Local Gapped Subforest Alignment and Its Application in Finding RNA Structural Motifs

Jesper Jansson, Ngo Trung Hieu, and Wing-Kin Sung

School of Computing, National University of Singapore, 3 Science Drive 2,
Singapore 117543
{jansson, ngotrung, ksung}@comp.nus.edu.sg

Abstract. We consider the problem of computing an optimal local alignment of two labeled ordered forests F_1 and F_2 where n_i and d_i, for $i \in \{1, 2\}$, denote the number of nodes in F_i and the degree of F_i, respectively; and its applications in finding RNA structural motifs. A previous result is the local closed subforest alignment problem, which can be solved in $O(n_1 n_2 d_1 d_2 (d_1 + d_2))$ time and $O(n_1 n_2 d_1 d_2)$ space. This paper generalizes the concept of a closed subforest to a *gapped subforest* and then presents an algorithm for computing the optimal local *gapped subforest alignment* of F_1 and F_2 in $O(n_1 n_2 d_1 d_2 (d_1 + d_2))$ time and $O(n_1 n_2 d_1 d_2)$ space. We show that our technique can improve the computation of the optimal local closed subforest alignment in $O(n_1 n_2 (d_1 + d_2)^2)$ time and $O(n_1 n_2 (d_1 + d_2))$ space. Furthermore, we prove that a special case of our local gapped subforest alignment problem is equivalent to a problem known in the literature as the local sequence-structure alignment problem (*lssa*). The previously best algorithm for *lssa* uses $O(n_1^2 n_2^2 (n_1 + n_2))$ time and $O(n_1 n_2)$ space; here, we show how to modify our main algorithm to obtain an algorithm for *lssa* running in $O(n_1 n_2 (d_1 + d_2)^2)$ time and $O(n_1 n_2 (d_1 + d_2))$ space.

1   Introduction

Many areas of computer science use labeled ordered trees to represent hierarchically structured information. It is often useful to measure the similarity between two or more such trees or to find parts of the trees that are similar. In computational molecular biology, labeled ordered trees are used to represent RNA molecules' secondary structures [14]. By measuring and comparing the similarity of secondary structure trees, researchers who investigate structural or evolutionary relationships between RNA molecules may obtain additional clues [4]. Furthermore, comparing RNAs lead to automated methods for discovering frequently recurring patterns in their secondary structures (also known as *motifs*) which are helpful when investigating the various functions in the cell of different types of RNA [8] or when predicting the secondary structure of a newly found RNA molecule.

Two ways to measure the similarity between two labeled ordered trees are by using the *tree edit distance* [15] or *alignments of trees* [11]. The problem of computing the optimal alignment of two trees can be viewed as a special

case of the tree edit distance problem [11]; indeed, the fastest known algorithms for optimal alignment between two trees have lower time complexities than the fastest known algorithms for the tree edit distance, both for unordered trees whose degrees are bounded by a constant [11, 19] and for ordered trees whose degrees are much smaller than their depths [11, 20].

Alignments between trees as defined in [11] consider similarities on *global* level only, in the sense that *every* node in the input trees must be paired off with either a node in the other tree or a space. [8] and [16] extended the concept of a global alignment of trees to a *local* alignment of trees by introducing problems in which the objective is to find two substructures of the input having the highest possible similarity, where the similarity between two substructures is defined using the maximum score of a global alignment between them.

In this paper, we improve the time and space complexities of the main algorithm presented in [8]. Moreover, we further extend the set of mathematical definitions and notations for local similarity in labeled forests by generalizing the concept of a closed subforest used in [8] to what we call a *gapped subforest*. Based on this new concept, we define a computational problem called the *local gapped subforest alignment problem* ($lgsf$) that can express even more general patterns of local similarities in two labeled ordered forests than the problem considered in [8], and give an efficient algorithm for solving it. Finally, we prove that a special case of $lgsf$ referred to as $lgsf_\beta$ is in fact equivalent to the *local sequence-structure alignment problem* ($lssa$) presented in [2], implying that a slightly modified version of our algorithm for $lgsf$ can be applied to solve $lssa$ much faster than the algorithm given in [2].

1.1 Problem Definitions

We first introduce some terminologies and notations used in this paper.

Let Σ be a set of symbols, the *alphabet*. A rooted ordered forest whose nodes are labeled by symbols in Σ is called a Σ-labeled forest (in short, forest). For any two nodes u and v in F, u and v are called *siblings* if and only if they have the same parents or both of them are roots of some trees in F. For any node u in F, let $e(u)$ be the rightmost sibling of u; let $l(u)$ and $r(u)$ be the sibling immediately to the left and to the right of u, respectively. u_L and u_R denote the leftmost and the rightmost children of u respectively. Given two siblings u and v, define $u..v$ as a sibling interval, i.e., the set of siblings which are to the right of u and to the left of v. If u and v are not siblings or if u is a right sibling of v, $u..v$ is an empty interval. Let $S(F)$ be the set of all sibling intervals of F. Note that the number of sibling intervals in $S(F)$ is $O(|F|deg(F))$ since, for each node u, there are at most $deg(F)$ sibling intervals $u..v$. Let $F[u..v]$ be the forest consisting of the subtrees rooted at the nodes in $u..v$.

Definition 1 (Closed Subforest). *Let F and F' be two Σ-labeled forests. F' is called a closed subforest of F if and only if $F' = F[u..v]$ for some siblings u, v in F. (see Fig. 1.)*
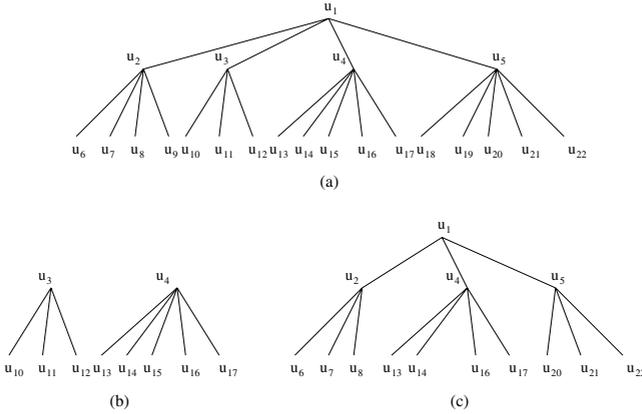
Fig. 1. (a) is an example of tree/forest F; (b) shows the closed subforest $F[u_3..u_4]$; (c) shows a gapped subforest at $F[u_1..u_1]$. This gapped subforest is formed by excluding the closed subforests $F[u_3..u_3]$, $F[u_9..u_9]$, $F[u_{15}..u_{15}]$, and $F[u_{18}..u_{19}]$. The forest in (b) is an α-gapped subforest at $F[u_3..u_5]$ since it only excludes $F[u_5..u_5]$. Also, (b) and (c) are β-gapped subforests at $F[u_3..u_4]$ and $F[u_1..u_1]$, respectively

Definition 2 (Gapped Subforest). *Let F and F' be two Σ-labeled forests. F' is called a gapped subforest of F if and only if there exists a sibling interval $u..v$ and k sibling intervals $x_i..y_i$, for $i = 1, 2, \ldots, k$, whose parents are p_i such that $p_i \neq p_j$ for all i, j; and the forest F' can be formed from F by excluding all k closed subforests $F[x_i..y_i]$ from $F[u..v]$. F' is called a gapped subforest at $F[u..v]$. We denote $gsf(F[u..v])$ or $gsf_*(F[u..v])$ as the set of all gapped subforests at $F[u..v]$.*

F' is called an α-gapped subforest at $F[u..v]$ if and only if we will not exclude any closed subforest $F[x..y]$ where $x = u$. We denote $gsf_\alpha(F[u..v])$ as the set of all α-gapped subforests at $F[u..v]$.

F' is called a β-gapped subforest at $F[u..v]$ if and only if we will not exclude any closed subforest $F[x..y]$ where $x..y \subseteq u..v$. We denote $gsf_\beta(F[u..v])$ as the set of all β-gapped subforests at $F[u..v]$. (see Fig. 1.)

Lemma 1. $gsf(F[u..u']) = gsf_\alpha(F[u..u']) \bigcup (\cup_{u'' \in u..u'} gsf_\beta[u''..u']))$

Given two Σ-labeled forests F and G, one way to measure their similarity is to compute their optimal global alignment score. The definition of *global alignment of forests* follows Jiang et al. [11]. Basically, a global alignment $\mathcal{A}$ of F and G is obtained by first inserting nodes labeled with spaces '-' into F and G such that the two resulting ordered forests F' and G' have the same structure, and then overlaying them. Here is the formal definition of the global alignment.

Definition 3 (Global Forest Alignment). *[8] Let F, G be two Σ-labeled forests. A $(\Sigma \cup \{-\})^2$-labeled forest $\mathcal{A}$ is called a global alignment of F and G if and only if $F = \pi(A_1)$ and $G = \pi(A_2)$, where A_1 and A_2 are left and right*

projections of $\mathcal{A}$ and $\pi(A_i)$ is a forest formed by successive deleting nodes labeled with '-' from A_i for $i = 1, 2$.

For every pair of labels $(u, v) \in (\Sigma \cup \{-\})^2$, we define a score $\sigma(u, v)$. The score of an alignment $\mathcal{A}$ is the sum of the scores of all pairs in the nodes of $\mathcal{A}$, that is, $\Sigma_{(u,v)\in\mathcal{A}}\sigma(u, v)$. The similarity $sim(F, G)$ of F and G is the score of optimal global alignment of F and G.

The local forest alignment problems focus on finding two local subforests of orginal forests such that they have optimal global alignment score. Here we define three problems of local forest alignment.

The *Local Gapped Subforest Alignment Problem* is to find two gapped sub-forests F' and G' of F and G, respectively, such that the global alignment score $sim(F', G')$ is maximized.

$$lgsf(F, G) = \max\{sim(F', G') \mid F' \in gsf(F), G' \in gsf(G)\}$$

The *Local β-Gapped Subforest Alignment Problem* is to find two β-gapped subforests F' and G' of F and G, respectively, maximizing the global alignment score $sim(F', G')$.

$$lgsf_\beta(F, G) = \max\{sim(F', G') \mid F' \in gsf_\beta(F), G' \in gsf_\beta(G)\}$$

The *Local Closed Subforest Alignment Problem* is to find two closed sub-forests F' and G' of F and G, respectively, such that the global alignment score $sim(F', G')$ is maximized.

$$lcsf(F, G) = \max\{sim(F[u..u'], G[v..v']) \mid u..u' \in S(F), v..v' \in S(G)\}$$

1.2   Previous Results

RNAs can be modeled as annotated sequences [5] or labeled ordered trees. For comparing annotated sequences, a number of results have been done on global edit distance and global alignment [1, 3, 5, 6, 7, 10, 12, 13]. The only known local alignment result is given in [2].

This paper focuses on labeled ordered trees comparison. For global tree edit distance, results include [15, 19, 20]. The first algorithm for *global* alignment of labeled ordered trees was proposed by Jiang, Wang, and Zhang [11]. Their al-gorithm computes the optimal global alignment between two labeled ordered trees T_1 and T_2 in $O(|T_1| \cdot |T_2| \cdot (\deg(T_1) + \deg(T_2))^2)$ time, where $|T_i|$ and $\deg(T_i)$ for $i \in \{1, 2\}$ denote the number of nodes in T_i and the degree of T_i, respectively. It was extended without affecting the asymptotic running time to the problem of optimally aligning two labeled ordered trees with gap penalties by Wang and Zhao [17]. In [17], Wang and Zhao also showed how to reduce the space complexity of the resulting algorithm from $O(|T_1| \cdot |T_2| \cdot (\deg(T_1) + \deg(T_2)))$ to $O(\log(|T_1|) \cdot |T_2| \cdot (\deg(T_1) + \deg(T_2)) \cdot \deg(T_1))$ at the expense of increasing the running time to $O(|T_1|^2 \cdot |T_2| \cdot (\deg(T_1) + \deg(T_2))^2)$. A modification to the algorithm of Jiang *et al.* which yields a lower running time for *similar* trees was given in [9].

As for computing *local alignments* of labeled ordered trees, Höchsmann *et al.* [8] gave an algorithm for *lcsf* (they termed it as the *local closed subforest similarity problem*). Backofen and Will [2] studied a problem that are called the *local sequence-structure alignment problem* (this problem is equivalent to our $lgsf_\beta$, as we prove in Section 4). The following table summarizes the complexities of the previously most efficient algorithms for *lcsf*, *lgsf*, and $lgsf_\beta$.

Problem	Time complexity	Space complexity
lcsf (See [8])	$O\big(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G) \cdot$ $(\deg(F) + \deg(G))\big)$	$O\big(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G)\big)$
lgsf	Not studied before	Not studied before
$lgsf_\beta$ (See [2])	$O\big(|F|^2 \cdot |G|^2 \cdot (|F| + |G|)\big)$	$O\big(|F| \cdot |G|\big)$

1.3 Our Results and Organization of the Paper

In Section 2, we introduce some additional notations and derive a number of recursive formulas which form the basis of our main dynamic programming-based algorithm for solving *lgsf*, presented in Section 2.4. Next, in Sections 3.1 and 3.2 we refine our algorithm for *lgsf* to solve $lgsf_\beta$ and *lcsf* even more efficiently. In Section 4, we prove that $lgsf_\beta$ is equivalent to the local sequence-structure problem considered in [2] and describe practical applications of *lgsf* related to finding structural motifs in RNA molecules. Finally, in Section 5, we discuss possible future extensions of our work.

The table below summarizes the complexities of our algorithms.

Problem	Time complexity	Space complexity
lcsf (Section 3.2)	$O\big(|F| \cdot |G| \cdot$ $(\deg(F) + \deg(G))^2\big)$	$O\big(|F| \cdot |G| \cdot (\deg(F) + \deg(G))\big)$
lgsf (Section 2.4)	$O\big(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G) \cdot$ $(\deg(F) + \deg(G))\big)$	$O\big(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G)\big)$
$lgsf_\beta$ (Section 3.1)	$O\big(|F| \cdot |G| \cdot$ $(\deg(F) + \deg(G))^2\big)$	$O\big(|F| \cdot |G| \cdot (\deg(F) + \deg(G))\big)$

2 The Local Gapped Subforest Alignment Problem

This section presents an algorithm to solve *lgsf*. To compute the similarity of two forests F and G through alignment, we consider the search space in a structurally recursive fashion. The following lemma acts as the base for our algorithm.

2.1 Base Lemma

Lemma 2. *[8] Let A be an alignment of two Σ-labeled forests F, G. If F or G is empty then the alignment A is an empty forest. If F and G are both non-empty*

forests where u and v are the roots of the leftmost trees of F and G, respectively, then the root a of the leftmost tree of A equals one of the following: (u, v), $(u, -)$ and $(-, v)$. We have three cases:

1. If $a = (u, v)$ then $A[a_L..a_R]$ is an alignment of $F[u_L..u_R]$ and $G[u_L..u_R]$, and $A[r(a)..e(a)]$ is an alignment of $F[r(u)..e(u)]$ and $G[r(v)..e(v)]$.
2. If $a = (u, -)$ then for some $v'' \in l(v)..e(v)$, $A[a_L..a_R]$ is an alignment of $F[u_L..u_R]$ and $G[v..v'']$, and $A[r(a)..e(a)]$ is an alignment of $F[r(u)..e(u)]$ and $G[r(v'')..e(v)]$.
3. If $a = (-, v)$ then for some $u'' \in l(u)..e(u)$, $A[a_L..a_R]$ is an alignment of $F[u..u'']$ and $G[v_L..v_R]$, and $A[r(a)..e(a)]$ is an alignment of $F[r(u'')..u']$ and $G[r(v)..e(v)]$.

2.2   Matrix Notations

From Lemma 1, we know that a gapped subforest at $F[u..u']$ can either be an α-gapped subforest at $F[u..u']$ or a β-gapped subforest at $F[u''..u']$, depending on its excluded interval in $u..u'$. Thus, given two Σ-labeled forests F and G, to find $lgsf(F, G)$, our algorithm computes 9 dynamic programming matrices depending on the types of the gapped subforests. For every $a, b \in \{\alpha, \beta, *\}$, we define the matrix D_{a-b} as follows:

Definition 4 (Matrix). *For every $a, b \in \{\alpha, \beta, *\}$, for every $u..u' \in S(F)$ and $v..v' \in S(G)$, $D_{a-b}[u..u'; v..v']$ is defined to be the maximum of all the global alignment scores of two forests F' and G', where $F' \in gsf_a(F[u..u'])$ and $G' \in gsf_b(G[v..v'])$. Precisely, we have*

$$D_{a-b}[u..u'; v..v'] = \max\{sim(F', G') | F' \in gsf_a(F[u..u']), G' \in gsf_b(G[v..v'])\}$$

2.3   Recursive Formulae

Given the above matrices, the local gapped subforest alignment score and the local β-gapped subforest alignment score of two forests F and G can be computed based on the following lemma.

Lemma 3. *Let F, G be two Σ-labeled forests. Then, we have:*

$$lgsf(F, G) = \max\{D_{*-*}[u..u'; v..v'] \mid u..u' \in S(F), v..v' \in S(G)\}$$

$$lgsf_\beta(F, G) = \max\{D_{\beta-\beta}[u..u'; v..v'] \mid u..u' \in S(F), v..v' \in S(G)\}.$$

The next step is to derive recursive formulae. First, the general matrix D_{*-*} is computed using the following lemma. The proof follows directly from Definitions 2 and 4 and together with Lemma 1, and is therefore omitted.

Lemma 4 (General Matrix).

$$D_{*-*}[u..u'; v..v'] = \max \begin{cases} D_{\alpha-\alpha}[u..u'; v..v'] \\ \max_{v'' \in v..r(v')} \{D_{*-\beta}[u..u'; v''..v']\} \\ \max_{u'' \in u..r(u')} \{D_{\beta-*}[u''..u'; v..v']\} \end{cases}$$

Also from Definitions 2 and 4 and Lemma 1, we can straightforwardly derive the formulae to compute the matrices $D_{\alpha-*}, D_{*-\alpha}, D_{\beta-*}, D_{*-\beta}$ as follows:

Lemma 5 (Special Matrices). *The recursive equations for* $D_{\alpha-*}, D_{*-\alpha}, D_{\beta-*},$ *and* $D_{*-\beta}$ *are:*

- $D_{\alpha-*}[u..u';v..v'] = \max\{D_{\alpha-\alpha}[u..u';v..v'], \max\limits_{v''\in v..r(v')}\{D_{\alpha-\beta}[u..u';v''..v']\}\}$
- $D_{*-\alpha}[u..u';v..v'] = \max\{D_{\alpha-\alpha}[u..u';v..v'], \max\limits_{u''\in u..r(u')}\{D_{\beta-\alpha}[u''..u';v..v']\}\}$
- $D_{\beta-*}[u..u';v..v'] = \max\{D_{\beta-\alpha}[u..u';v..v'], \max\limits_{v''\in v..r(v')}\{D_{\beta-\beta}[u..u';v''..v']\}\}$
- $D_{*-\beta}[u..u';v..v'] = \max\{D_{\alpha-\beta}[u..u';v..v'], \max\limits_{u''\in u..r(u')}\{D_{\beta-\beta}[u''..u';v..v']\}\}$

Now we proceed to the computations of $D_{\alpha-\alpha}, D_{\alpha-\beta},$ and $D_{\beta-\alpha}$. Because these formulae are more complicated, we provide a sketch of the proof.

Lemma 6 (Alpha-Alpha Matrix). $D_{\alpha-\alpha}[u..u';v..v'] =$

$$\max \begin{cases} \sigma(u,v) + D_{*-*}[u_L..u_R;v_L..v_R] + D_{*-*}[r(u)..u';r(v)..v'] \\ \sigma(u,-) + \max\limits_{v''\in l(v)..v'}\{D_{*-\beta}[u_L..u_R;v..v''] + D_{*-*}[r(u)..u';r(v'')..v']\} \\ \sigma(u,-) + \max\limits_{v''\in l(v)..v'}\{D_{*-\alpha}[u_L..u_R;v..v''] + D_{*-\beta}[r(u)..u';r(v'')..v']\} \\ \sigma(-,v) + \max\limits_{u''\in l(u)..u'}\{D_{\beta-*}[u..u'';v_L..v_R] + D_{*-*}[r(u'')..u';r(v)..v']\} \\ \sigma(-,v) + \max\limits_{u''\in l(u)..u'}\{D_{\alpha-*}[u..u'';v_L..v_R] + D_{\beta-*}[r(u'')..u';r(v)..v']\} \end{cases}$$

Proof. Let F' and G' be two α-gapped subforests at $F[u..u']$ and $G[v..v']$ such that their optimal global alignment A is optimal among those in $gsf_\alpha(F[u..u'])$ and $gsf_\alpha(G[v..v'])$. Let u^*, v^*, and a be the roots of the leftmost subtrees of F', G', and A respectively (i.e., $u^* = u$ and $v^* = v$, but when we refer to u^* and v^* we mean the roots in F' and G'). From Lemma 2, we consider 3 cases:

- Case 1: When $a = (u^*, v^*)$, by Lemma 2, $F'[u_L^*..u_R^*]$ is aligned with $G'[v_L^*..v_R^*]$, and $F'[r(u^*)..e(u^*)]$ is aligned with $G'[r(v^*)..e(v^*)]$.
  Since F' is an α-gapped subforest at $F[u..u']$, $F'[u_L^*..u_R^*]$ and $F'[r(u^*)..e(u^*)]$ are gapped subforests at $F[u_L..u_R]$ and $F[r(u)..u']$ respectively. Similarly, $G'[v_L^*..v_R^*]$ and $G'[r(v^*)..e(v^*)]$ are gapped subforests at $G[v_L..v_R]$ and $G[r(v)..v']$ respectively. Hence, the alignment score of A equals $\sigma(u,v) + A[u_L..u_R;v_L..v_R] + A[r(u)..u';r(v)..v']$.
- Case 2: When $a = (u^*, -)$, by Lemma 2, there exists some $v'' \in l(v^*)..e(v^*)$ such that $F'[u_L^*..u_R^*]$ is aligned with $G'[v^*..v'']$, and $F'[r(u^*)..e(u^*)]$ is aligned with $G'[r(v'')..e(v^*)]$.
  Since F' is an α-gapped subforest at $F[u..u']$, $F'[u_L^*..u_R^*]$ and $F'[r(u^*)..e(u^*)]$ are gapped subforests at $F[u_L..u_R]$ and $F[r(u)..u']$ respectively.
  Besides, by definition, since G' is an α-gapped subforest of $G[v..v']$, G' allows exclusion of $G[x..y]$ from $G[v..v']$ for at most one sibling interval $x..y \subseteq v..v'$. Depending on whether or not $x..y \subseteq v..v''$, we have two subcases.
  - Case 2a: $x..y \subseteq v..v''$. Then $G'[v^*..v'']$ is a β-gapped subforest at $G[v..v'']$, and $G'[r(v'')..e(v^*)]$ is a gapped subforest at $G[r(v'')..v']$. Hence, the alignment score of A equals $\sigma(u,-)+\max_{v''\in l(v)..v'}\{A_{*-\beta}[u_L..u_R;v..v'']+ A[r(u)..u';r(v'')..v']\}$.

- Case 2b: $x..y \not\subseteq v..v''$. Then $G'[v^*..v'']$ is an α-gapped subforest at $G[v..v'']$, and $G'[r(v'')..e(v^*)]$ is a β-gapped subforest at $G[r(v'')..v']$. Hence, the alignment score of A equals $\sigma(u,-) + \max_{v'' \in l(v)..v'} \{A_{*-\alpha}[u_L..u_R; v..v''] + A_{*-\beta}[r(u)..u'; r(v'')..v']\}$.
- Case 3: When $a = (-,v)$, the proof is symmetric to the proof for Case 2. From the above three cases, Lemma 6 thus follows.   □

In the same way as in the proof of Lemma 6, we can derive Lemmas 7 and 8 below for computing $D_{\alpha-\beta}[u..u', v..v']$, $D_{\beta-\alpha}[u..u', v..v']$, and $D_{\beta-\beta}[u..u', v..v']$.

Lemma 7 (Alpha-Beta matrix). $D_{\alpha-\beta}[u..u'; v..v'] =$

$$\max \begin{cases} \sigma(u,v) + D_{*-*}[u_L..u_R; v_L..v_R] + D_{*-\beta}[r(u)..u'; r(v)..v'] \\ \sigma(u,-) + \max_{v'' \in l(v)..v'} \{D_{*-\beta}[u_L..u_R; v..v''] + D_{*-\beta}[r(u)..u'; r(v'')..v']\} \\ \sigma(-,v) + \max_{u'' \in l(u)..u'} \{D_{\beta-*}[u..u''; v_L..v_R] + D_{*-\beta}[r(u'')..u'; r(v)..v']\} \\ \sigma(-,v) + \max_{u'' \in l(u)..u'} \{D_{\alpha-*}[u..u''; v_L..v_R] + D_{\beta-\beta}[r(u'')..u'; r(v)..v']\} \end{cases}$$

and analogously for $D_{\beta-\alpha}[u..u'; v..v']$.

Lemma 8 (Beta-Beta matrix). $D_{\beta-\beta}[u..u'; v..v'] =$

$$\max \begin{cases} \sigma(u,v) + D_{*-*}[u_L..u_R; v_L..v_R] + D_{\beta-\beta}[r(u)..u'; r(v)..v'] \\ \sigma(u,-) + \max_{v'' \in l(v)..v'} \{D_{*-\beta}[u_L..u_R; v..v''] + D_{\beta-\beta}[r(u)..u'; r(v'')..v']\} \\ \sigma(-,v) + \max_{u'' \in l(u)..u'} \{D_{\beta-*}[u..u''; v_L..v_R] + D_{\beta-\beta}[r(u'')..u'; r(v)..v']\} \end{cases}$$

2.4   The Main Algorithm and Its Complexity

From the recursive formulae, we can derive the algorithm to compute $lgsf(F,G)$ straightforwardly. Basically, the algorithm computes $D_{a-b}[u..u'; v..v']$ for all $a,b \in \{\alpha, \beta, *\}$, $u..u' \in S(F)$, and $v..v' \in S(G)$. With the 9 matrices D_{a-b} for $a,b \in \{*, \alpha, \beta\}$, the optimal alignment can be easily found using a simple traceback. The complexity will remain the same. The lemma below states its time and space complexity.

Lemma 9. *The algorithm Compute-lgsf runs in $O(|F||G|deg(F)deg(G)$ $(deg(F) + deg(G)))$ time and $O(|F||G|deg(F)deg(G))$ space.*

3   Algorithms for Two Variants of the Local Gapped Subforest Alignment Problem

3.1   Local β-Gapped Subforest Alignment Problem

To improve the efficiency of the algorithm for the Local β-Gapped Subforest Alignment Problem, we construct a new matrix $B[u,v]$ as follows:

$$B[u; v] = \max\{D_{\beta-\beta}[u..u'; v..v'] \mid u..u' \in S(F), v..v' \in S(G)\}$$

From the definitions of $D_{\beta-\beta}[u..u'; v..v']$ and of matrix B, we have:

Lemma 10. $B[u; v] = \max\{sim(F', G') \mid F' \in gsf_\beta[u..u'], G' \in gsf_\beta[v..v']\}.$

Together with Lemma 3, we also have:

Lemma 11. $lgsf_\beta(F, G) = \max\{B[u; v] \mid u \in F, v \in G\}.$

The computation of the matrix B is formulated as:

Lemma 12. $B[u; v] =$

$$\max \begin{cases} \sigma(u, v) + D_{*-*}[u_L..u_R; v_L..v_R] + \max\{B[r(u); r(v)], 0\} \\ \sigma(u, -) + \max_{v'' \in l(v)..v'} \{D_{*-\beta}[u_L..u_R; v..v''] + \max\{B[r(u); r(v'')], 0\}\} \\ \sigma(-, v) + \max_{u'' \in l(u)..u'} \{D_{\beta-*}[u..u''; v_L..v_R] + \max\{B[r(u''); r(v)], 0\}\} \end{cases}$$

To compute $lgsf_\beta(F, G)$, we need to compute all the entries in $B[u, v]$. In this situation, we observe that it is unnecessary to compute all of the entries in $D_{a-b}[u..u'; v..v']$ for all $a, b \in \{\alpha, \beta, *\}$, $u..u' \in S(F)$ and $v..v' \in S(G)$. We just need to fill in, for all $a, b \in \{\alpha, \beta, *\}$, the entries of the form $D_{a-b}[u..u'; v..v']$ where $u' = e(u)$ or $v' = e(v)$. Thus, for all $a, b \in \{\alpha, \beta, *\}$, each matrix $D_{a-b}[u..u'; v..v']$ is divided into two sub-matrices: $D'_{a-b}[u; v..v']$ and $D''_{a-b}[u..u'; v]$, where $D'_{a-b}[u; v..v'] = D_{a-b}[u..e(u); v..v']$ and $D''_{a-b}[u..u'; v] = D_{a-b}[u..u'; v..e(v)]$.

Lemma 13. *The* Local β-Gapped Subforest Alignment Problem *can be solved in* $O(|F||G|(deg(F) + deg(G))^2)$ *time and* $O(|F||G|(deg(F) + deg(G)))$ *space.*

3.2   Local Closed Subforest Alignment Problem

The *Local Closed Subforest Alignment Problem* has been proposed and solved by [8] in $O(|F||G|deg(F)deg(G)(deg(F)+deg(G)))$ time and $O(|F||G|deg(F)deg(G))$ space. We propose a faster and more space-saving algorithm for this problem. Using the same technique as in Section 3.1, we construct a new matrix $B[u..v]$:

$$B[u; v] = \max\{sim(F[u..u'], G[v..v']) \mid u..u' \in S(F), v..v' \in S(G)\}$$

Therefore, we have the following lemma:

Lemma 14. $csf(F, G) = \max\{B[u; v] \mid u \in F, v \in G\}$

The computation of the matrix B is formulated as following:

Lemma 15. $B[u; v] =$

$$\max \begin{cases} \sigma(u, v) + GD[u_L..u_R; v_L..v_R] + \max\{B[r(u); r(v)], 0\} \\ \sigma(u, -) + \max_{v'' \in l(v)..v'} \{GD[u_L..u_R; v..v''] + \max\{B[r(u); r(v'')], 0\}\} \\ \sigma(-, v) + \max_{u'' \in l(u)..u'} \{GD[u..u''; v_L..v_R] + \max\{B[r(u''); r(v)], 0\}\} \end{cases}$$

where $GD[u..u'; v..v']$ is the optimal global alignment score of two closed subforests $F[u..u']$ and $G[v..v']$. But we just need to fill in the entries $GD[u..u'; v..v']$ where either $u' = e(u)$ or $v' = e(v)$. These entries in the $GD[u..u'; v..v']$ matrix can be computed by [11] in $O(|F||G|(deg(F) + deg(G))^2)$ time. Therefore we have the following analysis:

Lemma 16. *The* Local Closed Subforest Alignment Problem *can be solved in* $O(|F||G|(deg(F) + deg(G))^2)$ *time and* $O(|F||G|(deg(F) + deg(G)))$ *space.*

4   An Application to Find Local RNA Sequence-Structure Motifs

An RNA secondary structure is a combination of an RNA sequence and a set of base pairings called *arcs* binded together by hydrogen bonds. An RNA secondary structure can be represented as an *annotated sequence* [1, 5, 6, 7, 10, 13], which is a tuple (S, P) where S is a sequence of bases $s_1 s_2 \ldots s_n$ and P is a set of arcs formed by the position pairs (i, j)'s. The majority of RNA secondary structures has the characteristic that no two arcs cross (i.e., there exists no two arcs (i, j) and (i', j') such that $i < i' < j < j'$). With this condition, it is also common to represent an RNA secondary structure as a labeled ordered forest F, where each node in F corresponds to a free base in the sequence (i.e., a base that does not pair with any other base) or an arc in the secondary structure such that:

- For any two nodes u and v in F, u is the parent of v iff u corresponds to the parent arc of v (i.e., the smallest arc enclosing the base/arc that v corresponds to).
- For any two nodes u and v in F, u is a right sibling of v iff u corresponds to a base/arc that lies to the right of the base/arc of v.

Biologists have noticed that two RNAs sharing similar local substructure (which is referred as motif) have similar functions. This observation motivates the problem of computing the maximum common local substructure of two RNAs. A number of ways to represent the local substructures of an RNA have been proposed. Among them, the local sequence-structure motif [8] is the most general one, because it can represent local RNA substructures whose bases are connected when represented as a motif graph. For example, the local sequence-structure motif can represent the putative SECIS-motif [18]. Formally, given an RNA represented by (S, P), (S', P') is called a local sequence-structure motif [8] of (S, P) if and only if:

- S' is a subsequence of S, and P' is a subset of P induced from S';
- S' is arc-complete for (S, P) (i.e., for every $(i, j) \in P$, either $i, j \in S'$ or $i, j \notin S'$); and
- any interval $s_k \ldots s_{k'}$ is called an exclusion of S' if $s_k \ldots s_l \notin S'$ and $s_{k-1}, s_{l+1} \in S'$. Every exclusion $s_k \ldots s_l$ of S' has an immediate successor, which is an arc $(i, j) \in P'$ such that $i < k < l < j$ and $j - i$ is minimized. Also, no two exclusions of S' share the same immediate successor.

Let F and F' be the forest representations of (S, P) and (S', P'). The lemma below shows that (S', P') is a local sequence-structure motif of (S, P) iff F' is a β-gapped subforest of F.

Lemma 17. *Consider two annotated sequences (S, P) and (S', P'). Let F and F' be the respective forest representations of them. We have (S', P') is a sequence-structure motif of (S, P) if and only if F' is a β-gapped subforest of F.*

Proof. ($\Rightarrow$) Suppose (S', P') is a local sequence-structure motif of (S, P). First, every exclusion in S' should be arc-complete. Otherwise, there exists some arc connecting a base in S' and a base in the exclusion, thus it contradicts to the fact that S' is arc-complete. Hence, every exclusion is arc-complete and corresponds to a closed subforest $F[x_i..y_i]$. In other words, F' is formed by excluding some $F[x_i..y_i]$'s from some $F[u..v]$. Then, since no exclusion has the same immediate successor, we conclude that all $F[x_i..y_i]$'s do not share the same parent. Thus, F' should be a gapped subforest of F. Finally, as every exclusion has an immediate successor, there is no sibling interval $x_i..y_i$ that can be excluded at the root level of F', because otherwise $F[x_i..y_i]$ would corresponds to an exclusion in S' with no immediate successor. Hence, F' is a β-gapped subforest of F'.

($\Leftarrow$) Suppose F' is a β-gapped subforest of F. F' is formed by excluding some closed subforests $F[x_i..y_i]$'s from a closed subforest $F[u..v]$. Hence, the corresponding S' is arc-complete. Since no sibling interval can be excluded at the root level of F', an excluded sibling interval (if one exists) has a parent in F'. Thus the corresponding exclusion has an immediate successor. Lastly, since there is no $x_i..y_i$'s sharing the same parent, all exclusions have different immediate successors. Hence, (S', P') is a local sequence-structure motif of (S, P). $\square$

Given Lemma 13 and the following lemma, the maximum local sequence-structure motif of two annotated sequences (S_1, P_1) and (S_2, P_2) can be computed in $O(|F||G|(deg(F) + deg(G))^2)$ time and $O(|F||G|deg(F)deg(G))$ space.

Lemma 18. *Given two annotated sequences (S_1, P_1) and (S_2, P_2), let F_1 and F_2 be their forest representations. The optimal local sequence-structure motif alignment of (S_1, P_1) and (S_2, P_2) is equivalent to the optimal restricted gapped subforest alignment of F_1 and F_2.*

5   Concluding Remarks

Our proposed problem and solutions motivate future development in local alignment of labeled ordered forests. One of the challenges is to find even more efficient algorithms for the local forest alignment problems. Any improvement can have a vital impact in RNA comparison and structure prediction applications. Another difficult task is to further generalize the local gapped subforest alignment problem by allowing exclusions of more than one closed subforest sharing the same parent. Lastly, new alignment models could be proposed to give more effective and efficient algorithms for RNA comparison and structure prediction problems.

References

1. J. Alber, J. Gramm, J. Guo, and R. Niedermeier. Computing the similarity of two sequences with nested arc annotations. *Theoretical Computer Science*, 312(2–3):337–358, 2004.
2. R. Backofen and S. Will. Local sequence-structure motifs in RNA. *Journal of Bioinformatics and Computational Biology*, to appear.

3. V. Bafna, S. Muthukrishnan, and R. Ravi. Computing similarity between RNA strings. In *Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching* (CPM 95), volume 937 of *LNCS*, pages 1–16. Springer, 1995.

4. L. J. Collins, V. Moulton, and D. Penny. Use of RNA secondary structure for studying the evolution of RNase P and RNase MRP. *Journal of Molecular Evolution*, 51(3):194–204, 2000.

5. P. A. Evans. *Algorithms and Complexity for Annotated Sequence Analysis.* PhD thesis, University of Victoria, Canada, 1999.

6. P. A. Evans. Finding common subsequences with arcs and pseudoknots. In *Proceedings of the 10th Annual Symposium on Combinatorial Pattern Matching* (CPM 99), volume 1645 of *LNCS*, pages 270–280. Springer, 1999.

7. J. Gramm, J. Guo, and R. Niedermeier. Pattern matching for arc-annotated sequences. In *Proceedings of the 22nd Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS 2002), volume 2556 of *LNCS*, pages 182–193. Springer, 2002.

8. M. Höchsmann, T. Töller, R. Giegerich, and S. Kurtz. Local similarity in RNA secondary structures. In *Proceedings of the Computational Systems Bioinformatics Conference* (CSB2003), pages 159–168, 2003.

9. J. Jansson and A. Lingas. A fast algorithm for optimal alignment between similar ordered trees. *Fundamenta Informaticae*, 56(1–2):105–120, 2003.

10. T. Jiang, G. Lin, B. Ma, and K. Zhang. The longest common subsequence problem for arc-annotated sequences. *Journal of Discrete Algorithms*, 2(2):257–270, 2004.

11. T. Jiang, L. Wang, and K. Zhang. Alignment of trees – an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148, 1995.

12. H.-P. Lenhof, K. Reinert, and M. Vingron. A polyhedral approach to RNA sequence structure alignment. In *Proceedings of the Annual International Conference on Computational Biology* (RECOMB 1998), pages 153–162, 1998.

13. G. Lin, Z.-Z. Chen, T. Jiang, and J. Wen. The longest common subsequence problem for sequences with nested arc annotations. *Journal of Computer and System Sciences*, 65(3):465–480, 2002.

14. B. A. Shapiro and K. Zhang. Comparing multiple RNA secondary structures using tree comparisons. *Computer Applications in the Biosciences*, 6(4):309–318, 1990.

15. K.-C. Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433, 1979.

16. J. T. L. Wang and K. Zhang. Identifying consensus of trees through alignment. *Information Sciences*, 126(1–4):165–189, 2000.

17. L. Wang and J. Zhao. Parametric alignment of ordered trees. *Bioinformatics*, 19(17):2237–2245, 2003.

18. R. Wilting, S. Schorling, B. C. Persson, and A. Böck. Selenoprotein synthesis in archaea: Identification of an mRNA element of Methanococcus jannaschii probably directing selenocysteine insertion. *Journal of Molecular Biology*, 266(4):637–641, 1997.

19. K. Zhang and T. Jiang. Some MAX SNP-hard results concerning unordered labeled trees. *Information Processing Letters*, 49(5):249–254, 1994.

20. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.