Computing Edit Distance between Rooted Labeled Caterpillars

Kohei Muraka
Graduate School of Computer Science and Systems Engineering
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820-8502, Japan
Email: muraka@dumbo.ai.kyutech.ac.jp

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

Abstract—A rooted labeled caterpillar is a rooted labeled tree transformed to a path after removing all the leaves in it. In this paper, we design the algorithm to compute the edit distance between root labeled caterpillars in $O(\lambda^2h^2)$ time, where $\lambda$ and $h$ are the maximum number of leaves and the maximum height in two caterpillars, respectively.

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 [9]. 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 [9].

Unfortunately, the problem of computing the edit distance between trees is MAX SNP-hard [13]. This statement also holds even if trees are binary or the maximum height of trees is at most 3 [1], [4].

Many variations of the edit distance have developed as more structurally sensitive distances, by introducing the restriction of Tai mappings (cf., [7], [11]). All the variations except those of an alignment distance [5] are metrics and the problem of computing them is tractable [10], [11], [12], [14]. In particular, the isolated-subtree distance (or constrained distance) [12], which is defined as the minimum cost of isolated-subtree mappings, is the most general tractable variation of the edit distance [11].

On the other hand, a caterpillar (cf. [3]) is a tree transformed to a 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 II in Appendix.

As a method to compare two caterpillars, we can adopt a complete subtree histogram distance, which is an $L_1$-distance between histograms consisting of complete subtrees in two trees [1]. The complete subtree histogram is computable in linear time and always a metric but it is greater than the edit distance in general [1]. In particular, as an extreme case, there exists two caterpillars such that the edit distance between them is one but the complete subtree histogram distance is the number of nodes in two caterpillars, consider two paths with the same length such that the labels of leaves are different.

As another method, we can also adopt a path histogram distance, which is an $L_1$-distance between histograms consisting of paths from the root to leaves in two trees [6]. The path histogram distance is computable in linear time and always a metric for caterpillars, which is not a metric for trees, but it is incomparable with the edit distance [6].

Since a caterpillar is an unordered tree, it remains open whether or not the problem of computing the edit distance between caterpillars is tractable. Hence, we discuss this problem.

First, we point out that there exists a Tai mapping between two caterpillars that is not an isolated-subtree mapping. Then, we cannot apply the algorithm to compute the isolated-subtree distance or its variations [10], [11], [12], [14] that are tractable variations of the edit distance, to compute the edit distance between caterpillars.

On the other hand, a caterpillar has the structural property that the children of a non-leaf node in a caterpillar consist of at most one caterpillar and leaves (possibly empty). Then, by deleting a non-leaf node in a caterpillar, we obtain at most one caterpillar and the set of leaves as a forest. Furthermore, once such leaves are obtained, then we can add them to the previous set of leaves.

Based on this property, in this paper, we design the algorithm to compute the edit distance between caterpillars in $O(\lambda^2h^2)$ time, where $\lambda$ and $h$ are the maximum number of leaves and the maximum height in two caterpillars, respectively. Furthermore, we point out that the structural restriction of caterpillars provides the limitation of tractable computing of the edit distance for unordered trees.

II. CATERPILLARS AND EDIT DISTANCE

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 sometimes 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

*The author would like to express thanks for support by Grant-in-Aid for Scientific Research 17H00762, 16H02670 and 16H01743 from the Ministry of Education, Culture, Sports, Science and Technology, Japan.
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 \((V', E')\) such that \( V' = \{v_1, \ldots, v_k\}, v_1 = r, v_k = v \) and \((v_i, v_{i+1}) \in E' \) for every \( i \leq i \leq k - 1 \), 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 call a node with no children a leaf and denote the set of all the leaves in \( T \) by \( \text{ln}(T) \).

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 \( \max\{|UP_r(v)| \mid w \in \text{ln}(T[v])\} \), 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 \( u \lor v \), if \( u \leq w \leq v \) and there exists no node \( w' \in T \) such that \( w' \leq u \) and \( w' \leq v \).

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 \( \text{pre}(u) \leq \text{pre}(v) \) for the preorder number \( \text{pre} \) in \( T \) and \( \text{post}(u) \leq \text{post}(v) \) for the postorder number \( \text{post} \) 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 \( \Sigma \). 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 catepillar (catepillar, for short) as follows, which this paper mainly deals with.

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

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

**Definition 2 (Edit operations [9]):** 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 \( v' \).

Let \( e \notin \Sigma \) denote a special blank symbol and define \( \Sigma_e = \Sigma \cup \{e\} \). Then, we represent each edit operation by \((l_1 \mapsto l_2)\), where \((l_1, l_2) \in (\Sigma_e \times \Sigma_e - \{(e, e)\}) \). The operation is a substitution if \( l_1 \neq e \) and \( l_2 \neq e \), a deletion if \( l_2 = e \), and an insertion if \( l_1 = e \). For nodes \( v \) and \( w \), we also denote \((l(v) \mapsto l(w))\) by \((v \mapsto w)\). We define a cost function \( \gamma : (\Sigma_e \times \Sigma_e \setminus \{(e, e)\}) \rightarrow \mathbb{R}^+ \) on pairs of labels. We often constrain a cost function \( \gamma \) 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 \( \gamma(l_1, l_2) + \gamma(l_2, l_3) \leq \gamma(l_1, l_3) \). 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 [9]):** For a cost function \( \gamma \), the cost of an edit operation \( e \) is \( l_1 \mapsto l_2 \) 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 \( \tau_{Ta}(T_1, T_2) \) between trees \( T_1 \) and \( T_2 \) is defined as follows:

\[
\tau_{Ta}(T_1, T_2) = \min \left\{ \gamma(E) \mid E \text{ is a sequence of edit operations transforming } T_1 \text{ to } T_2 \right\}.
\]

**Definition 4 (Tai mapping [9]):** 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) \( v_2 \leq w_2 \) iff \( v_1 \leq w_1 \) (ancestor condition).

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

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.

\[
\gamma(M) = \sum_{(v, w) \in M} \gamma(v, w) + \sum_{v \in I_M} \gamma(v, e) + \sum_{w \in J_M} \gamma(e, w).
\]

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

**Theorem 1** (Tai [9]): \( \tau_{TA}(T_1\!, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{TA}(T_1\!, T_2)\}. \)

Unfortunately, the following theorem is known for the problem of computing \( \tau_{TA} \).

**Theorem 2** ([11], [4], [13]): Let \( T_1 \) and \( T_2 \) be trees. Then, the problem of computing \( \tau_{TA}(T_1\!, T_2) \) is MAX SNP-hard. This statement also holds even if both \( T_1 \) and \( T_2 \) are binary or the maximum height of \( T_1 \) and \( T_2 \) is at most 3.

Finally, we introduce an isolated-subtree mapping and an isolated-subtree distance as the variations of the Tai mapping and the edit distance.

**Definition 5** (Isolated-subtree mapping and distance [12]): Let \( T_1 \) and \( T_2 \) be trees and \( M \in \mathcal{M}_{TA}(T_1\!, T_2) \). We say that \( M \) is an isolated-subtree mapping, denoted by \( M \in \mathcal{M}_{LS}(T_1\!, T_2) \), if \( M \) satisfies the following condition for every \((v_1, w_1), (v_2, w_2), (v_3, w_3) \in M\):

\[ v_3 < v_1 \cup v_2 \iff w_3 < w_1 \cup w_2. \]

Furthermore, we define an isolated-subtree distance \( \tau_{LS}(T_1\!, T_2) \) as follow.

\[ \tau_{LS}(T_1\!, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{LS}(T_1\!, T_2)\}. \]

It is obvious that \( \mathcal{M}_{LS}(T_1, T_2) \subseteq \mathcal{M}_{TA}(T_1, T_2) \) and then \( \tau_{TA}(T_1, T_2) \leq \tau_{LS}(T_1, T_2) \). In contrast to Theorem 2, the following theorem also holds.

**Theorem 3** (cf. [10]): Let \( T_1 \) and \( T_2 \) be trees. Then, we can compute \( \tau_{LS}(T_1\!, T_2) \) in \( O(n^d) \) time, where \( n = \max\{|T_1|, |T_2|\} \) and \( d = \min\{d(T_1), d(T_2)\} \).

It is known that \( \tau_{LS} \) is the most general tractable variation of \( \tau_{TA} \) [11].

**Example 1**: Consider two caterpillars \( C_1 \) and \( C_2 \) illustrated in Figure 2. Then, \( M \) illustrated in Figure 2 is the optimum mapping between \( C_1 \) and \( C_2 \). Here, it holds that \( M \notin \mathcal{M}_{LS}(C_1\!, C_2) \) and \( M \) is an alignable mapping corresponding to an alignment distance [7], [11]. Note that the problem of computing an alignment distance is MAX SNP-hard in general [5].

\[ J_M = \emptyset \text{ and } \gamma(M) = 0. \]

Figure 4 illustrates the recurrences of computing the edit distance \( \tau_{TA}(C_1[v]\!, C_2[w]) \) between two caterpillars \( C_1[v] \) and \( C_2[w] \), as \( \delta_{TA}(\emptyset \! \mid C_1[v]), \emptyset \! \mid C_2[w]) \). Here, we denote the string representation of the set \( L \) of leaves under the alphabetical order on \( \Sigma \) by \( s(L) \) and the string edit distance between two strings \( s_1 \) and \( s_2 \) [2] by \( \sigma(s_1, s_2) \).

**Theorem 4**: The recurrences in Figure 4 are correct to compute the edit distance \( \tau_{TA}(C_1[v]\!, C_2[w]) \) between \( C_1[v] \) and \( C_2[w] \) as \( \delta_{TA}(\emptyset \! \mid C_1[v]), \emptyset \! \mid C_2[w]) \).

**Proof**: It is obvious that the recurrences in (A) compute the edit distance when at least one forest is empty and the recurrence in (B) computes the edit distance between two forests such that both of them consist of just leaves.

For the recurrence (C), let \( M \) be the minimum cost mapping between \( \langle L_1 \mid \emptyset \rangle \) and \( \langle L_2 \mid C_2[w] \rangle \). By focusing on \( w \), \( M \) contains a pair of either \( (v, w) \) or \( (v, w) \) for some \( v \in L_1 \). See Figure 5.
\[
\begin{align*}
\delta_{\text{TM}}((L_1, C_1), \Phi) &= \sum_{v \in L_1} \delta(v, e) + \sum_{v \in C_1} \delta(e, v), \\
\delta_{\text{TM}}(\Phi, (L_2, C_2)) &= \sum_{v \in L_2} \delta(e, w) + \sum_{v \in C_2} \delta(e, w). 
\end{align*}
\]

(B) \( \delta_{\text{TM}}((L_1[0]), (L_2[0])) = \delta(\sigma(L_1), \sigma(L_2)). \)

(C) \( \delta_{\text{TM}}((L_1[0]), (L_2, C_2[w])) \)

\[
= \min \left\{ \begin{array}{l}
\gamma(e, w) + \delta_{\text{TM}}((L_1[0]), (L_2 \cup L_2(w) [B_2(w)])) \\
\gamma(e, w) + \delta_{\text{TM}}((L_2[0]), (L_2 \setminus \{e\})[0]) \\
\end{array} \right\}.
\]

(D) \( \delta_{\text{TM}}((L_1, C_1[w]), (L_2[0])) \)

\[
= \min \left\{ \begin{array}{l}
\gamma(e, w) + \delta_{\text{TM}}((L_1 \cup L_1(v) [B_1(v)], (L_2[0])) \\
\gamma(e, w) + \delta_{\text{TM}}((L_2[0]), (L_2 \setminus \{e\}[0])) \\
\end{array} \right\}.
\]

(E) \( \delta_{\text{TM}}((L_1, C_1[w]), (L_2, C_2[w])) \)

\[
= \min \left\{ \begin{array}{l}
\gamma(e, w) + \delta_{\text{TM}}((L_1 \cup L_1(v) [B_1(v)], (L_2, C_2[w])) \\
\gamma(e, w) + \delta_{\text{TM}}((L_1, C_1[w]), (L_2 \setminus L_2(w) [B_2(w)])) \\
\gamma(e, w) + \delta_{\text{TM}}((L_1, C_1[w]), (L_2 \setminus L_2(w) [B_2(w)])) \\
\end{array} \right\}.
\]

If \((e, w) \in M\), then \(M\) maps nodes in \(L_1[0]\) to those in \(L_2 \cup L_2(w) [B_2(w)]\), which is computed by \(\delta_{\text{TM}}((L_1[0]), (L_2 \cup L_2(w) [B_2(w)]))\). Hence, the formula (1) computes the cost of \(M\).

If \((v, w) \in M\) for some \(v \in L_1\), then \(M\) maps nodes in \(L_1 \setminus \{v\}[0]\) to those in \(L_2 \setminus L_2(w) [B_2(w)]\), which is computed by \(\delta_{\text{TM}}((L_1 \setminus \{v\}[0]), (L_2 \setminus L_2(w) [B_2(w)]))\). Since \(M\) is the minimum cost, it is necessary to minimize the value of \(\gamma(v, e) + \delta_{\text{TM}}((L_1 \setminus \{v\}[0]), (L_2 \setminus L_2(w) [B_2(w)]))\) for \(v \in L_1\). Furthermore, once \(M\) contains \((v, w)\) for some \(v \in L_1\), \(M\) touches no descendants of \(w\), that is, no nodes in \(L_2(w) [B_2(w)]\), which is computed by \(\delta_{\text{TM}}(\Phi, (L_2(w) [B_2(w)]))\). Hence, the formula (2) computes the cost of \(M\).

The recurrence (D) is correct as same as the recurrence (C). For the recurrence (E), let \(M\) be the minimum cost mapping between \((L_1, C_1[w])\) and \((L_2, C_2[w])\). By focusing on \(v\) and \(w\), \(M\) contains one of the pairs of \((v, w), (v, e)\), and \((e, w)\). See Figure 6.

If \((v, w) \in M\), then \(M\) maps \(L_1\) or \(L_2\) to \(B_2\) or \(B_2(w)\), and no nodes in \(L_2(w)\) to \(B_2(w)\). Then, \(M\) maps nodes in \(L_1\) to those in \(L_2\), which is computed by \(\delta_{\text{TM}}((L_1[0]), (L_2[0]))\). Also \(M\) maps nodes in \(L_1(v) [B_1(v)]\) to those in \(L_2(w) [B_2(w)]\), which is computed by \(\delta_{\text{TM}}((L_1(v) [B_1(v)]), (L_2(w) [B_2(w)]))\). Hence, the formula (5) computes the cost of \(M\).

If \((v, e) \in M\), then \(M\) maps nodes in \(L_1 \cup L_1(v) [B_1(v)]\) to those in \(L_2, C_2[w]\), which is computed by \(\delta_{\text{TM}}((L_1 \cup L_1(v) [B_1(v)]), (L_2, C_2[w]))\). Hence, the formula (6) computes the cost of \(M\).

If \((e, w) \in M\), then \(M\) maps nodes in \(L_1 \cup L_1(v) [B_1(v)]\) to those in \(L_2, C_2[w]\), which is computed by \(\delta_{\text{TM}}((L_1 \cup L_1(v) [B_1(v)]), (L_2, C_2[w]))\). Hence, the formula (7) computes the cost of \(M\).

**Example 2:** Consider two caterpillars \(C_1\) and \(C_2\) in Figure 2 in Example 1. By applying the recurrences in Figure 4, we obtain that the edit distance \(\tau_{\text{TM}}(C_1, C_2)\) between \(C_1\) and \(C_2\) is 3 as follows. Here, we represent a caterpillar as a term-like representation, that is, \(C_1 = (\lambda b, b, b)\) and
\[ C_2 = a(b, a(b(a, a))]. \]

\[ \tau_{\mathcal{D}}(C_1, C_2) = \delta_{\mathcal{D}}(\langle \emptyset | a(b, b) \rangle, \langle \emptyset | a(b, a(b(a, a))) \rangle) = \gamma(a, a) + \delta_{\mathcal{D}}(\langle \{b, b\} | \emptyset \rangle, \langle \{b, b\} | a(b(a, a))) \rangle) \]

\[ = \gamma(\varepsilon, a) + \delta_{\mathcal{D}}(\langle \{b, b\} | \emptyset \rangle, \langle \{b, b\} | a(b(a, a))) \rangle) \]

\[ = 1 + \gamma(b, b) + \delta_{\mathcal{D}}(\langle \{b, b\} | \emptyset \rangle, \langle \{b, b\} | \emptyset \rangle) \]

\[ = 1 + \gamma(\varepsilon, a) + \gamma(\varepsilon, a) \]

\[ = 3. \]

Hence, we can obtain the optimum mapping \( M \) between \( C_1 \) and \( C_2 \) illustrated in Figure 2, by collecting the pairs \( (L_1, L_2) \in C_1 \times C_2 \) such that \( l_1 \neq e \) and \( l_2 \neq e \) in \( \gamma(L_1, L_2) \).

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 \( par(w_i) = w_{i+1} (1 \leq j \leq m - 1) \). 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 4, Algorithm 1 illustrates the algorithm to compute the edit distance \( \tau_{\mathcal{D}}(C_1, C_2) \) between caterpillars \( C_1 \) and \( C_2 \). Here, the recurrence (A), (B), (C), (D) and (E) are corresponding to the lines 6 and 12, the line 3, the line 9, the line 15 and the line 19, respectively, in Algorithm 1.

**Theorem 5:** Let \( C_1 \) and \( C_2 \) be caterpillars. Then, we can compute the edit distance \( \tau_{\mathcal{D}}(C_1, C_2) \) between \( C_1 \) and \( C_2 \) in \( O(n^2 h^2) \) time, where \( h = \max \{ |\mathbf{v}(C_1)|, |\mathbf{v}(C_2)| \} \) and \( h = \max \{ h(C_1), h(C_2) \} \).

**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 \( \tau_{\mathcal{D}}(C_1, C_2) \) in Algorithm 1 calls every pair \( (v_i, w_j) \in bb(C_1) \times bb(C_2) \) just once. When computing \( \delta_{\mathcal{D}}(L_1(v_i), L_2(w_j)) \) for \( 2 \leq i \leq n \) and \( 2 \leq j \leq m \), it is necessary to construct the string representations \( s_1 = s(L_1(v_1) \cup \cdots \cup L_1(v_{i-1})) \) and \( s_2 = s(L_2(w_1) \cup \cdots \cup L_2(w_{j-1})) \) and compute the string edit
procedure $\tau_{TA}(C_1, C_2)$
\[ \begin{align*}
\text{if } C_1, C_2: \text{ categorpillars } & \gamma \\
\tau_{TA}(C_1, C_2) & \leftarrow \delta_{TA}(\emptyset | C_1, \emptyset | C_2); \\
\text{end if}
\end{align*} \]

procedure $\delta_{TA}(L_1 | C_1, L_2 | C_2)$
\[ \begin{align*}
\text{if } L_1, L_2: \text{ set of leaves, } C_1, C_2: \text{ categorpillars } & \gamma \\
\delta_{TA}(L_1 | \emptyset, L_2 | \emptyset) & \leftarrow \text{ compute the recurrence (B)}; \\
\text{else if } C_1 = \emptyset \text{ and } C_2 = \emptyset & \text{ then} \\
\delta_{TA}(\emptyset | L_1, \emptyset | L_2) & \leftarrow \text{ compute the recurrence (A)}; \\
\text{end if}
\end{align*} \]

for $j = 1 \text{ to } m$ do
\[ \delta_{TA}((L_1 | C_1), (L_2 | C_2)) \leftarrow \text{ compute the recurrence (C)}; \]

else if $C_1 \neq \emptyset$ and $C_2 = \emptyset$ then
\[ \begin{align*}
\text{if } L_2 = \emptyset & \text{ then} \\
\delta_{TA}(L_1 | C_1, \emptyset) & \leftarrow \text{ compute the recurrence (A)}; \\
\text{end if}
\end{align*} \]

else for $i = 1 \text{ to } n$ do
\[ \delta_{TA}((L_1 | C_1[v_i]), (L_2 | \emptyset)) \leftarrow \text{ compute the recurrence (D)}; \]

else
\[ \begin{align*}
\text{if } L_2 = \emptyset & \text{ then} \\
\delta_{TA}(L_1 | C_1, \emptyset) & \leftarrow \text{ compute the recurrence (A)}; \\
\text{end if}
\end{align*} \]

else for $i = 1 \text{ to } n$ do
\[ \delta_{TA}((L_1 | C_1[v_i]), (L_2 | C_2[w_j])) \leftarrow \text{ compute the recurrence (E)}; \]

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

(distance $\sigma(s_1, s_2)$. The running time to construct the string representations is $O(\lambda \log \lambda)$ time (as same as that of sorting) and to compute the string edit distance is $O(\lambda^2)$ time [2].

Hence, the total running time of Algorithm 1 is described as follows:
\[ \sum_{i=1}^{n} \sum_{j=1}^{m} (2O(\lambda \log \lambda) + O(\lambda^2)) = O(\lambda^2) \cdot mn \]
\[ \leq O(\lambda^2)(h-1)^2 = O(\lambda^2 n^2). \]

Theorem 5 also claims that the structural restriction of caterpillars provides the limitation of tractable computing the edit distance for unordered trees. 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 corollaries in the proof of [1] or [4].

Theorem 6 (cf. [11, [4]): The problem of computing the edit distance between generalized caterpillars is MAX SNPhard, even if the maximum height is at most 3.

Proof: It is straightforward from the proof of Corollary 4.3 in [1] or Theorem 1 in [4].

Finally, Table I illustrates the number of pairs $(C_1, C_2)$ of caterpillars such that $\tau_{TA}(C_1, C_2) < \tau_{BST}(C_1, C_2)$ and $\tau_{TA}(C_1, C_2) = \tau_{BST}(C_1, C_2)$, respectively, for all the pairs of 514 caterpillars in N-glycans (in Table II).

<table>
<thead>
<tr>
<th>$\tau_{TA}$</th>
<th>$\tau_{BST}$</th>
<th>$\tau_{TA} = \tau_{BST}$</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.218</td>
<td>130.618</td>
<td>131.841</td>
</tr>
</tbody>
</table>

Concerned with the 5 pairs in Table I, Figure 7 illustrates the caterpillars $C_1 = G04187$, $C_2 = G00698$, $C_3 = G00933$, $C_4 = G01221$, $C_5 = G01454$ and $C_6 = G11051$ in N-glycans such that $\tau_{BST}(C_1, C_4) = \tau_{TA}(C_1, C_4) = 2$ for $2 \leq i \leq 6$.

Fig. 7. The caterpillars $C_1 = G04187$, $C_2 = G00698$, $C_3 = G00933$, $C_4 = G01221$, $C_5 = G01454$ and $C_6 = G11051$.

Here, the following statements hold:
\[ \tau_{TA}(C_1, C_2) = 6, \tau_{BST}(C_1, C_2) = 8, \tau_{TA}(C_1, C_3) = 7, \tau_{BST}(C_1, C_3) = 9, \tau_{TA}(C_1, C_4) = 8, \tau_{BST}(C_1, C_4) = 10, \tau_{TA}(C_1, C_5) = 7, \tau_{BST}(C_1, C_5) = 9, \tau_{TA}(C_1, C_6) = 4, \tau_{BST}(C_1, C_6) = 6. \]

Figure 8 illustrates the optimum mappings $M_1 \in \mathcal{M}_{TA}(C_1, C_2)$ and $M_2 \in \mathcal{M}_{BST}(C_1, C_2)$ for caterpillars $C_1$ and $C_2$ in Figure 7, which is the reason that $\tau_{BST}(C_1, C_2) = 6$ (5 deleted nodes and 1 substituted node) and $\tau_{TA}(C_1, C_2) = 8$ (6 deleted nodes, 1 inserted node and 1 substituted node).

IV. CONCLUSION AND FUTURE WORKS

In this paper, we have designed the algorithm to compute the edit distance between caterpillars in $O(\lambda^2 n^2)$ time, which
is the limitation of tractable computing the edit distance for unordered trees.

Whereas we have given a small experimental result in the last of Section III, it is necessary to implement Algorithm 1 more efficiently. Then, it is a future work to evaluate running time from all the data of caterpillars in Appendix by comparing that of the algorithm of computing $\tau_{ST}$ [10], [11], [12] and to investigate the difference between $\tau_{TA}$ and $\tau_{BST}$. Also, it is a future work to analyze the correlation for caterpillars in real data between the edit distance and the complete subtree histogram distance [1] or the path histogram distance [6].

Concerned with Theorem 6, it is a future work to give the strict limitation of tractable computing of the edit distance. In other words, it is a future work to investigate whether or not the problem of computing the edit distance between a caterpillar and a generalized caterpillar or a standard tree is tractable. In particular, concerned with $D^-$ for $D \in \{\text{Auction, University, Protein, Nasa}\}$ in Table II in Appendix, it is a future work to investigate whether or not the problem of computing the edit distance between forests of caterpillars is tractable.

As the extension of the edit distance for rooted trees to that for rooted trees, Zhang et al. [14] have extend the degree-2 distance for rooted trees to that for unrooted trees. In their algorithm, first we select a pair of nodes in unrooted trees, compute the degree-2 distance between the rooted trees whose pair of the roots is the selected pair and then select the minimum value of the distances as the degree-2 distance. It is a future work to investigate whether or not we can apply this idea to the problem of computing the edit distance between unrooted caterpillars and, if so, design the algorithm to compute it.

**Appendix: Caterpillars in Real Data**

In this appendix, we point out how large the number of caterpillars in real data. Table II illustrates the number of caterpillars in N-glycans and all glycans from KEGG, CSLOGS, dhlp, SwissProt, TPC-H, Auction, University, Protein and Nasa.

<table>
<thead>
<tr>
<th>dataset</th>
<th>#cat</th>
<th>#data</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-glycans</td>
<td>514</td>
<td>2,142</td>
<td>23.996</td>
</tr>
<tr>
<td>all glycans</td>
<td>8,005</td>
<td>10,704</td>
<td>74.785</td>
</tr>
<tr>
<td>CSLOGS</td>
<td>41,592</td>
<td>59,691</td>
<td>69.679</td>
</tr>
<tr>
<td>dhlp</td>
<td>5,154,295</td>
<td>5,154,580</td>
<td>99.995</td>
</tr>
<tr>
<td>SwissProt</td>
<td>6,804</td>
<td>50,000</td>
<td>13.608</td>
</tr>
<tr>
<td>TPC-H</td>
<td>86,805</td>
<td>100,000</td>
<td>86.805</td>
</tr>
<tr>
<td>Auction</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>University</td>
<td>0</td>
<td>6,738</td>
<td>0</td>
</tr>
<tr>
<td>Protein</td>
<td>0</td>
<td>2,6225</td>
<td>0</td>
</tr>
<tr>
<td>Nasa</td>
<td>0</td>
<td>2,430</td>
<td>0</td>
</tr>
<tr>
<td>Auction'</td>
<td>259</td>
<td>259</td>
<td>100.000</td>
</tr>
<tr>
<td>University'</td>
<td>74,638</td>
<td>79,213</td>
<td>94.224</td>
</tr>
<tr>
<td>Protein'</td>
<td>1,874,583</td>
<td>2,204,000</td>
<td>85.857</td>
</tr>
<tr>
<td>Nasa'</td>
<td>21,245</td>
<td>27,921</td>
<td>76.089</td>
</tr>
</tbody>
</table>

Here, #cat is the number of caterpillars and #data is the total number of data. Furthermore, for $D \in \{\text{Auction, University, Protein, Nasa}\}$, $D^-$ 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 $D$.

**References**


1KOHEI MURAKA ET AL.: COMPUTING EDIT DISTANCE BETWEEN ROOTED LABELED CATERPILLARS 255

2http://www.kegg.jp/


4http://diseweb.cs.washington.edu/research/projects/xmlk/xml/indexdata/www

Repository.html