Alignment Distance of Regular Tree Languages

Yo-Sub Han[1] and Sang-Ki Ko[2]

[1] Department of Computer Science, Yonsei University
50 Yonsei-Ro, Seodaemun-Gu, Seoul 120-749, Republic of Korea
emmous@yonsei.ac.kr
[2] Department of Computer Science, University of Liverpool
Liverpool L69 3BX, UK
sangkiko@liverpool.ac.uk

Abstract. We consider the tree alignment distance problem between a tree and a regular tree language. The tree alignment distance is an alternative of the tree edit-distance, in which we construct an optimal alignment between two trees and compute its cost instead of directly computing the minimum cost of tree edits. The alignment distance is crucial for understanding the structural similarity between trees.
We, in particular, consider the following problem: given a tree t and a tree automaton recognizing a regular tree language L, find the most similar tree from L with respect to t under the tree alignment metric. Regular tree languages are commonly used in practice such as XML schema or bioinformatics. We propose an $O(mn)$ time algorithm for computing the (ordered) alignment distance between t and L when the maximum degree of t and trees in L is bounded by a constant, and $O(mn^2)$ time algorithm when the maximum degree of trees in L is not bounded, where m is the size of t and n is the size of finite tree automaton for L. We also study the case where a tree is not necessarily ordered, and show that the time complexity remains $O(mn)$ if the maximum degree is bounded and MAX SNP-hard otherwise.

Keywords: tree alignment, alignment edit-distance, regular tree languages, tree automata

1 Introduction

Measuring the similarity or dissimilarity between tree-structured data is essential in many fields such as XML document processing [14], RNA secondary structure alignment [6], pattern recognition [11]. In particular, much attention has been paid to research on various metrics for defining the similarity or dissimilarity of trees [7, 15, 20]. For example, the *tree edit-distance* between two ordered trees is the cost of the optimal edit script required to transform one tree into the other and is a natural extension of the Levenshtein distance [10]—often called the *edit-distance* in the literature—defined for strings.

The tree edit-distance problem have been extensively studied by many researchers [4, 8, 15]. Given two trees t and t' of size m and n (namely, there are

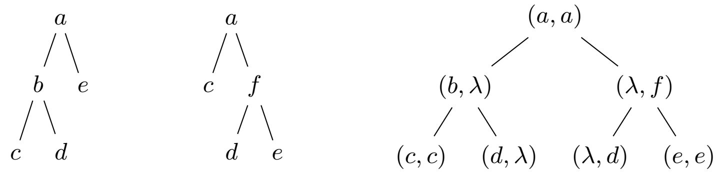

Fig. 1: Two trees t and t' and its optimal alignment $\mathcal{A}$

m nodes in t and n nodes in t'), the currently best known algorithm for computing the tree edit-distance between t and t' has been suggested by Demaine et al. [4] and runs in $O(m^2 n(1 + \log \frac{n}{m}))$ time, for $n \geq m$, using an optimal decomposition strategy. Similar questions for unordered trees also have been studied [18, 20]. Zhang et al. [20] showed that computing the tree edit-distance between unordered trees is NP-complete (in fact, MAX SNP-hard [19]).

Jiang et al. [7] introduced the *alignment distance* as an alternative to the tree edit-distance. Instead of considering the minimum number of tree editing operations, they considered the cost of an optimal tree alignment between two trees. They presented an $O(mnk^2)$ time algorithm for computing the alignment distance of two trees t and t', where m is the size of t, n is the size of t', and k is the maximum degree of t and t'. They also proved that computing the unordered alignment distance between two trees is MAX SNP-hard if the degree of one of the trees is not bounded. Lu et al. [13] proposed another constrained variant called the *less-constrained edit-distance* but Kuboyama et al. [9] proved that the less-constrained edit-distance is, in fact, equivalent to the alignment distance.

The alignment distance is useful in terms of visualization since we can obtain visualizable alignments for multiple trees whereas the tree edit-distance only cares optimal sequence of tree edits. See Fig. 1 for example. Höchsmann et al. [6] suggested a systematic approach for comparing RNA secondary structures based on the alignment distance since we can represent RNA secondary structures as trees by preserving their structural properties.

The problems of computing the tree edit-distance and its related variants have been extended to the case when we are given a tree t and a set L of trees—a regular tree language [2, 12]. Here we search for the most similar tree from L with respect to t under the considered distance metric. Note that in general L may be infinite and we need an efficient representation for such infinite L. Researchers suggested a regular tree grammar (RTG) and a tree automaton (TA) for recognizing a (infinite) set of trees preserving a certain regularity [3]. RTGs and TAs are widely used for denoting regular tree languages in several applications including XML schema [2], bioinformatics [16] and image recognition [11]. For example, we can formally define a set of RNA secondary structures excluding pseudoknots with a regular tree language. Xing [17] proposed an $O(mn \log n)$ time algorithm for computing the alignment distance between a tree and a regular tree grammar which recognizes a regular set of unranked trees. Unfortunately, the proposed algorithm cannot compute optimal alignments in all cases.

We extend the alignment distance problem to the alignment distance between a single tree and a regular tree language described as a TA. We separately consider two problems: the ranked case and the unranked case. For the ranked case where we fix the maximum degree of t and the maximum rank of A, we design an $O(mn)$ time algorithm for computing the alignment distance between a tree of size m and a ranked TA of size n. We also establish an $O(mn^2)$ time algorithm for unranked TAs.

We furthermore examine the unordered alignment distance between a tree and a regular tree language where the linear ordering of children is ignored. We show that the time complexity still remains polynomial by fixing the maximum degree of t and the maximum rank of A and otherwise, becomes MAX SNP-hard. The basic idea behind our algorithms is that we extend the classical dynamic programming algorithm [7] to operate with tree automata which can recognize regular sets of trees by their finite-state control. In order to employ the dynamic programming, we analyze the possible cases of alignments between a tree and a tree automaton and break the whole alignment problem into subproblems.

2 Preliminaries

A ranked alphabet Σ is a pair of a finite set of characters and a function $r : \Sigma \to rN \cup \{0\}$. We denote the set of elements of rank $m \geq 0$ by $\Sigma_m \subseteq \Sigma$. The set F_Σ consists of Σ-labelled trees, where a node labelled by $\sigma \in \Sigma_m$ for $m \geq 0$, always has m children. We denote the set of trees over Σ by F_Σ, which is the smallest set S satisfying the following condition: if $m \geq 0, \sigma \in \Sigma_m$ and $t, \ldots, t_m \in S$, then $\sigma(t, \ldots, t_m) \in S$.

A *nondeterministic bottom-up TA* A over a ranked alphabet Σ is specified by a tuple $A = (Q, \Sigma, F, \delta)$, where Q is a finite set of states, $F \subseteq Q$ is a set of final states, and δ associates to each $\sigma \in \Sigma_m$ a mapping $\sigma_\delta : Q^m \to 2^Q, m \geq 0$. For each tree $t = \sigma(t, \ldots, t_m) \in F_\Sigma$, we define inductively the set $t_\delta \subseteq Q$ by setting $q \in t_g$ if and only if there exist $q_i \in (t_i)_\delta$, for $1 \leq i \leq m$, such that $q \in \sigma_\delta(q_1, \ldots, q_m)$. Intuitively, t_δ consists of the states of Q that A may reach by reading the tree t. Thus, the tree language accepted by A is defined as follows: $L(A) = \{t \in F_\Sigma \mid t_\delta \cap F \neq \emptyset\}$. Given a state q of A, $A[q]$ denotes a new TA obtained from A by making $F = \{q\}$. We define the size $|A|$ of a ranked TA A to be $|Q| + \sum_{q \in \sigma_\delta(q_1, \ldots, q_m)} (r(\sigma) + 1)$.

Many modern applications of tree automata use automata operating on trees where the label of a node does not determine the number of children. For this reason we consider also unranked TAs.

A *nondeterministic unranked TA* is specified by a tuple $A = (\Sigma, Q, F, \delta)$, where Σ is an (unranked) alphabet, Q is a finite set of states, $F \subseteq Q$ is a set of final states, and δ is a transition relation defined in terms of horizontal languages that consist of regular sets of strings over Q. For each $q \in Q$ and $\sigma \in \Sigma$, we define $\delta(q, \sigma)$ to be the horizontal language associated with q and σ. We denote a finite-state automaton (FA) for the horizontal language $\delta(q, \sigma)$ of A by $H^A_{q,\sigma}$, which is called a *horizontal FA*. Note that an FA is specified by $A = (P, \Sigma, s, F, \delta)$, where

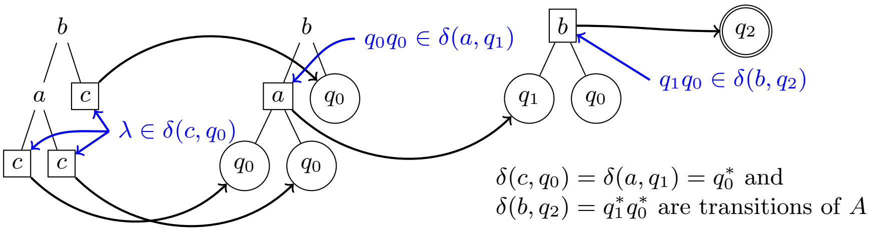
b
a
c
c
c
$\lambda \in \delta(c, q_0)$
b
$q_0q_0 \in \delta(a, q_1)$
a
q_0
q_0
q_0
b
q_2
q_1
q_0
$q_1q_0 \in \delta(b, q_2)$
$\delta(c, q_0) = \delta(a, q_1) = q_0^*$ and
$\delta(b, q_2) = q_1^* q_0^*$ are transitions of A

Fig. 2: An accepting run of an unranked TA $A = (\{q_0, q_1, q_2\}, \{a, b, c\}, \{q_2\}, \delta)$ for the tree t on the left-hand side. Note that horizontal languages are described as regular expressions.

P is a set of states, Σ is the input alphabet, s is the start state, $F \subseteq P$ is a set of final states, and δ is the transition function. Remind that horizontal FAs of the unranked TA $A = (\Sigma, Q, F, \delta)$ is defined over Q—the state set of A.

We denote an FA $H^A_{q,\sigma}[s_1, s_2] = (S_{q,\sigma}, Q, s_1, \{s_2\}, \gamma_{q,\sigma})$ obtained from $H^A_{q,\sigma}$ by having the initial state s_1 and the only final state s_2, where $s_1, s_2 \in S_{q,\sigma}$. Then, according to the transition relation δ, each $\sigma \in \Sigma$ defines a partial function $\sigma_\delta : Q^* \to Q$, where, for $w \in Q^*$, $q \in Q$, $q \in \sigma_\delta(w)$ if $w \in L(H^A_{q,\sigma})$. The transition relation is, in a natural way, extended as a binary relation on Σ-trees where some of the leaves can be labelled by elements of Q [3]. The tree language accepted by A is defined as follows: $L(A) = \{t \in T_\Sigma \mid t \xrightarrow{*} q_f \in F\}$. An accepting run of an unranked TA is described in Fig. 2. We define the size $|A|$ of an unranked TA A to be $|Q| + \sum_{q \in Q, \sigma \in \Sigma}(|H^A_{q,\sigma}| + 1)$. Naturally a ranked TA is a special case of an unranked TA, where for $\sigma \in \Sigma_m$ and $q \in Q$ we always have $L(H^A_{q,\sigma}) \subseteq Q^m$.

For a tree t, the *postorder traversal* of t is obtained by visiting all children in a left-to-right order and recursively visiting the subtrees rooted at the children, and then $\mathsf{root}(t)$. For a tree t, $t[i]$ denotes the ith node of t in postorder. When an ordering is specified for all nodes in a tree, the tree is called *ordered tree*. A *hedge* is a sequence of trees. We assume that all trees we discuss are ordered, unless explicitly stated otherwise. We denote a subhedge of a tree t that consists of the nodes from i to j by $t[i \dots j]$ where $i \leq j$. Here the nodes from i to j should satisfy one of the following two conditions: 1) a leaf or 2) all of its descendants are between i and j. We denote a hedge—a sequence $t_i, \dots, t_j$ of trees for $i \leq j$—by $h[t_i \dots t_j]$. A hedge formed from t by deleting the root node is denoted by $\hat{t}$. We denote the leftmost leaf descendant of node $t[i]$ by $l(i)$. Similarly, we denote the leftmost leaf descendant of a tree t by $l(t)$. We define $\mathsf{par}(t[i])$ to be the parent node of $t[i]$. Let $\mathsf{des}(t[i])$ be the set of all descendants of $t[i]$ including $t[i]$ itself. Thus, $t[l(i) \dots i]$ is the subtree rooted at $t[i]$, that is the subtree consisting of node i and all its descendants. Similarly, we define $\mathsf{anc}(t[i])$ to be the set of all ancestors of $t[i]$ including $t[i]$. We also denote the lowest common ancestor of $t[i]$ and $t[j]$ by $\mathsf{lca}(t[i], t[j])$. The size $|t|$ of t is the number of nodes in t and the degree $\mathsf{deg}(t)$ of t is the maximum number of children a node of t has. Let θ be the empty tree. We denote the character labelling a node $t[i]$ by $\sigma(i)$.

3 Distance Measures for Comparing Trees

Given an alphabet Σ, let $\Omega = \{(a \to b) \mid a, b \in \Sigma \cup \{\lambda\}\}$ be a set of edit operations. There are three edit operations: *deletion* $(a \to \lambda)$, *insertion* $(\lambda \to a)$ and *substitution* $(a \to b)$. We associate a non-negative edit cost to each edit operation $\omega_i \in \Omega$ as a function $\mathsf{c} : \Omega \to \mathbb{R}_+$. Note that the function c returns zero for the edit operations of trivial substitution $(a \to a)$, where $a \in \Sigma \cup \{\lambda\}$. We assume that c is a distance metric satisfying the following conditions:

1. $\mathsf{c}(a \to b) = 0$ if and only if $a = b$,
2. $\mathsf{c}(a \to b) = \mathsf{c}(b \to a)$, and
3. $\mathsf{c}(a \to c) \leq \mathsf{c}(a \to b) + \mathsf{c}(b \to c)$,

where $a, b, c \in \Sigma \cup \{\lambda\}$.

An *edit script* $S \in \Omega^*$ between two trees t and t' is a sequence of edit operations transforming t into t'. The cost $\mathsf{c}(S)$ of $S = s_1 s_2 \cdots s_n$ is $\mathsf{c}(S) = \sum_{i=1}^{n} \mathsf{c}(s_i)$. An *optimal edit script* between t and t' is an edit script of minimum cost and the minimum cost is the *tree edit-distance* between t and t'.

Definition 1. *We define the tree edit-distance $d(t, t')$ of two trees t and t' to be $d(t, t') = \min\{\mathsf{c}(S) \mid S$ is an edit script transforming t into $t'\}$. Namely, if S is an optimal edit script that transforms t into t', then $\mathsf{c}(S) = d(t, t')$.*

Let T and T' be sets of nodes in t and t', respectively. Define a triple (M, T, T') to be a mapping from t to t', where $M \subseteq T \times T'$ is a set of pairs of nodes (i, j) for $1 \leq i \leq |T|$ and $1 \leq j \leq |T'|$. We use M instead of (M, T, T') for simplicity when there is no confusion. We assume that trees are ordered in postorder. For any pair of (i_1, j_1) and (i_2, j_2) in M, the mapping M has the following restrictions:

1. $i_1 = i_2$ if and only if $j_1 = j_2$ (one-to-one)
2. $i_1 < i_2$ if and only if $j_1 < j_2$ (sibling order preserved)
3. $t[i_1] \in \mathsf{anc}(t[i_2])$ if and only if $t'[j_1] \in \mathsf{anc}(t'[j_2])$ (ancestor order preserved)

We consider the *alignment distance* between trees. Let t and t' be two labelled trees. We define an *alignment* $\mathcal{A}$ to be a tree where each node has a label from the set of edit operations. Let $\mathsf{left}(\mathcal{A})$ ($\mathsf{right}(\mathcal{A})$, resp.) be the left (right, resp.) projection of the alignment $\mathcal{A}$. Then, $\mathcal{A}$ is the alignment of t and t' if $\mathsf{left}(\mathcal{A}) = t$ and $\mathsf{right}(\mathcal{A}) = t'$. See Fig. 1 for example. We define the cost $\mathsf{c}(\mathcal{A})$ of an alignment $\mathcal{A}$ to be the sum of the costs of all pairs of labels in the alignment. In Fig. 1, the cost $\mathsf{c}(\mathcal{A})$ is 4 if we assume unit cost for all pairs in which two labels are different. We say that an alignment is optimal if the cost of the alignment is minimum over all possible alignments. Now we define the alignment distance between two trees as follows:

Definition 2. *We define the alignment distance $ad(t,t')$ of two trees t and t' to be $ad(t,t') = \min\{\mathsf{c}(\mathcal{A}) \mid \mathcal{A}$ is an alignment of t and $t'\}$. Note that the distance is symmetric $ad(t,t') = ad(t',t)$.*

We call a mapping corresponding to the alignment an *alignment mapping*. Now we formally define the additional restrictions required a mapping M to be an alignment mapping. For any triple of $(i_1,j_1),(i_2,j_2)$, and (i_3,j_3) in M, the alignment mapping M has the following additional restriction:

$$\text{if } \mathsf{lca}(t[i_1],t[i_3]) \in \mathsf{anc}(\mathsf{lca}(t[i_1],t[i_2])), \text{ then } \mathsf{lca}(t'[j_1],t'[j_3]) = \mathsf{lca}(t'[j_2],t'[j_3]).$$

Next we extend the alignment distance between trees to the distance between a tree and a set of trees—a tree language.

Definition 3. *We define the alignment distance $ad(t,L)$ between a tree t and a tree language L to be $ad(t,L) = \inf\{ad(t,t') \mid t' \in L\}$.*

We also consider an unordered variant for which we ignore the linear ordering of children called the *unordered alignment distance* and denote the distance between two trees t and t' by $uad(t,t')$.

4 Alignment Distance Problem

We study the alignment distance between a tree and a regular tree language. We tackle the following two cases separately: the ranked case and the unranked case. In the ranked case, we assume that the language is given by a ranked TA such that the number of children is fixed for every symbol in Σ. For the unranked case, the language is given by an unranked TA such that there is no restriction on the number of children. Our approach to these problems is based on the dynamic programming.

4.1 Ranked Case

We first establish the basis for our dynamic programming algorithm. For a tree t, we define the cost of a tree $\mathsf{c}(t)$ to be the minimum cost of inserting all nodes of the tree t. We denote the smallest cost among the costs of all trees in $L(A)$ by $\mathsf{mintree}(A) = \min\{\mathsf{c}(t) \mid t \in L(A)\}$. Then, given a tree t and a ranked TA $A = (\Sigma, Q, F, \Delta)$, we have the following equations for the basis:

1. $ad(\theta,\theta) = 0$,
2. $ad(t[1\ldots i],\theta) = ad(t[1\ldots i-1],\theta) + \mathsf{c}(\sigma(i),\lambda)$,
3. $ad(\theta, q_1 \ldots q_k) = \sum_{1\le i\le k} \mathsf{mintree}(A[q_i])$, where $q_i \in Q$ for $1 \le i \le k$.

It is straightforward to verify that the first two equations hold: The first case is when no edit operation is required and the second case is when we insert a node in a hedge $t[1\ldots i-1]$ to transform into $t[1\ldots i]$. Notice that $ad(\theta, t[1\ldots i]) = i$

if we assume unit cost for all edit-operations. For the third case, $ad(q_1 \dots q_k, \theta)$ is the alignment distance between an empty tree and the smallest hedge accepted by the sequence of states. Now we are ready to present a recursive formula of the distance between a sequence of states and a subhedge.

Lemma 4. *Given a tree t and a state q of a ranked TA $A = (\Sigma, Q, F, \delta)$, the alignment distance $ad(t,q)$ can be computed as follows:*

$$ad(t,q) = \min_{\substack{1 \le i_1 \le l;\\ 1 \le i_2 \le k;\\ q \in \sigma_\delta(q_1,\dots,q_k)}} \begin{cases} ad(t,\theta) + ad(t_{i_1}, q) - ad(t_{i_1}, \theta), \\ ad(\theta, q) + ad(t, q_{i_2}) - ad(\theta, q_{i_2}), \\ ad(h[t_1 \dots t_l], q_1 \dots q_k) + \mathsf{c}(\mathsf{root}(t) \to \sigma), \end{cases} \tag{1}$$

where $q_n \in Q$ for $1 \le n \le k$ and $\hat{t} = h[t_1 \dots t_l]$.

Proof. We prove the recurrence by considering an optimal alignment $\mathcal{A}_{t,t'}$ between two trees t and t', where t' is a tree accepted by reaching the state q in A. Especially, we consider the possible root node cases of the optimal alignment $\mathcal{A}_{t,t'}$ between t and t'. There are three possible cases to consider:

Case 1: The root node of $\mathcal{A}_{t,t'}$ is $(\mathsf{root}(t), \lambda)$. Then, the root node $\mathsf{root}(t')$ of t' is aligned with a descendant of t. Otherwise, $\mathcal{A}_{t,t'}$ has a node $(\lambda, \mathsf{root}(t'))$ and we can always have a better alignment by replacing $(\mathsf{root}(t), \lambda)$ and $(\lambda, \mathsf{root}(t'))$ with $(\mathsf{root}(t), \mathsf{root}(t'))$, which is considered in **Case 3**. Suppose that $\mathsf{root}(t')$ is aligned with a node in the ith subtree of t. Then the cost of the alignment can be written as follows: $\min_{1 \le i \le l}\{ad(t,\theta) + ad(t_i, t') - ad(t_i, \theta)\}$.

Since t' is accepted by reaching the state q in A, the first term in the recurrence captures this case.

Case 2: The optimal alignment $\mathcal{A}_{t,t'}$ has $(\lambda, \mathsf{root}(t'))$ as a root node. This case is completely symmetric with **Case 1** and described in the second term.

Case 3: The root node of $\mathcal{A}_{t,t'}$ is $(\mathsf{root}(t), \mathsf{root}(t'))$. Since the root nodes of two trees are aligned, it remains to compute an optimal alignment between two ordered sequences of trees under the root nodes. This also implies that we need to compute the alignment distance $ad(h[t_1 \dots t_l], h[t'_1 \dots t'_k])$ between two subhedges obtained from t and t' by removing the root nodes.

Now we define the alignment distance between two subhedges—two ordered sequences of trees. Since we are considering the alignment distance between a tree and a regular tree language, we use a sequence of states on the right-hand side in the distance function.

Lemma 5. *Given a subhedge $h[t_1 \dots t_l]$ of t and a sequence $q_1, \dots, q_k$ of states of a ranked TA $A = (\Sigma, Q, F, \delta)$, we can compute $ad(h[t_1 \dots t_l], q_1 \dots q_k)$ as follows:*

$$ad(h[t_1 \dots t_l], q_1 \dots q_k) =$$

$$\min_{\substack{1 \le i_1 \le l; \\ 1 \le i_2 \le k; \\ q_k \in \sigma_\delta(q'_1, \dots, q'_j)}} \begin{cases} ad(h[t_1 \dots t_l], q_1 \dots q_{k-1}) + ad(\theta, q_k), \\ ad(h[t_1 \dots t_{l-1}], q_1 \dots q_k) + ad(t_l, \theta), \\ ad(h[t_1 \dots t_{l-1}], q_1 \dots q_{i_2 - 1}) + ad(\hat{t_l}, q_{i_2} \dots q_k) + \mathsf{c}(\mathsf{root}(t_l) \to \lambda), \\ ad(h[t_1 \dots t_{i_1 - 1}], q_1 \dots q_{k-1}) + ad(h[t_{i_1} \dots t_l], q'_1 \dots q'_j) + \mathsf{c}(\lambda \to \sigma), \\ ad(h[t_1 \dots t_{l-1}], q_1 \dots q_{k-1}) + ad(\hat{t_l}, q'_1 \dots q'_j) + \mathsf{c}(\mathsf{root}(t_l) \to \sigma), \end{cases} \tag{2}$$

where $q_n, q'_m \in Q$ *for* $1 \le n \le k$ *and* $1 \le m \le j$.

Now we are ready to present an efficient algorithm for computing the alignment distance between a tree t and a regular tree language $L(A)$ described by a ranked TA A.

Theorem 6. *Given a tree* t *and a ranked TA* $A = (\Sigma, Q, F, \delta)$*, we can compute* $ad(t, L(A))$ *in* $O(mnk^2)$ *in the worst-case, where* $m = |t|$*,* $n = |A|$ *and* $k = \mathsf{deg}(t) + \max\{r(\sigma) \mid \sigma \in \Sigma\}$.

Proof. Notice that we need to compute the alignment distance for each subhedge $t[l(i) \dots i-1]$ of t and transition $q \in \sigma_g(q_1, \dots, q_l)$ of A. Let m_i be the number of children of a node $t[i]$ and denote the subhedge $t[l(i) \dots i-1]$ by $h[t_1 \dots t_{m_i}]$ for notational convenience.

Let us analyze the time complexity for computing $ad(h[t_1 \dots t_{m_i}], q_1 \dots q_l)$, which is the alignment distance between a sequence of states of length l and a hedge that consists of m_i trees. There are in total $m_i^2 l + m_i l^2$ values to compute and computing each ad value takes $O(m_i n_\delta + l)$ time in the worst-case, where n_δ is the number of transitions of A. Since $m_i + l \le k$, the time complexity required to compute each ad value is bounded by $O(n)$, where n is the size of A.

Hence, the total time complexity is bounded by

$$\sum_{i=1}^{m} O((m_i l) \cdot (m_i + l) \cdot n) \le \sum_{i=1}^{m} O(m_i k^2 n) \le O(mnk^2).$$

Note that if both the degree of t and the rank of A are bounded by a constant, the time complexity for computing $ad(t, L(A))$ is $O(mn)$. □

4.2 Unranked Case

We consider the case when we have an unranked TA for a regular tree language of unranked trees. Contrary to the ranked TAs where we consider a sequence of states in each bottom-up computation, we instead consider a horizontal language that contains a set of sequences of states. Let $A = (\Sigma, Q, F, \delta)$ be an unranked TA, $q \in Q$ and $\sigma \in \Sigma$. We define the alignment distance between a horizontal language $\delta(q, \sigma)$ and a subhedge $h[t_1 \dots t_l]$ of t as follows:

$$ad(h[t_1 \dots t_l], \delta(q, \sigma)) = \min\{\delta(h[t_1 \dots t_l], w) \mid w \in \delta(q, \sigma)\}.$$

Note that we use the definition for the alignment distance between a sequence of states and a subhedge given in Lemma 5.

We also define the following notation that is essential for computing the alignment distance $ad(h[t_1 \ldots t_l], \delta(q,\sigma))$ between a subhedge and a horizontal language. Let $H^A_{q,\sigma} = (S_{q,\sigma}, Q, s_{q,\sigma}, F_{q,\sigma}, \gamma_{q,\sigma})$ be a horizontal FA that accepts $\delta(q,\sigma)$, namely, $L(H^A_{q,\sigma}) = \delta(q,\sigma)$. We define the alignment distance between a subhedge $h[t_1 \ldots t_l]$ and two horizontal states $s_1, s_2 \in S_{q,\sigma}$ as follows:

$$ad(h[t_1 \ldots t_l], [s_1, s_2]) = \min\{ad(h[t_1 \ldots t_l], w) \mid w \in L(H^A_{q,\sigma}[s_1, s_2])\}.$$

Let $H^A_{q,\sigma} = (S_{q,\sigma}, Q, s_{q,\sigma}, F_{q,\sigma}, \gamma_{q,\sigma})$ be a horizontal FA. Then, the following holds: $ad(h[t_1 \ldots t_l], \delta(q,\sigma)) = \min\{ad(h[t_1 \ldots t_l], [s_{q,\sigma}, f_{q,\sigma}]) \mid f_{q,\sigma} \in F_{q,\sigma}\}$. Now we are ready to establish the alignment distance between a tree t and a state q of an unranked TA A as follows:

Lemma 7. *Given a tree t and a state q of an unranked TA $A = (\Sigma, Q, F, \delta)$, the alignment distance $ad(t,q)$ can be computed as follows:*

$$ad(t,q) = \min_{\substack{w \in \delta(q,\sigma);\\ 1 \le i_1 \le l;\\ 1 \le i_2 \le |w|;}} \begin{cases} ad(t,\theta) + ad(t_{i_1}, q) - ad(t_{i_1}, \theta), \\ ad(\theta, q) + ad(t, w_{i_2}) - ad(t_{i_1}, w_{i_2}), \\ ad(h[t_1 \ldots t_l], w) + \mathsf{c}(\mathsf{root}(t) \to \sigma), \end{cases} \tag{3}$$

where $\hat{t} = h[t_1 \ldots t_l]$.

We can see that the computation of Equation (3) requires the computation of the alignment distance between a subhedge of t and a horizontal language $\delta(q,\sigma)$.

Lemma 8. *Let $A = (\Sigma, Q, F, \delta)$ be an unranked TA and $H^A_{q,\sigma} = (S_{q,\sigma}, Q, s_{q,\sigma}, F_{q,\sigma}, \gamma_{q,\sigma})$ be a horizontal FA of A associated with a state $q \in Q$ and $\sigma \in \Sigma$.Given a subhedge $h[t_1 \ldots t_l]$ of t and two horizontal states s_1, s_2 of $H^A_{q,\sigma}$, the alignment distance $ad(h[t_1 \ldots t_l], [s_1, s_2])$ can be computed as follows: $ad(h[t_1 \ldots t_l], [s_1, s_2]) =$*

$$\min_{\substack{1 \le i \le l;\\ w \in \delta(q',\sigma);\\ s_2 \in \gamma_{q,\sigma}(s',q')}} \begin{cases} ad(h[t_1 \ldots t_l], [s_1, s']) + ad(\theta, q'), \\ ad(h[t_1 \ldots t_{l-1}], [s_1, s_2]) + ad(t_l, \theta), \\ ad(h[t_1 \ldots t_{l-1}], [s_1, r]) + ad(\hat{t_l}, [r, s_2]) + \mathsf{c}(\mathsf{root}(t_l) \to \lambda), \\ ad(h[t_1 \ldots t_{i-1}], [s_1, s']) + ad(h[t_i \ldots t_l], w) + \mathsf{c}(\lambda \to \sigma), \\ ad(h[t_1 \ldots t_{l-1}], [s_1, s']) + ad(\hat{t_l}, w) + \mathsf{c}(\mathsf{root}(t_l) \to \sigma), \end{cases}$$

where $s', r \in S_{q,\sigma}$ and $q' \in Q$.

Now we describe how we compute the alignment distance for the unranked case. We use the weighted directed graph for computing $ad(h[t_1 \ldots t_l], \delta(q,\sigma))$ between a hedge $h[t_1 \ldots t_l]$ and a horizontal language $\delta(q,\sigma)$.

Let $H^A_{q,\sigma} = (S_{q,\sigma}, Q, s_{q,\sigma}, F_{q,\sigma}, \gamma_{q,\sigma})$ be a horizontal FA recognizing $\delta(q,\sigma)$. We construct a weighted directed graph $\mathcal{W}(h[t_1 \ldots t_l], H^A_{q,\sigma}) = (V,E)$ where $V = S_{q,\sigma} \times \{0,1,\ldots,l\}$ is a set of vertices and $E \subseteq V \times \mathbb{N}_0 \times V$ is a set of weighted directed edges. For each transition $s_2 \in \gamma_{q,\sigma}(s_1, q)$ of $H^A_{q,\sigma}$, we define E to contain the following edges:

- $((s_1,i), ad(\theta,q), (s_2,i))$ for $0 \leq i \leq l$,
- $((s_1,i), ad(h[t_{i+1} \ldots t_j], \delta(q,\sigma)) + \mathsf{c}(\lambda \to \sigma), (s_2,j))$ for $0 \leq i < j \leq l$, and
- $((s_1,i), ad(\hat{t_{i+1}}, \delta(q,\sigma)) + \mathsf{c}(\mathsf{root}(t_{i+1}) \to \sigma), (s_2,i+1))$ for $0 \leq i \leq l-1$.

For each state $s \in S_{q,\sigma}$, we also define E to contain the following edges:

- $((s,i), ad(t_{i+1},\theta), (s,i+1))$ for $0 \leq i \leq l-1$.

Finally, for each pair of states $s_1, s_2 \in S_{q,\sigma}$, we add the following edges to E:

- $((s_1,i), ad(\hat{t_{i+1}}, [s_1,s_2]) + \mathsf{c}(\mathsf{root}(t_{i+1}) \to \lambda), (s_2,i+1))$ for $0 \leq i \leq l-1$.

By the construction, we know that the cost of the minimum cost path from (s_1,i) to (s_2,j), where $1 \leq i \leq j \leq l$, in $\mathcal{W}(h[t_1 \ldots t_l], H^A_{q,\sigma})$ implies $ad(h[t_{i+1} \ldots t_j], [s_1,s_2])$. Now we are ready to present a polynomial time algorithm in the unranked case.

Theorem 9. *Given a tree t and an unranked TA $A = (\Sigma, Q, F, \delta)$, we can compute $ad(t, L(A))$ in $O(mn^2k^2)$ in the worst-case, where $m = |t|$, $n = |A|$, and $k = \mathsf{deg}(t)$.*

Proof. We can compute the alignment distance between a horizontal language and a hedge by constructing a weighted directed graph and computing the minimum cost path from $(s_{q,\sigma}, 0)$ to $(f_{q,\sigma}, l)$ where $f_{q,\sigma} \in F_{q,\sigma}$.

Given that the size of $H^A_{q,\sigma}$ has x states and y transitions, the construction of the weighted directed graph $\mathcal{W}(h[t_1 \ldots t_l], H^A_{q,\sigma})$ yields $O(xl)$ vertices and $O(x^2l^2)$ edges. Note that Dijkstra's algorithm based on a min-priority queue for finding the minimum cost path runs in $O(|V|\log|V|+|E|)$ time [5] for a graph $G = (V,E)$ where V is a set of vertices and E is a set of edges. Therefore, we can find the minimum cost path in $\mathcal{W}(h[t_1 \ldots t_l], H^A_{q,\sigma})$ in $O(x^2l^2)$ time. Since we construct $\mathcal{W}(h[t_i \ldots t_l], H^A_{q,\sigma})$ for all $q \in Q, 1 \leq i \leq l$ and compute the minimum cost path, the total time complexity is upper bounded by

$$\sum_{i=1}^{m} \sum_{q \in Q} O(m_i \cdot x^2 m_i^2) \leq \sum_{i=1}^{m} \sum_{q \in Q} O(m_i \cdot x^2 k^2) \leq \sum_{q \in Q} O(mx^2k^2) \leq O(mn^2k^2).$$

Note that if the degree of t is bounded by a constant, the time complexity is upper bounded by $O(mn^2)$. □

5 Unordered Alignment Distance Problem

We study the unordered version of the alignment distance problem. The main difference from the ordered case is that here we treat sequences of trees (resp., sequences of states) as sets of trees (resp., multisets of states) because we do not care about the order of nodes.

Lemma 10. *Given a set $T_{1,l}$ of subtrees of t and a multiset $Q_{1,k}$ of states of a ranked TA $A = (\Sigma, Q, F, \delta)$, the unordered alignment distance $uad(T_{1,l}, Q_{1,k})$ can be computed as follows: $uad(T_{1,l}, Q_{1,k}) =$*

$$\min_{\substack{1 \le i_1 \le l;\\ 1 \le i_2 \le k;\\ q_{i_2} \in \sigma_\delta(q'_1, \ldots, q'_j)}} \begin{cases} uad(T_{1,l} \setminus \{t_{i_1}\}, Q_{1,k} \setminus Q') + uad(\hat{t_{i_1}}, Q') + \mathsf{c}(\mathsf{root}(t_{i_1}) \to \lambda),\\ uad(T_{1,l} \setminus T, Q_{1,k} \setminus \{q_{i_2}\}) + uad(T, P) + \mathsf{c}(\lambda \to \sigma),\\ uad(T_{1,l} \setminus \{t_{i_1}\}, Q_{1,k} \setminus \{q_{i_2}\}) + uad(\hat{t_{i_1}}, P) + \mathsf{c}(\mathsf{root}(t_{i_1}) \to \sigma), \end{cases}$$

where $T \subseteq T_{1,l}, Q' \subseteq Q_{1,k}$, and $P = \{q'_1, \ldots, q'_j\}$.

Proof. This lemma is an unordered extension of Lemma 5. The main difference between the ordered and unordered versions is that we treat sequences of subtrees (resp., sequences of states) as sets of trees (resp., multisets of states). Notice that a multiset (or bag) is a generalization of a set in which multiple instances of elements are allowed. Consider a transition $q \in \sigma_\delta(q_1, \ldots, q_k)$ of A. We cannot convert the sequence $q_1, \ldots, q_k$ of states into the set $\{q_1, \ldots, q_k\}$ as we may have multiple instances of a state. Therefore, we replace a sequence $q_1, \ldots, q_k$ of states by a multiset $\{q_1, \ldots, q_k\}$ of states and a hedge $h[t_1 \ldots t_l]$ by a set $\{t_1, \ldots, t_l\}$ of trees. For the sake of simplicity, we denote $\{q_1, \ldots, q_k\}$ by $Q_{1,k}$ and $\{t_1, \ldots, t_l\}$ by $T_{1,l}$.

For sequences of trees, it is enough to use sets as there is no chance of containing multiple instances of the same subtree. We also mention that the hedge $\hat{t_{i_1}}$ used in the equation also denotes the set of subtrees of t_{i_1}, not the sequence of subtrees of t_{i_1}. □

Based on Lemma 10, we design an algorithm that computes the unordered alignment distance $uad(T_{1,l}, Q_{1,k})$ between a set of subtrees of t and a multiset of states.

Theorem 11. *Given a tree t and a ranked TA $A = (\Sigma, Q, F, \delta)$, we can compute $uad(t, L(A))$ in $O(mn2^k)$ in the worst-case, where $m = |t|$, $n = |A|$ and $k = \mathsf{deg}(t) + \max\{r(\sigma) \mid \sigma \in \Sigma\}$.*

Proof. Note that the algorithm for computing the unordered alignment distance follows almost the same procedure except that we use the recurrence given in Lemma 10 instead of Lemma 5. This gives rise to the following time complexity:

$$\sum_{i=1}^{m} O(m_i n \cdot 2^{m_i + l}) \le \sum_{i=1}^{m} O(m_i n 2^k) \le O(mn2^k).$$

Note that the time complexity remains polynomial if we fix k to be a constant. □

Given two trees t and t', it is known that computing $uad(t, t')$ is MAX SNP-hard [7] if the degree of one of the two trees is not bounded by a constant. This means that unless P=NP there is no polynomial-time approximation scheme for the problem [1]. We can immediately obtain the following results.

Corollary 12. *Let t be a tree and A be a ranked TA. Then, we can compute the unordered alignment distance $uad(t, L(A))$ in $O(mn)$ time if $\mathsf{deg}(t) = k$ for some $k < \infty$, otherwise, the problem is MAX SNP-hard.*

Corollary 13. *Let t be a tree and A be an unranked TA. Then, the problem of computing the unordered alignment distance $uad(t, L(A))$ is MAX SNP-hard.*

References

1. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.
2. E. R. Canfield and G. Xing. Approximate matching of XML document with regular hedge grammar. *International Journal of Computer Mathematics*, 82(10):1191–1198, 2005.
3. H. Comon, M. Dauchet, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. 2007.
4. E. D. Demaine, S. Mozes, B. Rossman, and O. Weimann. An optimal decomposition algorithm for tree edit distance. *ACM Transactions on Algorithms*, 6(1):2:1–2:19, 2009.
5. M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.
6. M. Höchsmann, T. Töller, R. Giegerich, and S. Kurtz. Local similarity in RNA secondary structures. In *Proceedings of the 2nd IEEE Computer Society Conference on Bioinformatics*, pages 159–168, 2003.
7. T. Jiang, L. Wang, and K. Zhang. Alignment of trees — an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148, 1995.
8. P. N. Klein. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms*, pages 91–102, 1998.
9. T. Kuboyama, K. Shin, T. Miyahara, and H. Yasuda. A theoretical analysis of alignment and edit problems for trees. In *Proceedings of the 9th Italian Conference on Theoretical Computer Science*, pages 323–337, 2005.
10. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
11. D. López and S. España. Error-correcting tree language inference. *Pattern Recognition Letters*, 23(1-3):1–12, 2002.
12. D. López, J. M. Sempere, and P. García. Error correcting analysis for tree languages. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(03):357–368, 2000.
13. C. L. Lu, Z.-Y. Su, and C. Y. Tang. A new measure of edit distance between labeled trees. In *Proceedings of the 7th Annual International Conference on Computing and Combinatorics*, pages 338–348, 2001.

14. A. Nierman and H. V. Jagadish. Evaluating structural similarity in XML documents. In *Proceedings of the 5th International Workshop on the Web and Databases*, pages 61–66, 2002.
15. K.-C. Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433, 1979.
16. B. Voß, R. Giegerich, and M. Rehmsmeier. Complete probabilistic analysis of RNA shapes. *BMC Biology*, 4(1):1–23, 2006.
17. G. Xing. Approximate matching of XML documents with schemata using tree alignment. In *Proceedings of the 2014 ACM Southeast Regional Conference*, pages 43:1–43:4, 2014.
18. K. Zhang. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15(3):205–222, 1996.
19. K. Zhang and T. Jiang. Some MAX SNP-hard results concerning unordered labeled trees. *Information Processing Letters*, 49(5):249–254, 1994.
20. K. Zhang, R. Statman, and D. Shasha. On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3):133–139, 1992.