Local Gapped Subforest Alignment and Its Application in Finding RNA Structural Motifs[1]

JESPER JANSSON,[2] NGO TRUNG HIEU,[2] and WING-KIN SUNG[2]

ABSTRACT

RNA molecules whose secondary structures contain similar substructures often have similar functions. Therefore, an important task in the study of RNA is to develop methods for discovering substructures in RNA secondary structures that occur frequently (also referred to as *motifs*). In this paper, we consider the problem of computing an optimal local alignment of two given labeled ordered forests F_1 and F_2. This problem asks for a substructure of F_1 and a substructure of F_2 that exhibit a high similarity. Since an RNA molecule's secondary structure can be represented as a labeled ordered forest, the problem we study has a direct application to finding potential motifs. We generalize the previously studied concept of a closed subforest to a *gapped subforest* and present the first algorithm for computing the optimal local gapped subforest alignment of F_1 and F_2. We also show that our technique can improve the time and space complexity of the previously most efficient algorithm for optimal local closed subforest alignment. 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*) and modify our main algorithm to obtain a much faster algorithm for *lssa* than the one previously proposed. An implementation of our algorithm is available at *www.comp.nus.edu.sg/~bioinfo/LGSFAligner/*. Its running time is significantly faster than the original *lssa* program.

Key words: local forest alignment, gapped subforest, RNA secondary structure, local sequence-structure motif, dynamic programming.

1. INTRODUCTION

MANY AREAS OF COMPUTER SCIENCE USE labeled ordered trees to represent hierarchically structured information. It is often necessary to measure the similarity between two or more such trees or to identify parts of the trees that are similar, e.g., in software construction and maintenance applications (Baxter *et al.*, 1998; Tai, 1979; Yang, 1991) or to find structural changes between different versions of

---

electronic documents for information management and data archiving purposes (Chawathe *et al.*, 1996; Wang *et al.*, 1997). In computational molecular biology, labeled ordered trees can represent RNA molecules' secondary structures (Höchsmann *et al.*, 2003; Le *et al.*, 1989; Shapiro and Zhang, 1990). By measuring and comparing the similarity of secondary structure trees, researchers who investigate structural or evolutionary relationships between RNA molecules may obtain additional clues (Collins *et al.*, 2000; Höchsmann *et al.*, 2003; Shapiro and Zhang, 1990).[1] Automated methods for finding shared substructures in RNA secondary structure trees are also of great value; an important task in the study of RNA is to develop tools 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 or when predicting the secondary structure of a newly found RNA molecule (see, e.g., Backofen and Will [to appear], Höchsmann *et al.* [2003], and Le *et al.* [1989]).

Two ways to measure the overall similarity between two labeled ordered trees are by using the *tree edit distance* (Tai, 1979) or *alignments of trees* (Jiang *et al.*, 1995). The problem of computing the optimal alignment of two trees can be viewed as a special case of the tree edit distance problem (Jiang *et al.*, 1995); 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 (Jiang *et al.*, 1995; Zhang and Jiang, 1994) and for ordered trees whose degrees are much smaller than their depths (Jiang *et al.*, 1995; Zhang and Shasha, 1989). However, alignments between trees as defined in Jiang *et al.* (1995) consider similarities on the *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. Recently, Höchsmann *et al.* (2003) and Wang and Zhang (2000) 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 two input trees 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 by Höchsmann *et al.* (2003). 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 by Höchsmann *et al.* (2003) 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 by Höchsmann *et al.* (2003) and give an efficient algorithm for solving it. We also prove that a special case of *lgsf* which we refer to as *lgsf$_\beta$* can be used to express the *local sequence-structure alignment problem* (*lssa*) presented by Backofen and Will (to appear), implying that a slightly modified version of our algorithm for *lgsf* can be applied to solve *lssa* much faster than the algorithm given by Backofen and Will (to appear) at the cost of a minor increase in space complexity. Finally, we implement our improved algorithms for local subforest alignment problems and apply them to find structural motifs of RNA secondary structures. We apply a space-saving tabulation technique proposed by Höchsmann *et al.* (2003) to reduce the amount of memory used by the program and perform experiments on real RNA secondary structures to investigate the results of our algorithms in reality.

2. PROBLEM DEFINITIONS AND THE MOTIVATION

2.1. Preliminaries

We first introduce some basic terminology and notation used throughout this paper.

Labeled forests, closed subforests, and gapped subforests. Let Σ be a finite set of symbols. A rooted, ordered forest whose nodes are labeled by symbols in Σ is called a Σ-*labeled forest* (or in short, *forest*).

---

[1]This seems especially useful when the strings representing the primary structures of the molecules cannot be reliably aligned, as in the case of pRNA and mrpRNA studied by Collins *et al.* (2000). In general, if the RNA molecules to be compared have evolved for a long time, methods that also take into account secondary structure information are potentially more accurate than those that only rely upon the primary structure (Notredame *et al.*, 1997).

For any forest F, $|F|$ represents the number of nodes in F. To simplify the presentation below, from here on we will assume that the roots of the trees in any given forest share an imaginary (and arbitrarily labeled) parent node. The *degree of F*, denoted by $\deg(F)$, is the maximum number of children over all nodes in F and the imaginary parent node of the roots of the trees in F.

Let u and v be nodes in a forest F. Nodes u and v are called *siblings* if and only if they have the same parent node. We let $l(u)$ and $r(u)$ denote the sibling immediately to the left and to the right of u, respectively, and let $e(u)$ denote the right-most sibling of u. For technical reasons, we use the convention that $r(l(u)) = u$ even when $l(u)$ is undefined (e.g., if u has no left sibling). Furthermore, we let u_L and u_R denote the left-most and the right-most child of u. If u has no children, then we set $u_L = \emptyset$ and $u_R = \emptyset$.

Define the *sibling interval $u..v$* as follows. If $u = \emptyset$ or $v = \emptyset$ then $u..v = \emptyset$. Similarly, if u and v have different parents or if u is a right sibling of v, then $u..v = \emptyset$. Otherwise, $u..v$ is the set containing u, v, and all their siblings which are to the right of u and to the left of v. Denote by $\mathcal{S}(F)$ the set of all sibling intervals of the forest F, i.e., $\mathcal{S}(F) = \{u..v \mid u \in F, v \in F\}$. Observe that $\emptyset \in \mathcal{S}(F)$. Also observe that the set $\mathcal{S}(F)$ contains $O(|F| \cdot \deg(F))$ elements since there are at most $\deg(F)$ nonempty sibling intervals of the form $u..v$ for each node u in F. Finally, let $F[u..v]$ be the (possibly empty) forest consisting of all subtrees of F rooted at the nodes in $u..v$.

Next, we define two types of subforests: closed subforests (originally introduced by Höchsmann *et al.* [2003]) and gapped subforests. We need to distinguish two particular kinds of gapped subforests that we call α- and β-gapped subforests.

Definition 1 (closed subforest). *Let F and F' be two forests. F' is called a* closed subforest *of F if there exists two nodes u, v in F (possibly with $u = v$) such that $F' = F[u..v]$. When $u..v$ is not empty, the parent node of F' is defined to be the common parent node of u and v in F.*

Definition 2 (gapped subforest). *Let F and F' be two forests. F' is called a* gapped subforest *of F if there exists two nodes u, v in F (possibly with $u = v$) such that F' can be obtained by removing from $F[u..v]$ a set C of closed subforests where no two closed subforests belonging to C have the same parent node. In this case, F' is also called a* gapped subforest *at $F[u..v]$.*

F' is called an α-gapped subforest at $F[u..v]$ if F' is a gapped subforest at $F[u..v]$ and F' can be obtained from $F[u..v]$ without removing any closed subforest of the form $F[u..y]$ with $u..y \subseteq u..v$.

Lastly, F' is called a β-gapped subforest at $F[u..v]$ if F' is a gapped subforest at $F[u..v]$ and F' can be obtained from $F[u..v]$ without removing any closed subforest of the form $F[x..y]$ with $x..y \subseteq u..v$.

See Fig. 1 for some examples. For any closed subforest $F[u..v]$ of a forest F, we let $gsf(F[u..v])$ (also written as $gsf_*(F[u..v])$) denote the set of all gapped subforests at $F[u..v]$. Similarly, $gsf_\alpha(F[u..v])$ is the set of all α-gapped subforests at $F[u..v]$, and $gsf_\beta(F[u..v])$ is the set of all β-gapped subforests at $F[u..v]$. The next lemma relates gapped subforests, α-gapped subforests, and β-gapped subforests.

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

Proof. Suppose F' is a gapped subforest at $F[u..u']$. If F' is nonempty and F' contains the node u, then F' is an α-gapped subforest at $F[u..u']$. If F' is nonempty and F' does not contain u, then since F' is a gapped subforest, F' must be a β-gapped subforest at $F[r(u'')..u']$ for some $u'' \in u..l(u')$. Finally, if F' is the empty forest, then (by definition) F' is a β-gapped subforest at $F[r(u')..u']$. Conversely, every α-gapped subforest at $F[u..u']$ and every β-gapped subforest at $F[r(u'')..u']$ where $u'' \in u..u'$ must be a gapped subforest at $F[u..u']$. Thus, Lemma 1 follows. ■

Global alignment of two forests. Given two Σ-labeled forests F and G, one way to measure their similarity is by computing the score of an optimal global alignment between them, as defined below. Global alignment of forests was first considered by Jiang *et al.* (1995).
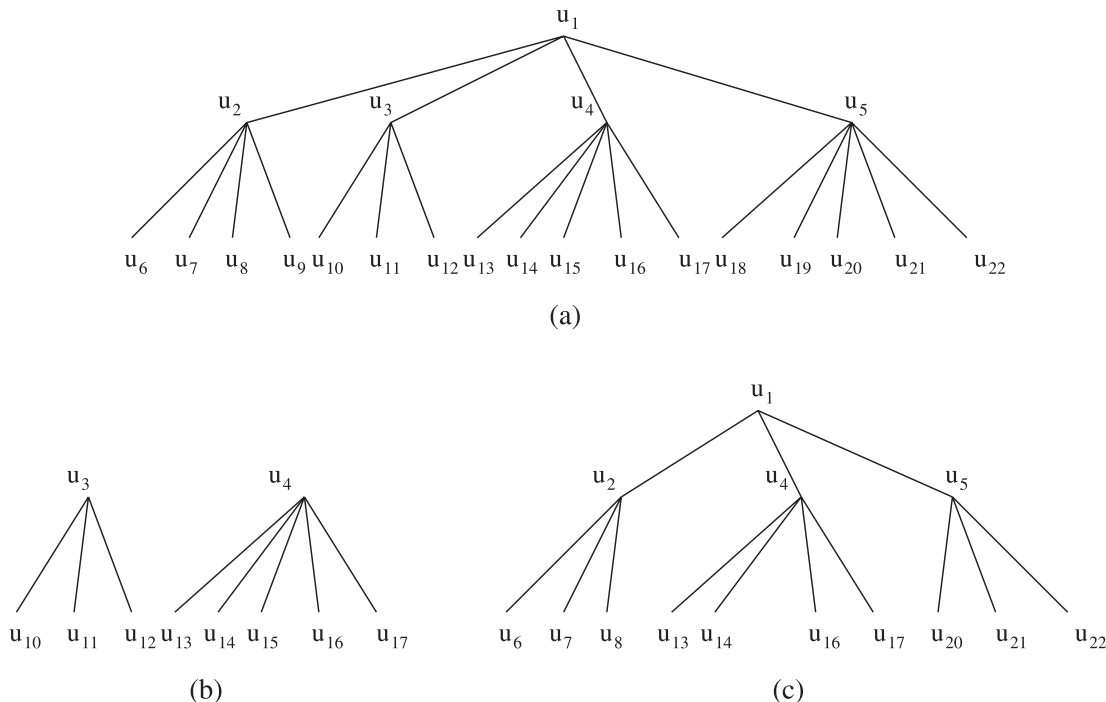
FIG. 1. (**a**) An example of a tree/forest F; (**b**) the closed subforest $F[u_3..u_4]$; (**c**) a gapped subforest at $F[u_1..u_1]$ formed by excluding the closed subforests $F[u_3..u_3]$, $F[u_9..u_9]$, $F[u_{15}..u_{15}]$, and $F[u_{18}..u_{19}]$. Note that the forest in (b) is an α-gapped subforest at $F[u_3..u_5]$ since it excludes only $F[u_5..u_5]$; however, it is not a β-gapped subforest at $F[u_3..u_5]$. Also note that (b) and (c) are β-gapped subforests at $F[u_3..u_4]$ and $F[u_1..u_1]$, respectively.

Let "$-$" be a special symbol that does not belong to Σ. An *insert operation* on a $(\Sigma \cup \{-\})$-labeled forest F adds a new node u labeled by "$-$," in such a way that u becomes the parent of a consecutive subsequence of children[2] of an existing node v and u becomes a child of v (here, we allow v to be the imaginary parent node shared by all the roots of the trees in F).

Let F and G be two Σ-labeled forests. A *global alignment between F and G* is any $(\Sigma \cup \{-\}) \times (\Sigma \cup \{-\})$-labeled forest that can be obtained by first performing insert operations on F and G so that the two resulting forests F' and G' are isomorphic when labels are ignored and then overlaying F' on G'. In addition, it is required that no node of the alignment corresponds to two nodes $x \in F'$ and $y \in G'$ which are both labeled by $-$. An example of a global alignment of two forests (in fact, two trees) is shown in Fig. 2. The *score* of an alignment is the sum of the scores of all pairs of aligned nodes, where the score of a pair of nodes is determined by a prespecified function μ defined on $(\Sigma \cup \{-\}) \times (\Sigma \cup \{-\})$. In the following, whenever we write $\mu(u, -)$, etc., where u is a node, we mean μ applied to the symbol that labels node u.

An *optimal global alignment* between a pair of Σ-labeled forests F and G is a global alignment between them achieving the highest possible score.[3] This score is also referred to as the *similarity of F and G* and is denoted by $sim(F, G)$.

2.2. Problem definitions

In general, the goal in a *local* forest alignment problem is to identify two subforests F' and G' of two given Σ-labeled forests F and G such that F' and G' conform to some specified structural requirements and the value of their optimal global alignment score $sim(F', G')$ is maximized. Here we define three closely related local forest alignment problems which are studied in this paper.

---

[2] Observe that a consecutive subsequence can consist of zero elements.

[3] Jiang *et al.* (1995) defined an optimal alignment as one with the *lowest* possible score.
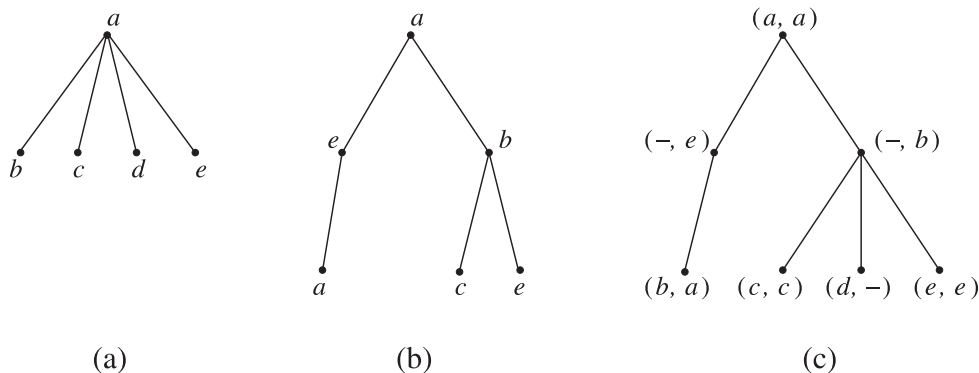
FIG. 2. Let $\Sigma = \{a, b, c, d, e\}$, and define the scoring function μ as follows: for every $x, y \in \Sigma$ with $x \neq y$, let $\mu(x, x) = 3$, $\mu(x, y) = -1$, and $\mu(x, -) = \mu(-, x) = -2$. Then the score of the global alignment in (c) between the two Σ-labeled forests shown in (a) and (b) is 2.

1. *The local closed subforest alignment problem* (*lcsf*): Find two closed subforests F' and G' of F and G, respectively, maximizing $sim(F', G')$. We have

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

2. *The local gapped subforest alignment problem* (*lgsf*): Find two gapped subforests F' and G' of F and G, respectively, maximizing $sim(F', G')$. We have

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

3. *The local β-gapped subforest alignment problem* (*lgsf$_\beta$*): Find two β-gapped subforests F' and G' of F and G, respectively, maximizing $sim(F', G')$. We have

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

2.3. Previous results

The first algorithm for optimal *global alignment* of two given labeled ordered trees was proposed by Jiang *et al.* (1995). Their algorithm computes an 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. It was subsequently extended without affecting the asymptotic running time to the problem of computing an optimal global alignment of two labeled ordered trees with gap penalties by Wang and Zhao (2003). 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 $O(\log(|T_1|) \cdot |T_2| \cdot (\deg(T_1) + \deg(T_2)) \cdot \deg(T_1))$ by 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 by Jansson and Lingas (2003). For some known results on computing the global *tree edit distance* between two labeled ordered trees, see Jiang *et al.* (1995), Tai (1979), Zhang and Jiang (1994), and Zhang and Shasha (1989).

As for computing optimal *local alignments* of labeled ordered trees, Höchsmann *et al.* (2003) gave an algorithm for *lcsf* (they termed it *the local closed subforest similarity problem*) running in $O(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G) \cdot (\deg(F) + \deg(G)))$ time and $O(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G))$ space. Backofen and Will (to appear) studied a problem which they called *the local sequence-structure alignment problem* (this problem is equivalent to our *lgsf$_\beta$*, as will be shown in Section 3) and gave an algorithm for solving it that runs in $O(|F|^2 \cdot |G|^2 \cdot (|F| + |G|))$ time and $O(|F| \cdot |G|)$ space.

A problem related to local alignments of labeled ordered trees is known as *the similar consensus problem* (Wang and Zhang, 2000). It asks for connected subgraphs T_1' and T_2' of two input labeled ordered trees T_1

TABLE 1. TIME AND SPACE COMPLEXITIES FOR SOME OF THE PREVIOUSLY MOST EFFICIENT
ALGORITHMS FOR *lcsf*, *lgsf*, AND *lgsf$_\beta$*

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

[a]See Höchsmann *et al.* (2003).
[b]See Backofen and Will (to appear).

and T_2 maximizing the optimal global alignment score between T_1' and T_2' such that T_1' is within a specified distance d of T_2'. The similar consensus problem was shown to be solvable in $O(d^2 \cdot |T_1| \cdot |T_2| \cdot (\deg(T_1) + \deg(T_2))^2)$ time by Wang and Zhang (2000).

We remark that in this paper, we focus on labeled ordered forest comparison. However, the secondary structure of an RNA molecule can also be modeled as an *annotated sequence* (see, e.g., Evans [1999a]). In the literature, there exist a number of results for problems involving constructing optimal global/local alignments of or computing the edit distance of two given annotated sequences (Alber *et al.*, 2004; Backofen and Will, to appear; Bafna *et al.*, 1995; Evans, 1999a, 1999b; Gramm *et al.*, 2002; Jiang *et al.*, 2002, 2004; Lenhof *et al.*, 1998; Lin *et al.*, 2002). See also Section 3 for a further discussion on the relationship between β-gapped subforests and a special type of annotated sequence.

Table 1 summarizes the time and space complexities of the previously most efficient algorithms for *lcsf*, *lgsf*, and *lgsf$_\beta$*.

2.4. Our results and organization of the paper

In Section 3, we prove that a special case of *lgsf$_\beta$* is equivalent to the local sequence-structure problem considered by Backofen and Will (to appear) and describe practical applications of *lgsf* related to finding structural motifs in RNA molecules. In Section 4, we introduce some additional matrix notations and derive a number of recursive formulae which form the basis of our main dynamic programming-based algorithm for *lgsf*, presented in Section 4.4. Next, in Sections 5.1 and 5.2 we refine our algorithm for *lgsf* to solve *lgsf$_\beta$* and *lcsf* more efficiently. We continue in Section 6 with the description of the implementation and experimental results of the *LGSFAligner* program. In this program, we implement our improved algorithms for local subforest alignment problems and apply them to find local structural motifs of RNA secondary structures. Finally, in Section 7, we discuss possible future extensions of our work.

Table 2 summarizes the time and space complexities of our algorithms.

3. AN APPLICATION TO FINDING LOCAL RNA SEQUENCE-STRUCTURE MOTIFS

An *annotated sequence* is defined as a tuple (S, P), where S is a sequence $s_1, s_2, \ldots, s_n$ of symbols from a finite alphabet Σ and where P is a set of unordered pairs of positions in S referred to as *arcs*. The secondary structure of an RNA molecule can be described using an annotated sequence over the

TABLE 2. TIME AND SPACE COMPLEXITIES OF OUR ALGORITHMS

Problem	Time complexity	Space complexity
lcsf (Section 5.2)	$O\big(|F| \cdot |G| \cdot (\deg(F) + \deg(G))^2\big)$	$O\big(|F| \cdot |G| \cdot (\deg(F) + \deg(G))\big)$
lgsf (Section 4.4)	$O\big(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G) \cdot (\deg(F) + \deg(G))\big)$	$O\big(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G)\big)$
lgsf$_\beta$ (Section 5.1)	$O\big(|F| \cdot |G| \cdot (\deg(F) + \deg(G))^2\big)$	$O\big(|F| \cdot |G| \cdot (\deg(F) + \deg(G))\big)$

alphabet $\{A, C, G, U\}$ by representing each hydrogen bond between a base pair in the sequence with an arc (see, e.g., Backofen and Will [to appear], Evans [1999a, 1999b], Jiang *et al.* [2004], and Lin *et al.* [2002]). The majority of all RNA secondary structures share the characteristic that no two arcs cross (i.e., there exist no two arcs (i, j) and (i', j') such that $i < i' < j < j'$); an annotated sequence containing no pair of crossing arcs is called *nested*.

Researchers have noticed that RNA molecules sharing similar local substructures (referred to as motifs) often have similar functions. This observation motivates the problem of computing the maximum common local substructure of two RNA molecules. A number of ways to represent local substructures of an RNA molecule's secondary structure have been proposed. Among them, the *local sequence-structure motif* (Backofen and Will, to appear) is one of the most effective. For example, it can represent the putative SECIS-motif (Wilting *et al.*, 1997) whereas many other methods fail (Backofen and Will, to appear). Given an RNA secondary structure represented as a nested annotated sequence (S, P), the annotated sequence (S', P') is called a *local sequence-structure motif of* (S, P) if and only if the following three conditions hold:

- S' is a subsequence of S, and P' is the subset of P induced from the subsequence S'.
- S' is *arc-complete for* (S, P); i.e., for every $(i, j) \in P$, either $i, j \in S'$ or $i, j \notin S'$.
- Each exclusion of S' has an immediate successor, where an interval $s_k..s_l$ of S is called an *exclusion of S'* if $s_k, \ldots, s_l \notin S'$ but $s_{k-1}, s_{l+1} \in S'$, and where the *immediate successor* of an exclusion $s_k..s_l$ of S' is defined as the smallest arc $(i, j) \in P'$ with $i < k < l < j$. Furthermore, no two exclusions of S' have the same immediate successor.

It is also common to represent an RNA secondary structure as a labeled ordered forest F in which every leaf corresponds to a *free base* (i.e., a base that does not pair with any other base) in the sequence and every internal node corresponds to an *arc* (i.e., two paired bases and the bond between them) in the secondary structure, such that:

- for any two nodes u and v in F, v is a right sibling of u if and only if v corresponds to a free base or an arc that lies to the right of the free base or the arc that u corresponds to, and
- for any two nodes u and v in F, v is the parent of u if and only if v corresponds to the smallest arc enclosing the free base or the arc that u corresponds to. In other words, v corresponds to the immediate successor of the free base or the arc that u corresponds to.

See, e.g., Fig. 6 and Fig. 7 of Höchsmann *et al.* (2003) for an example of this representation.

The following lemma shows that the local sequence-structure motifs of any nested annotated sequence (S, P) and the β-gapped subforests of the forest representation for (S, P) are equivalent.

Lemma 2. *Let (S, P) and (S', P') be two nested annotated sequences, and let F and F' be their respective forest representations. Motif (S', P') is a local 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 note that each exclusion I of S' is also arc-complete for (S, P) (this is because if some arc in P connects a base in I and a base in S' then S' would not be arc-complete for (S, P), and if some arc in P connects a base in I and a base in some other exclusion J of S' then (S, P) would not be a nested annotated sequence; in both cases, a contradiction). Hence, each exclusion I of S' corresponds to a closed subforest $F[x_i..y_i]$ of F', i.e., F' is obtained by removing some set C of closed subforests from some closed subforest $F[u..v]$ of F. Next, since no two exclusions of S' have the same immediate successor, we conclude that all closed subforests in C have different parent nodes, so F' is a gapped subforest of F. Finally, every exclusion of S' must have an immediate successor, implying that no closed subforest can be removed at the root level in F'. Hence, F' is a β-gapped subforest of F'.

$(\Leftarrow)$ Suppose F' is a β-gapped subforest of F. Then F' is obtained by removing some set C of closed subforests from some closed subforest $F[u..v]$ of F. Each arc (i, j) in P, along with the two bases at i and j, is represented by an internal node in F, so there is no way to delete just one of i and j from S,

i.e., the S' corresponding to F' must be arc-complete for (S, P). Next, since F' is a β-gapped subforest of F, no closed subforest of $F[u..v]$ has been removed at the root level in F', which means that every closed subforest belonging to C has a parent node in F', so each corresponding exclusion of S' has an immediate successor in P'. Lastly, all exclusions of S' have different immediate successors because no two closed subforests in C have the same parent node. Hence, (S', P') is a local sequence-structure motif of (S, P). ∎

Theorem 1. *Let (S_1, P_1) and (S_2, P_2) be two nested annotated sequences, and let F_1 and F_2 be their respective forest representations. The optimal local sequence-structure motif alignment of (S_1, P_1) and (S_2, P_2) (defined by Backofen and Will [to appear]) is equivalent to the optimal local β-gapped subforest alignment of F_1 and F_2.*

Proof. For $i \in \{1, 2\}$, by Lemma 2, it holds that each local sequence-structure motif of (S_i, P_i) corresponds uniquely to a β-gapped subforest of F_i. ∎

4. THE LOCAL GAPPED SUBFOREST ALIGNMENT PROBLEM (*lgsf*)

This section presents an algorithm to solve the local gapped subforest alignment problem.

4.1. Base lemma

The following lemma acts as the base for our algorithm.

Lemma 3. *(Lemma 1 of Höchsmann et al. [2003]) Let F and G be two Σ-labeled forests and let A be a global alignment of F and G. If F is an empty forest, then A and G are isomorphic when labels are ignored and each node in A is labeled $(-, \ell)$, where ℓ is the label of the corresponding node in G, and analogously if G is empty. If both of F and G are nonempty forests and u and v are the roots of the left-most trees of F and G, respectively, then the root a of the left-most tree of A is labeled by either (x, y), $(x, -)$, or $(-, y)$, where x is the label of u and y is the label of v. There are three cases:*

1. *If a is labeled by (x, y), then $A[a_L..a_R]$ is an alignment of $F[u_L..u_R]$ and $G[v_L..v_R]$, and $A[r(a)..e(a)]$ is an alignment of $F[r(u)..e(u)]$ and $G[r(v)..e(v)]$.*
2. *If a is labeled by $(x, -)$, 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 is labeled by $(-, y)$, 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 $F[r(u'')..e(u)]$ and $G[r(v)..e(v)]$.*

4.2. Matrix notations

In order to compute $lgsf(F, G)$ for the two given Σ-labeled forests F and G, our algorithm uses dynamic programming to fill in nine matrices corresponding to different types of gapped subforests. For every $a, b \in \{\alpha, \beta, *\}$, we define one matrix D_{a-b} of size $|\mathcal{S}(F)| \cdot |\mathcal{S}(G)|$ as follows:

Definition 3 (D-Matrix). *For every $a, b \in \{\alpha, \beta, *\}$, $u..u' \in \mathcal{S}(F)$, and $v..v' \in \mathcal{S}(G)$, the matrix element $D_{a-b}[u..u'; v..v']$ is defined as the maximum of all global alignment scores of two subforests F' and G', where $F' \in gsf_a(F[u..u'])$ and $G' \in gsf_b(G[v..v'])$, i.e.,*

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

The next lemma shows that the values of *lgsf* and *lgsf*$_\beta$ for F and G are given by the maximum element in D_{*-*} and $D_{\beta-\beta}$, respectively. Thus, once the above matrices have been computed, *lgsf* as well as *lgsf*$_\beta$ for F and G can be obtained directly.

Lemma 4. *Let F and G be two Σ-labeled forests. Then we have*

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

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

Proof. By Definition 3 above, $D_{*-*}[u..u'; v..v'] = \max\{sim(F', G') \mid F' \in gsf(F[u..u']),\ G' \in gsf(G[v..v'])\}$. Since $lgsf(F, G)$ is equal to the maximum value of $sim(F', G')$ taken over all possible $F' \in gsf(F[u..u'])$, $G' \in gsf(G[v..v'])$, $u..u' \in \mathcal{S}(F)$, and $v..v' \in \mathcal{S}(G)\}$, we obtain the equality for $lgsf$. The equality for $lgsf_\beta$ can be proved in the same way. ∎

4.3. Recursive formulae

The next step is to derive recursive formulae for each of the nine types of matrices. It is straightforward to compute the matrix entries when at least one of the two sibling intervals $u..u'$ and $v..v'$ is empty, so below we assume that both $u..u'$ and $v..v'$ are nonempty. First of all, the general matrix D_{*-*} can be expressed using the following lemma. The proof follows directly from Definitions 2 and 3 together with Lemma 1.

Lemma 5 (general matrix).

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

Also using Definitions 2 and 3 and Lemma 1, it is simple to derive formulae for computing the matrices $D_{\alpha-*}$, $D_{*-\alpha}$, $D_{\beta-*}$, and $D_{*-\beta}$:

Lemma 6 (*-matrix). *The recursive equations for $D_{\alpha-*}$, $D_{*-\alpha}$, $D_{\beta-*}$, and $D_{*-\beta}$ follow.*

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

Now we proceed to $D_{\alpha-\alpha}$, $D_{\alpha-\beta}$, $D_{\beta-\alpha}$, and $D_{\beta-\beta}$. These formulae are based on Lemma 3 and are slightly more complicated than the previous ones, so we give a proof for the $D_{\alpha-\alpha}$-case and illustrate it using Fig. 3.

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

$$\max \begin{cases} \mu(u, v) + D_{*-*}[u_L..u_R; v_L..v_R] + D_{*-*}[r(u)..u'; r(v)..v'] \\ \mu(u, -) + \max\limits_{v'' \in l(v)..v'} \{D_{*-\beta}[u_L..u_R; v..v''] + D_{*-\alpha}[r(u)..u'; r(v'')..v']\} \\ \mu(u, -) + \max\limits_{v'' \in l(v)..v'} \{D_{*-\alpha}[u_L..u_R; v..v''] + D_{*-\beta}[r(u)..u'; r(v'')..v']\} \\ \mu(-, v) + \max\limits_{u'' \in l(u)..u'} \{D_{\beta-*}[u..u''; v_L..v_R] + D_{\alpha-*}[r(u'')..u'; r(v)..v']\} \\ \mu(-, v) + \max\limits_{u'' \in l(u)..u'} \{D_{\alpha-*}[u..u''; v_L..v_R] + D_{\beta-*}[r(u'')..u'; r(v)..v']\} \end{cases}$$
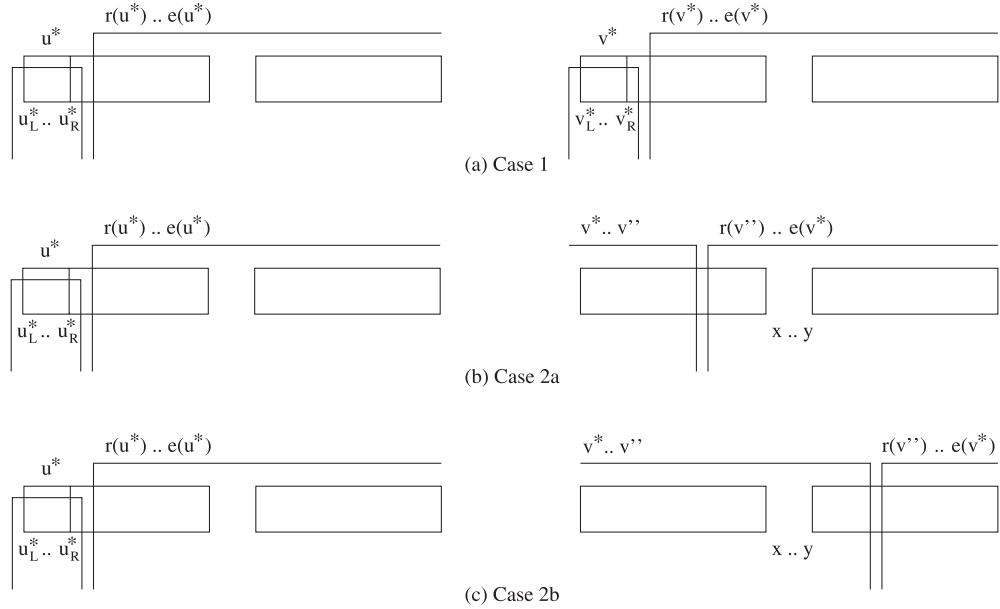
FIG. 3. Graphical illustration of Cases 1, 2a, and 2b in the proof of Lemma 7. The figures on the left and right represent the gapped subforests F' and G', respectively. The gap (which may be empty) in each figure corresponds to the sibling interval that is excluded from the original forest. The separating lines show how F' and G' are aligned in each case.

Proof. Let $F' \in gsf_\alpha(F[u..u'])$ and $G' \in gsf_\alpha(G[v..v'])$ be two α-gapped subforests at $F[u..u']$ and $G[v..v']$ that maximize the value of $sim(F', G')$, and let A be an optimal global alignment of F' and G' which attains this score. Let u^*, v^*, and a be the root of the left-most subtree 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 rather than the roots in F and G). In accordance with Lemma 3, we consider three cases:

- **Case 1:** When $a = (u^*, v^*)$, by Lemma 3, $F'[u_L^*..u_R^*]$ is aligned with $G'[v_L^*..v_R^*]$, and $F'[r(u^*)..e(u^*)]$ is aligned with $G'[r(v^*)..e(v^*)]$. See Fig. 3a. 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 $\mu(u,v) + D_{*-*}[u_L..u_R; v_L..v_R] + D_{*-*}[r(u)..u'; r(v)..v']$.
- **Case 2:** When $a = (u^*, -)$, by Lemma 3, 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 $G'[r(v'')..e(v^*)]$, and where $r(v'')$ is not a node that has been excluded from G. 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 $F[r(u)..u']$. Moreover, since G' is an α-gapped subforest at $G[v..v']$ and $r(v'')$ has not been excluded from G', at least one of the following two subcases must occur:
  - **Subcase 2a:** $G'[v^*..v'']$ is a β-gapped subforest at $G[v..v'']$ and $G'[r(v'')..e(v^*)]$ is an α-gapped subforest at $G[r(v'')..v']$. See Fig. 3b. Then the alignment score of A is equal to $\mu(u,-) + \max_{v'' \in l(v)..v'} \{D_{*-\beta}[u_L..u_R; v..v''] + D_{*-\alpha}[r(u)..u'; r(v'')..v']\}$.
  - **Subcase 2b:** $G'[v^*..v'']$ is an α-gapped subforest at $G[v..v'']$ and $G'[r(v'')..e(v^*)]$ is a β-gapped subforest at $G[r(v'')..v']$. See Fig. 3c. Then the alignment score of A is equal to $\mu(u,-) + \max_{v'' \in l(v)..v'} \{D_{*-\alpha}[u_L..u_R; v..v''] + D_{*-\beta}[r(u)..u'; r(v'')..v']\}$.
- **Case 3:** When $a = (-, v^*)$, the proof is symmetric to the proof for Case 2.

From the above three cases, the lemma follows. ∎

We can derive Lemmas 8 and 9 below for computing $D_{\alpha-\beta}[u..u'; v..v']$, $D_{\beta-\alpha}[u..u'; v..v']$, and $D_{\beta-\beta}[u..u'; v..v']$ in the same way as Lemma 7.

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

$$\max \begin{cases} \mu(u, v) + D_{*-*}[u_L..u_R; v_L..v_R] + D_{*-\beta}[r(u)..u'; r(v)..v'] \\[2mm] \mu(u, -) + \max_{v'' \in l(v)..v'} \{D_{*-\beta}[u_L..u_R; v..v''] + D_{*-\beta}[r(u)..u'; r(v'')..v']\} \\[2mm] \mu(-, v) + \max_{u'' \in l(u)..u'} \{D_{\beta-*}[u..u''; v_L..v_R] + D_{\alpha-\beta}[r(u'')..u'; r(v)..v']\} \\[2mm] \mu(-, v) + \max_{u'' \in l(u)..u'} \{D_{\alpha-*}[u..u''; v_L..v_R] + D_{\beta-\beta}[r(u'')..u'; r(v)..v']\} \end{cases}$$

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

Lemma 9 (beta-beta-matrix). $D_{\beta-\beta}[u..u'; v..v'] =$

$$\max \begin{cases} \mu(u, v) + D_{*-*}[u_L..u_R; v_L..v_R] + D_{\beta-\beta}[r(u)..u'; r(v)..v'] \\[2mm] \mu(u, -) + \max_{v'' \in l(v)..v'} \{D_{*-\beta}[u_L..u_R; v..v''] + D_{\beta-\beta}[r(u)..u'; r(v'')..v']\} \\[2mm] \mu(-, v) + \max_{u'' \in l(u)..u'} \{D_{\beta-*}[u..u''; v_L..v_R] + D_{\beta-\beta}[r(u'')..u'; r(v)..v']\} \end{cases}$$

Algorithm   *Compute-lgsf*

Input: Two Σ-labeled forests F and G.

Output: $lgsf(F, G)$.

1 For all $a, b \in \{\alpha, \beta, *\}$, initialize the base cases for $D_{a-b}[u..u'; v..v']$ (i.e., all matrix entries with $u..u' = \emptyset$ or $v..v' = \emptyset$).

2 **for** each $u..u' \in \mathcal{S}(F)$ according to the partial order $\prec$ **do**

2.1   **for** each $v..v' \in \mathcal{S}(G)$ according to the partial order $\prec$ **do**

2.1.1   Compute $D_{\alpha-\alpha}[u..u'; v..v']$, $D_{\alpha-\beta}[u..u'; v..v']$, $D_{\beta-\alpha}[u..u'; v..v']$, and

$D_{\beta-\beta}[u..u'; v..v']$ by using the expressions in Lemmas 7, 8, and 9.

2.1.2   Compute $D_{\alpha-*}[u..u'; v..v']$, $D_{*-\alpha}[u..u'; v..v']$, $D_{\beta-*}[u..u'; v..v']$, and

$D_{*-\beta}[u..u'; v..v']$ by Lemma 6.

2.1.3   Compute $D_{*-*}[u..u'; v..v']$ by Lemma 5.

        endfor

    endfor

3 Compute $lgsf(F, G)$ by Lemma 4 and **return** it.

End   *Compute-lgsf*

FIG. 4.   The dynamic programming algorithm for computing $lgsf(F, G)$.

4.4. The main algorithm

For any Σ-labeled forest F and $u..u', v..v' \in \mathcal{S}(F)$, we write $u..u' \prec v..v'$ if either the set of nodes in $u..u'$ are descendents of some node v'' in $v..v'$, or if $u..u'$ is a subset of $v..v'$. Note that $\prec$ is a partial order.

Our algorithm *Compute-lgsf* for computing the value of $lgsf(F, G)$ for two given Σ-labeled forests F and G is shown in Fig. 4. It uses Lemmas 5 to 9 to calculate $D_{a-b}[u..u'; v..v']$ for all $a, b \in \{\alpha, \beta, *\}$ and all sibling intervals $u..u' \in \mathcal{S}(F)$ and $v..v' \in \mathcal{S}(G)$ according to the partial order $\prec$, and then obtains $lgsf(F, G)$ by applying Lemma 4.

The next lemma states the time and space complexity of our algorithm.

Lemma 10. Compute-*lgsf runs in* $O(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G) \cdot (\deg(F) + \deg(G)))$ *time and* $O(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G))$ *space.*

Proof. Since there are $O(1)$ matrices of the form D_{a-b} for $a, b \in \{*, \alpha, \beta\}$, each of which uses $O(|\mathcal{S}(F)| \cdot |\mathcal{S}(G)|)$ space, and since $|\mathcal{S}(F)| = O(|F| \cdot \deg(F))$ and $|\mathcal{S}(G)| = O(|G| \cdot \deg(G))$, the total space complexity is $O(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G))$. Similarly, as we need to fill in a total of $O(|\mathcal{S}(F)| \cdot |\mathcal{S}(G)|)$ matrix entries and each entry can be computed in $O(\deg(F) + \deg(G))$ time, the total time complexity is $O(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G) \cdot (\deg(F) + \deg(G)))$. ∎

Having filled in all entries of the nine D_{a-b}-matrices for $a, b \in \{*, \alpha, \beta\}$, an optimal alignment can be easily found using a standard traceback technique. The complexity will remain the same.

Theorem 2. *The lgsf can be solved in* $O(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G) \cdot (\deg(F) + \deg(G)))$ *time and* $O(|F| \cdot |G| \cdot \deg(F) \cdot \deg(G))$ *space.*

5. ALGORITHMS FOR TWO VARIANTS OF THE LOCAL GAPPED SUBFOREST ALIGNMENT PROBLEM

5.1. The local β-gapped subforest alignment problem ($lgsf_\beta$)

To improve the efficiency of the algorithm in Section 4.4 for *lgsf* when applied to *lgsf$_\beta$*, we introduce a new matrix B of size $|F| \cdot |G|$, defined as follows.

Definition 4 (B-matrix). *For every node u in F and node v in G, the matrix element $B[u; v]$ is defined by $B[u; v] = \max\{sim(F', G') \mid F' \in gsf_\beta[u..u'], G' \in gsf_\beta[v..v'], u' \in F, v' \in G\}$.*

From the definitions of B and $D_{\beta-\beta}$, we immediately obtain the following lemma.

Lemma 11. $B[u; v] = \max\{D_{\beta-\beta}[u..u'; v..v'] \mid u' \in F, v' \in G\}$

Together with Lemma 4, we also have the following lemma.

Lemma 12. $lgsf_\beta(F, G) = \max\{B[u; v] \mid u \in F, v \in G\}$.

The computation of the elements of B can be recursively formulated as the following lemma.

Lemma 13. $B[u; v] =$

$$
\max \begin{cases} \mu(u, v) + D_{*-*}[u_L..u_R; v_L..v_R] + \max\{B[r(u); r(v)], 0\} \\[2mm] \mu(u, -) + \max_{v'' \in l(v)..v'} \{D_{*-\beta}[u_L..u_R; v..v''] + \max\{B[r(u); r(v'')], 0\}\} \\[2mm] \mu(-, v) + \max_{u'' \in l(u)..u'} \{D_{\beta-*}[u..u''; v_L..v_R] + \max\{B[r(u''); r(v)], 0\}\} \end{cases}
$$

Proof. The proof is similar to the proof of Lemma 7. Given $u \in F$ and $v \in G$, let F' and G' be two β-gapped subforests at $F[u..u']$ and $G[v..v']$, taken over all possible β-gapped subforests at $F[u..u']$ and $G[v..v']$ and all $u' \in F$ and all $v' \in G$, that maximize the value of $sim(F', G')$. Let A be an optimal global alignment of F' and G' which attains this score, and let u^*, v^*, and a be the root of the left-most subtree 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' rather than the roots in F and G). In accordance with Lemma 3, we consider three cases:

- **Case 1:** When $a = (u^*, v^*)$, by Lemma 3, $F'[u_L^*..u_R^*]$ is aligned with $G'[v_L^*..v_R^*]$, and $F'[r(u^*)..e(u^*)]$ is aligned with $G'[r(v^*)..e(v^*)]$. Since F' and G' are β-gapped subforests at $F[u..u']$ and $G[v..v']$, it holds that $F'[u_L^*..u_R^*]$ and $G'[v_L^*..v_R^*]$ are gapped subforests at $F[u_L..u_R]$ and $G[v_L..v_R]$, respectively, and $F'[r(u^*)..e(u^*)]$ and $G'[r(v^*)..e(v^*)]$ are β-gapped subforests at $F[r(u)..u']$ and $G[r(v)..v']$, respectively. The score of the subalignment A' of A between $F'[r(u^*)..e(u^*)]$ and $G'[r(v^*)..e(v^*)]$ equals $B[r(u); r(v)]$ if $B[r(u); r(v)] \geq 0$; otherwise, if $B[r(u); r(v)] < 0$ then by the optimality of A, we have $r(u^*)..e(u^*) = \emptyset$ and $r(v^*)..e(v^*) = \emptyset$ so that the score of A' must be 0. Hence, the alignment score of A equals $\mu(u, v) + D_{*-*}[u_L..u_R; v_L..v_R] + \max\{B[r(u); r(v)], 0\}$.
- **Case 2:** When $a = (u^*, -)$, by Lemma 3, 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 a β-gapped subforest at $F[u..u']$, we have that $F'[u_L^*..u_R^*]$ is a gapped subforest at $F[u_L..u_R]$, and $F'[r(u^*)..e(u^*)]$ is a β-gapped subforest at $F[r(u)..u']$. The score of the subalignment A' of A between $F'[r(u^*)..e(u^*)]$ and $G'[r(v'')..e(v^*)]$ equals $B[r(u); r(v'')]$ if $B[r(u); r(v'')] \geq 0$; otherwise, by the optimality of A, the score of A' must be 0. Hence, the alignment score of A equals $\mu(u, -) + \max_{v'' \in l(v)..v'}\{D_{*-\beta}[u_L..u_R; v..v''] + \max\{B[r(u); r(v'')], 0\}\}$.
- **Case 3:** When $a = (-, v^*)$, the proof is symmetric to the proof for Case 2.

From the above three cases, the lemma follows. ∎

To obtain $lgsf_\beta(F, G)$, we compute all entries in B and then apply Lemma 12. By inspecting the recursive equations in Lemma 13 and Lemmas 5–9, we note that it is unnecessary to fill in *all* entries in D_{a-b} for all $a, b \in \{\alpha, \beta, *\}$ to accomplish this. 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 $v' = e(v)$. This means that each of the nine D_{a-b}-matrices can be replaced by two smaller matrices D'_{a-b} and D''_{a-b} of size $|F| \cdot |\mathcal{S}(G)|$ and $|G| \cdot |\mathcal{S}(F)|$, respectively, defined through $D'_{a-b}[u; v..v'] = D_{a-b}[u..e(u); v..v']$ and $D''_{a-b}[u..u'; v] = D_{a-b}[u..u'; v..e(v)]$.

Theorem 3. *The $lgsf_\beta$ can be solved in $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G))^2)$ time and $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G)))$ space.*

Proof. First precompute all D'_{a-b}- and D''_{a-b}-matrices in the same way as all D_{a-b}-matrices were computed by algorithm *Compute-lgsf* in the previous section, then fill in all entries of B according to a bottom-up, right-to-left ordering of the nodes using Lemma 13, and finally apply Lemma 12 to get the solution to $lgsf_\beta$. Since there are only $O(|F| \cdot |\mathcal{S}(G)| + |G| \cdot |\mathcal{S}(F)|) = O(|F| \cdot |G| \cdot (\deg(F) + \deg(G)))$ entries of the form $D'_{a-b}[u; v..v']$ and $D''_{a-b}[u..u'; v]$, where $u..u' \in \mathcal{S}(F)$ and $v..v' \in \mathcal{S}(G)$, the total space required is reduced to $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G)))$. Each entry can be computed in $O(\deg(F) + \deg(G))$ time, so the total time complexity is $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G))^2)$. ∎

It follows from Theorem 1 and Theorem 3 that the local sequence-structure alignment problem (*lssa*) described by Backofen and Will (to appear) for two given annotated sequences (S_1, P_1) and (S_2, P_2) can be solved in $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G))^2)$ time and $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G)))$ space, where $|F| = O(|S_1|)$ and $\deg(F) = O(|S_1|)$, $|G| = O(|S_2|)$ and $\deg(G) = O(|S_2|)$.

5.2. *The local closed subforest alignment problem (lcsf)*

Here we present a faster and more space-efficent algorithm for *lcsf* than the one given by Höchsmann *et al.* (2003). Using the same technique as in Section 5.1, we define a matrix C of size $|F| \cdot |G|$ as follows:

Definition 5 (*C*-matrix). *For every node u in F and node v in G, the matrix element $C[u; v]$ is defined by $C[u; v] = \max\{sim(F[u..u'], G[v..v']) \mid u' \in F, v' \in G\}$.*

Then, we have the following lemma:

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

For convenience, we also define a matrix E of size $|\mathcal{S}(F)| \cdot |\mathcal{S}(G)|$ as follows.

Definition 6 (*E*-matrix). *For every $u..u' \in \mathcal{S}(F)$ and $v..v' \in \mathcal{S}(G)$, the matrix element $E[u..u'; v..v']$ is defined by $E[u..u'; v..v'] = sim(F[u..u'], G[v..v'])$.*

Now, the computation of the elements of C may be expressed recursively as the following lemma.

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

$$
\max \begin{cases}
\mu(u, v) + E[u_L..u_R; v_L..v_R] + \max\{C[r(u); r(v)], 0\} \\[2mm]
\mu(u, -) + \max_{v'' \in l(v)..v'} \{E[u_L..u_R; v..v''] + \max\{C[r(u); r(v'')], 0\}\} \\[2mm]
\mu(-, v) + \max_{u'' \in l(u)..u'} \{E[u..u''; v_L..v_R] + \max\{C[r(u''); r(v)], 0\}\}
\end{cases}
$$

To solve $lcsf(F, G)$, first construct C using Lemma 15 and then apply Lemma 14. To obtain the entries of C, we just need all entries of E of the form $E[u..u'; v..v']$ where $u' = e(u)$ or $v' = e(v)$. All of these entries can be precomputed by the algorithm of Jiang *et al.* (1995) in $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G))^2)$ time. Thus, we obtain the following theorem.

Theorem 4. *The lcsf can be solved in $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G))^2)$ time and $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G)))$ space.*

Proof. We compute $O(|F| \cdot |G|)$ entries of the form $C[u; v]$ and $O(|F| \cdot |\mathcal{S}(G)| + |G| \cdot |\mathcal{S}(F)|) = O(|F| \cdot |G| \cdot (\deg(F) + \deg(G)))$ entries of the form $E[u..u'; v..v']$, so the total space required is $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G)))$. As for the time complexity, it takes a total of $O(|F| \cdot |G| \cdot (\deg(F) + \deg(G))^2)$ time to first run the algorithm of Jiang *et al.* (1995) and then to fill in C since each element in C can be computed in $O(\deg(F) + \deg(G))$ time using Lemma 15. ∎

6. EXPERIMENTS

6.1. Implementation

In the experiments, our main purpose is to apply the improved algorithms for local subforest alignment problems to compare RNA secondary structures. We implemented the algorithm for $lgsf_\beta$ presented in Section 4 and Section 5.1 and tested the program on real RNA secondary structures. The source code is available from http://www.comp.nus.edu.sg/~bioinfo/LGSFAligner/. As discussed in Section 3, our *LGSFAligner* program has the same function as the *lssa* program implemented by Backofen and Will (to appear).

LGSFAligner takes as input two RNA secondary structures in FASTA format, represents them as labeled ordered forests, and computes their optimal β-gapped subforest alignment. In the program, we follow the scoring function suggested by *lssa*. Besides, we apply the space-saving tabulation technique proposed by Höchsmann *et al.* (2003) as follows. To store the entries of the matrices $D_{a-b}[u; v..v']$ and $D_{a-b}[u..u'; v]$, where $a, b \in \{*, \alpha, \beta\}; u, u' \in F; v, v' \in G$, a straightforward tabulation would use three-dimensional arrays, and it requires $O(|F| \cdot |G| \cdot \max(\deg(F), \deg(G)))$ space. It wastes space because $\max(\deg(F), \deg(G))$ may be much bigger than the degree of most of the nodes in F and G. A solution is to pack the entries

TABLE 3.   A COMPARISON OF THE RUNNING TIME (IN SECONDS) OF *LGSFAligner* AND *lssa*,
WHERE THE INPUTS ARE RNASE P ALIGNMENTS OF RNA SECONDARY STRUCTURES
TAKEN FROM BROWN (1999)

Genera/group	*Organisms*	*LGSFAligner*	*lssa*
Bacteria/Purple bacteria	Agrobacterium & Caulobacter	14	253
Bacteria/Gram-positive	Luteococcus & Terrabacter	11	19
Bacteria/Cyanobacteria	Nostoc & Tolythrix	18	240
Bacteria/Chlamydia	Chlamydia & Chlamydiophila	14	114
Bacteria/Planctomycetes	Pirellula & Planctomyces	16	66
Bacteria/Deinococci	Deinococcus & Thermus	14	270
Archaea/Euryarchaea	Haloferax & Thermococcus	12	298
Archaea/Crenarchaea	Acidianus & Sulfolobus	12	52
Eukaryotes/Mitochondrion	Reclinomonas & Porphyra	14	147
Eukaryotes/Plastid	Cyanophora & Nephroselmis	11	189

in D_{a-b} by defining an array *offset* as follows: $offset(root) = 1$, $offset(u) = offset(u_0) + nRSiblings(u_0)$, and refer to each sibling interval $u..u'$ by the index $offset(u) + i - 1$, where

- *root* is an imaginary root whose children are roots of the maximal trees in the forest,
- u_0 is the immediately previous node of u in the preorder traversal of the forest,
- $nRSiblings(u)$ is the number of right siblings of u, including itself, and
- u' is the i-th right sibling of u.

In other words, *offset* acts as a bijective function between the set of sibling intervals in $\mathcal{S}(F)$ and the set of integers in $\{1, \ldots, |\mathcal{S}(F)|\}$; i.e., each sibling interval $u..u'$ is indexed by a unique integer. This tabulation technique does not reduce the worst case space complexity, but in practice it considerably reduces the space required by the program. This is especially important because it enables the algorithms to be run with long RNA molecules, in which case the amount of memory usage increases rapidly.

6.2. Results

LGSFAligner was written in Java and was run with real RNA secondary structures, mainly taken from Brown (1999) and from Wilting *et al.* (1997). The running time and the space usage of *LGSFAligner* were compared with those of *lssa* program (Backofen and Will, to appear) written in C++. The experiments were conducted in a Windows XP environment; the system was an Intel Pentium 4 running at 1.8 GHz with 768 MB of RAM.

The first experiment performed on secondary structure elements with putative selenocysteine insertion function (Wilting *et al.*, 1997) showed that *LGSFAligner* correctly aligns their putative SECIS-motifs (see Fig. 3 of Wilting *et al.* [1997]). These RNAs are short and thus both programs compute their alignments in less than one second. The second experiment computed a number of RNase P alignments of RNA secondary structures taken from Brown (1999). The running time of *LGSFAligner* is significantly faster than *lssa*; however, *LGSFAligner* consumes a considerable amount of memory: about 250 MB are required to align two RNA secondary structures, each of which has about 400 bases, whereas *lssa* used only about 3 MB of memory. The memory bottleneck is due to two reasons: additional parameters in the forest representation of RNAs and many matrices required to store alignment scores and traceback information. Table 3 shows the comparison of the running time (in seconds) of *LGSFAligner* and *lssa*.

7. CONCLUDING REMARKS

In this paper, we have introduced a new problem called the local gapped subforest alignment problem (*lgsf*) and provided an efficient algorithm to solve it. Moreover, we have applied our techniques to the local closed subforest alignment problem (*lcsf*) defined by Höchsmann *et al.* (2003) as well as the problem

of finding local RNA sequence-structure motifs ($lgsf_\beta$) considered by Backofen and Will (to appear), improving on both the time and space complexity needed by the algorithm of Höchsmann *et al.* (2003) and greatly improving on the running time of the algorithm of Backofen and Will (to appear) by taking advantage of the forest representation of RNA secondary structures. Lastly, we have implemented the *LGSFAligner* program which is based on our improved algorithms for $lgsf_\beta$ to find RNA structral motifs. We have performed experiments on real RNA secondary structures and investigated the practical issues of our algorithms thoroughly.

Our proposed problem and solution motivate future developments in local alignments of labeled ordered forests. One of the challenges is to find even more efficient algorithms for these types of problems. Any improvement can have a vital impact on RNA comparison and structure prediction applications. Another interesting task is to further generalize our local gapped subforest alignment problem by allowing exclusions of more than one closed subforest sharing the same parent node. Lastly, new alignment models could be proposed to improve the accuracy for RNA comparison and structure prediction.

REFERENCES

Alber, J., Gramm, J., Guo, J., and Niedermeier, R. 2004. Computing the similarity of two sequences with nested arc annotations. *Theoret. Comput. Sci.* 312(2–3), 337–358.

Backofen, R., and Will, S. To appear. Local sequence-structure motifs in RNA. *J. Bioinformatics and Computational Biology*.

Bafna, V., Muthukrishnan, S., and Ravi, R. 1995. Computing similarity between RNA strings. *Proc. 6th Ann. Symp. on Combinatorial Pattern Matching (CPM 95)*, 1–16.

Baxter, I.D., Yahin, A., Moura, L., Sant'Anna, M., and Bier, L. 1998. Clone detection using abstract syntax trees. *Proc. IEEE Int. Conf. on Software Maintenance (ICSM 1998)*, 368–377.

Brown, J.W. 1999. The ribonuclease P database. *Nucl. Acids Res.* 27(1), 314.

Chawathe, S.S., Rajaraman, A., Garcia-Molina, H., and Widom, J. 1996. Change detection in hierarchically structured information. *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1996)*, 493–504.

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

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

Evans, P.A. 1999b. Finding common subsequences with arcs and pseudoknots. *Proc. 10th Ann. Symp. on Combinatorial Pattern Matching (CPM 99)*, 270–280.

Gramm, J., Guo, J., and Niedermeier, R. 2002. Pattern matching for arc-annotated sequences. *Proc. 22nd Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2002)*, 182–193.

Höchsmann, M., Töller, T., Giegerich, R., and Kurtz, S. 2003. Local similarity in RNA secondary structures. *Proc. Computational Systems Bioinformatics Conf. (CSB2003)*, 159–168.

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

Jiang, T., Lin, G., Ma, B., and Zhang, K. 2002. A general edit distance between RNA structures. *J. Comp. Biol.* 9(2), 371–388.

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

Jiang, T., Wang, L., and Zhang, K. 1995. Alignment of trees—an alternative to tree edit. *Theoret. Comput. Sci.* 143(1), 137–148.

Le, S.-Y., Nussinov, R., and Maizel, J.V. 1989. Tree graphs of RNA secondary structures and their comparisons. *Computers and Biomedical Res.* 22, 461–473.

Lenhof, H.-P., Reinert, K., and Vingron, M. 1998. A polyhedral approach to RNA sequence structure alignment. *J. Comp. Biol.* 5(3), 517–530.

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

Notredame, C., O'Brien, E.A., and Higgins, D.G. 1997. RAGA: RNA sequence alignment by genetic algorithm. *Nucl. Acids Res.* 25(22), 4570–4580.

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

Tai, K.-C. 1979. The tree-to-tree correction problem. *J. ACM* 26(3), 422–433.

Wang, J.T.L., Shasha, D., Chang, G.J.S., Relihan, L., Zhang, K., and Patel, G. 1997. Structural matching and discovery in document databases. *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1997)*, 560–563.

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

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

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

Yang, W. 1991. Identifying syntactic differences between two programs. *Software: Practice and Experience* 21(7), 739–755.

Zhang, K., and Jiang, T. 1994. Some MAX SNP-hard results concerning unordered labeled trees. *Inform. Proc. Lett.* 49(5), 249–254.

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

Address correspondence to:
Wing-Kin Sung
School of Computing
National University of Singapore
3 Science Drive 2
Singapore 117543

E-mail: Ksung@comp.nus.edu.sg