JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 14, Number 8, 2007

Pp. 1074–1087
DOI: 10.1089/cmb.2007.0062

Locality and Gaps in RNA Comparison

ROLF BACKOFEN,[1] SHIHYEN CHEN,[2] DANNY HERMELIN,[3] GAD M. LANDAU[3,4]
MIKHAIL A. ROYTBERG,[5] OREN WEIMANN,[6] and KAIZHONG ZHANG,[2]

ABSTRACT

Locality is an important and well-studied notion in comparative analysis of biological sequences. Similarly, taking into account affine gap penalties when calculating biological sequence alignments is a well-accepted technique for obtaining better alignments. When dealing with RNA, one has to take into consideration not only sequential features, but also structural features of the inspected molecule. This makes the computation more challenging, and usually prohibits the comparison only to small RNAs. In this paper we introduce two local metrics for comparing RNAs that extend the Smith-Waterman metric and its normalized version used for string comparison. We also present a global RNA alignment algorithm which handles affine gap penalties. Our global algorithm runs in $\mathcal{O}(m^2 n(1 + \lg \frac{n}{m}))$ time, while our local algorithms run in $\mathcal{O}(m^2 n(1 + \lg \frac{n}{m}))$ and $\mathcal{O}(n^2 m)$ time, respectively, where $m \leq n$ are the lengths of the two given RNAs. These time complexities are comparable to the time complexity of any known RNA alignment algorithm. Furthermore, both our global and local algorithms are robust to selections of arbitrary scoring schemes.

Key words: affine gap penalties, alignment, gaps, local alignment, pairwise comparison, RNA.

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 that are part of the nucleotides form hydrogen bonds within the same molecule leading to structure formation. 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 the cell control.

[1]Institute of Computer Science, Albert-Ludwigs Universität Freiburg, Freiburg, Germany.
[2]Department of Computer Science, University of Western Ontario London, Ontario, Canada.
[3]Department of Computer Science, University of Haifa, Haifa, Israel.
[4]Department of Computer and Information Science, Polytechnic University, New York, New York.
[5]Institute of Mathematical Problems in Biology, Russian Academy of Science, Pushchino, Moscow Region, Russia.
[6]Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, Massachusetts.

As an example, research on non-coding RNAs has been elected as the scientific breakthrough of 2002 by the readers of Science (Couzin, 2002).

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 (i.e., the set of all nucleotide pairings) (Moore, 1999). Second, it is known that the structure of an RNA is often more conserved than its sequence during evolution (Chartrand et al., 1999). 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 Zhang and Shasha (1989) which represented RNA sequences as rooted ordered trees (Fig. 1), 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 Zhang and Shasha (1989), there have been attempts at extending either the set of edit operations on trees (Alliali and Sagot, 2005; Guignon et al., 2005), or the set of allowed RNA alignments (Jiang et al., 2002), 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 (Backofen and Will, 2004; Chen et al., 2002; Currey et al., 1998; Giegerich et al., 2003; Wang and Zhang, 2000). These can be divided roughly into two main categories, depending on the exact notion of locality under consideration. The first category (Backofen and Will, 2004; Currey et al., 1998; Wang and Zhang, 2000) 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 (Chen et al., 2002; Giegerich et al., 2003) defines locality in the sequential sense, thus extending the well understood notion of locality in strings to RNA sequences.

Most of the biological justifications given for the importance of local similarity also apply for justifying the introduction of *affine gap penalties* to similarity calculation (Gotoh, 1982). This is enhanced by the fact that the deletion of a bounded number of consecutive nucleotides in an RNA sequence often occurs in a single mutation. Hence, one single long gap is more probable to occur than a multitude of separated short gaps at the same site. Affine gap penalties for RNA comparison was proposed and studied by Wang et al. (2001) and by Chen et al. (2002). Touzet (2003) considered structural gaps rather than sequential ones.

In this paper we introduce two sequentially-local similarity metrics for comparing RNA sequences that extend the Smith-Waterman metric and its normalized version used in strings. We generalize the familiar alignment graph used in string comparison to apply also for RNA sequences, and then utilize

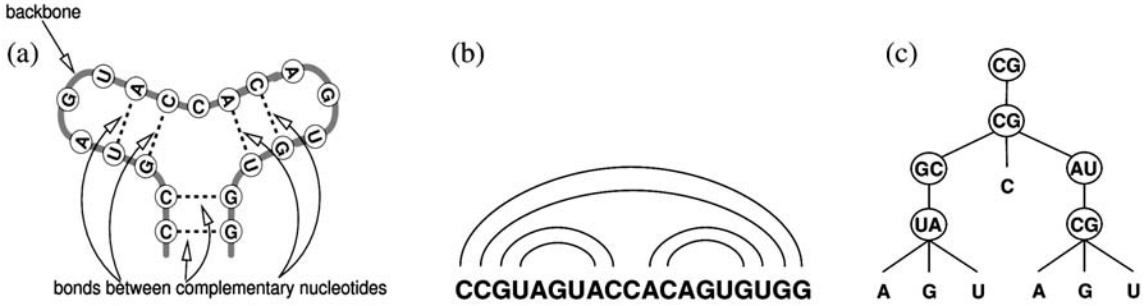

FIG. 1. Three different ways of viewing an RNA sequence. **(a)** A schematic 2-dimensional description of an RNA folding. **(b)** A linear representation of the RNA. **(c)** The RNA as a rooted ordered tree.

this generalization to devise two algorithms for computing local similarity according to our two suggested metrics. Following this, we extend the global RNA alignment metric of Zhang and Shasha (1989) to handle sequential affine gap penalties. We do so by modifying the standard recursions used for tree editing in a way inspired by Gotoh (1982) for string editing with affine gap penalties. Our global algorithm runs in $\mathcal{O}(m^2n(1+\lg\frac{n}{m}))$ time while our local algorithms run in $\mathcal{O}(m^2n(1+\lg\frac{n}{m}))$ and $\mathcal{O}(n^2m)$ time respectively, where $m \leq n$ are the lengths of the two given RNAs. Both our global and local algorithms can work with any arbitrary scoring schemes, including affine gap penalties.

1.1. Related work

Sequential local alignment of RNA sequences has been previously considered (Chen et al., 2002; Giegerich et al., 2003). In Chen et al. (2002), the authors considered locally aligning RNA sequences allowing gaps. When the gap length is set to zero, their definition is equivalent to our extension of the Smith-Waterman metric. The algorithm given in Chen et al. (2002) has worst-case running time of $\mathcal{O}(S_1S_2|R_1||R_2|)$, where $|R_1|$ and $|R_2|$ are the lengths of the given RNAs, and S_1 and S_2 denote the number of stems in these RNAs. In Giegerich et al. (2003), the authors considered local alignment with an extended set of edit operations, following the approach of Jiang et al. (2002). They gave an algorithm with worst-case running time of $\mathcal{O}(|T_1||T_2|deg(T_1)deg(T_2)(deg(T_1)+deg(T_2))$, where T_1 and T_2 are the two trees representing the given RNAs, and $|T_i|$ and $deg(T_i)$ denote the number of nodes and maximum degree of T_i for $i = 1, 2$. Finally, for normalized local alignment, the approach of Arslan et al. (2001) for string comparison can also be extended to RNA sequences. This is done by running a global RNA alignment algorithm $\mathcal{O}(\lg\Delta)$ times, where Δ is the difference between the highest and lowest scoring alignments of the two RNAs.

1.2. 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 global and local alignments of RNA sequences. In Section 3, we describe 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 4 to design two algorithms that compute the local alignment score between a pair RNA sequences according to the Smith-Waterman metric and its normalized version. In Section 5, we present our algorithm for computing global alignment scores with affine gap penalties. Finally, we discuss some conclusions in Section 6.

1.3. Notations

An RNA sequence $\mathcal{R}$ is an ordered pair (S, R), where $S = s_1 \cdots s_{|S|}$ is a string over the alphabet $\Sigma = \{A, C, G, U\}$, and $R \subseteq \{1, \ldots, |S|\} \times \{1, \ldots, |S|\}$ 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 (1989) and Zuker and Stiegler (1981), 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 $l \in \{i', \ldots, i\}$ 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 $\mathcal{R}_1 = (S_1, P_1)$ and $\mathcal{R}_2 = (S_2, P_2)$ 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 (Zhang and Shasha, 1989). 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 an ordered subset of* $\{1, \ldots, n\} \cup \{-\} \times \{1, \ldots, m\} \cup \{-\}$ *satisfying the following conditions:*

- $(-, -) \notin \mathcal{A}$.
- $\forall (i, j) \in \{1, \ldots, n\} \times \{1, \ldots, m\} : i$ *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* $\mathcal{R}_2$.
- $\forall (i', i) \in P_1, (j', j) \in P_2 : (i', j') \in \mathcal{A} \iff (i, j) \in \mathcal{A}$. *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* $\mathcal{A}$.

The ordering of the pairs in $\mathcal{A}$ *is required to be a linear extension of the natural ordering between pairs of integers. That is,* (i', j') *is before* (i, j) *iff* $i' < i$ *or* $j' < j$.

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 ordering between the pairs corresponds to the order of edit operations. 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. Figure 2 gives two example alignments for a pair of RNA sequences.

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 $(i', i) \in P_1$. 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.

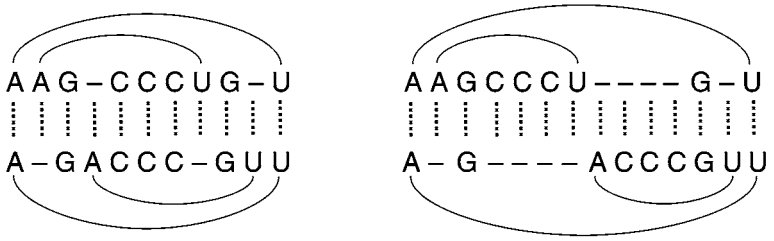

FIG. 2. Two example alignments for a pair of RNA sequences.

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:

$$\delta(\mathcal{A}) = \sum_{\substack{(i,j)\in\mathcal{A},\, i,j \text{ are not} \\ \text{arc endpoints}}} \delta_1(i, j) + \sum_{\substack{(i',j'),(i,j)\in\mathcal{A} \text{ s.t} \\ (i',i)\in P_1\cup\{(-,-)\} \text{ and} \\ (j',j)\in P_2\cup\{(-,-)\}.}} \delta_2(i', i, j', j).$$

2.1. RNA global 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 (Zhang and Shasha, 1989). Each arc (i, i') is identified with a set of ordered children which are all unpaired bases i'' such that $i < i'' < i'$, and all outermost arcs (l, l') with $i < l < l' < i'$ (Fig. 2). Zhang and Shasha (1989) suggested an algorithm for computing the edit distance between two ordered forests. Their algorithm was later improved by Klein (1998) to an $\mathcal{O}(m^2 n \lg n)$ algorithm, and recently by Demaine et al. (2007) to an $\mathcal{O}(m^2 n(1 + \lg \frac{n}{m}))$ algorithm, where $m \leq n$ denote the number of nodes in the two trees. All three algorithms of Demaine et al. (2007), Klein (1998), and Zhang and Shasha (1989) belong to the family of *decomposition strategy algorithms* (Dulucq and Touzet, 2003).

The decomposition strategy algorithms are all based on the following recursion: Given two ordered forests F and G, denote by v and w the two leftmost or two rightmost roots of F and G respectively, then $\delta(F, G)$ can be computed as follows:

- $\delta(\emptyset, \emptyset) = 0$.
- $\delta(F, \emptyset) = \delta(F - v, \emptyset) +$ cost of deleting v.
- $\delta(\emptyset, G) = \delta(\emptyset, G - w) +$ cost of deleting w.
- $\delta(F, G) = \min \begin{cases} \delta(F - v, G) + \text{cost of deleting } v, \\ \delta(F, G - w) + \text{cost of deleting } w, \\ \delta(F_v - v, G_w - w) + \delta(F - F_v, G - G_w) + \text{cost of relabeling } v \text{ to } w. \end{cases}$

where F_v is the subtree of F rooted at v, $F - v$ is the forest obtained by F after deleting the node v, and $F - F_v$ is the forest obtained by F after deleting the whole tree F_v (similar definitions for G and w). The only difference between two decomposition strategy algorithms is in the choice of whether v and w are the leftmost or the rightmost roots of F and G in each recursive call. Formally, a *strategy* is a mapping from pairs (F', G') of subforests of F and G to $\{\mathsf{left}, \mathsf{right}\}$. A lower bound of $\Omega(m^2 n(1 + \lg \frac{n}{m}))$ was shown in Demaine et al. (2007) for any decomposition strategy algorithm.

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, an important property of any decomposition strategy algorithm is that it computes the edit distance between every two root-deleted subtrees of F and G regardless of how we choose v and w.

Lemma 1 (Demaine et al., 2007). *Given two ordered forests F and G, any decomposition strategy algorithm that computes $\delta(F, G)$ will compute $\delta(F_v - v, G_w - w)$ for all $v \in F$ and $w \in G$.*

2.2. RNA local alignment

While the metric described in the previous sections 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 (Smith and Waterman, 1981) and normalization (Arslan et al., 2001) 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 arc (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 arc (l', l) in $\mathcal{R}_1[i', i]$ against '–'. Furthermore, we require that the total cost of aligning the left and right endpoints of an arc against '–' (in two different alignments) be equal to the cost of deleting this arc. That is, $\delta_2(l', -, -, -) + \delta_2(-, l, -, -) = \delta_2(l', l, -, -)$ and $\delta_2(-, -, l', -) + \delta_2(-, -, -, l) = \delta_2(-, -, l', l)$. Our definition of $\delta(\mathcal{A})$ naturally extends to the case where $\mathcal{A}$ is a local alignment.

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

Definition 3 ($OPT^{arc}_\delta(\mathcal{R}_1[i', i], \mathcal{R}_2[j', j])$). *For a pair of arcs $(i', i) \in P_1$ and $(j', j) \in P_2$, we set $OPT^{arc}_\delta(\mathcal{R}_1[i', i], \mathcal{R}_2[j', j]) = \delta_2(i', i, j', j) + OPT_\delta(\mathcal{R}_1[i' + 1, i - 1], \mathcal{R}_2[j + 1, j - 1])$.*

Lemma 1 now means that that $OPT^{arc}_\delta(\mathcal{R}_1[i', i], \mathcal{R}_2[j', j])$ is computed between all pairs of arcs $(i', i) \in P_1$ and $(j', j) \in P_2$ in a single execution of any decomposition strategy algorithm. The importance of this observation will become apparent later on.

3. AN ALIGNMENT GRAPH FOR RNA SEQUENCES

We next present an adaptation of the alignment graph that is used to describe string alignment (Gusfield, 1997), to an alignment graph that describes alignments of RNA sequences. Later, 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. A different graph (the *edit graph*) was suggested in Touzet (2005) as a tool for comparing similar ordered labeled trees. However, the edit graph of Touzet (2005) does not capture local alignments with broken arcs. 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 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. Figure 4a illustrates the alignment graph of two example strings.

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 (Gusfield, 1997). *An alignment of $S_1[i', i]$ and $S_2[j', j]$ has optimal score iff it corresponds to the heaviest path, i.e., the one with maximum total edge weight, 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 3a. 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 3b. 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 $\{(i-1, j), (i, j) \mid 0 \leq j \leq m\}$ 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 $\{(i, j-1), (i, j) \mid 0 \leq j \leq m\}$.

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

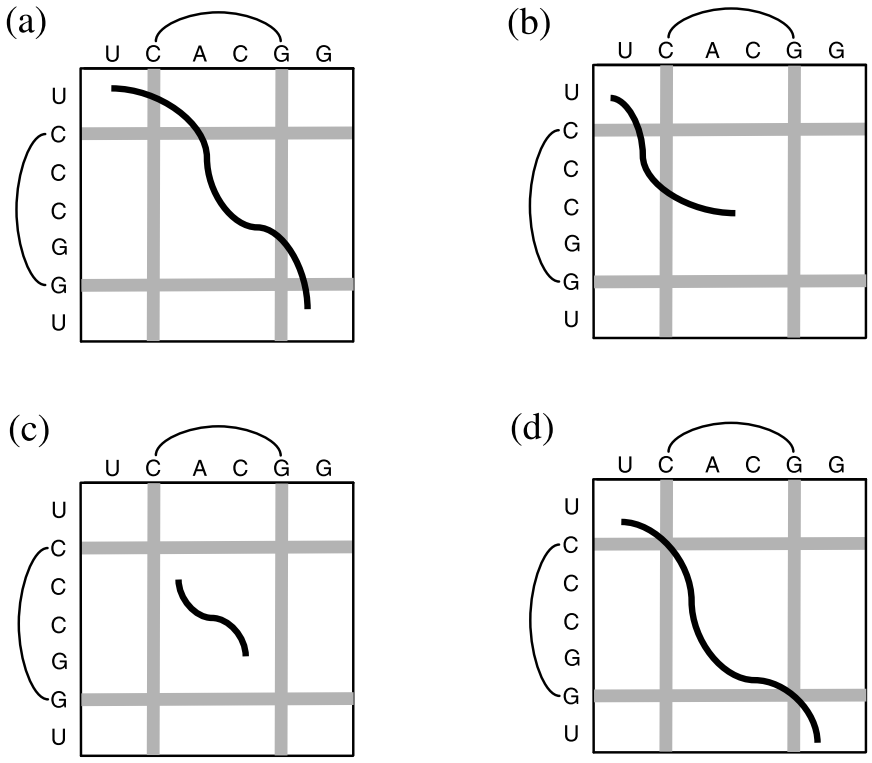

FIG. 3. Four different situations that occur when aligning RNA sequences.

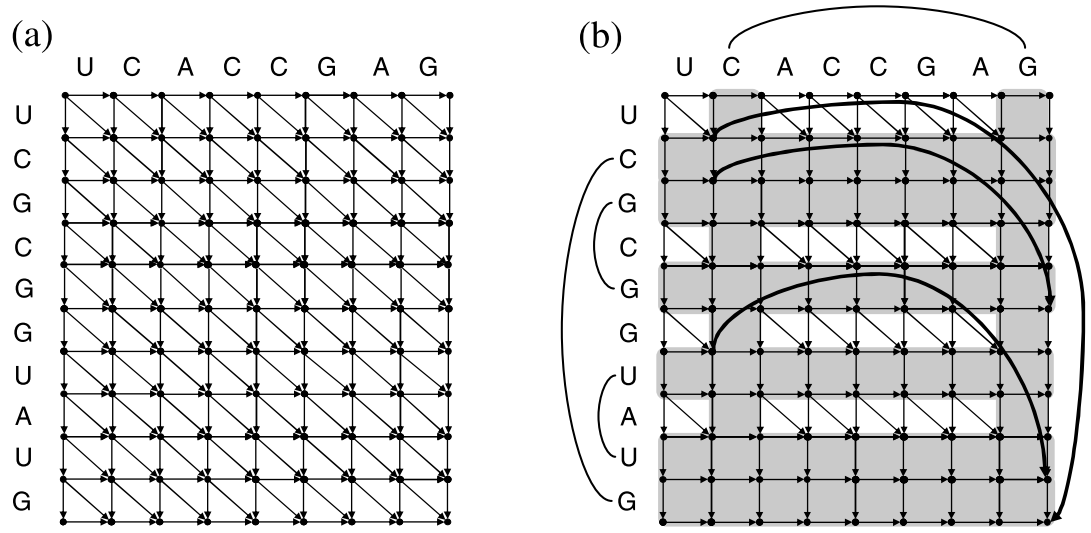

FIG. 4. The alignment graph before and after. **(a)** The alignment graph of two strings. **(b)** The alignment graph for RNA sequences after adding the shortcut edges and removing all diagonal edges from the shaded rows and columns.

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 2. *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.*

Proof. First note that the only edges missing from the original alignment graph are diagonal edges in rows and columns which correspond to arc endpoints. Therefore, any alignment which deletes all arc endpoints is represented by a path in the grid part, and vice versa. Consider such an alignment $\mathcal{A}$ between say $\mathcal{R}_1[i', i]$ and $\mathcal{R}_2[j', j]$, and let p be its corresponding path in the grid part. For any broken arc in $\mathcal{R}_1[i', i]$ and $\mathcal{R}_2[j', j]$, there exists exactly one edge in p with weight equal to the penalty of aligning the endpoint of this arc against "–." For every arc in $\mathcal{R}_1[i', i]$ and $\mathcal{R}_2[j', j]$ which is not broken, there are two edges in p with total weight which equals the score of deleting this arc. Therefore, since all other edges in p have original weights, the score of $\mathcal{A}$ equals the weight of p. ■

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 3d, that corresponds to an alignment which aligns $(i', i) \in P_1$ against $(j', j) \in P_2$. 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 2.2, 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 $\mathcal{R}_2$. Figure 4b depicts the final alignment graph of $\mathcal{R}_1$ 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 $\mathcal{R}_2$.*

Proof. Let p be the heaviest path from (i', j') to (i, j) in the alignment graph of $\mathcal{R}_1$ and $\mathcal{R}_2$. Decompose p into smaller subpaths such that each subpath is either a single shortcut edge or it contains no shortcut edges at all. By Lemma 2 and the correctness of our preprocessing step, each one of these subpaths corresponds to an optimal alignment between two consecutive subsequences of $\mathcal{R}_1[i'+1, i]$ and $\mathcal{R}_2[j'+1, j]$. We claim that the alignment $\mathcal{A}$ which is the union of all these alignments is an optimal alignment of $\mathcal{R}_1[i', i]$ and $\mathcal{R}_2[j', j]$.

Assume by contradiction that there exists an alignment $\mathcal{A}'$ of $\mathcal{R}_1[i'+1, i]$ and $\mathcal{R}_2[j'+1, j]$ such that $\delta(\mathcal{A}') > \delta(\mathcal{A})$. Call a pair of arcs $(i', i) \in P_1$ and $(j', j) \in P_2$ a maximal pair of arcs in $\mathcal{A}'$, if they are aligned against each other in $\mathcal{A}'$, and there are no two other arcs $(l_1', l_1) \in P_1$ and $(l_2', l_2) \in P_2$ aligned against each other in $\mathcal{A}'$ with $l_1' < i' < i < l_1$ and $l_2' < j' < j < l_2$. Decompose $\mathcal{A}'$ into alignments of common subsequences of $\mathcal{R}_1[i'+1, i]$ and $\mathcal{R}_2[j'+1, j]$ which are wrapped by pairs of maximal arcs in $\mathcal{A}'$, and into alignments of common subsequences in which all arcs are deleted. The first type of alignments correspond to shortcut edges, and therefore by the correctness of our preprocessing step, each has total weight which is at least the score of its corresponding alignment. The second type correspond to paths in the grid part, and therefore by Lemma 2, each has weight which equals exactly the score of its corresponding alignment. The concatenation of these paths is therefore a path p' from (i', i) to (j', j) with a total weight which is greater or equal to $\delta(\mathcal{A})$. But this contradicts p being the heaviest path, since p has weight $\delta(\mathcal{A}) < \delta(\mathcal{A}')$.

We thus proved that the heaviest path from (i', i) to (j', j) in the alignment graph of $\mathcal{R}_1$ and $\mathcal{R}_2$ corresponds to an optimal alignment of $\mathcal{R}_1[i'+1, i]$ and $\mathcal{R}_2[j'+1, j]$. Proving the converse direction is symmetric, and the theorem follows. ■

4. LOCAL ALIGNMENT ALGORITHMS

In the following section we present algorithms that utilize the alignment graph in order to determine the local similarity between RNA sequences. We start by defining two local similarity metrics.

4.1. Standard local alignment

The well-known Smith-Waterman technique (Smith and Waterman, 1981) 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 (Gusfield, 1997). In RNA terms, it is defined by:

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

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 $\mathcal{R}_1 = (S_1, P_1)$ and $\mathcal{R}_2 = (S_2, P_2)$, determine the standard local alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$.*

4.2. Normalized local alignment

According to Arslan et al. (2001), 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 (Arslan et al., 2001). In our terms, this *normalized alignment score* of $\mathcal{R}_1$ and $\mathcal{R}_2$

is defined by:

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

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.

When using a similarity metric for alignment score (Ma and Zhang, 2005), it would be desirable to have a normalized alignment score that is also a similarity metric. However, the above normalized alignment score may not have this property. To overcome this, we give the following alternative definition of the normalized alignment score:

$$\max_{\substack{1\le i'\le i\le|\mathcal{R}_1| \\ 1\le j'\le j\le|\mathcal{R}_2| \\ \delta(\mathcal{R}_1[i',i],\mathcal{R}_2[j',j])\ge I}} \left\{ \frac{\delta(\mathcal{R}_1[i',i],\mathcal{R}_2[j',j])}{\delta(\mathcal{R}_1[i',i],\mathcal{R}_1[i',i])+\delta(\mathcal{R}_2[j',j],\mathcal{R}_2[j',j])-\delta(\mathcal{R}_1[i',i],\mathcal{R}_2[j',j])\}} \right\} \tag{2}$$

Definition 5 (The normalized local alignment problem). *Given two RNA sequences $\mathcal{R}_1 = (S_1, P_1)$ and $\mathcal{R}_2 = (S_2, P_2)$, and an integer I, determine the normalized local alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$.*

4.3. Local alignment algorithms

We next describe two algorithms for computing local similarity of $\mathcal{R}_1$ and $\mathcal{R}_2$ according to the above similarity metrics. 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).

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 standard local alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$ is the maximum $s(i, j)$ over all $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, m\}$.

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

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

Proof. Assume that all values $s(i', j')$ have been correctly computed for $0 \le i' \le i$, $0 \le j' \le j$, and $(i', j') \ne (i, j)$. Then the heaviest path ending with an edge from say (i', j') to (i, j), has weight which is equal to $s(i', j') + w((i', j'), (i, j))$. The recursion above follows from the correctness of $s(i', j')$, and since the empty path has weight zero. ■

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 $\mathcal{O}(m^2 n(1 + \lg \frac{n}{m}))$ time (Demaine et al., 2007), and therefore our suggested algorithm solves the standard local alignment problem in $\mathcal{O}(m^2 n(1 + \lg \frac{n}{m}) + nm) = \mathcal{O}(m^2 n(1 + \lg \frac{n}{m}))$ time.

We next present an algorithm for computing the normalized local alignment score of $\mathcal{R}_1$ and $\mathcal{R}_2$. Our algorithm is written in terms of the first normalized metric (Equation (1)) but easily extends to the second one (Equation (2)). It 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 4. *The recursion below correctly computes* $s^k(i, j)$*:*

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

Proof. The shortest k weighting path that ends at vertex (i, j) can be decomposed into the last edge of the path outgoing from some $(i', j') \in N_{in}(i, j)$, and the shortest path with weight $k - w((i', j'), (i, j))$ that ends at (i', j'). The correctness of the recursion follows. ■

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 $\mathcal{O}(m^2 n(1 + \lg \frac{n}{m}))$ time (Demaine et al., 2007), our suggested algorithm therefore solves the normalized local alignment problem in $\mathcal{O}(m^2 n(1 + \lg \frac{n}{m}) + n^2 m) = \mathcal{O}(n^2 m)$ time.

5. GLOBAL ALIGNMENT WITH AFFINE GAP PENALTIES

In this section we introduce a method for extending any decomposition strategy algorithm to handle *affine gap penalties* in the same time and space bounds. Given an alignment $\mathcal{A}$ of $\mathcal{R}_1$ and $\mathcal{R}_2$, a *gap* is a consecutive sequence of pairs $(i,$"$-$"$) \in \mathcal{A}$ or a consecutive sequence of pairs ("$-$"$, j) \in \mathcal{A}$. For example, in the left alignment of Figure 2 the gaps are $(\mathrm{A}, -)$, $(-, \mathrm{A})$, $(\mathrm{U}, -)$ and $(-, \mathrm{U})$; whereas in the right alignment the gaps are $(\mathrm{A}, -)$, $(\mathrm{CCCU}, ----)$, $(----, \mathrm{ACCC})$ and $(-, \mathrm{U})$. If we view an alignment as a path in the alignment graph, then a gap corresponds to a subpath that consists only of vertical edges or only horizontal edges.

In biological context, one single long gap is more probable to occur than a multitude of separated short gaps at the same site. This situation can be modeled by using an affine gap penalty function $f(k) = g + k\alpha$, where $f(k)$ is the score of a gap of length k, g is an initiation penalty to a gap, and α is the cost of extending a gap by one. This was first introduced by Gotoh (1982) in the context of string editing, and we extend the technique to ordered tree editing as proposed by Wang et al. (2001) and by Chen et al. (2002).

As we focus on sequential-gaps, notice that deleting an arc means deleting its two endpoints, each of which might be a part of a different gap. To account for this, we slightly modify the familiar construction of a tree from an RNA sequence and the recursion presented in Section 2.1. To each node in the tree that corresponds to some arc (i, j) we add a leftmost child i^* and a rightmost child j^*. Both i^* and j^* are leaves, and relabeling them to anything costs ∞. In the recursion, when we delete the rightmost (respectively leftmost) root we also delete its rightmost (respectively leftmost) child, and when we match a node we remove both its leftmost and rightmost children.

In what follows, we modify the recursion of Section 2.1 to handle sequential affine gap penalties. This recursion can also be viewed as if it constructs the optimal path in the alignment graph. It does so by adding edges one at a time to the heads or tails of existing paths (depending on the strategy). In order to modify the recursion, we take a closer look at how a path is extended. We note that extending a path vertically or horizontally may either continue an existing gap or may start a new gap. Also, we need to consider extending a path diagonally by a diagonal edge or a shortcut edge (depending which of them is present). We keep track of these three options in both the head and the tail of the path.

For forests F and G, we define nine different "problems." Denoting by ℓ_F and r_F the leftmost and rightmost roots of a forest F, these problems are:

- $[F, G]$ is the edit distance with affine gap penalties between F and G.
- ${}_{\mathbf{F}}[F, G]$ (respectively: ${}_{\mathbf{G}}[F, G]$, $[F, G]_{\mathbf{F}}$, $[F, G]_{\mathbf{G}}$) is the edit distance with affine gap penalties between F and G *subject to the condition* that ℓ_F (respectively: ℓ_G, r_F, r_G) is deleted.
- ${}_{\mathbf{F}}[F, G]_{\mathbf{F}}$ (respectively: ${}_{\mathbf{F}}[F, G]_{\mathbf{G}}$, ${}_{\mathbf{G}}[F, G]_{\mathbf{F}}$, ${}_{\mathbf{G}}[F, G]_{\mathbf{G}}$) is the edit distance with affine gap penalties between F and G *subject to the condition* that both ℓ_F and r_F (respectively: ℓ_F and r_G, ℓ_G and r_F, ℓ_G and r_G) are deleted.

The following recursion computes the values of all the nine problems. Our recursion can follow any given decomposition strategy that determines the direction of the recursion in every recursive call. We present the recursion for a recursive call in which the strategy says left, the case in which the strategy says right is symmetric (simply replace the left and right subscripts of the "["s and "]"s). For every $X \in \{F, G, \varepsilon\}$ (where ε is used to denote the empty string):

- $$[F, G]_{\mathbf{X}} = \min \begin{cases} {}_{\mathbf{F}}[F, G]_{\mathbf{X}}, \\ {}_{\mathbf{G}}[F, G]_{\mathbf{X}}, \\ [F_{\ell_F} - \ell_F, G_{\ell_G} - \ell_G] + [F - F_{\ell_F}, G - G_{\ell_G}]_{\mathbf{X}} + \text{cost of relabeling } \ell_F \text{ to } \ell_G. \end{cases}$$
- $${}_{\mathbf{F}}[F, G]_{\mathbf{X}} = \min \begin{cases} {}_{\mathbf{F}}[F - \ell_F, G]_{\mathbf{X}} + \alpha, \\ [F - \ell_F, G]_{\mathbf{X}} + g + \alpha. \end{cases}$$
- $${}_{\mathbf{G}}[F, G]_{\mathbf{X}} = \min \begin{cases} {}_{\mathbf{G}}[F, G - \ell_G]_{\mathbf{X}} + \alpha, \\ [F, G - \ell_G]_{\mathbf{X}} + g + \alpha. \end{cases}$$

The halting conditions of this recursion are:

- $[\emptyset, \emptyset] = 0$
- ${}_{\mathbf{F}}[\emptyset, \emptyset] = {}_{\mathbf{G}}[\emptyset, \emptyset] = [\emptyset, \emptyset]_{\mathbf{F}} = [\emptyset, \emptyset]_{\mathbf{G}} = {}_{\mathbf{F}}[\emptyset, \emptyset]_{\mathbf{F}} = {}_{\mathbf{G}}[\emptyset, \emptyset]_{\mathbf{G}} = g$
- ${}_{\mathbf{F}}[\emptyset, \emptyset]_{\mathbf{G}} = {}_{\mathbf{G}}[\emptyset, \emptyset]_{\mathbf{F}} = 2g$
- $[\emptyset, G] = [\emptyset, G]_{\mathbf{G}} = {}_{\mathbf{G}}[\emptyset, G] = {}_{\mathbf{G}}[\emptyset, G]_{\mathbf{G}} = g + \alpha|G|$
- $[F, \emptyset] = [F, \emptyset]_{\mathbf{F}} = {}_{\mathbf{F}}[F, \emptyset] = {}_{\mathbf{F}}[F, \emptyset]_{\mathbf{F}} = g + \alpha|F|$
- $[F, \emptyset]_{\mathbf{G}} = {}_{\mathbf{G}}[F, \emptyset] = {}_{\mathbf{G}}[F, \emptyset]_{\mathbf{G}} = {}_{\mathbf{F}}[F, \emptyset]_{\mathbf{G}} = {}_{\mathbf{G}}[F, \emptyset]_{\mathbf{F}} = 2g + \alpha|F|$
- $[\emptyset, G]_{\mathbf{F}} = {}_{\mathbf{F}}[\emptyset, G] = {}_{\mathbf{F}}[\emptyset, G]_{\mathbf{F}} = {}_{\mathbf{F}}[\emptyset, G]_{\mathbf{G}} = {}_{\mathbf{G}}[\emptyset, G]_{\mathbf{F}} = 2g + \alpha|G|$

Time and space complexity. For every subproblem encountered by the decomposition strategy, our recursion encounters nine subproblems. Therefore, since the number of subproblems corresponds to the

time complexity, our recursion requires $\mathcal{O}(m^2 n(1 + \lg \frac{n}{m}))$ time by using the strategy of Demaine et al. (2007). Computing these subproblems can be done via dynamic programming in $\mathcal{O}(mn)$ space as shown in Demaine et al. (2007).

6. CONCLUSION

In this paper we considered alternatives to the RNA global alignment metric. The first alternative is a sequentially-local metric. We introduce two sequentially-local similarity metrics for comparing RNA sequences—the Smith-Waterman metric, and its normalized version. The second alternative is an extension of the global RNA alignment metric of Zhang and Shasha (1989) that can handle sequential affine gap penalties.

We defined the RNA equivalent of the well-known alignment graph used for strings, and described an efficient computation of it. Using this graph, we were able to devise two algorithms for computing local similarity according to our two sequentially-local metrics. These algorithms run in $\mathcal{O}(m^2 n(1 + \lg \frac{n}{m}))$ and $\mathcal{O}(n^2 m)$ time respectively, where $m \leq n$ are the lengths of the two given RNAs. In order to handle sequential affine gap penalties, we modified the standard recursions used for tree editing in a way inspired by Gotoh (1982) for string editing with affine gap penalties. This modification does not change the asymptotic of the tree editing algorithms.

Both our global and local algorithms can work with any arbitrary scoring schemes. Furthermore, our local algorithms can easily be adapted to also handle sequential affine gap penalties. This is done by running our new global algorithm once in the preprocessing step, and then computing the local score in a manner similar to Gotoh (1982).

ACKNOWLEDGMENTS

Preliminary work that led up to this paper can be found in Backofen et al. (2005, 2006). This work was partially supported by the Caesarea Edmond Benjamin de Rothschild Foundation Institute for Interdisciplinary Applications of Computer Science (grant to D.H.), by NSERC (grants to S.C. and K.Z.), by INTAS 05-1000008-8028 and RFBR-06-04-49249 (grant to M.R.), and by the Israel Science Foundation 35/05 (grant to G.L.).

REFERENCES

Alliali, J., and Sagot, M.-F. 2005. A new distance for high level RNA secondary structure comparison. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 2, 4–14.

Arslan, A.N., Eğecioğlu, Ö., and Pevzner, P.A. 2001. A new approach to sequence alignment: normalized sequence alignment. *Bioinformatics* 17, 327–337.

Backofen, R., Hermelin, D., Landau, G.M., et al. 2005. Normalized similarity of RNA sequences. *Proc. 12th Symp. String Processing Inform. Retrieval* 360–369.

Backofen, R., Hermelin, D., Landau, G.M., et al. 2006. Local alignment of RNA sequences with arbitrary scoring schemes. *Proc. 17th Annu. Symp. Combin. Pattern Matching*, pgs. 246–257.

Backofen, R., and Will, S. 2004. Local sequence-structure motifs in RNA. *J. Bioinform. Comput. Biol.* 2, 681–698.

Chartrand, P., Meng, X.-H., Singer, R.H., et al. 1999. Structural elements required for the localization of ASH1 mRNA and of a green fluorescent protein reporter particle *in vivo*. *Curr. Biol.* 9, 333–336.

Chen, S., Wang, Z., and Zhang, K. 2002. Pattern matching and local alignment for RNA structures. *Proc. Int. Conf. Math. Eng. Techn. Med. Biol. Sci.* 55–61.

Couzin, J. 2002. Breakthrough of the year. Small RNAs make big splash. *Science* 298, 2296–2297.

Currey, K.M., Sasha, D., Shapiro, B.A., et al. 1998. An algorithm for finding the largest approxemately common substructure of two trees. *IEEE Trans. Pattern Analysis Machine Intell.* 20, 889–895.

Demaine, E.D., Mozes, S., Rossman, B., et al. 2007. An optimal decomposition algorithm for tree edit distance. *Proc. 34th Int. Colloq. Automata Lang. Programm.*, pgs. 146–157.

Dulucq, S., and Touzet, H. 2003. Analysis of tree edit distance algorithms. *Proc. 14th Annu. Symp. Combin. Pattern Matching* 83–95.

Giegerich, R., Höchsmann, M., Kurtz, S., et al. 2003. Local similarity in RNA secondary structures. *Proc. Comput. Syst. Bioinform.* 159–168.

Gotoh, O. 1982. An improved algorithm for matching biological sequences. *J. Mol. Biol.* 162, 705–708.

Guignon, V., Chauve, C., and Hamel, S. 2005. An edit distance between RNA stem-loops. *Proc. 12th Symp. String Processing Inform. Retrieval* 335–347.

Gusfield, D. 1997. *Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology*. Press Syndicate of the University of Cambridge, Cambridge, UK.

Jiang, T., Lin, G., Ma, B., et al. 2002. A general edit distance between RNA structures. *J. Comput. Biol.* 9, 371–388.

Klein, P.N. 1998. Computing the edit-distance between unrooted ordered trees. *Proc. 6th Eur. Symp. Algorithms* 91–102.

Ma, B., and Zhang, K. 2005. The similarity metric and the distance metric. *Proc. 6th Atlantic Symp. Comput. Biol. Genome Inform.* 1239–1242.

Moore, P.B. 1999. Structural motifs in RNA. *Annu. Rev. Biochem.* 68, 287–300.

Smith, T.F., and Waterman, M.S. 1981. The identification of common molecular subsequences. *J. Mol. Biol.* 147, 195–197.

Touzet, H. 2003. Tree edit distance with gaps. *Inform. Processing Lett.* 85, 123–129.

Touzet, H. 2005. A linear tree edit distance algorithm for similar ordered trees. *Proc. 16th Annu. Symp. Combin. Pattern Matching* 334–345.

Wang, J., and Zhang, K. 2000. Identifying consensus of trees through alignment. *Inform. Sci.* 126, 165–189.

Wang, Z., and Zhang, K. 2001. Alignment between two RNA structures. *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pgs. 690–703.

Zhang, K., and Shasha, D. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.* 18, 1245–1262.

Zuker, M. 1989. On finding all suboptimal foldings of an RNA molecule. *Science* 244, 48–52.

Zuker, M., and Stiegler, P. 1981. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.* 9, 133–148.

Address reprint requests to:
Dr. Danny Hermelin
Department of Computer Science
University of Haifa
Haifa 31905, Israel

E-mail: danny@cri.haifa.ac.il