$$

In addition, we define $M^{rb}(i',j')$ to be the maximal matching containing the right partners i' and j' of the bonds only, i.e.

$$M^{rb}(i',j') = \max \left\{ |M| \;\middle|\; \begin{array}{l} M \in [i+1..i'] \times [j+1..j'] \\ \text{is a connected matching} \\ \text{with } (i',j') \in M \end{array} \right\}$$

$M^{loop}_{\preceq}(r,s)$ will be calculated for a matching of two loops associated with the bonds $i \overset{P_1}{\text{———}} i'$ and $j \overset{P_2}{\text{———}} j'$, given that $M^{loop}_{\preceq}$, M^{bb} and M^{rb} is already calculated for all bonds that are contained in the two loops. For calculating $M^{bb}(i,j)$, we use additional auxiliary variables. The variable $RDist$ stores the loop distance to the right-end of the loop. Thus, for given $RDist$, we consider elements r and s which have distance $RDist$ to i' and j', respectively. Looking from the right end (i',j') of the loop this implies that

$$r = \text{left}^{RDist}_{R_1}(i') \qquad \text{and} \qquad s = \text{left}^{RDist}_{R_2}(j').$$

First, we need to know whether there is a matching connecting (r,s) with the right ends of the loop (i',j'):

$$Reach^{r-end}(RDist) = \begin{cases} \text{true} & \text{if there is a connected matching} \\ & M \subseteq [i..i'] \times [j..j'] \text{ with } (r,s) \in M \\ & \text{and } (i',j') \in M \\ \text{false} & \text{otherwise} \end{cases} \tag{1}$$

Since we don't need the matrix entries any further, we only store the current value in the variable $Reach$. In addition, we store the size of the matching that is used in the definition of $Reach^{r-end}(RDist)$. If $Reach^{r-end}(RDist)$ is false, then we use the size of the last entry $Reach^{r-end}(RDist')$ with $RDist' < RDist$ and $Reach^{r-end}(RDist') = true$. Technically, this is achieved by an array $M^{r-end}(RDist)$ with

$$M^{r-end}(RDist) = \max \left\{ |M| \;\middle|\; \begin{array}{l} M \subseteq [i..i'] \times [j..j'] \text{ is a connected} \\ \text{matching with } (i',j') \in M \text{ and} \\ \forall (r',s') \in M \backslash \{(i,i'),(j,j')\} \text{ we} \\ \text{have } (r,s) \preceq (r',s') \end{array} \right\} \tag{2}$$

Proposition 4.4 *If $Reach^{r-end}(RDist) = true$, then $M^{r-end}(RDist) = M^{loop}_{\preceq}(r,s)$, where $r = \text{left}^{RDist}_{R_1}(i')$ and $s = \text{left}^{RDist}_{R_2}(j')$.*

5 Recursion Equations

For the recursion equations, we introduce the following predicates. Given two positions i and j, then

$$\text{match}(i,j) = \Big[S(i) = S(j) \wedge (\text{lbd}_{P_1}(i) \leftrightarrow \text{lbd}_{P_2}(j)) \wedge (\text{rbd}_{P_1}(i) \leftrightarrow \text{rbd}_{P_2}(j)) \Big]$$

denotes bases that satisfy the structure and base conditions (see Def. 3.1). The predicate match($i \overset{P_1}{\text{---}} i', j \overset{P_2}{\text{---}} j'$) denotes basepairs that can be matched. If we do not require that the matching preserves the bond type (i.e. if we are considering matchings according to Def. 3.1 only), then this predicate is defined by

$$[i \overset{P_1}{\text{---}} i'] \wedge [j \overset{P_2}{\text{---}} j']$$

Otherwise, if we consider matchings according to Definitions 3.1 and 3.2, then we have

$$[i \overset{P_1}{\text{---}} i'] \wedge [j \overset{P_2}{\text{---}} j'] \wedge [S_1(i) = S_2(j)] \wedge [S_1(i') = S_2(j')].$$

Note that for the rest of the paper, it does not make any difference which definition of match($i \overset{P_1}{\text{---}} i', j \overset{P_2}{\text{---}} j'$) we are using.

The auxiliary matrices and arrays can be easily calculated via the following recursion equations. For $M_{\preceq}^{loop}(r,s)$ we have

$$M_{\preceq}^{loop}(r,s) = \tag{3}$$
$$\begin{cases} M^{bb}(r,s) + M_{\preceq}^{loop}(r'+1,s'+1) & \text{if match}(r \overset{P_1}{\text{---}} r', s \overset{P_2}{\text{---}} s') \\ M^{rb}(r,s) + M_{\preceq}^{loop}(r+1,s+1) & \text{else if } \mathrm{rbd}_{P_1}(r) \wedge \mathrm{rbd}_{P_2}(s) \wedge \text{match}(r,s) \\ 1 + M_{\preceq}^{loop}(r+1,s+1) & \text{else if match}(r,s) \\ 0 & \text{otherwise} \end{cases}$$

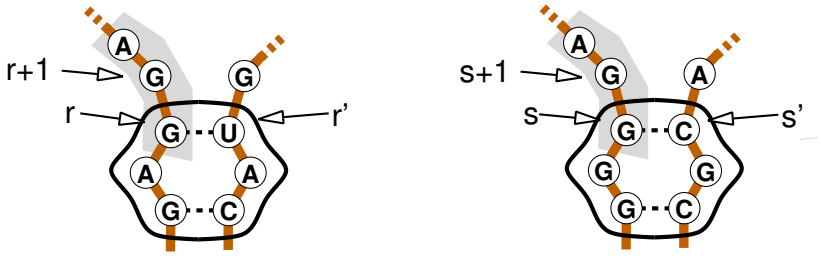

Figure 6. Extension to next loop.

Note that if r and s are the left ends of the bonds $r \overset{P_1}{\text{---}} r' \wedge s \overset{P_2}{\text{---}} s'$, but the bonds are not matchable, then this case is covered by the third case. Here, $r+1$ and $s+1$ are *not* in the same loop as r, s. Therefore, we consider the case where the maximal matching extends to the next loop via the left ends of two bonds. This case is depicted in Figure 6. r and s do match, whereas the bonding partners r' and s' do not match. The currently considered loop is encircled. Since $r+1$ and $s+1$ in the contiguous loop do match, we know that we can calculate $M_{\preceq}^{loop}(r,s)$ recursively by calculating $M_{\preceq}^{loop}(r+1,s+1)$.

As the next step, we define the auxiliary arrays $Reach^{r-end}(RDist)$ as well as $M^{r-end}(RDist)$ for a given loop. $RDist$ is the distance to the right end of the

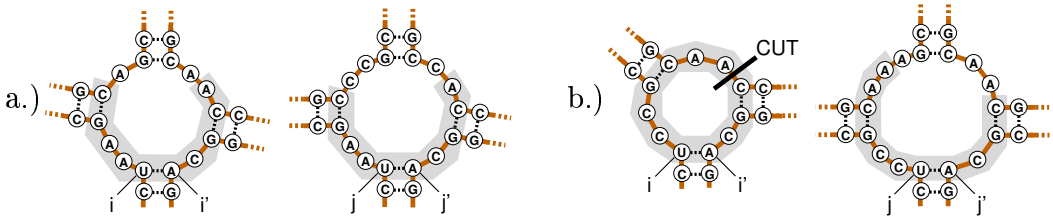

Figure 7. The two possible cases for $M^{bb}(i \overset{P_1}{\text{—}} i', j \overset{P_2}{\text{—}} j')$

closing bond. Consider the case where we want to match two loops associated with the bonds $i \overset{P_1}{\text{—}} i'$ and $j \overset{P_2}{\text{—}} j'$. Let len be the minimum of the two loop lengths, and $0 \leq RDist < len$. Then

$$Reach^{r\text{–}^{end}}(0) = \begin{cases} true & \text{if match}(i', j') \\ false & \text{otherwise} \end{cases} \text{ and } M^{r\text{–}^{end}}(0) = \begin{cases} 1 & \text{if match}(i', j') \\ 0 & \text{otherwise} \end{cases}$$

For $1 \leq RDist \leq len_{min}$, let $r = \text{left}_{R_1}^{RDist}(i')$ and $s = \text{left}_{R_2}^{RDist}(j')$ be the two positions with distance $RDist$ to the right end of the considered loops. Then we obtain

$$Reach^{r\text{–}^{end}}(RDist) = Reach^{r\text{–}^{end}}(RDist-1) \wedge \text{match}(r, s)$$
$$M^{r\text{–}^{end}}(RDist) = \begin{cases} M_{\preceq}^{loop}(r, s) & \text{if } Reach^{r\text{–}^{end}}(RDist) \\ M^{r\text{–}^{end}}(RDist-1) & \text{otherwise.} \end{cases}$$

The matrix $M^{rb}(i', j')$ then is simply

$$\max_{0 \leq Rdist < len_{min}} \left\{ M^{r\text{–}^{end}}(RDist) \right\}$$

For the M^{bb} matrix, there are two different cases as shown in Figure 7. In the first case a.), the extensions from the initial matching (i, i') to the right, and the extension from (j, j') to the left do not overlap, whereas they do overlap in the second case b). For the second case, we do not know exactly how to match the overlapping part. Hence, we have to consider all possible *cuts* in the smaller loop, marking the corresponding ends of the extensions from the left ends and from the right ends of the loop. The extensions from the right ends are already calculated in the $M^{r\text{–}^{end}}$ matrix. Only for the definition of the recursion equation, we define $M^{l\text{–}^{end}}(LDist)$ and $Reach^{l\text{–}^{end}}(LDist)$ analogously to equations (2) and (1), respectively. For the implementation, we need to store only the current values $M^{l\text{–}^{end}}$ and $Reach^{l\text{–}^{end}}$.

Now let $len_{i,i'}$ (resp. $len_{j,j'}$) be $|\text{loop}(i \overset{P_1}{\text{—}} i')|$ (resp. $|\text{loop}(i \overset{P_1}{\text{—}} i')|$), and

let $len_{min} = \min\{len_{i,i'}, len_{j,j'}\}$. Then we have

$$M^{bb}(i,j) = \max_{\substack{0 \le LDist < Len_{Min} \\ \text{with right}_{P_1}^{LDist}(i) \text{ is not} \\ \text{a left end of a bond}}} \left\{ \begin{array}{c} M^{l\_end}(LDist) \\ + M^{r\_end}(RDist) \end{array} \right\} \tag{4}$$

$$\text{where } RDist = \begin{cases} len_{i,i'} - LDist & \text{if } len_{min} = len_{i,i'}, \\ len_{i,i'} + (len_{i,i'} - len_{j,j'}) - LDist & \text{otherwise.} \end{cases}$$

The condition $\text{right}_{P_1}^{LDist}(i)$ *is not a left end of a bond* guarantees that we do not cut in the middle of a bond, which is excluded since we are considering bond-preserving matchings only. The term $(len_{i,i'} - len_{j,j'})$ in the second part of the definition of $RDist$ is to compensate for the longer length of the first loop. [1]

Finally, we consider the $M(r,s)$ entries. Let r and s again be two bases of the loops defined $i \overset{P_1}{\text{---}} i'$ and $j \overset{P_2}{\text{---}} j'$ with distance $RDist$ to the right loop ends i' and j', respectively. The values of $M(r,s)$ and $Mloop(r,s)$ are equal for all entries $M(r,s) \neq 0$. $M(r,s)$ is zero if there is some $(r',s') \prec (r,s)$ that is matchable. This leads to the following equation: $M(r,s) = 0$ if $\neg\text{match}(r,s)$ or $\text{match}(\text{left}_{R_1}(r), \text{left}_{R_2}(s))$ or $Reach^{r\_end}(RDist)$, and $M_{\preceq}^{loop}(r,s)$ otherwise.

6 Pseudo-Code

The main procedure consists of two for-loops, each calling a base-pair from the first and second RNA, and performs the pattern search from inner to outer loops. It calls the procedure START-LOOP-WALKING which initiates the calculation of all matrices except $M^{bb}(i,j)$ for two bonds $i \overset{P_1}{\text{---}} i'$ and $j \overset{P_2}{\text{---}} j'$, assuming that all matrix entries for loops above are already calculated. In addition, it calculates the loop length of the smaller loop and the distance of the two loop lengths (which is done in the sub-procedure CALC-REMAIN-LOOP-LEN, see appendix).

The real calculation of these matrices is done in the sub-procedure LOOP-WALKING, which traverses the loop from right to left (via the application of left.$(\cdot)$ function). The function LOOP-WALKING has two modes concerning whether we started the loop-traversal with both right ends i', j' or not. In

[1] In the case that $i \overset{P_1}{\text{---}} i'$ is the smaller loop, then overlapping of the left and right match extensions is already excluded by definition, and we do not need to compensate for it

```
1: procedure START-LOOP-WALKING(i, i', j, j')
2:     reach = INIT-LOOP-MATRICES(i', j', i', j')
3:     (loop_size, loop_dist) := LOOP-WALKING(i', j', i, j, i', j', reach, true)
4:     k := i'
5:     while k > i + 1 do
6:         k := left_{R_1}(k)
7:         INIT-LOOP-MATRICES(k, j', i', j')
8:         LOOP-WALKING(k, j', i, j, i', j', false, false)
9:     end while
10:    l := j'
11:    while l > j + 1 do
12:        l := left_{R_2}(l)
13:        INIT-LOOP-MATRICES(i', l, i', j')
14:        LOOP-WALKING(i', l, i, j, i', j', false, false)
15:    end while
16:    return (loop_size, loop_dist)
17: end procedure
```

Figure 8. Starting points of loop walking

the first mode (initiated in line of START-LOOP-WALKING), we calculate also the array M^{r-end}, and move the $M(r,s)$ down to (i',j') for all (r,s) where $Reach^{r-end}$ is true. This part is done by the sub-procedure LOOP-REACH (listed in the appendix). In the second mode, when LOOP-WALKING is called with only one right end (lines 8 and 14 of START-LOOP-WALKING), then we know the right ends cannot be in any matching considered there. In this

```
1: procedure LOOP-WALKING(r, s, i, j, i', j', reach, right_ends)
2:     RDist = 0
3:     while r > i ∧ s > j do
4:         r' := r;   s' := s;   r := left_{R_1}(r');   s := left_{R_2}(s')
5:         RDist = RDist + 1
6:         if BASE-MATCH(r, s) ∨ BOND-MATCH(r^r, r, s^r, s) then
7:             MLOOP-RECURSION(r^r, r, s^r, s')
8:             M(r, s) := M_≤^loop(r, s);   M(r', s') := 0
9:             If right_ends then LOOP-REACH(r, s, i, j, i', j', reach, RDist)
10:        else
11:            M_≤^loop(r, s) := 0;   M(r, s) := 0;   reach := false
12:            If right_ends then M^{r-end}(RDist) := M^{r-end}(RDist − 1)
13:        end if
14:    end while
15:    if right_ends then
16:        return CALC-REMAIN-LOOP-LEN(r, s, i, j, RDist)
17:    end if
18: end procedure
```

Figure 9. The procedure loop walking is going from one base to the next

case, we may not calculate the M^{r_end} array. The sub-procedure MLOOP-RECURSION is just an implementation of recursion Equation (3) for $M^{loop}_{\preceq}$.

The sub-procedure INIT-LOOP-MATRICES of START-LOOP-WALKING just initializes the matrices for the starting points. In most cases, the initial values are 0 (since we cannot have a match if we do not start with the right-ends due to the structure condition). The only exception is if we start with both right ends, and these rights ends do match. In this case, we initialize the corresponding matrix entries with 1. The sub-procedure INIT-LOOP-MATRICES is listed in the appendix.

The next step is to calculate $M^{bb}(i,j)$, which is done by the procedure LOOP-MATCHING. LOOP-MATCHING is called *after* START-LOOP-WALKING is finished. In principle, this is just an implementation of the recursion equation (4). Since we do not want to maintain another array $M^{l\_end}(LDist)$, we store only the value for the current $LDist$ in the variable M^{l_end}. The procedure maintains three neighboring cells (r^l, s^l), (r,s) and (r^r, s^r). (r^l, s^l) corresponds to $LDist-1$, and (r,s) to $LDist$. The cut will be between (r,s) and (r^r, s^r).

```
 1: procedure LOOP-MATCHING(i, i', j, j', i_i'_lens, lens_dist)
 2:     LDist := 0
 3:     if BOND-MATCH(i, i', j, j') then
 4:         M^{l_end} := 0;   Reach^{l_end} := true
 5:         r^r := i;   r := i;   r^l := i;   s^r := j;   s := j;   s^l := j
 6:         while r^r < i' ∧ s^r < j' ∧ Reach^{l_end} := true do
 7:             r^l := r; r := r^r; r^r := right_{R_1}(r^r); s^l := s; s := s^r; s^r :=
                    right_{R_2}(s^r);
 8:             if BASE-MATCH(r, s) ∨ BOND-MATCH(r^r, r, s^r, s) then
 9:                 M^{l_end} = MLEND-RECURSION(r^l, r, r^r, s^l, s, s^r, M^{l_end})
10:             else Reach^{l_end} := false endif
11:             if Reach^{l_end} ∧ ¬BOND-MATCH(r^l, r, s^l, s) then
12:                 FILL-MBB(i, i', j, j', M^{l_end}, LDist, i_i'_len, lens_dist)
13:             end if
14:             LDist := LDist + 1
15:         end while
16:     else M^{bb}(i, j) := 0 end if
17: end procedure
```

Figure 10. Calculation of M^{bb}

The sub-procedure FILL-MBB sets the entry of $M^{bb}(i,j)$ to the current maximum according to equation 4 (see appendix). The sub-procedure MLEND-RECURSION is in principle only an implementation of the recursion equation for M^{l_end} *under* the condition that $Reach^{l\_end}$ is true. It is assumed that M^{l_end} is already calculated up to (r^l, s^l) when MLEND-RECURSION is called, and that we want to calculate it for the right neighbors (r,s) of (r^l, s^l). The currently considered cut in LOOP-MATCHING is between (r,s) and their right

neighbors (r^r, s^r).

As can be seen from the definition of $M^{r}{}_{-}{}^{end}$ in Equation (2), the recursion equation under this condition is in principle analogous to the recursion equation for $M^{loop}_{\preceq}$ given in Equation (3). There are two differences. First, the analogous case (with right and left exchanged) shown in Figure 6 cannot happen since we have $Reach^{l}{}_{-}{}^{end}$. Since this case is not treated separately in Equation (3), we do not see any differences here. Second, the recursion requires to evaluate three cells (first case in Equation (3). These three cells are here given by (r^l, s^l), (r, s) and (r^r, s^r), respectively. But we do not have the corresponding three $Reach^{l}{}_{-}{}^{end}$ entries. We have only the current $Reach^{l}{}_{-}{}^{end}$ (corresponding to cell (r^l, s^l)), and the one we are currently calculating (corresponding to cell (r, s)). This can be solved by looking in the "future", which is done in line 5 of procedure MLEND-RECURSION.

```
 1: procedure MLEND-RECURSION(r^l, r, r^r, s^l, s, s^r, M^l_-end)
 2:     if BOND-MATCH(r^l, r, s^l, s) then
 3:         M^l_-end := M^l_-end + M^bb(r, s)
 4:     else if lbd_P1(r) ∧ lbd_P2(s) ∧ BASE-MATCH(r, s) then
 5:         if ¬BOND-MATCH(r, r^r, s, s^r) then
 6:             M^l_-end := M^l_-end + 1 + M^loop_≼(r + 1, s + 1))
 7:         end if
 8:     else if BASE-MATCH(r, s) then
 9:         M^l_-end := M^l_-end + 1
10:     end if
11:     return M^l_-end
12: end procedure
```

Figure 11. Auxiliary function MLEND-RECURSION

The maximally extended matchings are finally calculated from the $M(r, s)$ matrix by an usual traceback. The space complexity of the algorithm is $O(nm)$. The time complexity is $O(nm)$ for the following reason. Every pair (r, s) with $1 \leq r \leq |S_1|$ and $1 \leq s \leq |S_2|$ is considered at most twice in START-LOOP-WALKING and LOOP-WALKING, with an $O(1)$ complexity for calculating the corresponding matrix entries. Similarly, every pair (r, s) is considered at most twice in LOOP-WALKING. Since there are $O(nm)$ many pairs (r, s), we get a total complexity of $O(nm)$.

7 Conclusion

We have presented a fast dynamic programming approach in time $O(nm)$ and space $O(nm)$ for detecting common sequence/structure patterns between two RNAs given by their sequence and secondary structures. These patterns

are derived from exact matchings and can be used for local alignments ([2]). The most promising advantage is clearly to investigate large RNAs of several thousand bases in reasonable time. Here, one can think of detecting local sequence/structure regions of several RNAs sharing the same biological function.

References

[1] Victor Ambros. The functions of animal microRNAs. *Nature*, 431(7006):350–5, 2004.

[2] Rolf Backofen and Sebastian Will. Local sequence-structure motifs in RNA. *Journal of Bioinformatics and Computational Biology (JBCB)*, 2004. accepted for publication.

[3] V. Bafna, S. Muthukrishnan, and R. Ravi. Computing similarity between rna strings. In *Proc. 6th Symp. Combinatorical Pattern Matching*, pages –16, 1995.

[4] Vineet Bafna and Shaojie Zhang. Fastr: Fast database search tool for non-coding rna. pages 52–61. IEEE Computer Society, 2004.

[5] A. Bateman, E. Birney, L. Cerruti, R. Durbin, L. Etwiller, S. R. Eddy, S. Griffiths-Jones, K. L. Howe, M. Marshall, and E. L. Sonnhammer. The Pfam protein families database. *Nucleic Acids Res*, 30(1):276–80, 2002.

[6] Jennifer Couzin. Breakthrough of the year. Small RNAs make big splash. *Science*, 298(5602):2296–7, 2002.

[7] David Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms & Applications*, 3(3):1–27, 1999.

[8] P. Gendron, D. Gautheret, and F. Major. Structural ribonucleic acid motifs identification and classification. In *High Performance Computing Systems and Applications*. Kluwer Academic Press, 1998.

[9] Gramm, Guo, and Niedermeier. Pattern matching for arc-annotated sequences. *FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, 22, 2002.

[10] I. L. Hofacker, S. H. Bernhart, and P. F. Stadler. Alignment of RNA base pairing probability matrices. *Bioinformatics*, 2004.

[11] I. L. Hofacker, B. Priwitzer, and P. F. Stadler. Prediction of locally stable RNA secondary structures for genome-wide surveys. *Bioinformatics*, 20(2):186–190, 2004.

[12] Ivo L. Hofacker, Walter Fontana, Peter F. Stadler, Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte Chemie*, 125:167–188, 1994.

[13] Matthias Höchsmann, Thomas Töller, Robert Giegerich, and Stefan Kurtz. Local similarity in rna secondary structures. In *Proceedings of Computational Systems Bioinformatics (CSB 2003)*, 2003.

[14] Tao Jiang, Guo-Hui Lin, Bin Ma, and Kaizhong Zhang. The longest common subsequence problem for arc-annotated sequences. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (CPM2000)*, 2000.

[15] Tao Jiang, Guohui Lin, Bin Ma, and Kaizhong Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2):371–88, 2002.

[16] Rune B. Lyngso and Christian N. S. Pedersen. 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.

[17] B. Morgenstern, K. Frech, A. Dress, and T. Werner. DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics*, 14(3):290–4, 1998.

[18] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, January 1976.

[19] Juris Viksna and David Gilbert. Pattern matching and pattern discovery algorithms for protein topologies. In O. Gascuel and B. M. E. Moret, editors, *Proceedings of the First International Workshop on Algorithms in Bioinformatics (WABI 2001)*, number 2149 in LNCS, pages 98–111, 2001.

[20] Jason Tsong-Li Wang, Bruce A. Shapiro, Dennis Shasha, Kaizhong Zhang, and Kathleen M. Currey. An algorithm for finding the largest approximately common substructures of two trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):889–895, 1998.

Acknowledgment

We thank the anonymous reviewers for their comments. They helped us to improve the quality of the paper.

A Listing of Sub-Procedures

A.1 Sub-Procedures for Loop-Walking

```
1: procedure Loop-Reach(r, s, i, j, i', j', reach, RDist)
2:     if reach then
3:         M(i', j') := M(r, s);    M(r, s) := 0
4:         M^{r_end}(RDist) := M^{loop}_{≼}(r, s)
5:         M^{rb}(i', j') := M^{r_end}(RDist)
6:     else
7:         M^{r_end}(RDist) := M^{r_end}(RDist − 1)
8:     end if
9: end procedure
```

```
1: procedure Init-Loop-Matrices(k, l, i', j')
2:     right_ends := (k = i' ∧ l = j')
3:     if right_ends ∧ base-match(i', j') then
4:         M^{loop}_{≼}(k, l) := 1;    M^{rb}(k, l) := 1;    M(k, l) = 1;
5:         M^{r_end}(0) := 1;    reach := true
6:     else if right_ends then
7:         M^{loop}_{≼}(k, l) := 0;    M^{rb}(k, l) := 0;    M(k, l) = 0;
8:         M^{r_end}(0) := 0;    reach := false
9:     else
10:        M^{loop}_{≼}(k, l) := 0;    M(k, l) = 0;    reach := false
11:    end if
12:    return (reach)
13: end procedure
```

```
1: procedure Calc-Remain-Loop-Len(r, s, i, j, LDist)
2:     Loop_Len_Dist := 0
3:     if r = i then
4:         while s > j do
5:             s = left_{R_2}(s)
6:             Loop_Len_Dist := Loop_Len_Dist + 1
7:         end while
8:         return (LDist − 1, Loop_Len_Dist)
9:     else
```

10: **while** $r > i$ **do**
11: $r = \text{left}_{R_1}(r)$
12: $Loop\_Len\_Dist := Loop\_Len\_Dist + 1$
13: **end while**
14: **return** $(LDist + Loop\_Len\_Dist - 1, -Loop\_Len\_Dist)$
15: **end if**
16: **end procedure**

A.2 Sub-Procedures for LOOP-MATCHING

1: **procedure** FILL-MBB($i, i', j, j', M^{l\_end}, LDist, i\_i'\_len, lens\_dist$)
2: **if** $lens\_dist \geq 0$ **then**
3: $RDist = i\_i'\_len - LDist$ ▷ Bond i—i' has smaller loop
4: **else**
5: $RDist = i\_i'\_len + lens\_dist - LDist$ ▷ Bond j—j' has smaller loop
6: **end if**
7: $M^{bb}(i,j) := \max \left\{ \begin{array}{l} M^{bb}(i,j) \\ M^{l\_end} + M^{r\_end}(RDist) \end{array} \right\}$
8: $M(i,j) = M^{bb}(i,j)$
9: $M(i',j') = 0$
10: **end procedure**

B Examples

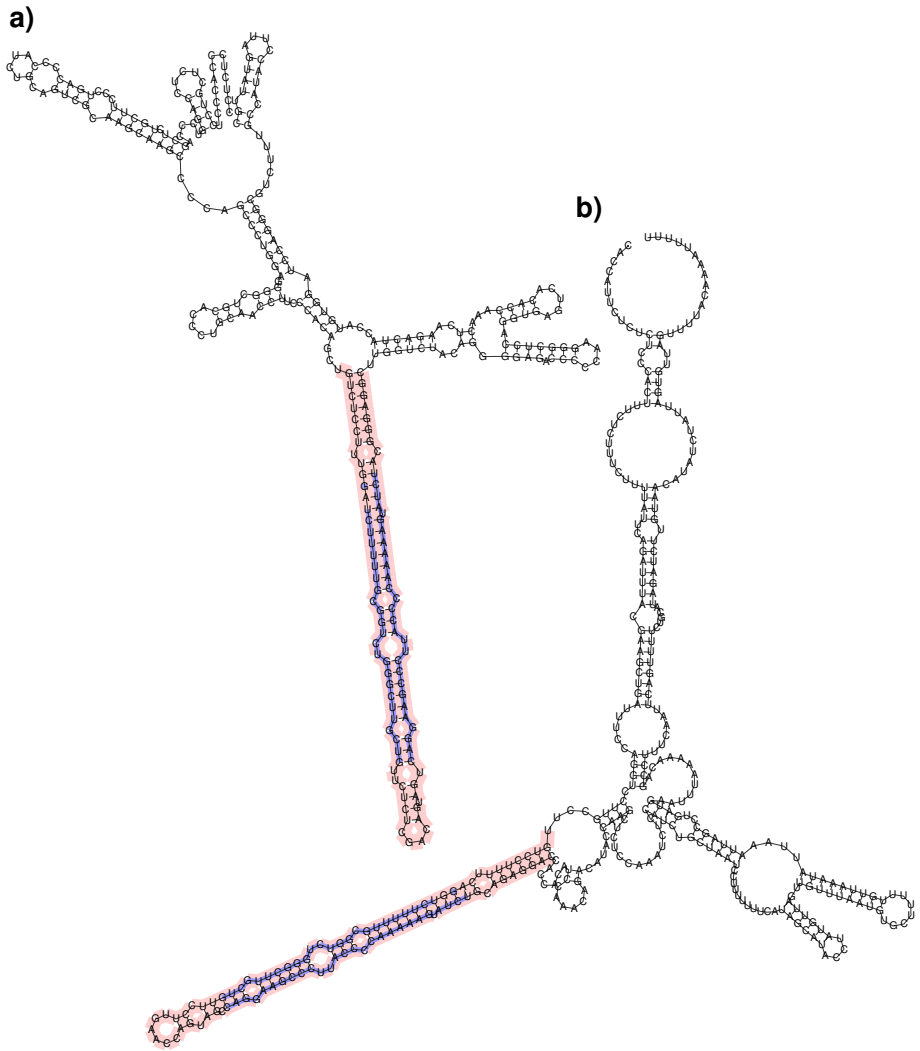

Figure B.1. Two microRNAs from the mir-129 precursor family (accession numbers: BX649263.6,AC072048.4, Rfam database [5]). The two RNAs are folded into their optimal structures using RNAfold [12]. The largest exact pattern common to both RNAs is highlighted in blue. The largest approximate pattern is highlighted in pink. The last pattern contains in addition to the exact pattern bonded pairs of nucleotides not satisfying the bond condition in definition 3.2.

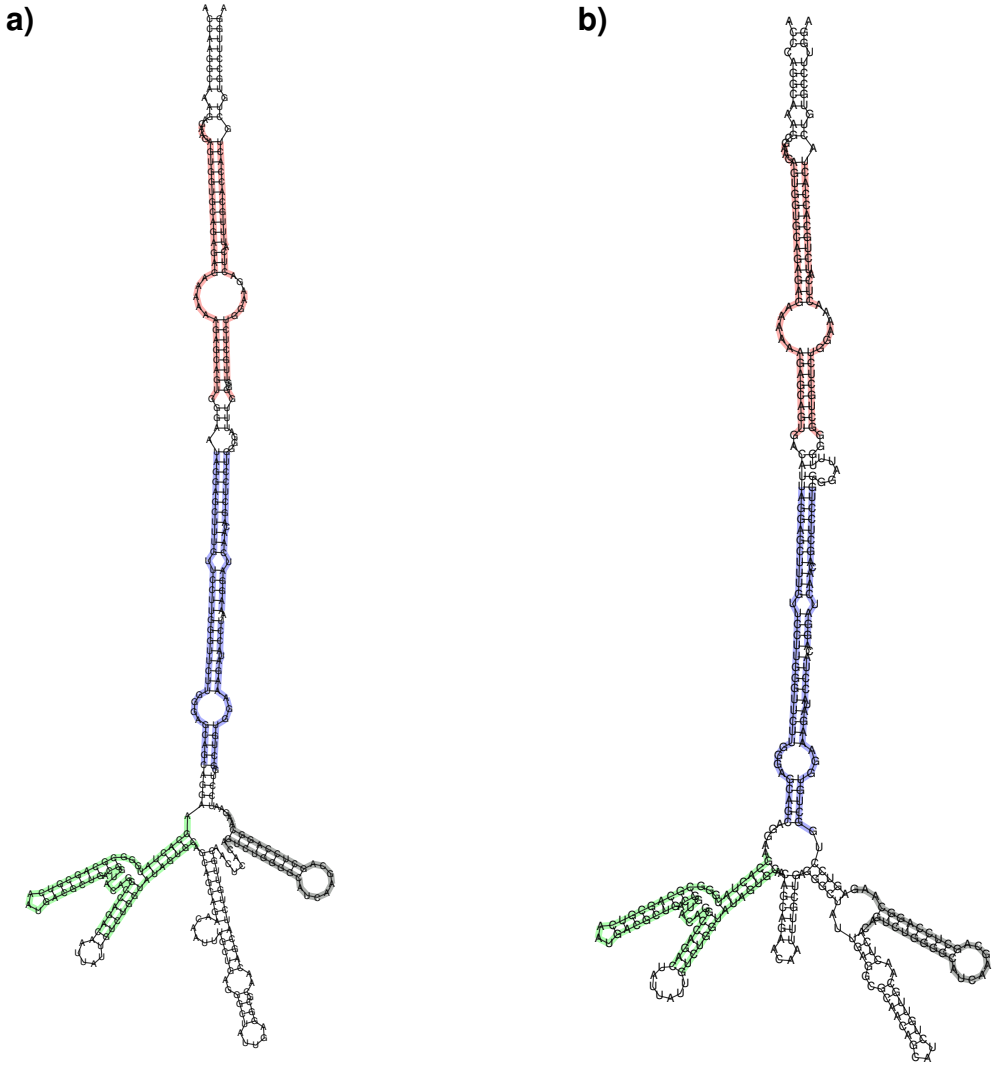

Figure B.2. Two Rev Response elements (RRE), (accession numbers M14100, U36876, Rfam database [5]) encoded within the HIV-env gene. Our program was performed on their optimal structures. The four largest, exact patterns are highlighted. A clearly visible one-to-one correspondence between the patterns shown as different colors detects a high similarity among those RNAs. Notice that the running time and the space complexity are only $O(nm)$.