Position Papers of the Federated Conference on Computer Science and Information Systems pp. 19–25
DOI: 10.15439/2019F338
ISSN 2300-5963 ACSIS, Vol. 19

Alignment for Rooted Labeled Caterpillars

Yoshiyuki Ukita
Graduate School of Computer Science and Systems Engineering
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820-8502, Japan
Email: o231014y@mail.kyutech.jp

Takuya Yoshino, Kouichi Hirata*
Department of Artificial Intelligence
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820-8502, Japan
Email: {yoshino,hirata}@ai.kyutech.ac.jp

***Abstract*—A *rooted labeled caterpillar* (*caterpillars*, for short) is a rooted labeled tree transformed to a rooted path after removing all the leaves in it. In this paper, we design the algorithm to compute the *alignment* distance between caterpillars in $O(h^2\lambda^3)$ time under the general cost function and in $O(h^2\lambda)$ time under the unit cost function, where h is the maximum height and λ is the maximum number of leaves in caterpillars.**

I. Introduction

COMPARING tree-structured data such as HTML and XML data for web mining or RNA and glycan data for bioinformatics is one of the important tasks for data mining. The most famous distance measure [2] between *rooted labeled unordered trees* (*trees*, for short) is the *edit distance* [10]. The edit distance is formulated as the minimum cost of *edit operations*, consisting of a *substitution*, a *deletion* and an *insertion*, applied to transform a tree to another tree. It is known that the edit distance is always a metric and coincides with the minimum cost of *Tai mappings* [10].

Unfortunately, the problem of computing the edit distance between trees is MAX SNP-hard [15]. This statement also holds even if trees are binary or the maximum height of trees is at most 3 [1], [4].

A *caterpillar* (*cf.* [3]) is a tree transformed to a rooted path after removing all the leaves in it. Whereas the caterpillars are very restricted and simple, there are some cases containing many caterpillars in real dataset, see Table I in Appendix. Recently, Muraka *et al.* [8] have proposed the algorithm to compute the edit distance between caterpillars in $O(h^2\lambda^3)$ time under the general cost function and in $O(h^2\lambda)$ time under the unit cost function, where h is the maximum height and λ is the maximum number of leaves in caterpillars[1]. They have also introduced the efficient comparable distances to approximate the edit distance between caterpillars [9].

An *alignment distance* is an alternative distance measure between trees, introduced by Jiang *et al.* [5]. The alignment distance between two trees is formulated as the minimum cost of possible *alignments* (as trees) obtained by first inserting nodes labeled with spaces into two trees so that the resulting trees have the same structure and then overlaying them. In operational, the alignment distance is regarded as an edit distance such that every insertion precedes to deletions. Hence, the alignment distance between trees is not always equal to the edit distance and regarded as a variation of the edit distance. Furthermore, Kuboyama [6] has shown that the alignment distance coincides with the minimum cost of *less-constrained mappings* [7], which is the restriction of the Tai mapping.

As same as the edit distance, the problem of computing the alignment distance between trees is also MAX SNP-hard [5]. On the other hand, it is tractable if the degrees are bounded by some constant [5]. Since a caterpillar is not a bounded-degree tree, it is still open whether or not the problem of computing the alignment distance is tractable,

In this paper, first we point out that there exists a pair of caterpillars whose minimum cost less-constrained mapping is not an isolated-subtree mapping and whose minimum cost Tai mapping is not a less-constrained mapping. Then, we can apply the algorithm of computing neither the isolated-subtree distance or its variations [12], [13], [14], [16] nor the edit distance [8] to compute the alignment distance between caterpillars.

Next, we design the algorithm to compute the alignment distance between caterpillars in $O(h^2\lambda^3)$ time under the general cost function and in $O(h^2\lambda)$ time under the unit cost function. Here, it is necessary to adopt the edit distance for multisets (*cf.*, [9]) to compute the alignment distance between sets of leaves. Furthermore, as same as the edit distance [8], we point out the structural restriction of caterpillars provides the limitation of tractable computing of the alignment distance for unordered trees.

II. Preliminaries

In this section, we prepare the notions necessary to discuss the later sections.

A *tree* T is a connected graph (V, E) without cycles, where V is the set of vertices and E is the set of edges. We denote V and E by $V(T)$ and $E(T)$. The *size* of T is $|V|$ and denoted by $|T|$. We sometime denote $v \in V(T)$ by $v \in T$. We denote an empty tree $(\emptyset, \emptyset)$ by $\emptyset$. A *rooted tree* is a tree with one node r chosen as its *root*. We denote the root of a rooted tree T by $r(T)$.

Let T be a rooted tree such that $r = r(T)$ and $u, v, w \in T$. We denote the unique path from r to v, that is, the tree

*The author would like to express thanks for support by Grant-in-Aid for Scientific Research 17H00762, 16H02870 and 16H01743 from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

[1]This time complexity is different from the result in [8], because it contains some errors. See Appendix.

(V', E') such that $V' = \{v_1, \ldots, v_k\}$, $v_1 = r$, $v_k = v$ and $(v_i, v_{i+1}) \in E'$ for every i $(1 \leq i \leq k-1)$, by $UP_r(v)$.

The *parent* of $v(\neq r)$, which we denote by $par(v)$, is its adjacent node on $UP_r(v)$ and the *ancestors* of $v(\neq r)$ are the nodes on $UP_r(v) - \{v\}$. We say that u is a *child* of v if v is the parent of u and u is a *descendant* of v if v is an ancestor of u. We denote the set of children of v by $ch(v)$. We call a node with no children a *leaf* and denote the set of all the leaves in T by $lv(T)$.

The *degree* of v, denoted by $d(v)$, is the number of children of v, and the *degree* of T, denoted by $d(T)$, is $\max\{d(v) \mid v \in T\}$. The *height* of v, denoted by $h(v)$, is $\max\{|UP_v(w)| \mid w \in lv(T[v])\}$, and the *height* of T, denoted by $h(T)$, is $\max\{h(v) \mid v \in T\}$.

We use the ancestor orders $<$ and $\leq$, that is, $u < v$ if v is an ancestor of u and $u \leq v$ if $u < v$ or $u = v$. We say that w is the *least common ancestor* of u and v, denoted by $u \sqcup v$, if $u \leq w$, $v \leq w$ and there exists no node $w' \in T$ such that $w' \leq w$, $u \leq w'$ and $v \leq w'$. Let T be a rooted tree (V, E) and v a node in T. A *complete subtree of T at v*, denoted by $T[v]$, is a rooted tree $T' = (V', E')$ such that $r(T') = v$, $V' = \{u \in V \mid u \leq v\}$ and $E' = \{(u, w) \in E \mid u, w \in V'\}$.

We say that u is *to the left of* v in T if $pre(u) \leq pre(v)$ for the preorder number *pre* in T and $post(u) \leq post(v)$ for the postorder number *post* in T. We say that a rooted tree is *ordered* if a left-to-right order among siblings is given; *unordered* otherwise. We say that a rooted tree is *labeled* if each node is assigned a symbol from a fixed finite alphabet Σ. For a node v, we denote the label of v by $l(v)$, and sometimes identify v with $l(v)$. In this paper, we call a rooted labeled unordered tree a *tree* simply. Furthermore, we call a set of trees a *forest*.

As the restricted form of trees, we introduce a *rooted labeled caterpillar* (*caterpillar*, for short) as follows, which this paper mainly deals with.

Definition 1 (Caterpillar (cf., [3])): We say that a tree is a *caterpillar* if it is transformed to a rooted path after removing all the leaves in it. For a caterpillar C, we call the remained rooted path a *backbone* of C and denote it by $bb(C)$.

It is obvious that $r(C) = r(bb(C))$ and $V(C) = bb(C) \cup lv(C)$ for a caterpillar C, that is, every node in a caterpillar is either a leaf or an element of the backbone.

Next, we introduce an *edit distance* and a *Tai mapping* between trees.

Definition 2 (Edit operations for trees [10]): The *edit operations* of a tree T are defined as follows, see Figure 1.

1) *Substitution*: Change the label of the node v in T.
2) *Deletion*: Delete a node v in T with parent v', making the children of v become the children of v'. The children are inserted in the place of v as a subset of the children of v'. In particular, if v is the root in T, then the result applying the deletion is a forest consisting of the children of the root.
3) *Insertion*: The complement of deletion. Insert a node v as a child of v' in T making v the parent of a subset of the children of v'.

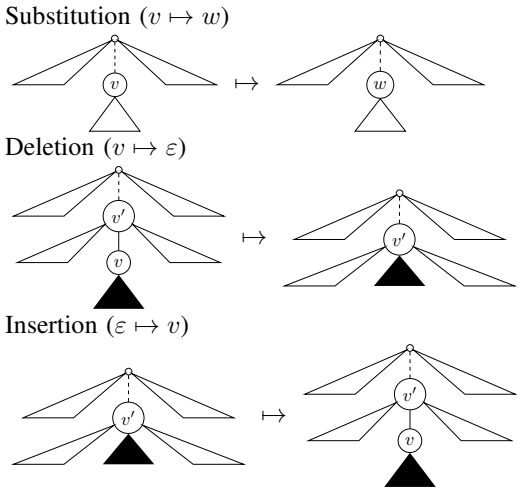

Fig. 1. Edit operations for trees.

Let $\varepsilon \notin \Sigma$ denote a special *blank* symbol and define $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$. Then, we represent each edit operation by $(l_1 \mapsto l_2)$, where $(l_1, l_2) \in (\Sigma_\varepsilon \times \Sigma_\varepsilon - \{(\varepsilon, \varepsilon)\})$. The operation is a substitution if $l_1 \neq \varepsilon$ and $l_2 \neq \varepsilon$, a deletion if $l_2 = \varepsilon$, and an insertion if $l_1 = \varepsilon$. For nodes v and w, we also denote $(l(v) \mapsto l(w))$ by $(v \mapsto w)$.

We define a *cost function* $\gamma : (\Sigma_\varepsilon \times \Sigma_\varepsilon \setminus \{(\varepsilon, \varepsilon)\}) \mapsto \mathbf{R}^+$ on pairs of labels. For $(v, w) \in V(T_1) \times V(T_2)$, we also denote $\gamma(l(v), l(w))$ by $\gamma(v, w)$ simply.

We often constrain a cost function γ to be a *metric*, that is, $\gamma(l_1, l_2) \geq 0$, $\gamma(l_1, l_2) = 0$ iff $l_1 = l_2$, $\gamma(l_1, l_2) = \gamma(l_2, l_1)$ and $\gamma(l_1, l_3) \leq \gamma(l_1, l_2) + \gamma(l_2, l_3)$. In particular, we call the cost function that $\gamma(l_1, l_2) = 1$ if $l_1 \neq l_2$ a *unit cost function*.

Definition 3 (Edit distance for trees [10]): For a cost function γ, the *cost* of an edit operation $e = l_1 \mapsto l_2$ is given by $\gamma(e) = \gamma(l_1, l_2)$. The *cost* of a sequence $E = e_1, \ldots, e_k$ of edit operations is given by $\gamma(E) = \sum_{i=1}^{k} \gamma(e_i)$. Then, an *edit distance* $\tau_{\text{TAI}}(T_1, T_2)$ between trees T_1 and T_2 is defined as follows:

$$\tau_{\text{TAI}}(T_1, T_2) = \min \left\{ \gamma(E) \;\middle|\; \begin{array}{l} E \text{ is a sequence} \\ \text{of edit operations} \\ \text{transforming } T_1 \text{ to } T_2 \end{array} \right\}.$$

Definition 4 (Tai mapping [10]): Let T_1 and T_2 be trees. We say that a triple (M, T_1, T_2) is a *Tai mapping* (a *mapping*, for short) from T_1 to T_2 if $M \subseteq V(T_1) \times V(T_2)$ and every pair (v_1, w_1) and (v_2, w_2) in M satisfies the following conditions.

1) $v_1 = v_2$ iff $w_1 = w_2$ (one-to-one condition).
2) $v_1 \leq v_2$ iff $w_1 \leq w_2$ (ancestor condition).

We will use M instead of (M, T_1, T_2) when there is no confusion denote it by $M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)$.

Let M be a mapping from T_1 to T_2. Let I_M and J_M be the sets of nodes in T_1 and T_2 but not in M, that is, $I_M = \{v \in T_1 \mid (v, w) \notin M\}$ and $J_M = \{w \in T_2 \mid (v, w) \notin M\}$. Then, the *cost* $\gamma(M)$ of M is given as follows.

$$\gamma(M) = \sum_{(v,w) \in M} \gamma(v, w) + \sum_{v \in I_M} \gamma(v, \varepsilon) + \sum_{w \in J_M} \gamma(\varepsilon, w).$$

Trees T_1 and T_2 are *isomorphic*, denoted by $T_1 \equiv T_2$, if there exists a mapping $M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)$ such that $I_M = J_M = \emptyset$ and $\gamma(M) = 0$.

Theorem 1 (Tai [10]): $\tau_{\text{TAI}}(T_1, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)\}$.

III. Alignment Distance

In this section, we introduce the alignment distance and characterize it by using the variation of Tai mappings.

Definition 5 (Alignment [5]): Let T_1 and T_2 be trees. Then, an *alignment* between T_1 and T_2 is a tree $\mathcal{T}$ obtained by the following steps.

1) Insert new nodes labeled by ε into T_1 and T_2 so that the resulting trees T_1' and T_2' are isomorphic with ignoring labels and $l(\phi(v)) \neq \varepsilon$ whenever $l(v) = \varepsilon$ for an isomorphism ϕ from T_1' to T_2' and every node $v \in T_1'$.
2) Set $\mathcal{T}$ to a tree T_1' obtained by relabeling a label $l(v)$ for every node $v \in T_1'$ with $(l(v), l(\phi(v)))$. (Note that $(\varepsilon, \varepsilon) \notin \mathcal{T}$.)

Let $\mathcal{A}(T_1, T_2)$ denote the set of all possible alignments between trees T_1 and T_2.

For a cost function γ, the *cost* of an alignment $\mathcal{T}$, denoted by $\gamma(\mathcal{T})$, is the sum of the costs of all labels in $\mathcal{T}$.

Definition 6 (Alignment distance [5]): Let T_1 and T_2 be trees and γ a cost function. Then, an *alignment distance* $\tau_{\text{ALN}}(T_1, T_2)$ between T_1 and T_2 is defined as follows.

$$\tau_{\text{ALN}}(T_1, T_2) = \min\{\gamma(\mathcal{T}) \mid \mathcal{T} \in \mathcal{A}(T_1, T_2)\}.$$

Also we call an alignment between T_1 and T_2 with the minimum cost an *optimal alignment* and denote it by $\mathcal{A}^*(T_1, T_2)$.

The notion of the alignment can be easily extended to forests. The only change is that it is now possible to insert a node (as the root) of trees in the forest. We denote the set of all possible alignments between forests F_1 and F_2 by $\mathcal{A}(F_1, F_2)$ and an optimal alignment by $\mathcal{A}^*(F_1, F_2)$.

Example 1: For two caterpillars C_1 and C_2 illustrated in Figure 2, $\mathcal{A}^*(C_1, C_2)$ is the optimal alignment between C_1 and C_2. Also, for two caterpillars C_3 and C_4 illustrated in Figure 2, $\mathcal{A}^*(C_3, C_4)$ is the optimal alignment between C_3 and C_4. Under the unit cost function, it holds that $\tau_{\text{ALN}}(C_1, C_2) = 3$ and $\tau_{\text{ALN}}(C_3, C_4) = 3$.

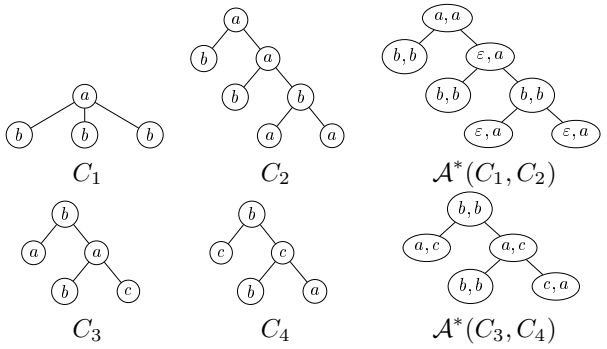

Fig. 2. Two caterpillars C_1, C_2, C_3 and C_4 and the optimal alignments $\mathcal{A}^*(C_1, C_2)$ and $\mathcal{A}^*(C_1, C_2)$. in Example 1.

Next, we introduce the variations of Tai mappings, including the mapping characterizing the alignment distance.

Definition 7 (Variations of Tai mapping): Let T_1 and T_2 be trees and $M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)$.

1) We say that M is a *less-constrained mapping* [7], denoted by $M \in \mathcal{M}_{\text{LESS}}(T_1, T_2)$, if M satisfies the following condition for every $(v_1, w_1), (v_2, w_2), (v_3, w_3) \in M$:
$$(v_1 \sqcup v_2 < v_1 \sqcup v_3) \Longrightarrow (w_2 \sqcup w_3 = w_1 \sqcup w_3).$$
Also we define a *less-constrained distance* $\tau_{\text{LESS}}(T_1, T_2)$ as the minimum cost of all the less-constrained mappings, that is:
$$\tau_{\text{LESS}}(T_1, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\text{LESS}}(T_1, T_2)\}.$$
2) We say that M is an *isolated-subtree mapping* [11] (or a *constrained mapping* [14]), denoted by $M \in \mathcal{M}_{\text{ILST}}(T_1, T_2)$, if M satisfies the following condition for every $(v_1, w_1), (v_2, w_2), (v_3, w_3) \in M$:
$$(v_3 < v_1 \sqcup v_2) \iff (w_3 < w_1 \sqcup w_2).$$
Also we define an *isolated-subtree distance* $\tau_{\text{ILST}}(T_1, T_2)$ as the minimum cost of all the isolated-subtree mappings, that is:
$$\tau_{\text{ILST}}(T_1, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\text{ILST}}(T_1, T_2)\}.$$

Theorem 2: Let T_1 and T_2 be trees, where $n = \max\{|T_1|, |T_2|\}$ and $d = \min\{d(T_1), d(T_2)\}$.

1) It holds that $\tau_{\text{ALN}}(T_1, T_2) = \tau_{\text{LESS}}(T_1, T_2)$ [6]. Also it holds that $\tau_{\text{TAI}}(T_1, T_2) \leq \tau_{\text{ALN}}(T_1, T_2) \leq \tau_{\text{ILST}}(T_1, T_2)$ but the equations always do not hold (*cf.*, [5], [6], [14]).
2) The problem of computing $\tau_{\text{TAI}}(T_1, T_2)$ is MAX SNP-hard [15]. This statement holds even if both T_1 and T_2 are binary, the maximum height of T_1 and T_2 is at most 3 or the cost function is the unit cost function [1], [4].
3) The problem of computing $\tau_{\text{ALN}}(T_1, T_2)$ is MAX SNP-hard. On the other hand, if the degrees of T_1 and T_2 are bounded by some constants, then we can compute $\tau_{\text{ALN}}(T_1, T_2)$ in polynomial time with respect to n [5].
4) We can compute $\tau_{\text{ILST}}(T_1, T_2)$ in $O(n^2 d)$ time (*cf.*, [12]).

Example 2: Consider two caterpillars C_1 and C_2 in Figure 2 in Example 1 and assume the unit cost function. Then, M_1 and M_2 illustrated in Figure 3 are the minimum cost mappings in $\mathcal{M}_{\text{LESS}}(C_1, C_1) (= \mathcal{M}_{\text{TAI}}(C_1, C_2))$ and $\mathcal{M}_{\text{ILST}}(C_1, C_2)$. Here, it holds that $M_1 \notin \mathcal{M}_{\text{ILST}}(C_1, C_2)$. Then, it holds that $\tau_{\text{TAI}}(C_1, C_2) = \tau_{\text{ALN}}(C_1, C_2) = 3 < 5 = \tau_{\text{ILST}}(C_1, C_2)$.

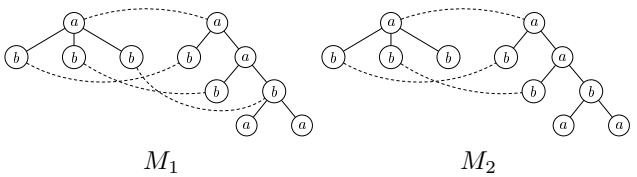

Fig. 3. The minimum cost mappings $M_1 \in \mathcal{M}_{\text{LESS}}(C_1, C_2)$ and $M_2 \in \mathcal{M}_{\text{ILST}}(C_1, C_2)$ in Example 2.

Example 3: Consider two caterpillars C_3 and C_4 in Figure 2 in Example 1 and assume the unit cost function.

Then, M_3 and M_4 illustrated in Figure 4 are the minimum cost mappings in $\mathcal{M}_{\text{TAI}}(C_3, C_4)$ and $\mathcal{M}_{\text{LESS}}(C_3, C_4)$. Here, it holds that $M_3 \notin \mathcal{M}_{\text{LESS}}(C_3, C_4)$. Hence, it holds that $\tau_{\text{TAI}}(C_3, C_4) = 2 < 3 = \tau_{\text{ALN}}(C_3, C_4)$.

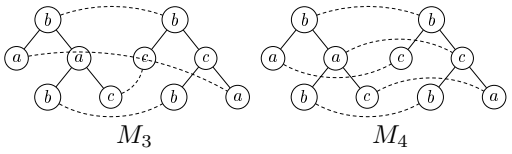

Fig. 4. The minimum cost mappings $M_3 \in \mathcal{M}_{\text{TAI}}(C_3, C_4)$ and $M_4 \in \mathcal{M}_{\text{LESS}}(C_3, C_4)$ in Example 3.

Example 2 shows that there exists a pair of caterpillars whose minimum cost less-constrained mapping is not an isolated-subtree mapping. Then, we cannot use the algorithm to compute the isolated-subtree distance between caterpillars [12], [13], [14], [16] to compute their alignment distance. Also Example 3 shows that there exists a pair of caterpillars whose minimum cost Tai mapping is not a less-constrained mapping. Then, we cannot use the algorithm to compute the edit distance between caterpillars [8] to compute their alignment distance. Furthermore, it still remains open whether or not Theorem 2.3 holds for caterpillars.

Hence, in the next section, we discuss the problem of computing the alignment distance between caterpillars.

IV. The Algorithm of Computing Alignment Distance between Caterpillars

In this section, we design the algorithm to compute the alignment distance τ_{ALN} between caterpillars.

A. Edit distance for multisets

In order to compute the edit distance between the sets of leaves, it is necessary to introduce an edit distance for *multisets* on labels occurring in the set of leaves. Then, we prepare the notions of the edit distance for multisets according to [9].

A *multiset* on an alphabet Σ is a mapping $S : \Sigma \to \mathbf{N}$. For a multiset S on Σ, we say that $a \in \Sigma$ is an *element* of S if $S(a) > 0$ and denote it by $a \in S$ (like as a standard set). The *cardinality* of S, denoted by $|S|$, is defined as $\sum_{a \in \Sigma} S(a)$.

Definition 8 (Edit operations for multisets): Let $a, b \in \Sigma$ such that $S(a) > 0$ and $a \neq b$. Then, a *substitution* $(a \mapsto b)$ operates $S(a)$ to $S(a) - 1$ and $S(b)$ to $S(b) + 1$, a *deletion* $(a \mapsto \varepsilon)$ operates $S(a)$ to $S(a) - 1$ and an *insertion* $(\varepsilon \mapsto b)$ operates $S(b)$ to $S(b) + 1$.

Also we assume a cost function γ as in Section II.

Definition 9 (Edit distance for multisets): Let S_1 and S_2 be multisets on Σ and γ a cost function. Then, an *edit distance* $\mu(S_1, S_2)$ between S_1 and S_2 is defined as follows.

$$\mu(S_1, S_2) = \min \left\{ \gamma(E) \;\middle|\; \begin{array}{l} E \text{ is a sequence} \\ \text{of edit operations} \\ \text{transforming } S_1 \text{ to } S_2 \end{array} \right\}.$$

For multisets S_1 and S_2 on Σ, we define the *difference* $S_1 \setminus S_2$ between S_1 and S_2 as a multiset satisfying that $(S_1 \setminus S_2)(a) = \max\{S_1(a) - S_2(a), 0\}$ for every $a \in \Sigma$.

Lemma 1 ([9]): Let Π_1 be the set of all the injections from S_1 to S_2 when $|S_1| \leq |S_2|$ and Π_2 the set of all the injections from S_2 to S_1 when $|S_1| > |S_2|$. Then, we can compute $\mu(S_1, S_2)$ as follows:

$$\mu(S_1, S_2) = \begin{cases} \min_{\pi \in \Pi_1} \left\{ \sum_{a \in S_1} \gamma(a, \pi(a)) + \sum_{b \in S_2 \setminus \pi(S_1)} \gamma(\varepsilon, b) \right\}, \\ \qquad \text{if } |S_1| \leq |S_2|, \\ \min_{\pi \in \Pi_2} \left\{ \sum_{b \in S_2} \gamma(\pi(b), b) + \sum_{a \in S_1 \setminus \pi(S_2)} \gamma(a, \varepsilon) \right\}, \\ \qquad \text{otherwise.} \end{cases}$$

Furthermore, if we adopt the unit cost function, then we can compute $\mu(S_1, S_2)$ as follows:

$$\mu(S_1, S_2) = \max\{|S_1 \setminus S_2|, |S_2 \setminus S_1|\}.$$

In this case, $\mu(S_1, S_2)$ coincides with a famous *bag distance* (*cf.*, [2]) between multisets S_1 and S_2.

Lemma 2 ([9]): Let $m = \max\{|S_1|, |S_2|\}$. Then, we can compute $\mu(S_1, S_2)$ in $O(m^3)$ time under the general cost function. If we adopt the unit cost function, then we can compute $\mu(S_1, S_2)$ in $O(m)$ time.

B. Recurrences

Let L be the set of leaves and C a non-leaf caterpillar. Then, every forest obtained by deleting the root from a caterpillar is one of the forms of $\{C\}$, L or $L \cup \{C\}$. As same as [8], we denote these forests by $\langle \emptyset \,|\, C \rangle$, $\langle L \,|\, \emptyset \rangle$ and $\langle L \,|\, C \rangle$, respectively. In particular, we denote an empty forest $\langle \emptyset \,|\, \emptyset \rangle$ by Φ simply.

Let $C[v]$ be a caterpillar with the root v, where $L(v)$ denotes a (possibly empty) set of leaves as the children of v and $B(v)$ denotes at most one caterpillar of the child v. Then, $C[v]$ is one of the forms in Figure 5. Furthermore, by deleting v from $C[v]$, we obtain one of the forests of $\langle \emptyset \,|\, B(v) \rangle$, $\langle L(v) \,|\, \emptyset \rangle$ and $\langle L(v) \,|\, B(v) \rangle$, respectively.

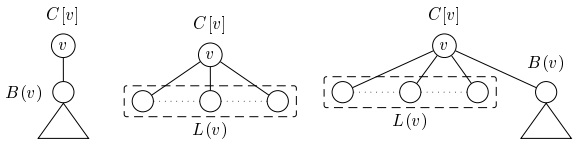

Fig. 5. The representation of a caterpillar $C[v]$.

Figure 6 illustrates the recurrences of computing the alignment distance $\tau_{\text{ALN}}(C_1[v], C_2[w])$ between two caterpillars $C_1[v]$ and $C_2[w]$. Here, we regard a set L of leaves as a multiset of labels on Σ occurring in L, which we denote by $\widetilde{L}$. Also $\delta_{\text{ALN}}(\langle L_1 \mid B_1 \rangle, \langle L_2 \mid B_2 \rangle)$ describes the alignment distance between forests $\langle L_1 \,|\, B_1 \rangle$ and $\langle L_2 \,|\, B_2 \rangle$. Furthermore, assume that the forest obtained by deleting v (*resp.*, w) from $C_1[v]$ (*resp.*, $C_2[w]$) is $\langle L_1(v) \,|\, B_1(v) \rangle$ (*resp.*, $\langle L_2(w) \,|\, B_2(w) \rangle$).

Theorem 3: The recurrences in Figure 6 are correct to compute the alignment distance $\tau_{\text{ALN}}(C_1[v], C_2[w])$ between $C_1[v]$ and $C_2[w]$.

$$\tau_{\text{ALN}}(\emptyset,\emptyset) = 0. \quad (T_0)$$
$$\tau_{\text{ALN}}(C_1[v],\emptyset) = \gamma(v,\varepsilon) + \delta_{\text{ALN}}(\langle L_1(v)\,|\,B_1(v)\rangle, \Phi). \quad (T_1)$$
$$\tau_{\text{ALN}}(\emptyset, C_2[w]) = \gamma(\varepsilon,w) + \delta_{\text{ALN}}(\Phi, \langle L_2(w)\,|\,B_2(w)\rangle). \quad (T_2)$$
$$\delta_{\text{ALN}}(\langle L_1\,|\,C_1\rangle, \Phi) = \sum_{v\in L_1}\gamma(v,\varepsilon) + \sum_{v\in C_1}\gamma(v,\varepsilon). \quad (F_1)$$
$$\delta_{\text{ALN}}(\Phi, \langle L_2\,|\,C_2\rangle) = \sum_{w\in L_2}\gamma(\varepsilon,w) + \sum_{w\in C_2}\gamma(\varepsilon,w). \quad (F_2)$$
$$\delta_{\text{ALN}}(\langle L_1\,|\,\emptyset\rangle, \langle L_2\,|\,\emptyset\rangle) = \mu(\widetilde{L_1}, \widetilde{L_2}). \quad (F_3)$$
(A) $\tau_{\text{ALN}}(C_1[v], C_2[w])$
$$= \min\left\{\begin{array}{ll}\gamma(v,w) & \\ \quad + \delta_{\text{ALN}}(\langle L_1(v)\,|\,B_1(v)\rangle, \langle L_2(w)\,|\,B_2(w)\rangle), & (T_3)\\ \gamma(v,\varepsilon) + \tau_{\text{ALN}}(B_1(v), C_2[w]) & \\ \quad + \delta_{\text{ALN}}(\langle L_1(v)\,|\,\emptyset\rangle, \Phi), & (T_4)\\ \gamma(\varepsilon,w) + \tau_{\text{ALN}}(C_1[v], B_2(w)) & \\ \quad + \delta_{\text{ALN}}(\Phi, \langle L_2(w)\,|\,\emptyset\rangle) & (T_5)\end{array}\right\}.$$
(B) $\delta_{\text{ALN}}(\langle L_1\,|\,\emptyset\rangle, \langle L_2\,|\,C_2[w]\rangle)$
$$= \min\left\{\begin{array}{ll}\gamma(\varepsilon,w) & \\ \quad +\delta_{\text{ALN}}(\langle L_1\,|\,\emptyset\rangle, \langle L_2 \cup L_2(w)\,|\,B_2(w)\rangle), & (F_4)\\ \min\limits_{v\in L_1}\{\gamma(v,w) + \delta_{\text{ALN}}(\langle L_1 \setminus \{v\}\,|\,\emptyset\rangle, \langle L_2\,|\,\emptyset\rangle)\} & \\ \quad +\delta_{\text{ALN}}(\Phi, \langle L_2(w)\,|\,B_2(w)\rangle) & (F_5)\end{array}\right\}.$$
(C) $\delta_{\text{ALN}}(\langle L_1\,|\,C_1[v]\rangle, \langle L_2\,|\,\emptyset\rangle)$
$$= \min\left\{\begin{array}{ll}\gamma(v,\varepsilon) & \\ \quad +\delta_{\text{ALN}}(\langle L_1 \cup L_1(v)\,|\,B_1(v)\rangle, \langle L_2\,|\,\emptyset\rangle), & (F_6)\\ \min\limits_{w\in L_2}\{\gamma(v,w) + \delta_{\text{ALN}}(\langle L_1\,|\,\emptyset\rangle, \langle L_2 \setminus \{w\}\,|\,\emptyset\rangle)\} & \\ \quad +\delta_{\text{ALN}}(\langle L_1(v)\,|\,B_1(v)\rangle, \Phi) & (F_7)\end{array}\right\}.$$
(D) $\delta_{\text{ALN}}(\langle L_1\,|\,C_1[v]\rangle, \langle L_2\,|\,C_2[w]\rangle)$
$$= \delta_{\text{ALN}}(\langle L_1\,|\,\emptyset\rangle, \langle L_2\,|\,\emptyset\rangle) + \tau_{\text{ALN}}(C_1[v], C_2[w]). \quad (F_8)$$

Fig. 6. The recurrences of computing the alignment distance $\tau_{\text{ALN}}(C_1[v], C_2[w])$ between $C_1[v]$ and $C_2[w]$.

Proof: The recurrences of (T_0), (T_1), (T_2), (F_1), (F_2) and (F_3) are obvious.

First, consider the recurrences for τ_{ALN}. Let $\mathcal{T}$ be the optimal alignment (tree) $\mathcal{A}^*(C_1[v], C_2[w])$. Then, for the label in $\mathcal{T}$, one of the following four cases holds.

1) (v,w) is a label in $\mathcal{T}$.
2) (v,ε) and (v',w) are labels in $\mathcal{T}$.
3) (ε,w) and (v,w') are labels in $\mathcal{T}$.
4) (v,ε) and (ε,w) are labels in $\mathcal{T}$.

It is not necessary to consider the case 4) because the resulting alignment to delete the two nodes and then add (v,w) as the new root, which is the case 1), has a smaller cost.

For the case 1), the root of $\mathcal{T}$ is (v,w). By the forms of $C_1[v]$ and $C_2[w]$, it holds that $\tau_{\text{ALN}}(C_1[v], C_2[w]) = \gamma(v,w) + \delta_{\text{ALN}}(\langle L_1(v)\mid B_1(v)\rangle, \langle L_2(w)\mid B_2(w)\rangle)$, which is the recurrence (T_3).

For the case 2), the root of $\mathcal{T}$ is (v,ε). By the form of $C_1[v]$, since $|B_1(v)| \geq 2$, $B_1(v)$, not $L_1(v)$, contains the node v' corresponding to w in $C_2[w]$. Then, $\mathcal{T}$ contains a label (v'',ε) for every $v'' \in L_1(v)$. Hence, it holds that $\tau_{\text{ALN}}(C_1[v], C_2[w]) = \gamma(v,\varepsilon) + \tau_{\text{ALN}}(B_1(v), C_2[w]) + \delta_{\text{ALN}}(\langle L_1(v)\mid\emptyset\rangle, \Phi)$, which is the recurrence (T_4). The case 3) is similar to the case 2), which is the recurrence (T_5).

Next, consider the recurrences for δ_{ALN}.

Let $\mathcal{F}$ be the optimal alignment (forest) $\mathcal{A}^*(\langle L_1\,|\,\emptyset\rangle, \langle L_2\,|\,C_2[w]\rangle)$. Then, for the label in $\mathcal{F}$, one of the following two cases holds.

1) (ε,w) is a label in $\mathcal{F}$.
2) (v,w) for some $v \in L_1$ is a label in $\mathcal{F}$.

For the case 1), by deleting w from $C_2[w]$, $\langle L_2\mid C_2[w]\rangle$ is transformed to $\langle L_2 \cup L_2(w)\,|\,B_2(w)\rangle$. Hence, it holds that $\delta_{\text{ALN}}(\langle L_1\mid\emptyset\rangle, \langle L_2\mid C_2[w]\rangle) = \gamma(\varepsilon,w) + \delta_{\text{ALN}}(\langle L_1\mid\emptyset\rangle, \langle L_2 \cup L_2(w)\mid B_2(w)\rangle)$, which is the recurrence (F_4).

For the case 2), once (v,w) for some $v \in L_1$ becomes a label in $\mathcal{F}$, every label in $\mathcal{F}$ for every $w' \in \langle L_2(w)\,|\,B_2(w)\rangle$ is always of the form (ε,w'). Also the labels concerned with leaves except $v \in L_1$ in $\mathcal{F}$ can be computed as $\delta_{\text{ALN}}(\langle L_1\setminus\{v\}\mid\emptyset\rangle, \langle L_2\mid\emptyset\rangle)$. Hence, by selecting $v \in L_1$ with the minimum cost, we obtain the recurrence (F_5).

By using the same discussion, for the case that $\mathcal{F}$ is the optimal alignment (forest) $\mathcal{A}^*(\langle L_1\,|\,C_1[v]\rangle, \langle L_2\,|\,\emptyset\rangle)$, we obtain the recurrences (F_6) and (F_7).

Let $\mathcal{F}$ be the optimal alignment (forest) $\mathcal{A}^*(\langle L_1\mid C_1[v]\rangle, \langle L_2\mid C_2[w]\rangle)$. Since $|C_1[v]| \geq 2$ and $|C_2[w]| \geq 2$, $\mathcal{F}$ contains labels for the alignment of $C_1[v]$ and $C_2[w]$ and that of L_1 and L_2. Hence, it holds that $\delta_{\text{ALN}}(\langle L_1\,|\,C_1[v]\rangle, \langle L_2\mid C_2[w]\rangle) = \delta_{\text{ALN}}(\langle L_1\,|\,\emptyset\rangle, \langle L_2\,|\,\emptyset\rangle + \tau_{\text{ALN}}(C_1[v], C_2[w])$, which is the recurrence (F_8). ■

Example 4: Consider two caterpillars C_1 and C_2 in Figure 2 in Example 1 and assume the unit cost function. By applying the recurrences in Figure 6, we obtain that the alignment distance $\tau_{\text{ALN}}(C_1, C_2)$ between C_1 and C_2 is 3 illustrated in Figure 7. Here, we represent a multiset as a sequence enclosed by "[" and "]" and a caterpillar as a term-like representation with "[" and "]", that is, $C_1 = a[b,b,b]$ and $C_2 = a[b,a[b,b[a,a]]]$.

$$\begin{array}{lll}
& \tau_{\text{ALN}}(C_1, C_2) & \\
= & \underbrace{\gamma(a,a)}_{=0} + \delta_{\text{ALN}}(\langle [b,b,b]\,|\,\emptyset\rangle, \langle [b]\,|\,a[b,b[a,a]]\rangle) & (T_3)\\
= & \underbrace{\gamma(\varepsilon,a)}_{=1} + \delta_{\text{ALN}}(\langle [b,b,b]\,|\,\emptyset\rangle, \langle [b,b]\,|\,b[a,a]\rangle) & (F_4)\\
= & 1 + \underbrace{\gamma(b,b)}_{=0} + \delta_{\text{ALN}}(\langle [b,b]\,|\,\emptyset\rangle, \langle [b,b]\,|\,\emptyset\rangle) & \\
& \quad + \delta_{\text{ALN}}(\Phi, \langle [a,a]\,|\,\emptyset\}\rangle) & (F_5)\\
= & 1 + \underbrace{\mu([b,b],[b,b])}_{=0} + \underbrace{\gamma(\varepsilon,a)}_{=1} + \underbrace{\gamma(\varepsilon,a)}_{=1} & (F_2), (F_3)\\
= & 3. &
\end{array}$$

Fig. 7. The result of computing $\tau_{\text{ALN}}(C_1, C_2)$ in Example 4.

Example 5: Consider two caterpillars $C_3 = a[a, d[b,c]]$ and $C_4 = a[c, e[b,a]]$ in Figure 2 in Example 1 and assume the unit cost function. By applying the recurrences in Figure 6, we obtain that the alignment distance $\tau_{\text{ALN}}(C_3, C_4)$ between C_3 and C_4 is 3 illustrated in Figure 8.

$$
\begin{aligned}
& \tau_{\mathrm{ALN}}(C_3, C_4) \\
= \; & \underbrace{\gamma(b,b)}_{0} + \delta_{\mathrm{ALN}}(\langle [a] \,|\, a[b,c] \rangle, \langle [c] \,|\, c[b,a] \rangle) && (T_3) \\
= \; & \underbrace{\gamma(a,c)}_{1} + \delta_{\mathrm{ALN}}(\langle [a] \,|\, \emptyset \rangle, \langle [c] \,|\, \emptyset \rangle) \\
& + \delta_{\mathrm{ALN}}(\langle [b,c] \,|\, \emptyset \rangle, \langle [b,a] \,|\, \emptyset \rangle) && (F_8) \\
= \; & 1 + \underbrace{\mu([a],[c])}_{=1} + \underbrace{\mu([b,c],[a,b])}_{=1} && (F_3) \\
= \; & 3.
\end{aligned}
$$

Fig. 8. The result of computing $\tau_{\mathrm{ALN}}(C_3, C_4)$ in Example 5.

C. Algorithm and time complexity

Let $C_1[v]$ and $C_2[w]$ be caterpillars. Then, we denote $bb(C_1[v])$ by a sequence $v_1, \ldots, v_n$ such that $v_n = v$ and $par(v_i) = v_{i+1}$ $(1 \leq i \leq n-1)$ and $bb(C_2[w])$ by a sequence $w_1, \ldots, w_m$ such that $w_m = w$ and $par(w_j) = w_{j+1}$ $(1 \leq j \leq m-1)$. In this case, we denote by $bb(C_1[v]) = [v_1, \ldots, v_n]$ and $bb(C_2[w]) = [w_1, \ldots, w_m]$. Also we use the same notations of $L_1(v_i)$ and $B_1(v_i)$ for $1 \leq i \leq n$ and $L_2(w_j)$ and $B_2(w_j)$ for $1 \leq j \leq m$.

Based on the recurrences in Figure 6, Algorithm 1 illustrates the algorithm to compute the alignment distance $\tau_{\mathrm{ALN}}(C_1, C_2)$ between caterpillars C_1 and C_2. Here, the statement "$v \leftarrow$ (A)" means to substitute the value of computing the right side of the recurrence (A) to v, for example.

Theorem 4: Let C_1 and C_2 be caterpillars, where $h = \max\{h(C_1), h(C_2)\}$ and $\lambda = \max\{|lv(C_1)|, |lv(C_2)|\}$. Then, we can compute the alignment distance $\tau_{\mathrm{ALN}}(C_1, C_2)$ between C_1 and C_2 in $O(h^2\lambda^3)$ time. Furthermore, if we adopt the unit cost function, then we can compute it in $O(h^2\lambda)$ time.

Proof: Let $bb(C_1) = [v_1, \ldots, v_n]$ and $bb(C_2) = [w_1, \ldots, w_m]$. Then, it is obvious that $h(C_1) = n + 1$ and $h(C_2) = m + 1$, so it holds that $m \leq h - 1$ and $n \leq h - 1$.

The algorithm of computing $\tau_{\mathrm{ALN}}(C_1, C_2)$ calls every pair $(v_i, w_j) \in bb(C_1) \times bb(C_2)$ just once. When computing $\delta_{\mathrm{ALN}}(\langle L_1(v_{i-1}) \,|\, C_1[v_i] \rangle, \langle L_2(w_{j-1}) \,|\, C_2[w_j] \rangle)$ for $2 \leq i \leq n$ and $2 \leq j \leq m$, it is possible to construct multisets $S_1 = \widetilde{L_1(v_1)} \sqcup \cdots \sqcup \widetilde{L_1(v_{i-1})}$ and $S_2 = \widetilde{L_2(w_1)} \sqcup \cdots \sqcup \widetilde{L_2(w_{j-1})}$ and compute the edit distance $\mu(S_1, S_2)$ between multisets in the worst case. By Lemma 2, we can compute it in $O(\lambda^3)$ time under the general cost function and in $O(\lambda)$ time under the unit cost function.

Hence, the total running time of computing $\tau_{\mathrm{ALN}}(C_1, C_2)$ under the general cost function is described as follows:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} O(\lambda^3) = O(\lambda^3)mn \leq O(\lambda^3)(h-1)^2 = O(h^2\lambda^3).$$

By replacing $O(\lambda^3)$ with $O(\lambda)$, this time complexity is reduced to $O(h^2\lambda)$ time under the unit cost function. ■

Theorem 4 also claims that the structural restriction of caterpillars provides the limitation of tractable computing the alignment distance for unordered trees as follows. We say that a tree is a *generalized caterpillar* if it is transformed to a caterpillar after removing all the leaves in it. Then, the following theorem also holds as a corollary of [8].

```
procedure τ_ALN(C1, C2)
    /* C1, C2: caterpillars, bb(C1) = [v1, ..., vn],
       bb(C2) = [w1, ..., wm], vn = r(C1), wm = r(C2) */
1   τ_ALN(∅, ∅) ← 0; /* (T0) */
2   for i = 1 to n do τ_ALN(C1[vi], ∅) ← (T1);
3   for j = 1 to m do τ_ALN(∅, C2[wj]) ← (T2);
4   for i = 1 to n do
5       for j = 1 to m do
6           τ_ALN(C1[vi], C2[wj]) ← (A);

procedure δ_ALN(⟨L1 | C1⟩, ⟨L2 | C2⟩)
    /* L1, L2 : set of leaves, C1, C2: caterpillars */
7   if C1 = ∅ and C2 = ∅ then
8       δ_TAI(⟨L1 | ∅⟩, ⟨L2 | ∅⟩) ← (F3);
9   else if C1 ≠ ∅ and C2 = ∅ then
        /* bb(C1) = [v1, ..., vn], vn = r(C1) */
10      if L2 = ∅ then δ_TAI(⟨L1 | C1⟩, Φ) ← (F1);
11      else
12          for i = 1 to n do
13              δ_TAI(⟨L1 | C1[vi]⟩, ⟨L2 | ∅⟩) ← (B);
14  else if C1 = ∅ and C2 ≠ ∅ then
        /* bb(C2) = [w1, ..., wm], wm = r(C2) */
15      if L1 = ∅ then δ_TAI(Φ, ⟨L2 | C2⟩) ← (F2);
16      else
17          for j = 1 to m do
18              δ_TAI(⟨L1 | ∅⟩, ⟨L2 | C2[wj]⟩) ← (C);
19  else
        /* bb(C1) = [v1, ..., vn], bb(C2) = [w1, ..., wm],
           vn = r(C1), wm = r(C2) */
20      for i = 1 to n do
21          for j = 1 to m do
22              δ_TAI(⟨L1 | C1[vi]⟩, ⟨L2 | C2[wj]⟩) ← (D);
```

Algorithm 1: $\tau_{\mathrm{ALN}}(C_1, C_2)$

Theorem 5 (cf., [1], [4]): The problems of computing the alignment distance τ_{ALN} between generalized caterpillars are MAX SNP-hard, even if the maximum height is at most 3 and the cost function is the unit cost function.

Proof: It is straightforward from the proof of Corollary 4.3 in [1] or Theorem 1 in [4] and because the Tai mapping constructed in their proof is a less-constrained mapping. ■

V. Conclusion and Future Works

In this paper, we have designed the algorithm to compute the alignment distance τ_{ALN} between caterpillars in $O(h^2\lambda^3)$ time under the general cost function and in $O(h^2\lambda)$ time under the unit cost function.

It is an important future work to implement the algorithms and then give experimental results to compute τ_{ALN}, with comparing the results of τ_{TAI} in [8] with those of τ_{ALN}. Since the proof in Theorem 4 is rough, it is possible to improve the time complexity, together with that of computing the edit distance between multisets under the general cost function (Lemma 2), which is also a future work.

REFERENCES

[1] T. Akutsu, D. Fukagawa, M. M. Halldórsson, A. Takasu, K. Tanaka: *Approximation and parameterized algorithms for common subtrees and edit distance between unordered trees*, Theoret. Comput. Sci. **470**, 10–22 (2013). https://doi.org/10.1016/j.tcs.2012.11.017.

[2] M. M. Deza, E. Deza: *Encyclopedia of distances* (4th ed.) Springer (2016). https://doi.org/10.1007/978-3-662-52844-0.

[3] J. A. Gallian: *A dynamic survey of graph labeling*, Electorn. J. Combin., DS6 (2018).

[4] K. Hirata, Y. Yamamoto, T. Kuboyama: *Improved MAX SNP-hard results for finding an edit distance between unordered trees*, Proc. CPM'11, LNCS **6661**, 402–415 (2011). https://doi.org/10.1007/978-3-642-21458-5_34.

[5] T. Jiang, L. Wang, K. Zhang: *Alignment of trees – an alternative to tree edit*, Theoret. Comput. Sci. **143**, 137–148 (1995). https://doi.org/10.1016/0304-3975(95)80029-9.

[6] T. Kuboyama: *Matching and learning in trees*, Ph.D thesis, University of Tokyo (2007).

[7] C. L. Lu, Z.-Y. Su, C. Y. Yang: *A new measure of edit distance between labeled trees*, Proc. COCOON'01, LNCS **2108**, 338–348 (2001). https://doi.org/10.1007/3-540-44679-6_37.

[8] K. Muraka, T. Yoshino, K. Hirata: *Computing edit distance between rooted labeled caterpillars*, Proc. FedCSIS'18, 245–252 (2018). http://dx.doi.org/10.15439/2018F179.

[9] K. Muraka, T. Yoshino, K. Hirata: *Vertical and horizontal distance to approximate edit distance for rooted labeled caterpillars*, Proc. ICPRAM'19, 590–597 (2019). https://dx.doi.org/10.5220/0007387205900597.

[10] K.-C. Tai: *The tree-to-tree correction problem*, J. ACM **26**, 422–433 (1979). https://doi.org/10.1145/322139.322143.

[11] J. T. L. Wang, K. Zhang: *Finding similar consensus between trees: An algorithm and a distance hierarchy*, Pattern Recog. **34**, 127–137 (2001). https://doi.org/10.1016/50031-3203(99)00199-5.

[12] Y. Yamamoto, K. Hirata, T. Kuboyama: *Tractable and intractable variations of unordered tree edit distance*, Internat. J. Found. Comput. Sci. **25**, 307–330 (2014). https://doi.org/10.1142/50129054114500154.

[13] T. Yoshino, K. Hirata: *Tai mapping hierarchy for rooted labeled trees through common subforest*, Theory Comput. Sys. **60**, 759–783 (2017). https://doi.org/10.1007/s00224-016-9705-1.

[14] K. Zhang: *A constrained edit distance between unordered labeled trees*, Algorithmica **15**, 205–222 (1996). https://doi.org/10.1007/BF01975866.

[15] K. Zhang, T. Jiang: *Some MAX SNP-hard results concerning unordered labeled trees*, Inform. Process. Lett. **49**, 249–254 (1994). https://doi.org/10.1016/0020-0190(94)90062-0.

[16] K. Zhang, J. Wang, D. Shasha: *On the editing distance between undirected acyclic graphs*, Internat. J. Found. Comput. Sci. **7**, 45–58 (1996). https://doi.org/10.1142/S0129054196000051.

APPENDIX

In this appendix, we point out the number of caterpillars in real data and revise the result for the edit distance between caterpillars in [8].

A. Caterpillars in real data

Table I, which is represented in [8], illustrates the number of caterpillars in N-glycans and all glycans from KEGG[2], CSLOGS[3], dblp[4], and SwissProt, TPC-H, Auction, Nasa, Protein and University from UW XML Repository[5]. Here, #cat is the number of caterpillars and #data is the total number of data. For $D \in \{\text{Auction, Nasa, Protein, University}\}$, D^- denotes the trees obtained by deleting the root for every tree in D. Since one tree in D produces some trees in D^-, the total number of trees in D^- is greater than that of D. Hence, there are some cases containing many caterpillars in real dataset.

TABLE I
THE NUMBER OF CATERPILLARS IN N-GLYCANS AND ALL GLYCANS FROM KEGG, CSLOGS, DBLP, SWISSPROT, TPC-H, AUCTION, UNIVERSITY, PROTEIN AND NASA.

dataset	#cat	#data	%
N-glycans	514	2,142	23.996
all glycans	8,005	10,704	74.785
CSLOGS	41,592	59,691	69.679
dblp	5,154,295	5,154,530	99.995
SwissProt	6,804	50,000	13.608
TPC-H	86,805	86,805	100.000
Auction	0	37	0
Nasa	0	2,430	0
Protein	0	262,625	0
University	0	6,738	0
Auction$^-$	259	259	100.000
Nasa$^-$	21,245	27,921	76.089
Protein$^-$	1,874,703	2,204,068	85.057
University$^-$	74,638	79,213	94.224

B. The revision of the edit distance for caterpillars

Muraka *et al.* [8] have designed the algorithm to compute the edit distance $\tau_{\text{TAI}}(C_1, C_2)$. Then, they have pointed out that its time complexity is $O(h^2\lambda^2)$ time, where $h = \max\{h(C_1), h(C_2)\}$ and $\lambda = \max\{|lv(C_1)|, |lv(C_2)|\}$.

Note that their recurrence between the set of leaves is based on the string edit distance. However, as similar as the alignment distance in this paper, in order to compute the edit distance between the set of leaves, it is necessary to adopt the edit distance for multisets.

Let $s(L)$ be the string representation of the set L of leaves and σ the string edit distance. Then, Muraka *et al.* [8] have introduced the following recurrence to compute $\tau_{\text{TAI}}(C_1, C_2)$:

$$\delta_{\text{TAI}}(\langle L_1 \,|\, \emptyset\rangle, \langle L_2 \,|\, \emptyset\rangle) = \sigma(s(L_1), s(L_2)).$$

On the other hand, as stated above, it is necessary to replace this recurrence with the following recurrence to compute $\tau_{\text{TAI}}(C_1, C_2)$ as same as Figure 6:

$$\delta_{\text{TAI}}(\langle L_1 \,|\, \emptyset\rangle, \langle L_2 \,|\, \emptyset\rangle) = \mu(\widetilde{L_1}, \widetilde{L_2}).$$

Consider the time complexity of $O(h^2\lambda^2)$ presented in [8]. The part $O(\lambda^2)$ follows from the time complexity of computing the string edit distance between the set of (all the) leaves in two caterpillars. On the other hand, by replacing the recurrence as above, it is necessary to revise this part based on the time complexity of computing the multiset edit distance, that is, revise to $O(\lambda^3)$ under the general cost function and $O(\lambda)$ under the unit cost function by Lemma 2. Furthermore, we can improve the proof of [8] to the similar proof of Theorem 4.

Hence, we can compute $\tau_{\text{TAI}}(C_1, C_2)$ in $O(h^2\lambda^3)$ time under the general cost function and $O(h^2\lambda)$ time under the unit cost function, which is the same result of $\tau_{\text{ALN}}(C_1, C_2)$.

[2]Kyoto Encyclopedia of Genes and Genomes, http://www.kegg.jp/

[3]http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/Software

[4]http://dblp.uni-trier.de/

[5]http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html