Local Alignment of RNA Sequences with Arbitrary Scoring Schemes

Rolf Backofen[1], Danny Hermelin[⋆2],
Gad M. Landau[⋆⋆2,3], and Oren Weimann[4]

[1] Institute of Computer Science,
Albert-Ludwigs Universität Freiburg, Freiburg - Germany.
`backofen@informatik.uni-freiburg.de`
[2] Department of Computer Science,
University of Haifa, Haifa - Israel.
`danny@cri.haifa.ac.il, landau@cs.haifa.ac.il`
[3] Department of Computer and Information Science,
Polytechnic University, New York - USA.
[4] Computer Science and Artificial Intelligence Laboratory,
MIT, Cambridge MA - USA.
`oweimann@mit.edu`

Abstract. Local similarity is an important tool in comparative analysis of biological sequences, and is therefore well studied. In particular, the Smith-Waterman technique and its normalized version are two established metrics for measuring local similarity in strings. In RNA sequences however, where one must consider not only sequential but also structural features of the inspected molecules, the concept of local similarity becomes more complicated. First, even in global similarity, computing global sequence-structure alignments is more difficult than computing standard sequence alignments due to the bi-dimensionality of information. Second, one can view locality in two different ways, in the sequential or structural sense, leading to different problem formulations.

In this paper we introduce two sequentially-local similarity metrics for comparing RNA sequences. These metrics combine the global RNA alignment metric of Shasha and Zhang [16] with the Smith-Waterman metric [17] and its normalized version [2] used in strings. We generalize the familiar alignment graph used in string comparison to apply also for RNA sequences, and then utilize this generalization to devise two algorithms for computing local similarity according to our two suggested metrics. Our algorithms run in $\mathcal{O}(m^2 n \lg n)$ and $\mathcal{O}(m^2 n \lg n + n^2 m)$ time respectively, where $m \leq n$ are the lengths of the two given RNAs. Both algorithms can work with any arbitrary scoring scheme.

1 Introduction

Ribonucleic acids (RNAs) are polymers consisting of the four nucleotides Adenine, Cytosine, Guanine, and Uracil, which are linked together by their phosphodiester bonds. Bases which are part of the nucleotides form hydrogen bonds

[⋆] Partially supported by the Israel Science Foundation grant 282/01.

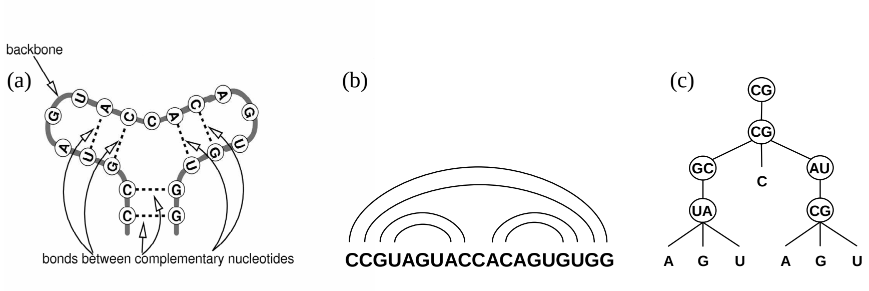

Fig. 1. Three different ways of viewing an RNA sequence. In (a), a schematic 2-dimensional description of an RNA folding. In (b), a linear representation of the RNA. In (c), the RNA as a rooted ordered tree.

within the same molecule leading to structure formation. These hydrogen bonds are referred to as base pairs, and the set of all base pairs is called the secondary structure of the RNA. The role of RNA in biological systems was largely underestimated for a long time. Today, RNA enjoys increasing attention due to recent developments such as the discovery of ribozymes (RNA-molecules with enzymatic properties), and the observation that non-coding RNA molecules play an enormous role in cell control. As an example, research on non-coding RNAs has been elected as the scientific breakthrough of 2002 by the readers of Science [6].

One major challenge of research on RNAs is to find common patterns since these suggest functional similarities in the inspected 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 is determined by its secondary structure [15]. Second, it is known that the structure of an RNA is often more conserved than its sequence during evolution [4]. Thus, two RNA sequences with their corresponding secondary structure are aligned using both sequential and structural information for scoring the alignment.

There have been quite a few approaches for defining alignments in terms of RNAs. The first one is due to the seminal paper of Shasha and Zhang [16] which represented RNA sequences as rooted ordered trees, and defined editing operations on trees which correspond to editing operations on RNA sequences. In this way, an alignment of two RNA sequences corresponds to a sequence of editing operations on two corresponding trees, and any tree editing algorithm can be used to compute the optimal alignment of two RNAs. Furthermore, this approach allows base pairs to be considered as whole entities, meaning that one can require any base pair to either be deleted (resulting in a removal of two nucleotides) or be aligned against another base pair in the opposite RNA. Since [16], there have been attempts at extending either the set of edit operations on trees [1, 11], or the set of allowed RNA alignments [13], in order to model certain biological mutations that weaken and ultimately break bonds between base pairs. Usually, these extensions introduce an increase in the time complexities of the algorithms required to compute them.

In RNA sequences, as in many other biological applications, searching for local similarities is at least as important as determining global similarity. In contrast, most RNA sequence-structure alignment methods are global. To our knowledge, there are only a few exceptions for this, namely [3, 5, 7, 10, 18]. These can be divided roughly into two main categories, depending on the exact notion of locality under consideration. The first category of [3, 7, 18] defines locality in the structural sense, thus allowing large gaps in the sequences not to be considered as relevant in the alignment score. The second category of [5, 10] defines locality in the sequential sense, thus extending the well understood notion of locality in strings to RNA sequences.

In this paper we introduce two sequentially-local metrics for RNA local alignment. The first one is a natural extension of the Smith-Waterman metric used in strings [17]. The second one is a a normalized variant of the first metric, where one divides the alignment score of two local regions by the sum of their lengths. This metric was suggested for string comparison by [2], and was dealt also in [9].

Our results: We give two algorithms for computing the optimal local alignment score of two RNA sequences of lengths m and n, $m \leq n$. The first algorithm computes in $\mathcal{O}(m^2 n \lg n)$ time the optimal local alignment score according to our extension of the Smith-Waterman metric. The second one computes the optimal normalized local alignment score in $\mathcal{O}(m^2 n \lg n + n^2 m)$ time. Both algorithms work with any arbitrary scoring scheme.

Roadmap: The rest of this paper is organized as follows. We next introduce notations and terminology that will be used throughout the paper. Following this, in Section 2, we discuss the notion of alignment for RNA sequences. In Section 3, we discuss local alignment and introduce two new local similarity metrics that we will be dealing with throughout the paper. Section 4 then describes an adaptation of the familiar alignment graph used in string comparison to an alignment graph for RNA sequences. This adapted graph is then used in Section 5 to design two algorithms that compute the local alignment score between a pair RNA sequences according to our two suggested metrics. Due to space limitations, all proofs are omitted from this version of the paper.

Notations: An RNA sequence $\mathcal{R}$ is an ordered pair (S, A), where $S = s_1 \cdots s_{|S|}$ is a string over the alphabet $\Sigma = \{A, C, G, U\}$, and $A \subseteq \{1, \ldots, |S|\} \times \{1, \ldots, |S|\}$ is the set of hydrogen bonds between bases of $\mathcal{R}$ (*i.e.* the secondary structure). Any base in $\mathcal{R}$ can bond with at most one other base, therefore we have $\forall\ (i'_1, i_1), (i'_2, i_2) \in A$, $i'_1 = i'_2 \Leftrightarrow i_1 = i_2$. Furthermore, following Zuker [19, 20], we assume a model where the bonds in A are *non crossing*, *i.e.* for any $(i'_1, i_1), (i'_2, i_2) \in A$, we cannot have $i'_1 < i'_2 < i_1 < i_2$ nor $i'_2 < i'_1 < i_2 < i_1$. We refer to a bond $(i', i) \in A$, $i' < i$, as an *arc*, and i' and i are referred to as the *left* and *right endpoints* of this arc. Also, we let $|\mathcal{R}|$ denote the number of nucleotides in $\mathcal{R}$, *i.e.* $|\mathcal{R}| = |S|$.

We will require a notion similar to that of a substring for RNA sequences. Therefore, for any $1 \leq i' \leq i \leq |\mathcal{R}|$, we let $\mathcal{R}[i', i] = (S[i', i], A[i', i])$, the *consecutive subsequence* of $\mathcal{R}$, be the RNA with $S[i', i] = S_{i'} \cdots S_i$ and $A[i', i] = A \cap \{i', \ldots, i\} \times \{i', \ldots, i\}$. If $(i', i) \in A$, then we say that arc (i', i) *wraps* $\mathcal{R}[i', i]$. Also, for convenience purposes, we slightly abuse notation and let $\mathcal{R}[i+1, i] = (\emptyset, \emptyset)$ denote the empty RNA for any $1 \leq i \leq |\mathcal{R}|$. Note that arcs of $\mathcal{R}$ with one endpoint in $\mathcal{R}[i', i]$ are absent in $\mathcal{R}[i', i]$. These arcs are said to be *broken* in $\mathcal{R}[i', i]$. A position $l \in \{i', ..., i\}$ is considered an arc endpoint in $\mathcal{R}[i', i]$, even if it is an arc endpoint of an arc which is broken in $\mathcal{R}[i', i]$.

This paper deals with comparing two RNA sequences. We denote these two RNAs by $\mathcal{R}_1 = (S_1, A_1)$ and $\mathcal{R}_2 = (S_2, A_2)$ throughout the paper, and we set $|\mathcal{R}_1| = |S_1| = n$ and $|\mathcal{R}_2| = |S_2| = m$. Furthermore, we assume $m \leq n$.

2 RNA Alignment

As in the case of strings, RNA alignment is analogous to the edit distance of two RNAs, *i.e.* the minimum number of edit operations necessary in order to transform one RNA into the other [16]. The edit operations defined for RNA molecules are similar to those defined for strings, except that here we can perform editing operations on arcs as well as on unpaired nucleotides. The allowed edit operations are therefore insertion, deletion, and relabeling of arcs and nucleotides on either one of the given RNAs. Defining separate operations on arcs and unpaired nucleotides captures the notion of arcs and unpaired bases being different entities.

An *alignment* of $\mathcal{R}_1$ and $\mathcal{R}_2$ is another way of viewing a sequence of edit operations on these two RNAs. Formally, it is defined as follows:

Definition 1 (Alignment). *An* alignment $\mathcal{A}$ *of* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *is a subset of* $\{1, \ldots, n\} \cup \{-\} \times \{1, \ldots, m\} \cup \{-\}$ *satisfying the following conditions:*

- $(-, -) \notin \mathcal{A}$.
- $\forall (i, j) \in \mathcal{A} \cap \{1, \ldots, n\} \times \{1, \ldots, m\}$: *i and j appear exactly once in $\mathcal{A}$.*
- $\forall (i', j'), (i, j) \in \mathcal{A} \cap \{1, \ldots, n\} \times \{1, \ldots, m\} : i' < i \iff j' < j$. *That is, any two pairs in $\mathcal{A}$ are non-crossing.*
- $\forall (i, j) \in \mathcal{A} \cap \{1, \ldots, n\} \times \{1, \ldots, m\}$: *i is a left (resp. right) arc endpoint in $\mathcal{R}_1 \iff j$ is a left (resp. right) arc endpoint in $\mathcal{R}_2$.*
- $\forall (i', i) \in A_1, (j', j) \in A_2 : (i', j') \in \mathcal{A} \iff (i, j) \in \mathcal{A}$. *That is, the left endpoints of any pair of arcs are aligned against each other in $\mathcal{A}$ iff their right endpoints are also aligned against each other in $\mathcal{A}$.*

In terms of editing operations, a pair $(i, j) \in \mathcal{A} \cap \{1, \ldots, n\} \times \{1, \ldots, m\}$ corresponds to relabeling the ith nucleotide (unpaired or not) of $\mathcal{R}_1$ so it would match the jth nucleotide of $\mathcal{R}_2$, while pairs $(i, -)$ and $(-, j)$ corresponds to deleting the ith and jth nucleotides in $\mathcal{R}_1$ and $\mathcal{R}_2$. The first three conditions in the above definition require any position in $\mathcal{R}_1$ and $\mathcal{R}_2$ to be aligned, and that $(-,-) \notin \mathcal{A}$, since $(-,-)$ does not correspond to any valid edit operation. The next condition enforces the order of the subsequences to be preserved in $\mathcal{A}$, and the last two conditions restrict any arc to be either deleted or aligned against another arc in the opposite RNA.

Let $\Sigma' = \Sigma \cup \{-\}$. A *scoring scheme* $\delta = (\delta_1, \delta_2)$ for alignments of $\mathcal{R}_1$ and $\mathcal{R}_2$ is an ordered pair of two separate scoring functions $\delta_1 : \Sigma' \times \Sigma' \longrightarrow \mathbb{Z}$ and $\delta_2 : \Sigma'^2 \times \Sigma'^2 \longrightarrow \mathbb{Z}$, one which measures the quality of aligning a single unpaired nucleotide of $\mathcal{R}_1$ against another unpaired nucleotide of $\mathcal{R}_2$, and the other for measuring the quality of aligning pairs of arcs of the two RNA sequences. Hence $\delta_2((S_1[i'], S_1[i]), (-,-))$ denotes, for example, the deletion of the arc $(i', i) \in A_1$. We assume the scoring scheme is a similarity metric, and so a high score is given for aligning similar arcs or similar unpaired nucleotides, while different penalties are given in all other possible cases.

For brevity of notation, let us write $\delta_1(i,j)$ to denote the value $\delta_1(S_1[i], S_2[j])$ if $i, j \neq -$, and $\delta_1(S_1[i], -)$ (resp. $\delta_1(-, S_2[j])$) if j (resp. i) is the blank symbol '$-$'. Also, we write $\delta_2(i', i, j', j)$ instead of $\delta_2((i', i), (j', j))$. The *score* of an alignment $\mathcal{A}$ of $\mathcal{R}_1$ and $\mathcal{R}_2$ with respect to δ is given by:

$$\delta(\mathcal{A}) \;=\; \sum_{\substack{(i,j)\in\mathcal{A},\ i,j \text{ are not} \\ \text{arc endpoints}}} \delta_1(i,j) \qquad + \qquad \sum_{\substack{(i',j'),(i,j)\in\mathcal{A}, \\ (i',i)\in A_1 \wedge (j',j)\in A_2}} \delta_2(i', i, j', j).$$

Definition 2 ($\boldsymbol{OPT_\delta(\mathcal{R}'_1, \mathcal{R}'_2)}$)**.** *Given two RNA sequences $\mathcal{R}'_1$ and $\mathcal{R}'_2$ and a scoring scheme $\delta = (\delta_1, \delta_2)$, $OPT_\delta(\mathcal{R}'_1, \mathcal{R}'_2)$ denotes the highest score of any alignment of $\mathcal{R}'_1$ and $\mathcal{R}'_2$ with respect to δ.*

2.1 RNA alignment via ordered tree editing

The non crossing formation formed by the arcs in both $\mathcal{R}_1$ and $\mathcal{R}_2$ conveniently allows representing these RNAs as rooted ordered trees [16]. Each arc (i, i') is identified with a set of ordered children which are all unpaired bases i'' such that $i < i'' < i'$, and arcs (l, l') such that $i < l < l' < i'$ (see Figure 1). In [16], Shasha and Zhang suggested an algorithm for computing the edit distance between two ordered trees. Their algorithm was later improved by Klein [14] to an $\mathcal{O}(m^2 n \lg n)$ algorithm, where $m \leq n$ denote the number of nodes in the two trees. (Recently, Demaine *et al.* [8] presented an $\mathcal{O}(m^2 n(1+\lg \frac{n}{m}))$ improvement to this algorithm. Using their algorithm improves the results of this paper to $\mathcal{O}(m^2 n(1 + \lg \frac{n}{m}))$ time for the Smith-Waterman metric, and $\mathcal{O}(n^2 m)$ for the normalized metric).

Not by chance, the edit operations defined for trees are analogous to the ones defined for RNA sequences. For this reason, any tree editing algorithm can be used to determine the global alignment score of two RNAs, with the slight modification that a penalty of ∞ is assigned for relabeling a node which corresponds to an unpaired nucleotide by a label corresponding to a base pair, and vise versa. Furthermore, as a side effect of the recursions used in [14, 16], these algorithms compute the optimal alignment between every pair of rooted subtrees of the two given trees, assuming this alignment matches the roots of the two subtrees. In our setting, this means that $\delta_2(i', i, j', j) + OPT_\delta(\mathcal{R}_1[i' + 1, i-1], \mathcal{R}_2[j+1, j-1])$ is computed between all pairs of arcs $(i', i) \in A_1$ and $(j', j) \in A_2$ in a single execution of either algorithms. The importance of this property will become apparent later on.

Definition 3 ($OPT_\delta^{arc}(\mathcal{R}_1[i', i], \mathcal{R}_2[j', j])$). *For a pair of arcs $(i', i) \in A_1$ and $(j', j) \in A_2$, we set $OPT_\delta^{arc}(\mathcal{R}_1[i', i], \mathcal{R}_2[j', j]) = \delta_2(i', i, j', j) + OPT_\delta(\mathcal{R}_1[i' + 1, i-1], \mathcal{R}_2[j+1, j-1])$.*

3 Local Alignment

While the metric described in the previous section is suitable for measuring global similarity of two RNA sequences, in many applications two RNAs may not be very similar when both considered as a whole, but may contain many regions of high similarity. The goal is then to extract a pair of regions, one from each RNA, which admits a strong degree of similarity. This is known as *local similarity.* In the following section we introduce two metrics for measuring local similarity between RNA sequences. These metrics are extensions of the Smith-Waterman [17] and normalization [2] techniques used for strings.

A *local alignment* of $\mathcal{R}_1$ and $\mathcal{R}_2$, one which corresponds to a pair of contiguous regions in the RNAs, is an alignment of two consecutive subsequences $\mathcal{R}_1[i', i]$ and $\mathcal{R}_2[j', j]$. Note that the last condition of Definition 1 implies that any arc endpoint of a broken arc in each subsequence must be aligned with the blank symbol '$-$'. However, we need to distinguish between this situation and a deletion of an unpaired nucleotide. Therefore, we use $\delta_2(l', -, -, -)$ (resp. $\delta_2(-, l, -, -)$) to denote the cost of aligning the left (resp. right) endpoint of a broken edge (l', l) in $\mathcal{R}_1[i', i]$ against '$-$', and symmetrically, $\delta_2(-, -, l', -)$ and $\delta_2(-, -, -, l)$ are used to denote the costs of aligning the endpoints of a broken edge (l', l) in $\mathcal{R}_1[i', i]$ against '$-$'. Furthermore, we require that the total cost of aligning the left and right endpoints of a broken edge (in two different alignments) be equal to the cost of deleting this edge. That is, $\delta_2(l', -, -, -) + \delta_2(-, l, -, -) = \delta_2(l', l, -, -)$ and $\delta_2(-, -, l', -) + \delta_2(-, -, -, l) = \delta_2(-, -, l', l)$.

3.1 Standard local alignment

The well known Smith-Waterman [17] technique for computing local similarity between strings has been extensively studied in the literature. It is defined as the highest scoring alignment between any pair of substrings of the input strings. The simplicity of this definition has gained it wide applicability in many biological settings [12]. In our terms, it is defined by:

$$\max \left\{ OPT_\delta(\mathcal{R}_1[i', i], \mathcal{R}_2[j', j]) \;\middle|\; \begin{array}{l} 1 \leq i' \leq i \leq |\mathcal{R}_1|, \\ 1 \leq j' \leq j \leq |\mathcal{R}_2| \end{array} \right\}.$$

We refer to the Smith-Waterman metric as the *standard local alignment score* of $\mathcal{R}_1$ and $\mathcal{R}_2$. The computational problem corresponding to this metric is then defined as follows:

Definition 4 (The standard local alignment problem). *Given two RNA sequences $\mathcal{R}_1 = (S_1, A_1)$ and $\mathcal{R}_2 = (S_2, A_2)$, determine the standard local alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$.*

3.2 Normalized local alignment

According to [2], the Smith-Waterman technique has two weaknesses that make it non optimal as a local similarity measure. The first weakness is called the *mosaic effect*. This term describes the algorithm's inability to discard poorly conserved intermediate segments, although it can discard poor prefixes or suffixes of a segment. The second weakness is known as the *shadow effect*. This term describes the tendency of the algorithm to lengthen long alignments with a high score rather than shorter alignments with a lower score and a higher degree of similarity.

One way to overcome these weaknesses is to normalize the alignment score of two substrings by dividing it with their total length [2]. In our terms, the *normalized alignment score* of $\mathcal{R}_1$ and $\mathcal{R}_2$ is defined by:

$$\max\left\{ \frac{OPT_\delta(\mathcal{R}_1[i,i'], \mathcal{R}_2[j,j'])}{|\mathcal{R}_1[i,i']| + |\mathcal{R}_2[j,j']|} \;\middle|\; \begin{array}{l} 1 \leq i \leq i' \leq |\mathcal{R}_1|, \\ 1 \leq j \leq j' \leq |\mathcal{R}_2|, \\ OPT_\delta(\mathcal{R}_1[i,i'], \mathcal{R}_2[j,j']) \geq I \end{array} \right\}.$$

Where $I \in \mathbb{N}$ is an integer regulating the minimum score (before normalization) of solution alignments, predefined according to the application at hand. Note that this additional parameter is necessary for preventing trivial solutions (*e.g.* a single match) from being optimal.

Definition 5 (The normalized local alignment problem). *Given two RNA sequences $\mathcal{R}_1 = (S_1, A_1)$ and $\mathcal{R}_2 = (S_2, A_2)$, and an integer I, determine the normalized local alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$.*

4 An Alignment Graph for RNA Sequences

We next present an adaptation of the alignment graph that is used to describe string alignment [12], to an alignment graph that describes alignments of RNA sequences. Later, in Section 5, this adapted graph will be utilized for computing the local similarity score of $\mathcal{R}_1$ and $\mathcal{R}_2$ according to our two suggested metrics. We begin with a brief description of the alignment graph used for strings, and then proceed to explain in further detail the modifications necessary for our case.

Let S_1 and S_2 be two strings over any given alphabet, and δ_1 be a given scoring function over this alphabet. The *alignment graph* for S_1 and S_2 is a weighted directed graph with $(|S_1|+1)(|S_2|+1)$ vertices, each indexed by a distinct pair $(i, j) \in \{0, \ldots, |S_1|\} \times \{0, \ldots, |S_2|\}$. For each vertex (i, j), the alignment graph contains a directed edge from (i, j) to each of the vertices $(i, j+1)$, $(i+1, j)$, and $(i+1, j+1)$, provided these vertices exist. These edges are called the *horizontal*, *vertical*, and *diagonal* edges of (i, j) respectively, and their weights are given

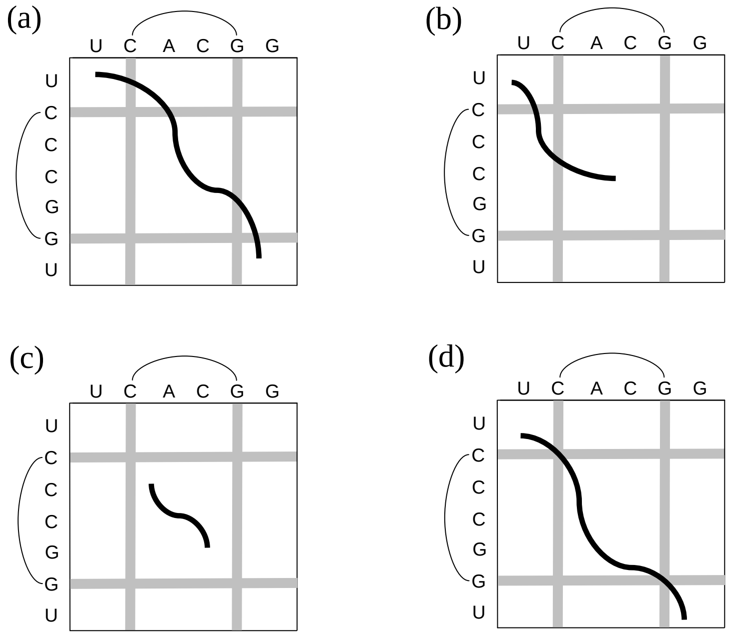

Fig. 2. Four different situations that occur when aligning RNA sequences.

by $\delta_1(-,j)$, $\delta_1(i,-)$, and $\delta_1(i,j)$. In this way, the alignment graph captures the standard dynamic programming used in string alignment.

The central property of the alignment graph of S_1 and S_2 is that any path from say (i',j') to (i,j) corresponds to an alignment between $S_1[i'+1,i]$ and $S_2[j'+1,j]$ with a score which equals the total sum of weights of the edges in the path. Conversely, any alignment between $S_1[i'+1,i]$ and $S_2[j'+1,j]$ with score w has a corresponding w-weighted path in the alignment graph.

Theorem 1 ([12]). *An alignment of $S_1[i',i]$ and $S_2[j',j]$ has optimal score iff it corresponds to the heaviest path from (i',i) to (j',j).*

Let us now consider alignments of RNA sequences. The main difference when aligning RNA sequences is that now we must consider arcs and unpaired nucleotides as different entities which must be aligned separately. There are four different cases that we should each address accordingly:

- *Case 1* corresponds to arc deletions and is depicted in Figure 2(a). Note that the path in the figure deletes the left and right endpoints in both arcs of the RNAs, and therefore it corresponds to deleting the two arcs. Note that this would also be the case even if the path had passed through the diagonal edge that corresponds to aligning the right endpoints of the arcs.
- *Case 2* corresponds to alignments which break arcs and is depicted in Figure 2(b). Such alignments must be penalized accordingly.

- *Case 3* corresponds to alignments in areas which do not contain arcs. In contrast to the previous case, these types of alignments do not break any arcs, and therefore they should not be penalized.
- *Case 4* corresponds to alignments which align arcs against each other.

Note that there are also alignments which combine the first two cases, by deleting one arc and breaking the other.

We now turn to describe the necessary modifications for adapting the alignment graph to RNA sequence-structure alignment. We begin by focusing on the first three cases in the example above. The last case will be dealt with separately. For a given $i \in \{1, \ldots, n\}$, we refer to the set of all edges connecting a pair of vertices in $\{(i-1, j), (i, j) \,|\, 0 \leq j \leq m\}$ as the *ith row* of the alignment graph. Hence, the ith row corresponds to all edges that represent an edit operation which involves the ith position of $\mathcal{R}_1$. The *jth column*, $j \in \{1, \ldots, n\}$, is defined symmetrically to be the set of all edges connecting pairs of vertices in $\{(i, j-1), (i, j) \,|\, 0 \leq j \leq m\}$.

Consider some position i in $\mathcal{R}_1$, along with the ith row corresponding to this position in the alignment graph. If i is not an arc endpoint, then no modifications are necessary on this row, since all editing operations on i are equivalent to those in strings. Otherwise, when i is an endpoint, we wish to model the three cases discussed above. For this, we first remove all diagonal edges. This is done to ensure that i is not aligned against any position in $\mathcal{R}_2$, as we are only concerned with arc deletions at the moment. Following this, we set the weights of all vertical edges to the penalty of breaking the arc of which i is an endpoint of. If i a left endpoint, we set these weights to $\delta_2(i, -, -, -)$, and otherwise we set them to $\delta_2(-, i, -, -)$. This takes care of the second case described above. All other edges in the row, *i.e.* the horizontal edges, are left untouched. For a position j in $\mathcal{R}_2$ the modifications are symmetric. Here the weights of the horizontal edges are modified, while the vertical edges remain unmodified. We mention that all our modifications could be done on the scoring scheme (by adding additional letters to the alphabet which represent different types of arc endpoints) rather then on the alignment graph.

After applying the above modifications to each column and row which corresponds to arc endpoints, we obtain the *grid part* of our adapted alignment graph.

Lemma 1. *Paths in the grid part are in bijective correspondence with alignments of consecutive subsequences of $\mathcal{R}_1$ and $\mathcal{R}_2$ in which all arcs are deleted.*

What is left now, is to take care of alignments which align arcs against each other, *i.e.* the last case in the example above. Consider a path, as in Figure 2(d), that corresponds to an alignment which aligns $(i', i) \in A_1$ against $(j', j) \in A_2$. This path must pass through the nodes $(i'-1, j'-1)$ and (i, j) in the alignment graph (the two intersections of the shaded rows and columns). This means that this path consists of a prefix which ends at $(i'-1, j'-1)$, a middle part from $(i'-1, j'-1)$ to (i, j), and a suffix which starts at (i, j). As was explained in Section 2.1, the optimal score of the middle part is given by $OPT_{\delta}^{arc}(\mathcal{R}_1[i',i],\mathcal{R}_2[j',j])$, and it is computed in the preprocessing step. Therefore, if this path is optimal, its weight equals $OPT_{\delta}^{arc}(\mathcal{R}_1[i',i],\mathcal{R}_2[j',j])$ plus the combined weights of the suffix and prefix. We represent the middle part of any optimal path that aligns (i',i) against (j',j) by adding a single edge from $(i'-1,j'-1)$ to (i,j) in the alignment graph, and setting its weight to $OPT_{\delta}^{arc}(\mathcal{R}_1[i',i],\mathcal{R}_2[j',j])$ accordingly. We refer to this new edge as a *shortcut* edge, and we add such shortcut edges for each pair of arcs in $\mathcal{R}_1$ and $\mathcal{R}_2$.

Theorem 2. *An alignment of $\mathcal{R}_1[i'+1,i]$ and $\mathcal{R}_2[j'+1,j]$ is optimal iff it corresponds to the heaviest path from (i',i) to (j',j) in the alignment graph of $\mathcal{R}_1$ and $\mathcal{R}_2$.*

5 Local Alignment Algorithms

We next describe two algorithms for computing local similarity of $\mathcal{R}_1$ and $\mathcal{R}_2$ according to the two metrics defined in Section 3. For simplicity, we focus only on computing the score of an optimal alignment rather than computing an actual alignment. One can easily obtain the latter within the same time and space bounds in both algorithms.

As a consequence of Theorem 2, computing optimal local alignments of $\mathcal{R}_1$ and $\mathcal{R}_2$ reduces to computing locally optimal paths in the alignment graph of $\mathcal{R}_1$ and $\mathcal{R}_2$. Therefore, both our algorithms initially construct the alignment graph of $\mathcal{R}_1$ and $\mathcal{R}_2$, and then perform all computations on this graph. For any edge in the alignment graph, from say (i',j') to (i,j), we let $w((i',j'),(i,j))$ denote the weight of the edge. For any vertex (i,j) in the graph, we let $N_{in}(i,j)$ denote the set of vertices with an edge to (i,j), that is, the set of *in-neighbors* of (i,j).

5.1 Standard local alignment algorithm

For computing the standard local alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$, we define $s(i,j)$ to be the weight of the heaviest path, including the empty one, that ends at vertex (i,j). Note that by Theorem 2, this value equals the highest scoring alignment achievable by any pair of consecutive substrings $\mathcal{R}_1[i',i]$ and $\mathcal{R}_2[j',j]$ with $i' \in \{1,\ldots,i\}$ and $j' \in \{1,\ldots,j\}$. Therefore, the maximum $s(i,j)$ over all $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,m\}$ equals the standard local alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$.

Lemma 2. *The recursion below correctly computes $s(i,j)$:*

$$s(i,j) = \max \begin{cases} s(i',j') + w((i',j'),(i,j)) \ \textit{where}\ (i',j') \in N_{in}(i,j) \\ 0 \end{cases}$$

Time Complexity: Using standard dynamic programming, once the alignment graph of $\mathcal{R}_1$ and $\mathcal{R}_2$ is constructed, we can compute $s(i,j)$ for every $i \in \{1,\ldots,n\}$

and $j \in \{1,\ldots,m\}$ in $\mathcal{O}(nm)$ time, since the in-degree of every vertex is at most three. The preprocessing step takes $\mathcal{O}(m^2 n \lg n)$ time [14], and therefore our suggested algorithm solves the standard local alignment problem in $\mathcal{O}(m^2 n \lg n + nm) = \mathcal{O}(m^2 n \lg n)$ time.

5.2 Normalized local alignment algorithm

We next present an algorithm for computing the normalized local alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$. This algorithm works by computing all optimal (in terms of length) local alignments of any possible score, and which end at any possible pair of positions in $\mathcal{R}_1$ and $\mathcal{R}_2$. For this purpose, it is convenient to slightly abuse the graph-theoretic notion of path lengths, and define the *length* of any path from (i',j') to (i,j) in the alignment graph as the value $\Delta((i',j'),(i,j)) = j'-j+i'-i$ rather than the number of edges in this path. In other words, the length of any path from (i',j') to (i,j) is defined to be the combined lengths of the two consecutive subsequences $\mathcal{R}_1[i'+1,i]$ and $\mathcal{R}_2[j'+1,j]$.

For computing the normalized local alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$, we define $s^k(i,j)$ to be the length of the shortest path that ends at vertex (i,j) and has weight equal to k, or ∞ if no such path exists. Note that the normalized score of such a path is given by $k/s^k(i,j)$.

Lemma 3. *The recursion below correctly computes $s^k(i,j)$:*

$$s^k(i,j) = \min\left\{ s^{k'}(i',j') + \Delta((i',j'),(i,j)) \;\middle|\; \begin{array}{l} (i',j') \in N_{in}(i,j), \\ k' = k - w((i',j'),(i,j)) \end{array} \right\}.$$

Notice that if our scoring scheme contains values which are not constant, we could use a similar recursion in which the roles of lengths and scores are reversed. This is done by defining $s^k(i,j)$ to be the weight of the heaviest length k path that ends at vertex (i,j). The advantage of defining $s^k(i,j)$ as in the presentation above is that one can stop the computation once a satisfying solution is found.

Time Complexity: Let δ_{min} and δ_{max} be the minimum and maximum score of a single edit operation in our given scoring scheme $\delta = (\delta_1, \delta_2)$. Notice that $|(m+n)\delta_{max}|$ and $-|(m+n)\delta_{min}|$ are upper and lower bounds on the global alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$. Set $\hat{k} = |(m+n)\delta_{max}|$ and $\check{k} = -|(m+n)\delta_{min}|$. Using standard dynamic programming, once the alignment graph of $\mathcal{R}_1$ and $\mathcal{R}_2$ is constructed, we can compute $s^k(i,j)$ for every $i \in \{1,\ldots,n\}$, $j \in \{1,\ldots,m\}$, and $k \in \{\check{k},\ldots,\hat{k}\}$, in $\mathcal{O}(nm(\hat{k}-\check{k}))$ time, which is $\mathcal{O}(n^2 m)$ assuming δ_{max} and δ_{min} are both constants. Also, The bounds on k follow from the integrality of the scoring scheme. Note that if either δ_{max} or δ_{min} are not constants, or if the scoring scheme is not integral, we can use the alternative definition of $s^k(i,j)$ given above to obtain the same complexity bounds. With a preprocessing stage of $\mathcal{O}(m^2 n \lg n)$ time [14], our suggested algorithm therefore solves the normalized local alignment problem in $\mathcal{O}(m^2 n \lg n + n^2 m)$ time.

References

1. J. Alliali and M-F. Sagot. A new distance for high level RNA secondary structure comparison. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(1):4–14, 2005.
2. A.N. Arslan, Ö. Eğecioğlu, and P.A. Pevzner. A new approach to sequence alignment: normalized sequence alignment. *Bioinformatics*, 17(4):327–337, 2001.
3. R. Backofen and S. Will. Local sequence-structure motifs in RNA. *Journal of Bioinformatics and Computational Biology (JBCB)*, 2(4):681–698, 2004.
4. P. Chartrand, X-H. Meng, R.H. Singer, and R.M. Long. Structural elements required for the localization of ASH1 mRNA and of a green fluorescent protein reporter particle *in vivo*. *Current Biology*, 9:333–336, 1999.
5. S. Chen, Z. Wang, and K. Zhang. Pattern matching and local alignment for RNA structures. In *international conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS)*, pages 55–61, 2002.
6. J. Couzin. Breakthrough of the year. Small RNAs make big splash. *Science*, 298(5602):2296–2297, 2002.
7. K.M. Currey, D. Sasha, B.A. Shapiro, J. Wang, and K. Zhang. An algorithm for finding the largest approxemately common substructure of two trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):889–895, 1998.
8. E.D. Demaine, S. Mozes, B. Rossman, and O. Weimann. An $O(n^3)$-time algorithm for tree edit distance. Technical Report arXiv:cs.DS/0604037, Cornell University, 2006.
9. N. Efraty and G.M. Landau. Sparse normalized local alignment. In *15th Combinatorial Pattern Matching conference (CPM)*, pages 333–346, 2004.
10. R. Giegerich, M. Höchsmann, S. Kurtz, and T. Töller. Local similarity in RNA secondary structures. In *Computational Systems Bioinformatics (CSB)*, pages 159–168, 2003.
11. V. Guignon, C. Chauve, and S. Hamel. An edit distance between RNA stem-loops. In *12th Symposium on String Processing and Information Retrieval (SPIRE)*, pages 335–347, 2005.
12. D. Gusfield. *Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology.* Press Syndicate of the University of Cambridge, 1997.
13. T. Jiang, G. Lin, B. Ma, and K. Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2):371–88, 2002.
14. P.N. Klein. Computing the edit-distance between unrooted ordered trees. In *6th European Symposium on Algorithms (ESA)*, pages 91–102, 1998.
15. P.B. Moore. Structural motifs in RNA. *Annual review of biochemistry*, 68:287–300, 1999.
16. D. Shasha and K. Zhang. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.
17. T.F. Smith and M.S. Waterman. The identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
18. J. Wang and K. Zhang. Identiffying consensus of trees through alignment. *Information Sciences*, 126:165–189, 2000.
19. M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244(4900):48–52, 1989.
20. M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981.