Theoretical Computer Science 143 (1995) 137–148

Theoretical Computer Science

Note

# Alignment of trees – an alternative to tree edit

Tao Jiang[a,1], Lusheng Wang[b,2], Kaizhong Zhang[c,*,3]

[a] *Department of Computer Science, McMaster University, Hamilton, Ont, Canada L8S 4K1*
[b] *Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ont., Canada L8S 4K1*
[c] *Department of Computer Science, University of Western Ontario, London, Ont., Canada N6A 5B7.*

Received March 1994; revised August 1994
Communicated by A. Apostolico

**Abstract**

In this paper, we propose the *alignment of trees* as a measure of the similarity between two *labeled* trees. Both *ordered* and *unordered* trees are considered. An algorithm is designed for ordered trees. The time complexity of this algorithm is $O(|T_1|\cdot|T_2|\cdot(\deg(T_1)+\deg(T_2))^2)$, where $|T_i|$ is the number of nodes in $T_i$ and $\deg(T_i)$ is the degree of $T_i$, $i = 1, 2$. The algorithm is faster than the best known algorithm for tree edit when $\deg(T_1)$ and $\deg(T_2)$ are smaller than the depths of $T_1$ and $T_2$. For unordered trees, we show that the alignment problem can be solved in polynomial time if the trees have a bounded degree and becomes MAX SNP-hard if one of the trees is allowed to have an arbitrary degree. In contrast, the edit problem for unordered trees is MAX SNP-hard even if both trees have a bounded degree (Zhang and Jiang, 1994). Finally, multiple alignment of trees is discussed.

## 1. Introduction

In many fields such as RNA secondary structures comparison, syntactic pattern recognition, image clustering, genetics, and chemical structure analysis, one often faces the problem of finding the similarity of two *labeled trees* [5–7, 9, 10, 12, 13, 15]. For instance, the comparison of *ordered* trees is very useful in the study of RNA

* Corresponding author.
A preliminary version of this paper appeared in the Proc. of CPM'94, Springer, Lecture Notes in Computer Science, Vol. 807, 1994. Submission of this paper in final form to Theoret. Comput. Sci. was invited in view of the strong endorsement contained in the conference reviews.
[1] Supported in part by NSERC Research Grant OGP0046613 E-mail: jiang@maccs.mcmaster.ca.
[2] Supported in part by NSERC Research Grant OGP0046613. E-mail: lwang@maccs.mcmaster.ca.
[3] Supported in part by NSERC Research Grant OGP0046373. E-mail: kzhang@csd.uwo.ca.

secondary structures. In general, we can decompose an RNA secondary structure into components of fives types: stem (S), hairpin (H), bulge (B), interior loop (I), and multi-branch loop (M). The secondary structure can be conveniently expressed as a tree in which each node is labeled by a letters S, H, B, I, or M, and the left-to-right order among siblings is significant [5, 10, 11, 17]. The comparison of RNA secondary structure trees can help identify conserved structural motifs in an RNA folding process [6] and construct taxonomy trees [11]. The comparison of *unordered trees* has applications to the morphological problems arising in genetics (e.g., determining genetic diseases based on ancestry tree patterns) and other fields [12, 13, 15].

As in the case of sequence comparisons, there are many ways to measure the similarity between two trees. For instance, one could use the *largest common subtree*, the *smallest common supertree*, *tree edit distance*, and the *transferable ratio* between two trees to describe the degree of similarity [4, 5, 9, 11, 14, 16]. Although edit distance and transferable ratio are both sensible measures of the distance between RNA secondary structures [5, 9], each of them only represents a certain approximation of the true functional similarity. Thus, more realistic and feasible measures would be of interest. Here, we introduce the notion of *alignment of trees* as another measure of similarity of labeled trees. The notion is a natural extension of alignment of sequences to trees.

An *ordered tree* is a rooted tree in which the children of each node are ordered. That is, if a node has $k$ children, then we can designate them as the first child, the second child, and so on up to the $k$th child. An *unordered tree* is a rooted tree in which the children of each node are unordered, i.e., they are viewed as a set.

Let us recall the definition of an insertion operation in tree edit [14, 17]. Let $T$ be an ordered (or unordered) tree. Inserting a node $u$ into $T$ means that for some node $v$ in $T$, we make $u$ the parent of a consecutive subsequence (or a subset, respectively) of the children of $v$ and then $v$ the parent of $u$. A deletion is just the complement of an insertion. Let $T_1$ and $T_2$ be two labeled trees. An alignment $\mathscr{A}$ of $T_1$ and $T_2$ is obtained by first inserting nodes labeled with *spaces* into $T_1$ and $T_2$ such that the two resulting trees $T'_1$ and $T'_2$ have the same structure, i.e., they are identical if the labels are ignored, and then *overlaying* $T'_1$ on $T'_2$. An example alignment is shown in Fig. 1. A score is defined for each pair of labels. The *value* of alignment $\mathscr{A}$ is the sum of the scores of all pairs of opposing labels. An *optimal* alignment is one that minimizes the

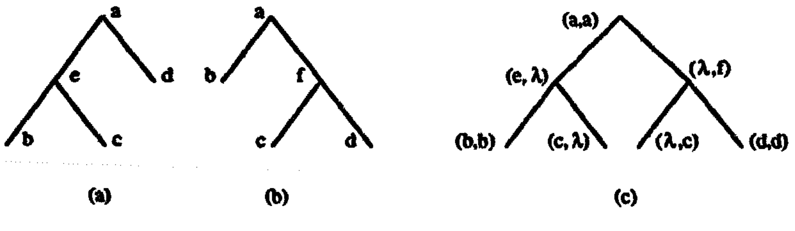

Fig. 1. (a) Tree $T_1$. (b) Tree $T_2$. (c) The optimal alignment of $T_1$ and $T_2$.

value over all possible alignments. The *alignment distance* between $T_1$ and $T_2$ is the value of an optimal alignment of $T_1$ and $T_2$.

We present an algorithm for computing the alignment distance between ordered trees. The time complexity of this algorithm is $O(|T_1| \cdot T_2 \cdot (\deg(T_1) + \deg(T_2))^2)$, where $|T_i|$ and $\deg(T_i)$ are the size and degree of $T_i$, respectively. (The degree of a tree is the maximum number of children of any node in the tree.) We also show that the alignment distance between two unordered trees can be computed in polynomial time if the trees have bounded degrees and becomes MAX SNP-hard if one of the trees is allowed to have an arbitrary degree. Thus, it is very unlikely that the alignment problem for unordered trees has a polynomial-time-approximation scheme, not to mention a polynomial-time algorithm.

## 1.1. Alignment of trees vs. tree edit

It is well known that edit and alignment are two equivalent notions for sequences. In particular, for any two sequences $x_1$ and $x_2$, the edit distance between $x_1$ and $x_2$ equals the value of an optimal alignment of $x_1$ and $x_2$. However, edit and alignment turn out to be very different for trees. The following are some interesting comparisons between alignment of trees and tree edit.

1. The edit distance and alignment distance between two trees can be different. For example, assume that each edit operation (i.e., insertion, deletion, or replacement) costs 1 and consequently each pair of distinct letters has a score 1. Consider the two ordered trees shown in Fig. 1. To optimally edit $T_1$ into $T_2$, we simply delete $e$ from $T_1$ and insert $f$ into the new tree. Thus, the edit distance between $T_1$ and $T_2$ is 2. The optimal alignment of the two trees is unique and is shown in Fig. 1(c), with a value 4. The difference between edit distance and alignment distance can be made arbitrarily large by adding subtrees below nodes $b, c, d$ in both trees. It is easy to see that in general the edit distance is smaller than the alignment distance for trees. This is because each alignment of trees actually corresponds to a restricted tree edit in which all the insertions precede all the deletions. Note that, the order of edit operations is not important for sequences. Also, it seems that alignment charges more for the structural dissimilarity at the top levels of the trees than at the lower levels, whereas edit treats all the levels the same.

2. The best algorithm for computing the edit distance between ordered trees runs in time $O(|T_1| \cdot |T_2| \cdot \min\{\text{depth}(T_1), \text{leaves}(T_1)\} \cdot \min\{\text{depth}(T_2), \text{leaves}(T_2)\}$, where $\text{depth}(T_i)$ and $\text{leaves}(T_i)$ are the depth and number of leaves of tree $T_i$, $i = 1, 2$ [17]. Clearly, $\deg(T_i) \leqslant \text{leaves}(T_i)$. In practice (e.g., RNA secondary structures), $\deg(T_i) \ll \text{leaves}(T_i)$ and $\deg(T_i) \ll \text{depth}(T_i)$. Hence, our above result shows that it is easier (faster) to align ordered trees than to edit. In particular, we can align trees with bounded degrees in time $O(|T_1| \cdot |T_2|)$.

3. The difference in time complexity is even bigger for unordered trees. As mentioned earlier, unordered trees with bounded degrees can be aligned in polynomial time (in fact, in time $O(|T_1| \cdot |T_2|)$). On the other hand, editing unordered trees with bounded degrees is NP-hard [18]. In fact, it is MAX SNP-hard [16].

In Section 2, we present the algorithm for aligning ordered trees. Section 3 contains some results on unordered trees. We briefly discuss multiple alignment of trees with SP-score in Section 4.

## 2. An efficient algorithm for aligning ordered trees

We need some definitions. The notion of alignment can be easily extended to ordered forests. The only change is that it is now possible to insert a node (as the root) to join a consecutive subsequence of trees in the forest. Denote the alignment distance between forests $F_1$ and $F_2$ as $D(F_1, F_2)$. Let $\theta$ denote the empty tree, $\lambda$ denote space, and $\mu(a,b)$ denote the score of the opposing letters $a$ and $b$. A standard assumption is that the score scheme $\mu$ satisfies *triangle inequality*, i.e., for any three letters $a$, $b$, and $c$, $\mu(a,c) \leqslant \mu(a,b) + \mu(b,c)$. The nodes in an ordered tree of size $n$ are numbered 1–$n$ according to the *postorder*. Let $T_1$ and $T_2$ be two fixed ordered labeled trees throughout this section. Denote the label of node $j$ in tree $T_i$ as $l_i[j]$ and the subtree of $T_i$ rooted at node $j$ as $T_i[j]$.

In the following, let $i$ be a node of $T_1$ and $j$ a node of $T_2$. Suppose that the degrees of $i$ and $j$ are $m_i$ and $n_j$, respectively. Denote the children of $i$ as $i_1, \ldots, i_{m_i}$ and the children of $j$ as $j_1, \ldots, j_{n_j}$. For any $s, t$, $1 \leqslant s \leqslant t \leqslant m_i$, let $F_1[i_s, i_t]$ represent the forest consisting of the subtrees $T_1[i_s], \ldots, T_1[i_t]$. For convenience, $F_1[i_1, i_{m_i}]$ is also denoted $F_1[i]$. Note that $F_1[i] \neq F_1[i,i]$. $F_2[j_s, j_t]$ and $F_2[j]$ are defined similarly.

### 2.1. Properties of the alignment distance

The following lemmas form the basis of our algorithm. The first lemma is trivial.

**Lemma 1.** $D(\theta,\theta) = 0$; $D(F_1[i],\theta) = \sum_{k=1}^{m_i} D(T_1[i_k],\theta)$; $D(T_1[i],\theta) = D(F_1[i],\theta) + \mu(l_1[i],\lambda)$; $D(\theta,F_2[j]) = \sum_{k=1}^{n_j} D(\theta,T_2[j_k])$; $D(\theta,T_2[j]) = D(\theta,F_2[j]) + \mu(\lambda,l_2[j])$.

**Lemma 2.**

$$D(T_1[i],T_2[j]) = \min \begin{cases} D(\theta,T_2[j]) + \min_{1 \leqslant r \leqslant n_j}\{D(T_1[i],T_2[j_r]) - D(\theta,T_2[j_r])\}, \\ D(T_1[i],\theta) + \min_{1 \leqslant r \leqslant m_i}\{D(T_1[i_r],T_2[j]) - D(T_1[i_r],\theta)\}, \\ D(F_1[i],F_2[j]) + \mu(l_1[i],l_2[j]). \end{cases}$$

**Proof.** Consider an optimal alignment (tree) $\mathscr{A}$ of $T_1[i]$ and $T_2[j]$. There are four cases: (1) $(l_1[i], l_2[j])$ is a label in $\mathscr{A}$, (2) $(l_1[i],\lambda)$ and $(l_1[k], l_2[j])$ are labels in $\mathscr{A}$ for some $k$, (3) $(l_1[i], l_2[k])$ and $(\lambda, l_2[j])$ are labels in $\mathscr{A}$ for some $k$, (4) $(l_1[i],\lambda)$ and $(\lambda, l_2[j])$ are labels in $\mathscr{A}$. We actually need not consider Case 4 since in this case we can delete the two nodes and then add $(l_1[i], l_2[j])$ as the new root, resulting in a better alignment.

*Case* 1: The root of $\mathscr{A}$ must be labeled as $(l_1[i], l_2[j])$. Clearly, $D(T_1[i], T_2[j]) = D(F_1[i], F_2[j]) + \mu(l_1[i], l_2[j])$.

*Case* 2: The root of $\mathscr{A}$ must be labeled as $(l_1[i], \lambda)$. In this case $k$ must be a node in $T_1[i_r]$ for some $1 \leqslant r \leqslant m_i$. Therefore, $D(T_1[i], T_2[j]) = D(T_1[i], \theta) + \min_{1 \leqslant r \leqslant m_i}\{D(T_1[i_r], T_2[j]) - D(T_1[i_r], \theta)\}$.

*Case* 3: Similar to Case 2. □

Note, the above implies that $D(F_1[i], F_2[j])$ is required for computing $D(T_1[i], T_2[j])$.

**Lemma 3.** *For any $s, t$ such that $1 \leqslant s \leqslant m_i$ and $1 \leqslant t \leqslant n_j$,*

$$D(F_1[i_1, i_s], F_2[j_1, j_t]) =$$
$$\min \begin{cases} D(F_1[i_1, i_{s-1}], F_2[j_1, j_t]) + D(T_1[i_s], \theta), \\ D(F_1[i_1, i_s], F_2[j_1, j_{t-1}]) + D(\theta, T_2[j_t]), \\ D(F_1[i_1, i_{s-1}], F_2[j_1, j_{t-1}]) + D(T_1[i_s], T_2[j_t]), \\ \mu(\lambda, l_2[j_t]) + \min_{1 \leqslant k < s}\{D(F_1[i_1, i_{k-1}], F_2[j_1, j_{t-1}]) + D(F_1[i_k, i_s], F_2[j_t])\}, \\ \mu(l_1[i_s], \lambda) + \min_{1 \leqslant k < t}\{D(F_1[i_1, i_{s-1}], F_2[j_1, j_{k-1}]) + D(F_1[i_s], F_2[j_k, j_t])\}. \end{cases}$$

**Proof.** Consider an optimal alignment (forest) $\mathscr{A}$ of $F_1[i_1, i_s]$ and $F_2[j_1, j_t]$. The root of the rightmost tree in $\mathscr{A}$ is labeled by either $(l_1[i_s], l_2[j_t])$, $(l_1[i_s], \lambda)$, or $(\lambda, l_2[j_t])$.

*Case* 1: *The label is* $(l_1[i_s], l_2[j_t])$. In this case, the rightmost tree must be an optimal alignment of $T_1[i_s]$ and $T_2[j_t]$. Therefore $D(F_1[i_1, i_s], F_2[j_1, j_t]) = D(F_1[i_1, i_{s-1}], F_2[j_1, j_{t-1}]) + D(T_1[i_s], T_2[j_t])$.

*Case* 2: *The label is* $(l_1[i_s], \lambda)$. In this case, there is a $k$, $0 \leqslant k \leqslant t$, such that $T_1[i_s]$ is aligned with the subforest $F_2[j_{t-k+1}, j_t]$. A key observation here is the fact that subtree $T_2[j_{t-k+1}]$ is not split by the alignment with $T_1[i_s]$. There are three subcases.

*Case* 2.1: $k = 0$. I.e., $F_2[j_{t-k+1}, j_t] = \theta$. Therefore,

$$D(F_1[i_1, i_s], F_2[j_1, j_t]) = D(F_1[i_1, i_{s-1}], F_2[j_1, j_t]) + D(T_1[i_s], \theta).$$

*Case* 2.2: $k = 1$. I.e., $F_2[j_{t-k+1}, j_t] = T_2[j_t]$. This is the same as in Case 1.

*Case* 2.3: $k \geqslant 2$. This is the most general case. It is easy to see that

$$D(F_1[i_1, i_s], F_2[j_1, j_t]) = \mu(l_1[i_s], \lambda) + \min_{1 \leqslant k < t}\{D(F_1[i_1, i_{s-1}], F_2[j_1, j_{k-1}]) + D(F_1[i_s], F_2[j_k, j_t])\}.$$

*Case* 3: *The label is* $(\lambda, l_2[j_t])$. Similar to Case 2. □

### 2.2. *The algorithm*

It follows from the above discussion that, for each pair of subtrees $T_1[i]$ and $T_2[j]$, we have to compute $D(F_1[i], F_2[j_s, j_t])$ for all $1 \leqslant s \leqslant t \leqslant n_j$, and $D(F_1[i_s, i_t], F_2[j])$ for all $1 \leqslant s \leqslant t \leqslant m_i$. That is, we need align $F_1[i]$ with each subforest of $F_2[j]$, and conversely align $F_2[j]$ with each subforest of $F_1[i]$. Note that we do not have to align an arbitrary forest of $F_1[i]$ with an arbitrary forest of $F_2[j]$. Otherwise the time complexity would be higher.

For each fixed $s$ and $t$, where either $s = 1$ or $t = 1$, $1 \leqslant s \leqslant m_i$ and $1 \leqslant t \leqslant n_j$, the procedure in Fig. 2 computes $\{D(F_1[i_s, i_p], F_2[j_t, j_q]) | s \leqslant p \leqslant m_i, t \leqslant q \leqslant n_j\}$, assuming that all $D(F_1[i_k], F_2[j_p, j_q])$ are known, where $1 \leqslant k \leqslant m_i$ and $1 \leqslant p \leqslant q \leqslant n_j$, and all $D(F_1[i_p, i_q], F_2[j_k])$ are known, where $1 \leqslant p \leqslant q \leqslant m_i$ and $1 \leqslant k \leqslant n_j$.

Hence we can obtain $D(F_1[i], F_2[j_s, j_t])$ for all $1 \leqslant s \leqslant t \leqslant n_j$ by calling Procedure 1 $n_j$ times, and $D(F_1[i_s, i_t], F_2[j])$ for all $1 \leqslant s \leqslant t \leqslant m_i$ by calling Procedure 1 $m_i$ times. Our algorithm to compute $D(T_1, T_2)$ is given in Fig. 3.

```
Input: $F_1[i_s, i_{m_i}]$ and $F_2[j_t, j_{n_j}]$.
$D(F_1[i_s, i_{s-1}], F_2[j_t, j_{t-1}]) := 0$;
for $p := s$ to $m_i$
    $D(F_1[i_s, i_p], F_2[j_t, j_{t-1}]) := D(F_1[i_s, i_{p-1}], F_2[j_t, j_{t-1}]) + D(T_1[i_p], \theta)$;
for $q := t$ to $n_j$
    $D(F_1[i_s, i_{s-1}], F_2[j_t, j_q]) := D(F_1[i_s, i_{s-1}], F_2[j_t, j_{q-1}]) + D(\theta, T_2[j_q])$;
for $p := s$ to $m_i$
    for $q := t$ to $n_j$
        Compute $D(F_1[i_s, i_p], F_2[j_t, j_q])$ as in Lemma 3.
Output: $\{D(F_1[i_s, i_p], F_2[j_t, j_q]) | s \leqslant p \leqslant m_i, t \leqslant q \leqslant n_j\}$.
```

Fig. 2. Procedure 1: Computing $\{D(F_1[i_s, i_p], F_2[j_t, j_q]) | s \leqslant p \leqslant m_i, t \leqslant q \leqslant n_j\}$ for fixed $s$ and $t$.

```
Input: $T_1$ and $T_2$.
$D(\theta, \theta) := 0$;
for $i := 1$ to $|T_1|$
    Initialize $D(T_1[i], \theta)$ and $D(F_1[i], \theta)$ as in Lemma 1;
for $j := 1$ to $|T_2|$
    initialize $D(\theta, T_2[j])$ and $D(\theta, F_2[j])$ as in Lemma 1;
for $i := 1$ to $|T_1|$
    for $j := 1$ to $|T_2|$
        for $s := 1$ to $m_i$
            Call Procedure 1 on $F_1[i_s, i_{m_i}]$ and $F_2[j]$;
        for $t := 1$ to $n_j$
            Call Procedure 1 on $F_1[i]$ and $F_2[j_t, i_{n_j}]$;
        Compute $D(T_1[i], T_2[j])$ as in Lemma 2.
Output: $D(T_1[|T_1|], T_2[|T_2|])$.
```

Fig. 3. Algorithm 1: Computing $D(T_1, T_2)$.

### 2.3. The time complexity

For an input $F_1[i_s, i_{m_i}]$ and $F_1[j_t, j_{n_j}]$, the running time of Procedure 1 is bounded by

$$O((m_i - s)\cdot(n_j - t)\cdot(m_i - s + n_j - t)) = O(m_i\cdot n_j\cdot(m_i + n_j)).$$

So, for each pair $i$ and $j$, Algorithm 1 spends $O(m_i\cdot n_j\cdot(m_i + n_j)^2)$ time. Therefore, the time complexity of Algorithm 1 is

$$\begin{aligned}&\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O(m_i\cdot n_j\cdot(m_i + n_j)^2)\\ &\leqslant \sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O(m_i\cdot n_j\cdot(\deg(T_1) + \deg(T_2))^2)\\ &\leqslant O\left((\deg(T_1) + \deg(T_2))^2\cdot\sum_{i=1}^{|T_1|} m_i\cdot\sum_{j=1}^{|T_2|} n_j\right)\\ &\leqslant O(|T_1|\cdot|T_2|\cdot(\deg(T_1) + \deg(T_2))^2).\end{aligned}$$

If both $T_1$ and $T_2$ have degrees bounded by some constant, the time complexity becomes $O(|T_1|\cdot|T_2|)$. Note that the algorithm actually computes $D(T_1[i], T_2[j])$, $D(F_1[i], F_2[j])$, $D(F_1[i_s, i_t], F_2[j])$ and $D(F_1[i], F_2[j_s, j_t])$. With these data, an actual optimal alignment can be easily found using a simple back-tracking technique. The complexity will remain the same.

## 3. The alignment of unordered trees

In this section we consider unordered labeled trees, i.e., the order among the siblings is insignificant. It is known that computing the edit distance for unordered trees is MAX SNP-hard even if both trees have bounded degrees [16]. We will show that aligning unordered trees with bounded degrees can be done in polynomial time, and give a simple algorithm to align unordered binary trees. Finally, we prove that aligning unordered trees becomes MAX SNP-hard if one of the input trees can have an arbitrary degree.

### 3.1. Unordered trees with bounded degrees

When the degrees are bounded, we can compute the alignment distance using a modified version of Algorithm 1. Lemmas 1 and 2 still work. The only difference is in the computation of $D(F_1[i], F_2[j])$. We have to revise the recurrence relation in Lemma 3 as follows: for each (forest) $A \subseteq \{T_1[i_1], \ldots, T_1[i_{m_i}]\}$ and each (forest)

$B \subseteq \{T_2[j_1], \ldots, T_2[j_{n_j}]\}$.

$$D(A, B) =$$
$$\min \begin{cases} \min_{T_1[i_p] \in A, T_2[j_q] \in B} D(A - \{T_1[i_p]\}, B - \{T_2[j_q]\}) + D(T_1[i_p], T_2[j_q]), \\ \min_{T_1[i_p] \in A, B' \subseteq B} D(A - \{T_1[i_p]\}, B - B') + D(F_1[i_p], B') + \mu(l_1[i_p], \lambda), \\ \min_{A' \subseteq A, T_2[j_q] \in B} D(A - A', B - \{T_2[j_q]\}) + D(A', F_2[j_q]) + \mu(\lambda, l_2[j_q]). \end{cases}$$

Since $m_i$ and $n_j$ are bounded, $D(A, B)$ can be computed in polynomial time. If $T_1$ and $T_2$ are both in fact binary trees, the algorithm can be much simplified, as shown in Fig. 4. It is easy to see that the time complexity of this algorithm is $O(|T_1| \cdot |T_2|)$.

## 3.2. *The hardness of aligning unordered trees*

In this subsection we show that the problem of determining the optimal alignment of two unordered trees is MAX SNP-hard. It follows from [1] that this problem does not have a polynomial-time-approximation scheme (PTAS), unless P = NP. (A problem has a PTAS if for every fixed $\varepsilon > 0$, the problem can be approximate with ratio $1 + \varepsilon$ in polynomial time.)

```
Input: $T_1$ and $T_2$.
for $i := 1$ to $|T_1|$
  for $j := 1$ to $|T_1|$
    $D(F_1[i], F_2[j]) := \min\{\mu(l_1[i_2], \lambda) + D(F_1[i_2], F_2[j]) + D(T_1[i_1], \theta),$
                               $\mu(l_1[i_1], \lambda) + D(F_1[i_1], F_2[j]) + D(T_1[i_2], \theta),$
                               $\mu(\lambda, l_2[j_2]) + D(F_1[i], F_2[j_2]) + D(\theta, T_2[j_1]),$
                               $\mu(\lambda, l_2[j_1]) + D(F_1[i], F_2[j_1]) + D(\theta, T_2[j_2]),$
                               $D(T_1[i_1, T_2[j_1]) + D(T_1[i_2], T_2[j_2]),$
                               $D(T_1[i_1, T_2[j_2]) + D(T_1[i_2], T_2[j_1])\};$

    $D(T_1[i], T_2[j]) := \min\{\mu(l_1[i], l_2[j]) + D(F_1[i], F_2[j]),$
                               $\mu(l_1[i], \lambda) + D(T_1[i_1], T_2[j]) + D(T_1[i_2], \theta),$
                               $\mu(l_1[i], \lambda) + D(T_1[i_2], T_2[j]) + D(T_1[i_1], \theta),$
                               $\mu(\lambda, l_2[j]) + D(T_1[i], T_2[j_1]) + D(\theta, T_2[j_2]),$
                               $\mu(\lambda, l_2[j]) + D(T_1[i], T_2[j_2]) + D(\theta, T_2[j_1])\};$
Output: $D(T_1[|T_1|], T_2[|T_2|])$;
```

Fig. 4. Algorithm 2: Aligning unordered binary trees.

We need the concept of an L-reduction [8]. Suppose that $\Pi_1$ and $\Pi_2$ are two optimization problems. We say that $\Pi_1$ *L-reduces* to $\Pi_2$ if there are two polynomial-time algorithms $f$ and $g$ and constants $\alpha, \beta > 0$ such that, for any instance $I$ of $\Pi_1$,

1. $OPT(f(I)) \leqslant \alpha \cdot OPT(I)$.
2. Given any solution of $f(I)$ with weight $s_2$, algorithm $g$ produces in polynomial time a solution of $I$ with weight $s_1$ satisfying $|s_1 - OPT(I)| \leqslant \beta \cdot |s_2 - OPT(f(I))|$.

A problem is MAX SNP-hard if every problem in the class MAX SNP [8] L-reduces to it. Since the composition of two L-reductions is also an L-reduction, to show that a problem $\Pi$ is MAX SNP-hard, it suffices to L-reduce a MAX SNP-hard problem to $\Pi$.

We will L-reduce maximum bounded cover by 3-sets (MAX 3SC-3) [3] to alignment of unordered trees. MAX 3SC-3 is an optimization version of exact cover by 3-sets [2]. Kann showed that this problem is MAX SNP-hard [3].

**Problem MAX 3SC-3.** Given a collection $C = \{c_1, c_2, \dots, c_n\}$ of subsets of a finite set $A = \{a_1, a_2, \dots, a_m\}$, where every $c \in C$ contains three elements and every element $a \in A$ is contained in at most 3 of the subsets in $C$, find a largest *exact partial cover* $C' \subseteq C$ of $A$. (An exact partial cover of $A$ is a collection of mutually disjoint subsets of $C$.)

Our reduction is actually similar to the ones in [16]. Let $A = \{a_1, a_2, \dots, a_m\}$ and $C = \{c_1, c_2, \dots, c_n\}$, where $c_i = \{c_{i,1}, c_{i,2}, c_{i,3}\}$ and $c_{i,j} \in A$, be an instance of the MAX 3SC-3 problem. Without loss of generality we assume that $m \leqslant 3n$. Let $C'$ be a largest exact partial cover of $A$.

We construct two trees as in Fig. 5. The alphabet of labels is $A \cup \{r, s, t\}$, assuming letters $r, s, t$ are not contained in $A$. Let $\mu(a,b) = 0$ if $a = b$ or 2 if $a, b \neq \lambda$ and $a \neq b$, $\mu(a, \lambda) = 1$, and $\mu(\lambda, b) = 1$. The degree of $T_1$ is bounded by 3. Each $T_{1,i}$ is a subtree, which corresponds to the subset $c_i$ in $C$. The sequence of nodes with label $s$ in each $T_{1,i}$ is called *upper segment* and the three branches are called *lower segment*. The root of $T_2$ has $m + n$ children.

It is clear that $T_1$ and $T_2$ can be constructed from an instance of MAX 3SC-3 in polynomial time. These two trees are considered to be an instance of alignment of

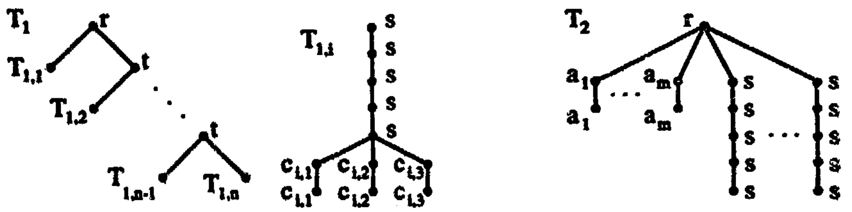

Fig. 5. The reduction.

unordered trees. We call this transformation from MAX 3SC-3 to alignment of unordered trees $f$.

Let $\mathscr{A}'$ be an optimal alignment of $T_1$ and $T_2$ in Fig. 5. The following lemma gives the relationship between $C'$ and $\mathscr{A}'$.

**Lemma 4.** $v(\mathscr{A}') = |T_1| + |T_2| - 2(|C'| + 5n + 1)$.

**Proof.** In an optimal alignment of $T_1$ an $T_2$, for each subtree $T_{1,i}$, either its entire upper segment or its entire lower segment is "matched" with the corresponding identical parts of $T_2$. Matching the upper segment saves a cost of 10, whereas matching the lower segment saves a cost of 12. Thus, an optimal alignment matches as many lower segments as possible.

The roots of both trees must be matched in an optimal alignment, saving a cost of 2. The $|C'|$ lower segments of $T_{1,i}$ such that $c_i \in C'$ are all matched, saving a cost of $12|C'|$. The $n - |C'|$ upper segments of $T_{1,i}$ such that $c_i \notin C'$ are all matched, saving a cost of $10(n - |C'|)$. Therefore $v(\mathscr{A}') = |T_1| + |T_2| - 12|C'| - 10(n - |C'|) - 2 = |T_1| + |T_2| - 2(|C'| + 5n + 1)$. □

Given an alignment $\mathscr{A}''$ of $T_1$ and $T_2$, we can construct, in polynomial time, an exact partial cover $C''$ as follows: for subtree $T_{1,i}$ of $T_1$, if its entire lower segment is matched with the corresponding identical parts of $T_2$, let $c_i$ be in $C''$. We call this polynomial-time algorithm $g$.

**Lemma 5.** $v(\mathscr{A}'') \geq |T_1| + |T_2| - 2(|C''| + 5n + 1)$.

**Proof.** With $|C''|$ subtrees of $T_{1,i}$ matched their entire lower segments, the best way to align the rest of the subtrees would be to align their entire upper segments. Therefore we have $v(\mathscr{A}'') \geq |T_1| + |T_2| - 2(6|C''| + 5(n - |C''|) + 1) = |T_1| + |T_2| - 2(|C''| + 5n + 1)$. □

**Theorem 6.** *Computing the optimal alignment of two unordered trees is* MAX SNP-*hard if one of the trees can have an arbitrary degree.*

**Proof.** We will show that $(f, g)$ defined above forms an L-reduction. Assume $I$ is an instance of MAX 3SC-3. We have to show the following inequalities:

(1) $OPT(f(I)) \leq \alpha OPT(I)$ for some constant $\alpha$.

By Lemma 4, $OPT(f(I)) = |T_1| + |T_2| - 2(OPT(I) + 5n + 1) = 11n + n - 1 + 2m + 5n + 1 - 2(OPT(I) + 5n + 1) \leq 13n - 2 \cdot OPT(I) - 2$.

Since each element $a \in A$ is contained in at most three of the subsets in $C$, there is an exact partial cover of size at least $n/7$. This exact partial cover $C_1$ can be constructed as follows: pick an arbitrary subset $c = \{a_i, a_j, a_k\}$ from $C$, put it in $C_1$, delete from $C$ the subset containing $a_i$, $a_j$, or $a_k$. We can delete at most six subsets. Repeat above process until there is no subset left in $C$. Now $C_1$ is an exact partial cover of $A$ and $n/7 \leq |C_1| \leq OPT(I)$.

Now $OPT(f(I)) = 13n - 2 \cdot OPT(I) - 2 \leqslant 13n \leqslant 91 \cdot OPT(I)$. Thus, we can let $\alpha = 91$ in the above inequality.

(2) For any alignment of $f(I)$ of cost $s_2$, we can in polynomial time find an exact partial cover of $I$ of size $s_1$ such that $|OPT(I) - s_1| \leqslant \beta |OPT(f(I)) - s_2|$ for some constant $\beta$.

Given an alignment $\mathscr{A}''$ of $f(I)$ of cost $s_2 = v(\mathscr{A}'')$, by Lemma 5 we can construct, by using $g$, an exact partial cover $C''$ of $I$ of size $s_1 = |C''|$ such that $s_2 = v(\mathscr{A}'') \geqslant |T_1| + |T_2| - 2(|C''| + 5n + 1) = |T_1| + |T_2| - 2(s_1 + 5n + 1)$.

Now, $s_2 - OPT(f(I)) \geqslant |T_1| + |T_2| - 2(s_1 + 5n + 1) - OPT(f(I)) = |T_1| + |T_2| - 2(s_1 + 5n + 1) - (|T_1| + |T_2| - 2(OPT(I) + 5n + 1)) = 2 \cdot (OPT(I) - s_1)$. Thus, $\beta = \frac{1}{2}$ in the above inequality. □

## 4. Conclusion

It is interesting to observe that the *tree inclusion* problem defined in [4] is actually a special case of alignment of trees. Again, let $\mu(a,b) = 0$ if $a = b$ or 2 if $a, b \neq \lambda$ and $a \neq b$, $\mu(a, \lambda) = 1$, and $\mu(\lambda, b) = 1$. Then $T_1$ is included in $T_2$ if and only if $D(T_1, T_2) = |T_2| - |T_1|$. The complexity of Algorithm 1 is slightly higher than that for ordered tree inclusion [4], but alignment of trees is more difficult than tree inclusion. Also, under the above cost/score scheme, an optimal edit from $T_1$ to $T_2$ yields a largest common subtree of $T_1$ and $T_2$, and an optimal alignment yields a smallest common supertree [16]. So, edit and alignment form a "dual" in this sense.

Whether the complexity of Algorithm 1 can be improved is a rather hard question. A direct approach would be to prove that the alignment distance between two forests satisfies either quadrangle inequality $(D(F_1[i], F_2[j_k, j_m]) + D(F_1[i], F_2[j_l, j_n]) \leqslant D(F_1[i], F_2[j_l, j_m]) + D(F_1[i], F_2[j_k, j_n])$, for all $k \leqslant l \leqslant m \leqslant n)$ or inverse quadrangle inequality $(D(F_1[i], F_2[j_k, j_m]) + D(F_1[i], F_2[j_l, j_n]) \geqslant D(F_1[i], F_2[j_l, j_m]) + D(F_1[i], F_2[j_k, j_n])$, for all $k \leqslant l \leqslant m \leqslant n)$. If this were true, one could reduce the complexity for computing $D(F_1[i], F_2[j_s, j_t])$ and $D(F_1[i_s, i_t], F_2[j])$, using a matrix search technique. Unfortunately, we can show that neither of these inequalities hold.

It is well known that the quadrangle inequality is true for sequence edit distance. Since a sequence can be considered as an ordered forest where each tree contains only one node, it is clear that inverse quadrangle inequality is not true. As for the quadrangle inequality, consider the counterexample in Fig. 6. Assume that $\mu(a,b) = 0$ if $a = b$ or 2 if $a, b \neq \lambda$ and $a \neq b$, $\mu(a, \lambda) = 1$, and $\mu(\lambda, b) = 1$. It is easy to see that $D(F_1, F_2) = 5$, $D(F_1, F_3) = 5$, $D(F_1, F_4) = 3$, $D(F_1, F_5) = 5$. Therefore, $D(F_1, F_3) + D(F_1, F_5) > D(F_1, F_4) + D(F_1, F_2)$ and quadrangle inequality does not hold.

It also remains open if the alignment and edit problems for unordered trees are approximable. At the moment, no nontrivial approximation algorithm is known for either problem.

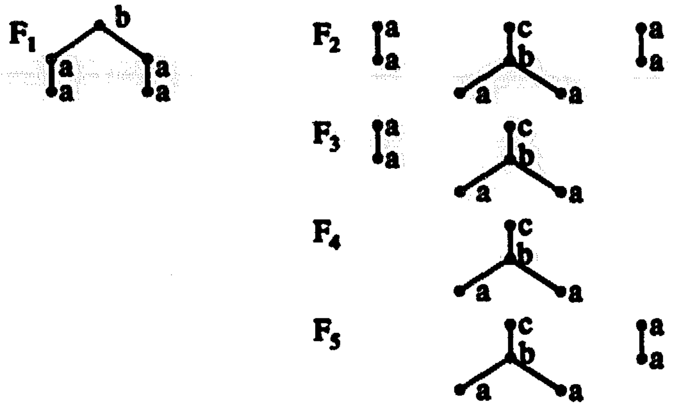

Fig. 6. Quadrangle inequality is not true: $D(F_1, F_3) + D(F_1, F_5) > D(F_1, F_4) + D(F_1, F_2)$.

## References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, Proof verification and hardness of approximation problems, in: *Proc. 33rd IEEE Symp. Found. Comp. Sci.* (1992) 14–23.

[2] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Freeman, New York, 1979).

[3] V. Kann, Maximum bounded 3-dimensional matching is MAX SNP-complete, *Inform. Process. Lett.* 37 (1991) 27–35.

[4] P. Kilpelainen and H. Mannila, Ordered and unordered tree inclusion, Report A-1991-4, Dept. of Comp. Sci. Univ. of Helsinki, August 1991; *SIAM J. Comput.*, to appear.

[5] S.-Y. Le, R. Nussinov and J.V. Maizel, Tree graphs of RNA secondary structures and their comparisons, *Comput. Biomed. Res.* 22 (1989) 461–473.

[6] S.-Y. Le, J. Owens, R. Nussinov, J.-H. Chen, B. Shapiro and J.V. Maizel, RNA secondary structures: comparison and determination of frequently recurring substructures by consensus, *Comp. Appl. Biosci.* 5 (1989) 205–210.

[7] S.Y. Lu, A tree-tree distance and its application to cluster analysis, *IEEE Trans. Pattern Anal. Mach. Intell.* 1 (1979) 219–224.

[8] C.H. Papadimitriou and M. Yannakakis, Optimization, approximation and complexity classes, *J. Comput. System Sci.* 43 (1991) 425–440.

[9] D. Sankoff and J. Kruskal (eds.), *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison* (Addison Wesley, Reading, MA, 1983).

[10] B. Shapiro, An algorithm for comparing multiple RNA secondary structures, *Comput. Appl. Biosci.* (1988) 387–393.

[11] B. Shapiro and K. Zhang, Comparing multiple RNA secondary structures using tree comparisons, *Comput. Appl. Biosci.* 6 (1990) 309–318.

[12] F.Y. Shih, Object representation and recognition using mathematical morphology model, *J. System Integration* 1 (1991) 235–256.

[13] F.Y. Shih and O.R. Mitchell, Threshold decomposition of grayscale morphology into binary morphology, *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-11 (1989) 31–42.

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

[15] Y. Takahashi, Y. Satoh, H. Suzuki and S. Sasaki, Recognition of largest common structural fragment among a variety of chemical structures, *Analyt. Sci.* 3 (1987) 23–28.

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

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

[18] K. Zhang, R. Statman and D. Shasha, On the editing distance between unordered labeled trees, *Inform. Process. Lett.* 42 (1992) 133–139.