Computing the Variations of Edit Distance for Rooted Labaled Caterpillars

Manami Hagihara, Takuya Yoshino and Kouich Hirata
Kyushu Institute of Technology, Kawazu 680-4, Iizuka 820-8502, Japan

Keywords: Edit Distance, Rooted Labeled Caterpillar, Rooted Labeled Unordered Tree, Top-down Distance, Bottom-up Distance, LCA-preserving Distance.

Abstract: In this paper, we pay our attention to *top-down distance*, *LCA-preserving distance* and *bottom-up distance* for *rooted labeled caterpillars* (*caterpillars*, for short), as the variations of the edit distance. Here, the top-down distance is the edit distance that the deletion and the insertion are allowed to just leaves, the LCA-preserving distance is one to just either leaves or vertices with one child and the bottom-up distance is one to just the root. Then, we show that the top-down and the bottom-up distances for caterpillars can be computed in $O(n)$ time and the LCA-preserving distance for caterpillars in $O(n^2)$ time. Furthermore, we give experimental results of computing these variations for caterpillars in real data.

1 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 (Deza and Deza, 2016) between *rooted labeled unordered trees* (*trees*, for short) is the *edit distance* τ_{TAI} (Tai, 1979). 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* (Tai, 1979). Unfortunately, the problem of computing the edit distance between trees is MAX SNP-hard (Zhang and Jiang, 1994), even if trees are binary or the maximum height of trees is at most 3 (Akutsu et al., 2013; Hirata et al., 2011).

Whereas the edit distance is the standard measure for comparing trees, it is too general for several applications. Therefore, more structurally sensitive distances of the edit distance such as the *top-down* (or *degree*-1) distance τ_{TOP} (Chawathe, 1999; Selkow, 1977), the *LCA-preserving* (or *degree*-2) distance τ_{LCA} (Zhang et al., 1996) and the *bottom-up* distance τ_{BOT} (Valiente, 2001) required for these applications. Such distances are formulated as the minimum cost of the variations of the Tai mapping such as a *top-down mapping* (Chawathe, 1999; Selkow, 1977), an *LCA-preserving mapping* (Zhang et al., 1996) and a *bottom-up mapping* (Kuboyama, 2007; Valiente, 2001) respectively.

As operational, the top-down distance is the edit distance that the deletion and the insertion are allowed to just leaves, the LCA-preserving distance is one to just either leaves or vertices with one child and the bottom-up distance is one to just the root. Yoshino and Hirata (Yoshino and Hirata, 2017) have summarized and characterized the other variations of the Tai mapping as a Tai mapping hierarchy.

For trees, we can compute the top-down and the LCA-preserving distances in $O(n^2d)$ time (Yamamoto et al., 2014; Zhang et al., 1996), where n is the maximum number of vertices and d is the minimum degree in two trees. On the other hand, the problems of computing the bottom-up distance is MAX SNP-hard (Yamamoto et al., 2014).

A *caterpillar* (*cf.* (Gallian, 2007)) 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 (*cf.*, (Muraka et al., 2018; Ukita et al., 2021)). Recently, Muraka *et al.* (Muraka et al., 2018) have proposed the algorithm to compute the edit distance between caterpillars in $O(n^2\lambda)$ time under the unit cost function, where λ is the maximum number of leaves in caterpillars[1].

[1]This time complexity is different from the result in

Hagihara, M., Yoshino, T. and Hirata, K.
Computing the Variations of Edit Distance for Rooted Labaled Caterpillars.
DOI: 10.5220/0010826100003122
In *Proceedings of the 11th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2022)*, pages 272-279
ISBN: 978-989-758-549-4; ISSN: 2184-4313

Hence, in this paper, we pay our attention to the top-down, the LCA-preserving and bottom-up distances for caterpillars as the variations of the edit distance. Then, we design the algorithm to compute them and show that the top-down distance and the bottom-up distance for caterpillars can be computed in $O(n)$ time and the LCA-preserving distance for caterpillars in $O(n^2)$ time, see Table 1.

Table 1: The time complexity of computing τ_{TAI}, τ_{TOP}, τ_{LCA} and τ_{BOT} for trees and caterpillars. Here, n is the maximum number of vertices, d is the minimum degree and λ is the maximum number of leaves in two trees or caterpillars.

distance	tree	caterpillar
τ_{TAI}	MAX SNP-hard (Zhang and Jiang, 1994)	$O(n^2\lambda)$ (Muraka et al., 2018)
τ_{TOP}	$O(n^2d)$ (Yamamoto et al., 2014)	$O(n)$ Theorem 4
τ_{LCA}	$O(n^2d)$ (Yamamoto et al., 2014)	$O(n^2)$ Theorem 5
τ_{BOT}	MAX SNP-hard (Yamamoto et al., 2014)	$O(n)$ Theorem 6

Also, we give experimental results of computing these variations for caterpillars in real data. In particular, we compare the running time of the algorithms in this paper with the previous algorithms of computing the top-down and the LCA-preserving distances for trees (Yamamoto et al., 2014).

2 PRELIMINARIES

A *tree* is a connected graph without cycles. For a tree $T=(V,E)$, we denote V and E by $V(T)$ and $E(T)$. We sometimes denote $v \in V(T)$ by $v \in T$. A *rooted tree* is a tree with one vertex r chosen as its *root*, which we denote by $r(T)$.

For each vertex v in a rooted tree with the root r, let $UP_r(v)$ be the unique path from v to r. The *parent* of $v(\neq r)$, which we denote by $par(v)$, is its adjacent vertex on $UP_r(v)$ and the *ancestors* of $v(\neq r)$ are the vertices 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 all children of v by $ch(v)$. Two vertices with the same parent are called *siblings*. A *leaf* is a vertex having no children. We denote the set of all leaves in a tree T by $lv(T)$.

We denote $u<v$ if v is an ancestor of u, and we denote $u \leq v$ if either $u<v$ or $u=v$. Also 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 w' such that $u \leq w'$, $v \leq w'$ and $w' \leq w$. 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 $E'=\{(u,w) \in E \mid u,w \in V'\}$.

The *height* $h(v)$ of v is defined as $|UP_r(v)|-1$ and the *height* $h(T)$ of T is the maximum height for every vertex $v \in T$. The *degree* $d(v)$ of v is the number of the children of $v \in T$. and the *degree* $d(T)$ of T is the maximum degree for every vertex in T.

We say that a rooted tree is *ordered* if a left-to-right order among siblings is given; *Unordered* otherwise. Also we say that a tree is *labeled* over Σ if each vertex is assigned a symbol from a fixed finite alphabet Σ, where we denote the label of a vertex v by $l(v)$, and sometimes identify v with $l(v)$. In this paper, we call a rooted labeled unordered tree over Σ a *tree*, simply.

As the restricted form of trees, we introduce a *rooted labeled caterpillar* (*caterpillar*, for short).

Definition 1. We say that a tree is a *caterpillar* (*cf.* (Gallian, 2007)) 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 vertex in a caterpillar is either a leaf or an element of the backbone.

Next, we introduce a *tree edit distance* and a *Tai mapping*.

Definition 2 (Edit operations (Tai, 1979))**.** 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 subsequence in the left-to-right order 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 consecutive subsequence a subset of the children of v'.

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

(Muraka et al., 2018), because it contains some errors. See (Ukita et al., 2021) in more detail.

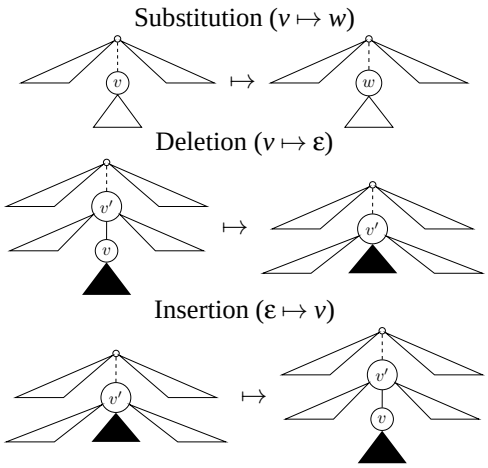

Figure 1: Edit operations for trees.

pairs of labels. 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 (Tai, 1979))**.** 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* $\tau_{\mathrm{TAI}}(T_1,T_2)$ between trees T_1 and T_2 is defined as follows:

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

Definition 4 (Tai mapping (Tai, 1979))**.** 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 $M \in \mathcal{M}_{\mathrm{TAI}}(T_1,T_2)$.

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 without labels*, denoted by $T_1 \equiv_l T_2$, if there exists a mapping $M \in \mathcal{M}_{\mathrm{TAI}}(T_1,T_2)$ such that $I_M = J_M = \emptyset$, and *isomorphic*, denoted by $T_1 \equiv T_2$, if there exists a mapping $M \in \mathcal{M}_{\mathrm{TAI}}(T_1,T_2)$ such that $I_M = J_M = \emptyset$ and $\gamma(M) = 0$.

Theorem 1. (Tai, 1979) *It holds that:*

$$\tau_{\mathrm{TAI}}(T_1,T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\mathrm{TAI}}(T_1,T_2)\}.$$

Furthermore, we introduce the variations of the Tai mapping and the edit distance, which are main topics in this paper.

Definition 5. Let T and S be trees and suppose that $M \in \mathcal{M}_{\mathrm{TAI}}(T,S)$. We define M^- as $M \setminus \{r(T),r(S)\}$.

1. We say that M is a *top-down mapping* (Chawathe, 1999; Selkow, 1977), which we denote by $M \in \mathcal{M}_{\mathrm{TOP}}(T,S)$, if $(par(v),par(w)) \in M$ for every $(v,w) \in M^-$.
2. We say that M is an *LCA-preserving mapping* (or *degree-2 mapping*) (Zhang et al., 1996), which we denote by $M \in \mathcal{M}_{\mathrm{LCA}}(T,S)$ if $(v \sqcup v', w \sqcup w') \in M$ for every $(v,w),(v',w') \in M$.
3. We say that M is a *bottom-up mapping* (Valiente, 2001), which we denote by $M \in \mathcal{M}_{\mathrm{BOT}}(T,S)$, if the following condition holds for every $(v,w) \in M$.

$$\left(\forall v' \in T[v] \exists w' \in S[w] \Big((v',w') \in M \Big)\right) \wedge \left(\forall w' \in S[w] \exists v' \in T[v] \Big((v',w') \in M \Big)\right).$$

Furthermore, for $\circ \in \{\mathrm{TOP}, \mathrm{LCA}, \mathrm{BOT}\}$, we define the distance $\tau_\circ(T,S)$ between T and S as the minimum cost of all the mappings in $\mathcal{M}_\circ(T,S)$, that is:

$$\tau_\circ(T,S) = \min\{\gamma(M) \mid M \in \mathcal{M}_\circ(T,S)\}.$$

Here, we call τ_{TOP}, τ_{LCA} and τ_{BOT} a *top-down distance*, an *LCA-preserving distance* and a *bottom-up distance*, respectively.

As the time complexity of the variations of the edit distance in Definition 5, the following theorem holds, also see Table 1 in Section 1.

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

1. The problem of computing $\tau_{\mathrm{TAI}}(T,S)$ is MAX SNP-hard (Zhang and Jiang, 1994). This statement also holds even if both T and S are binary trees or the maximum height of trees is at most 3 (Akutsu et al., 2013; Hirata et al., 2011).
2. We can compute $\tau_{\mathrm{TOP}}(T,S)$ and $\tau_{\mathrm{LCA}}(T,S)$ in $O(n^2 d)$ time (Yamamoto et al., 2014).
3. The problem of computing $\tau_{\mathrm{BOT}}(T,S)$ is MAX SNP-hard. This statement also holds even if both T and S are binary trees (Yamamoto et al., 2014).

It is know the following theorem for caterpillars.

Theorem 3. (Muraka et al., 2018) *Let C and D be caterpillars, where $n = \max\{|C|,|D|\}$ and $\lambda = \min\{|lv(C)|,|lv(D)|\}$. Then, we can compute $\tau_{\mathrm{TAI}}(C,D)$ in $O(n^2\lambda)$ time under the unit cost function.*

3 COMPUTING THE VARIATIONS FOR CATERPILLARS

Let C and D be caterpillars. We regard a backbone $bb(C)$ as a sequence $[v_1,\dots,v_n]$, where $v_1 = r(C)$ and $(v_i,v_{i+1}) \in E(C)$, and a backbone $bb(D)$ as a sequence $[w_1,\dots,w_m]$, where $w_1 = r(D)$ and $(w_j,w_{j+1}) \in E(D)$.

Let L_i $(1 \le i \le n)$ denote the set of leaves in $ch(v_i)$, that is, $L_i = ch(v_i) \setminus \{v_{i+1}\}$ for $1 \le i \le n-1$ and $L_n = ch(v_n)$. Also Let K_j $(1 \le j \le m)$ denote the set of leaves in $ch(w_j)$, that is, let $K_j = ch(w_j) \setminus \{w_{j+1}\}$ for $1 \le j \le m-1$ and $K_m = ch(w_m)$.

Recall that $C[v]$ denotes the (complete) subcaterpillar of C rooted at v. Also $C(v)$ denotes the forest obtained by deleting the root v in $C[v]$. For a Caterpillar C and a subcaterpillar C' of C, we denote the caterpillar obtained by deleting C' from C by $C \setminus C'$.

When designing the algorithm to compute the variations of edit distance for caterpillars, we use a *multiset* of labels on an alphabet Σ. A *multiset* on 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 $\sum_{a\in\Sigma} S(a)$.

Let S_1 and S_2 be multisets on Σ. Then, we define the *intersection* $S_1 \sqcap S_2$, the *union* $S_1 \sqcup S_2$ and the *difference* $S_1 \setminus S_2$ as multisets satisfying that $(S_1 \sqcap S_2)(a) = \min\{S_1(a),S_2(a)\}$, $(S_1 \sqcup S_2)(a) = \max\{S_1(a),S_2(a)\}$ and $(S_1 \setminus S_2)(a) = \max\{S_1(a) - S_2(a),0\}$ for every $a \in \Sigma$. Note that $S_1 \setminus S_2 = S_1 \setminus (S_1 \sqcap S_2)$ and $|S_1 \setminus S_2| = |S_1 \setminus (S_1 \sqcap S_2)| = |S_1| - |S_1 \sqcap S_2|$.

We can compute the edit distance μ between multisets S and S' in $O(|S| + |S'|)$ time, since $\mu(S,S') = \max\{|S \setminus S'|, |S' \setminus S|\}$ under the unit cost function (*cf.*, (Ukita et al., 2021)).

Let S be a set of vertices. Then, we denote the multiset of labels on Σ occurring in S by $\widetilde{S}$. Also, we denote $\sum_{v\in S} \gamma(v,\varepsilon)$ by $del(S)$ and $\sum_{w\in S} \gamma(\varepsilon,w)$ by $ins(S)$.

3.1 Top-down Distance

First, we consider the equation to compute the top-down distance $\tau_{\mathrm{TOP}}(C,D)$ between caterpillars C and D, illustrated in Figure 2.

Theorem 4. *Let C and D be caterpillars, where $n = \max\{|C|,|D|\}$. Then, we can compute $\tau_{\mathrm{TOP}}(C,D)$ in $O(n)$ time under the unit cost function.*

Proof. First, we show that the equations in Figure 2 is correct. Suppose that $M \in \mathcal{M}_{\mathrm{TOP}}(C,D)$. Note that $bb(C) = [v_1,\dots,v_n]$ and $bb(D) = [w_1,\dots,w_m]$.

$$\tau_{\mathrm{TOP}}(C,D) = \sum_{i=1}^{\min\{n,m\}-2} \Big(\gamma(v_i,w_i) + \mu(\widetilde{L_i},\widetilde{K_i})\Big) + d_{n,m}(C,D),$$

$$d_{n,m}(C,D) = \begin{cases} \gamma(v_{m-1},w_{m-1}) + \\ \min\left\{\begin{array}{l}\gamma(v_m,w_m) + \mu(\widetilde{ch(v_m)},\widetilde{K_m}) + del(C(v_{m+1})),\\ \mu(ch(v_{m-1}),ch(w_{m-1})) + del(C(v_m)) + ins(K_m)\end{array}\right\} \\ \quad \text{if } n > m, \\ \gamma(v_{n-1},w_{n-1}) + \\ \min\left\{\begin{array}{l}\gamma(v_n,w_n) + \mu(\widetilde{L_n},\widetilde{K_n}),\\ \mu(\widetilde{ch(v_{n-1})},\widetilde{ch(w_{n-1})}) + del(L_n) + ins(K_n)\end{array}\right\} \\ \quad \text{if } n = m, \\ \gamma(v_{n-1},w_{n-1}) + \\ \min\left\{\begin{array}{l}\gamma(v_n,w_n) + \mu(\widetilde{L_n},\widetilde{ch(w_n)}) + ins(D(w_{n+1})),\\ \mu(\widetilde{ch(v_{n-1})},\widetilde{ch(w_{n-1})}) + ins(D(w_n)) + del(L_n)\end{array}\right\} \\ \quad \text{if } n < m. \end{cases}$$

Figure 2: The equations of computing τ_{TOP}.

It is obvious that $(v_1,w_1) \in M$. Also, if $(v_i,w) \in M$ (*resp.*, $(v,w_j) \in M$), then it holds that $w = w_i$ (*resp.*, $v = v_j$). Hence, there exists an index h such that $1 \le h \le \min\{n,m\}$ and M contains the pairs $(v_1,w_1),\dots,(v_h,w_h)$. Furthermore, if M is the minimum cost, then M contains the pairs (v_h,w_h) as many as possible, so such an h is $\min\{n,m\}$ and such an M implies $\tau_{\mathrm{TOP}}(C,D)$.

For every i $(1 \le i \le h-2)$, we can compute the correspondences in M between the leaves in L_i and the leaves in K_i as $\mu(\widetilde{L_i},\widetilde{K_i})$, where $\widetilde{L_i} \sqcap \widetilde{K_i}$ implicitly represents such correspondences. Then, it holds that $\tau_{\mathrm{TOP}}(C \setminus C[h], D \setminus D[h]) = \sum_{i=1}^{h-2} \Big(\gamma(v_i,w_i) + \mu(\widetilde{L_i},\widetilde{K_i})\Big)$, which is computed in the formula of $\tau_{\mathrm{TOP}}(C,D)$ except $d_{n,m}(C,D)$ in Figure 2.

Consider the case that $i = h-1$, that is, consider the formula $d_{n,m}(C,D)$.

If $n = m$, then we can compute $\tau_{\mathrm{TOP}}(C[v_{n-1}],D[v_{n-1}])$ as the sum of $\gamma(v_{n-1},w_{n-1})$ and the minimum value of $\gamma(v_n,w_n) + \mu(\widetilde{L_n},\widetilde{K_n})$ (if v_n is corresponding to w_n) and $\mu(\widetilde{ch(v_{n-1})},\widetilde{ch(w_{n-1})}) + del(L_n) + ins(K_n)$ (otherwise), which is realized as the second formula in $d_{n,m}(C,D)$ in Figure 2.

Suppose that $n > m$. Then, we can compute $\tau_{\mathrm{TOP}}(C[v_{n-1}],D[v_{n-1}])$ as the sum of $\gamma(v_{m-1},w_{m-1})$ and the minimum value of the upper and the lower formulas in the first formula in $d_{n,m}(C,D)$ in Figure 2.

If v_m is corresponding to w_m, then its cost is $\gamma(v_m,w_m)$ and the leaves in K_m are possible to correspond to not only the leaves in L_m but also v_{m+1}, that is, $ch(v_m)$. Such correspondences are computed as $\mu(\widetilde{ch(v_m)},\widetilde{K_m})$. Furthermore, the remained vertices in $C(v_{m+1})$ are deleted, which is realized as

$del(C(v_{m+1}))$. Hence, the upper formula is correct.

Otherwise, that is, if v_m is not corresponding to w_m, then the vertices in $ch(v_{m-1})$ are corresponding to the vertices in $ch(w_{m-1})$. Such correspondences are computed as $\mu(\widetilde{ch(v_{m-1})}, \widetilde{ch(w_{m-1})})$. Furthermore, the remained vertices in $C(v_m)$ are deleted and the remained vertices in K_m are inserted, which is realized as $del(C(v_m)) + ins(K_m)$. Hence, the lower formula is correct. Therefore, the first formula is correct.

Similarly, for the case that $n < m$, the third formula in $d_{n,m}(C,D)$ in Figure 2 is also correct.

Since the equations traverse at most once for every vertex in C and D with traversing (v_i, w_i) and the processing for (v_i, w_i) runs in $O(1)$ time, the total running time is $O(|C| + |D|) = O(n)$. □

3.2 LCA-preserving Distance

Next, we consider the recurrences of computing the LCA-preserving distance $\tau_{\mathrm{LCA}}(C,D)$ between caterpillars C and D illustrated in in Figure 3. Here, we regard $\widetilde{C[v_{i+1}]}$ and $\widetilde{D[w_{j+1}]}$ in the recurrences as the multisets of labels occurring in all the vertices in $C[v_{i+1}]$ and $D[w_{j+1}]$.

$$\tau_{\mathrm{LCA}}(C[v_i], D[w_j]) =$$
$$\min \left\{ \begin{array}{l} \gamma(v_i, w_j) + \mu(\widetilde{L_i}, \widetilde{K_j}) + \tau_{\mathrm{LCA}}(C[v_{i+1}], D[w_{j+1}]), \\ \gamma(v_i, \varepsilon) + del(L_i) + \tau_{\mathrm{LCA}}(C[v_{i+1}], D[w_j]), \\ \gamma(\varepsilon, w_j) + ins(K_j) + \tau_{\mathrm{LCA}}(C[v_i], D[w_{j+1}]) \end{array} \right\}$$
$$\text{if } 1 \le i < n \text{ and } 1 \le j < m,$$
$$\tau_{\mathrm{LCA}}(C[v_n], D[w_j]) =$$
$$\min \left\{ \begin{array}{l} \gamma(\varepsilon, w_j) + ins(K_j) + \tau_{\mathrm{LCA}}(C[v_n], D[w_{j+1}]), \\ \gamma(v_n, w_j) \\ \quad + \min_{v \in L_n} \left\{ \mu(\widetilde{L_n} \setminus \widetilde{\{v\}}, \widetilde{K_j}) + \mu(\widetilde{\{v\}}, \widetilde{D[w_{j+1}]}) \right\}, \\ \gamma(v_n, w_j) + \mu(\widetilde{L_n}, \widetilde{K_j}) + ins(D[w_{j+1}]) \end{array} \right\}$$
$$\text{if } 1 \le j < m,$$
$$\tau_{\mathrm{LCA}}(C[v_i], D[w_m]) =$$
$$\min \left\{ \begin{array}{l} \gamma(v_i, \varepsilon) + del(L_i) + \tau_{\mathrm{LCA}}(C[v_{i+1}], D[w_j]), \\ \gamma(v_i, w_m) \\ \quad + \min_{w \in K_m} \left\{ \mu(\widetilde{L_i}, \widetilde{K_m} \setminus \widetilde{\{w\}}) + \mu(\widetilde{C[v_{i+1}]}, \widetilde{\{w\}}) \right\}, \\ \gamma(v_i, w_m) + \mu(\widetilde{L_i}, \widetilde{K_m}) + del(C[v_{i+1}]) \end{array} \right\}$$
$$\text{if } 1 \le i < n,$$
$$\tau_{\mathrm{LCA}}(C[v_n], D[w_m]) = \gamma(v_n, w_m) + \mu(\widetilde{L_n}, \widetilde{K_m}).$$

Figure 3: The recurrences of computing τ_{LCA}.

We start the following simple lemma.

Lemma 1. *Let C be a caterpillar. For distinct vertices $v, w \in C$, it holds that $v \sqcup w \in bb(C)$.*

Proof. If $v \in bb(C)$ and $w \in bb(C)$, then it holds that $v \sqcup w = v$ or w, which implies that $v \sqcup w \in bb(C)$. If $v \in bb(C)$ and $w \in lv(C)$, then it holds that $v \sqcup w = v \sqcup par(w)$, which implies that $v \sqcup w \in bb(C)$. By the same reason, it holds that $v \sqcup w \in bb(C)$ if $v \in lv(C)$ and $w \in bb(C)$. If $v \in lv(C)$ and $w \in lv(C)$, then it holds that $v \sqcup w = par(v) \sqcup par(w)$, which implies that $v \sqcup w \in bb(C)$. □

Theorem 5. *Let C and D be caterpillar, where $n = \max\{|C|, |D|\}$. Then, we can compute $\tau_{\mathrm{LCA}}(C,D)$ in $O(n^2)$ time under the unit cost function.*

Proof. The first recurrence in Figure 3 computes that, for $M \in \mathcal{M}_{\mathrm{LCA}}(C,D)$, (1) if $(v_i, w_j) \in M$, then $(v,w) \in L_i \times K_j$ such that $v, w \in \widetilde{L_i} \sqcap \widetilde{K_j}$ are added to M and next it computes $\tau_{\mathrm{LCA}}(C[v_{i+1}], D[w_{j+1}])$, (2) if v_i is deleted, then all the leaves in L_i are deleted and next it computes $\tau_{\mathrm{LCA}}(C[v_{i+1}], D[w_j])$, or (3) if w_j is inserted, then all the leaves in K_j are inserted and next it computes $\tau_{\mathrm{LCA}}(C[v_i], D[w_{j+1}])$, for $1 \le i < n$ and $1 \le j < m$. Then, the pairs added to M are obtained from just the case (1), and the pairs consist of some $(v,w) \in L_i \times K_j$ and (v_i, w_j). Since $par(v) = v_i$ and $par(w) = w_j$ and by Lemma 1, M is LCA-preserving.

By the same reason, the mapping obtained from the last recurrence in Figure 3 is LCA-preserving.

Consider the second recurrence in Figure 3, that is, the case that $\tau_{\mathrm{LCA}}(C[v_n], D[w_j])$ for $1 \le j < m$ and $M \in \mathcal{M}_{\mathrm{LCA}}(C,D)$. The first formula means to insert w_j and L_j, and next compute $\tau_{\mathrm{LCA}}(C[v_n], D[w_{j+1}])$, no pairs are added to M.

The second and third formulas mean to add (v_n, w_j) to M. Then, the second formula means that some $v \in L_n$ is corresponding to some vertex $w \in D[w_{j+1}]$ (and (v,w) is added to M and the remained vertices in $D[w_{j+1}]$ are inserted), and then $L_n \setminus \{v\}$ is corresponding to K_j as possible (and the corresponding pairs are added to M). On the other hand, the third formula means that no $v \in L_n$ is corresponding to $D[w_{j+1}]$. In this case, L_n is corresponding to K_j as possible (and the corresponding pairs are added to M) and the vertices in $D[w_{j+1}]$ are inserted.

For both formulas, by Lemma 1, it holds that $(v_1 \sqcup v_2, w_1 \sqcup w_2) = (v_n, w_{j+1}) \in M$ for distinct pairs $(v_1, w_1), (v_2, w_2) \in M \cap (C[v_n] \times D[w_{j+1}])$. Then, M is LCA-preserving.

By the same reason, the mapping obtained from the third recurrence in Figure 3 is LCA-preserving.

Hence, the mapping obtained from the recurrences in Figure 3 is LCA-preserving.

By traversing C and D at once in $O(n)$ time, we can obtain the information of v_i, w_i, L_i and K_i. Then, in computing $\tau_{\mathrm{LCA}}(C[v_i], D[w_j])$ for a fixed i and j, the running time is $O(1)$. Since the recurrences compute $\tau_{\mathrm{LCA}}(C[v_i], D[w_j])$ for $1 \le i \le n$ and $1 \le j \le m$, the total running time is $O(n^2)$. □

3.3 Bottom-up Distance

When considering the algorithm of computing the bottom-up distance $\tau_{\text{BOT}}(C,D)$ between caterpillars C and D, we deal with the reversal of backbones, that is, $bb(C) = [v_1,\ldots,v_n]$ for and $bb(D) = [w_1,\ldots,w_m]$, where $r(C) = v_n$, $(v_i,v_{i+1}) \in E(C)$, $r(D) = w_m$ and $(w_j,w_{j+1}) \in E(D)$. Then, we design the algorithm BOTCAT in Algorithm 1.

```
   procedure BOTCAT(C,D)
       /* C,D: caterpillars */
       /* bb(C) = [v_1,...,v_n], r(C) = v_n */
       /* bb(D) = [w_1,...,w_m], r(D) = w_m */
1      for h = 1 to min{n,m} do
2          if |L_h| ≠ |K_h| then break;
3      h--; /* |L_i| = |K_i| for 1 ≤ i ≤ h */
4      A ← bb(C); B ← bb(D); L ← lv(C); K ← lv(D);
5      d ← μ(L̃,K̃) + del(A) + ins(B);
6      if h > 0 then
7          d_0 ← d; L'_0 ← L; K'_0 ← K;
8          for i = 1 to h do
9              L'_i ← L'_{i-1} \ L_i; K'_i ← K'_{i-1} \ K_i;
10             d_i ←
               d_{i-1} − γ(v_i,ε) − γ(ε,w_i) + γ(v_i,w_i) −
               μ(L'_{i-1}~, M'_{i-1}~) + μ(L̃_i,K̃_i) + μ(L̃'_i,K̃'_i);
11             d ← min{d,d_i};
12     return d;
```

Algorithm 1: BOTCAT.

Theorem 6. *Let C and D be caterpillars, where $n = \max\{|C|,|D|\}$. Then, we can compute $\tau_{\text{BOT}}(C,D)$ in $O(n)$ time under the unit cost function.*

Proof. Since C and D are caterpillars, if $|L_i| = |K_i|$ for $1 \le i \le h$ but $|L_{h+1}| \ne |K_{h+1}|$, then it holds that $C[v_h] \equiv_l D[w_h]$ but $C[v_{h+1}] \not\equiv_l D[w_{h+1}]$. The algorithm BOTCAT first finds such an h in lines 1, 2 and 3. In this case, we can obtain bottom-up mapping $M \in \mathcal{M}_{\text{BOT}}(C,D)$ between $C[w_h]$ and $D[w_h]$ by (1) adding (v_i,w_i) to M, (2) adding (v,w) to M for $v \in L_i$, $w \in K_i$ and $l(v) = l(w)$ and (3) adding (v,w) to M for the remained $v \in L_i$ and $w \in K_i$ for $1 \le i \le h$. We can compute the distance concerned with the above (2) and (3) as $\mu(\widetilde{L}_i,\widetilde{K}_i)$. Note that the remained vertices in C are deleted and those in D are inserted.

After obtaining the above h, the algorithm BOTCAT computes the bottom-up distance whose bottom-up mapping $M \in \mathcal{M}_{\text{BOT}}(C,D)$ contain no pair in $bb(C) \times bb(D)$ as d in line 6. Then, in for-loop in lines from 7 to 12, the algorithm BOTCAT updates d as the minimum value of the current d and the newly obtained d_i such that $(v_i,w_i) \in M$. Here, d_i is the distance that $v_i \in bb(C)$ is corresponding to $w_i \in bb(D)$, by adding $\gamma(v_i,w_i)$ instead of $\gamma(v_i,\varepsilon) + \gamma(\varepsilon,w_i)$, and L_i are corresponding to K_i, by adding $\mu(\widetilde{L}_i,\widetilde{K}_i)$ instead of $\mu(\widetilde{L'_{i-1}},\widetilde{K'_{i-1}})$. This is realized at line 11, that is, by using the following formula.

$$\begin{aligned} d_i \leftarrow\ & d_{i-1} - \gamma(v_i,\varepsilon) - \gamma(\varepsilon,w_i) + \gamma(v_i,w_i) \\ & - \mu(\widetilde{L'_{i-1}},\widetilde{M'_{i-1}}) + \mu(\widetilde{L}_i,\widetilde{K}_i) + \mu(\widetilde{L}'_i,\widetilde{K}'_i). \end{aligned}$$

In other words, for $1 \le i \le h$, the bottom-up mapping $M \in \mathcal{M}_{\text{BOT}}(C,D)$ is updated by adding (v_i,w_i) and the correspondence between L_i and K_i to M for every i, after removing the correspondences between $L_1 \cup \cdots \cup L_i$ and $K_1 \cup \cdots \cup K_i$ in M. Hence, the algorithm BOTCAT is correct.

By traversing C and D at once in $O(n)$ time, we can obtain the information of $bb(C)$, $bb(D)$, $lv(C)$ and $lv(D)$ (so v_i, w_i, L_i and K_i). Then, each of lines 2, 4, 5, 7 and 9 to 11 runs in $O(1)$ time. Hence, the total running time of the algorithm BOTCAT is $O(n)$. □

4 EXPERIMENTAL RESULTS

In this section, we give the experimental results of computing τ_{TOP}, τ_{LCA} and τ_{BOT}. Here, the computer environment is that OS is Ubuntu 14.04.6, CPU is Intel Xeon E5-1650 v3(3.50GHz) and RAM is 15GB.

We deal with caterpillars for N-glycans from KEGG[2], the largest 5,154 caterpillars (0.1%) in dblp[3] (refer to $\text{dblp}_{0.1\%}$), SwissProt and non-isomorphic caterpillars in TPC-H (refer to TPC-$\text{H}_\circ$) from UW XML Repository[4]. Also we deal with caterpillars obtained by deleting the root in Auction (refer to Auction$^-$) and non-isomorphic caterpillars obtained by deleting the root in Nasa (refer to $\text{NASA}_\circ^-$), Protein (refer to $\text{Protein}_\circ^-$) and University (refer to $\text{University}_\circ^-$) from UW XML Repository. Table 2 illustrates the information of such caterpillars. Here, #, n, d, h, λ and β are the number of caterpillars, the average number of vertices, the average degree, the average height, the average number of leaves and the average number of labels.

Then, we use all the pairs in the caterpillars in Table 2, of which the number is $\frac{\# \times (\# - 1)}{2}$. Table 3 illustrates the number (#pairs) of all the pairs in caterpillars in Table 2.

Table 4 illustrates the running time to compute τ_{TOP}, τ_{LCA} and τ_{BOT}, as comparing with τ_{TAI} by the algorithm in (Muraka et al., 2018).

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

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

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

Table 2: The information of caterpillars.

data	#	n	d	h	λ	β
N-glycans	514	6.40	1.84	4.22	2.18	4.50
$\text{dblp}_{0.1\%}$	5,154	41.74	40.73	1.01	40.73	10.61
SwissProt	6,804	35.10	24.96	2.00	33.10	16.79
TPC-$\text{H}_\circ$	8	8.63	7.63	1.00	7.63	8.63
Auction^-	259	4.29	3.00	0.71	3.57	4.29
$\text{Nasa}_\circ^-$	33	7.27	5.15	1.64	5.64	3.18
$\text{Protein}_\circ^-$	5,150	4,97	3.63	1.16	3.81	4.57
$\text{University}_\circ^-$	26	1.35	0.35	0.19	1.15	1.35

Table 3: The number (#pairs) of all the pairs in caterpillars in Table 2.

data	#pairs	data	#pairs
N-glycans	131,841	Auction^-	33,411
$\text{dblp}_{0.1\%}$	13,279,281	$\text{Nasa}_\circ^-$	528
SwissProt	23,143,806	$\text{Protein}_\circ^-$	13,258,675
TPC-$\text{H}_\circ$	28	$\text{University}_\circ^-$	325

Table 4: The running time (sec.) to compute τ_{TAI}, τ_{TOP}, τ_{LCA} and τ_{BOT}.

data	τ_{TAI}	τ_{TOP}	τ_{LCA}	τ_{BOT}
N-glycans	753.33	1.23	2,804.82	2.57
$\text{dblp}_{0.1\%}$	7,525.28	343.70	1,505.05	737.96
SwissProt	82,031.10	1,594.42	9,819.62	2,138.54
TPC-$\text{H}_\circ$	$5.78_{\times 10^{-3}}$	$0.64_{\times 10^{-3}}$	$1.77_{\times 10^{-3}}$	$1.43_{\times 10^{-3}}$
Auction^-	4.55	0.23	0.87	0.94
$\text{Nasa}_\circ^-$	$20.93_{\times 10^{-2}}$	$0.34_{\times 10^{-2}}$	$4.91_{\times 10^{-2}}$	$0.57_{\times 10^{-2}}$
$\text{Protein}_\circ^-$	2,055.77	118.20	433.22	327.66
$\text{University}_\circ^-$	$14.22_{\times 10^{-3}}$	$0.40_{\times 10^{-3}}$	$2.84_{\times 10^{-3}}$	$6.58_{\times 10^{-3}}$

Table 4 shows that, whereas the time complexity of computing τ_{TOP} is same as that of computing τ_{BOT}, the running time of computing τ_{TOP} is slightly smaller than that of computing τ_{BOT}. On the other hand, the running time of computing τ_{LCA} is smaller than that of computing τ_{TAI} except N-glycan. The reason is that the depth of caterpillars in N-glycan is much larger than other caterpillars.

Table 5 illustrates the number (#cases) of cases that $\tau_{\text{TAI}} < \tau_{\text{TOP}}$, $\tau_{\text{TAI}} < \tau_{\text{LCA}}$ and $\tau_{\text{TAI}} < \tau_{\text{BOT}}$ with their ratios (%) in all the pairs (#pairs), where "max." is the maximum difference from τ_{TAI}. Since it always holds $\tau_{\text{LCA}} \leq \tau_{\text{TOP}}$, we omit the cases that $\tau_{\text{TAI}} < \tau_{\text{LCA}}$ in Table 5 when the number of cases that $\tau_{\text{TAI}} < \tau_{\text{TOP}}$ is 0.

Table 5 show that, for caterpillars in $\text{dblp}_{0.1\%}$, TPC-$\text{H}_\circ$, Auction^- and $\text{University}_\circ^-$, τ_{TOP} is an alternative and much faster distance to τ_{TAI}. Also, whereas τ_{LCA} is an improved distance of τ_{TOP} for caterpillars in N-glycans, τ_{LCA} and τ_{TOP} are not changed for the other caterpillars. Furthermore, τ_{BOT} is insufficient to approximate to τ_{TAI} since the number of cases that $\tau_{\text{TAI}} < \tau_{\text{BOT}}$ are much larger than the number of cases that $\tau_{\text{TAI}} < \tau_{\text{TOP}}$.

Table 5: The number (#cases) of cases that $\tau_{\text{TAI}} < \tau_{\text{TOP}}$, $\tau_{\text{TAI}} < \tau_{\text{LCA}}$ and $\tau_{\text{TAI}} < \tau_{\text{BOT}}$ with their ratios (%) in all the pairs (#pairs) with the maximum difference (max.)

$\tau_{\text{TAI}} < \tau_{\text{TOP}}$

data	#pairs	#cases	%	max.
N-glycans	131,841	64,467	48.90	10
$\text{dblp}_{0.1\%}$	13,279,281	0	0.00	0
SwissProt	23,143,806	5,933,179	25.64	30
TPC-$\text{H}_\circ$	28	0	0	0
Auction^-	33,411	0	0	0
$\text{Nasa}_\circ^-$	528	104	19.70	9
$\text{Protein}_\circ^-$	13,258,675	697,697	5.26	50
$\text{University}_\circ^-$	325	0	0	0

$\tau_{\text{TAI}} < \tau_{\text{LCA}}$

data	#pairs	#cases	%	max.
N-glycans	131,841	5,490	4.16	2
SwissProt	23,143,806	5,933,179	25.64	29
$\text{Nasa}_\circ^-$	528	56	10.61	1
$\text{Protein}_\circ^-$	132,586,75	348,119	2.63	10

$\tau_{\text{TAI}} < \tau_{\text{BOT}}$

data	#pairs	#cases	%	max.
N-glycans	131,841	117,657	89.24	16
$\text{dblp}_{0.1\%}$	13,279,281	12,667,501	95.39	4
SwissProt	23,143,806	23,019,607	99.46	4
TPC-$\text{H}_\circ$	28	27	96.43	2
Auction^-	33,411	4,107	12.29	1
$\text{Nasa}_\circ^-$	528	403	76.33	4
$\text{Protein}_\circ^-$	13,258,675	8,828,524	66.59	5
$\text{University}_\circ^-$	325	5	1.54	1

On the other hand, by focusing on the maximum difference, for caterpillars in SwissProt and $\text{Protein}_\circ^-$, the maximum difference of $\tau_{\text{BOT}} - \tau_{\text{TAI}}$ is much smaller than that of $\tau_{\text{TOP}} - \tau_{\text{TAI}}$ and $\tau_{\text{LCA}} - \tau_{\text{TAI}}$. Then, for these caterpillars, whereas the number of cases that $\tau_{\text{TAI}} < \tau_{\text{BOT}}$ is larger than the number of cases that $\tau_{\text{TAI}} < \tau_{\text{TOP}}$ and $\tau_{\text{TAI}} < \tau_{\text{LCA}}$, τ_{BOT} is more appropriate to characterize the forms of caterpillars than τ_{TOP} and τ_{LCA}.

In order to improve the results in Table 5, Table 6 summarizes the case that $\min\{\tau_{\text{TOP}}, \tau_{\text{BOT}}\}$.

By comparing with Table 5, Table 6 shows that the usage of $\min\{\tau_{\text{TOP}}, \tau_{\text{BOT}}\}$ succeeds to decrease the maximum difference with slightly decreasing the ratio. Hence, $\min\{\tau_{\text{TOP}}, \tau_{\text{BOT}}\}$ provides to fast approximate to τ_{TAI} for caterpillars.

Finally, we compare the algorithms in this paper

Table 6: The number (#cases) of cases that $\tau_{\mathrm{TAI}} < \min\{\tau_{\mathrm{TOP}}, \tau_{\mathrm{BOT}}\}$ with their ratios (%) in all the pairs (#pairs) with the maximum difference (max.).

data	#pairs	#cases	%	max.
N-glycans	131,841	59,921	45.45	9
$\mathrm{dblp}_{0.1\%}$	13,279,281	0	0.00	0
SwissProt	23,143,806	5,933,179	25.64	2
TPC-$\mathrm{H}_{\circ}$	28	0	0	0
$\mathrm{Auction}^{-}$	33,411	0	0	0
$\mathrm{Nasa}^{-}_{\circ}$	528	94	17.80	1
$\mathrm{Protein}^{-}_{\circ}$	13,258,675	637,773	4.81	2
$\mathrm{University}^{-}_{\circ}$	325	0	0	0

for caterpillars with the algorithms designed by (Yamamoto et al., 2014) for standard trees. Table 7 illustrates the running time of computing τ_{TOP} and τ_{LCA} by using such algorithms which refer to τ^{T}_{TOP} and τ^{T}_{LCA}. Here, "–" denotes time out over 10,000 seconds.

Table 7: The running time (sec.) of computing τ_{TOP} and τ_{LCA} by using the algorithms in this paper and the algorithms τ^{T}_{TOP} and τ^{T}_{LCA} in (Yamamoto et al., 2014).

data	τ_{TOP}	τ_{LCA}	τ^{T}_{TOP}	τ^{T}_{LCA}
N-glycans	1.23	2,804.82	11.77	25.64
$\mathrm{dblp}_{0.1\%}$	343.70	1,505.05	–	–
SwissProt	1,594.42	9,819.62	–	–
TPC-$\mathrm{H}_{\circ}$	$0.64_{\times 10^{-3}}$	$1.77_{\times 10^{-3}}$	$3.77_{\times 10^{-3}}$	$7.45_{\times 10^{-3}}$
$\mathrm{Auction}^{-}$	0.23	0.87	1.20	2.12
$\mathrm{Nasa}^{-}_{\circ}$	$0.34_{\times 10^{-2}}$	$4.91_{\times 10^{-2}}$	$5.64_{\times 10^{-2}}$	$10.68_{\times 10^{-2}}$
$\mathrm{Protein}^{-}_{\circ}$	118.20	433.22	628.79	1156.32
$\mathrm{University}^{-}_{\circ}$	$0.40_{\times 10^{-3}}$	$2.84_{\times 10^{-3}}$	$2.93_{\times 10^{-3}}$	$2.19_{\times 10^{-3}}$

Table 7 shows that the algorithm of computing τ_{TOP} in this paper is much faster than τ^{T}_{TOP}. Also, except N-glycans and $\mathrm{University}^{-}_{\circ}$, the algorithm of computing τ_{LCA} in this paper is faster than τ^{T}_{LCA}.

5 CONCLUSION

In this paper, we have designed the algorithms of computing τ_{TOP} and τ_{BOT} for caterpillars in $O(n)$ time and τ_{LCA} in $O(n^2)$ time. Also, we have given experimental results of computing τ_{TOP}, τ_{LCA} and τ_{BOT} for caterpillars in real data. Then, the usage of $\min\{\tau_{\mathrm{TOP}}, \tau_{\mathrm{BOT}}\}$ have provided to fast approximate to τ_{TAI} for caterpillars. Also, the algorithms in this paper have been almost fast and faster than the previous algorithms for trees (Yamamoto et al., 2014).

Since the algorithm of computing τ_{LCA} for caterpillars is slow for N-glycan, it is a future work to improve the implementation, in particular, to apply to larger number of caterpillars such as all-glycans in KEGG and CSLOGS[5]. Also it is a future work to investigate the other variations of the edit distance for caterpillars presented in (Yoshino and Hirata, 2017).

REFERENCES

Akutsu, T., Fukagawa, D., Halldórsson, M. M., Takasu, A., and Tanaka, K. (2013). Approximation and parameterized algorithms for common subtrees and edit distance between unordered trees. *Theoret. Comput. Sci.*, 470:10–22.

Chawathe, S. S. (1999). Comparing hierarchical data in external memory. In *Proc. VLDB'99*, pages 90–101.

Deza, M. M. and Deza, E. (2016). *Encyclopedia of distances (4th ed.)*. Springer.

Gallian, J. A. (2007). A dynamic survey of graph labeling. *Electorn. J. Combin.*, 14:DS6.

Hirata, K., Yamamoto, Y., and Kuboyama, T. (2011). Improved MAX SNP-hard results for finding an edit distance between unordered trees. In *Proc. CPM'11 (LNCS 6661)*, pages 402–415.

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

Muraka, K., Yoshino, T., and Hirata, K. (2018). Computing edit distance between rooted labeled caterpillars. In *Proc. FedCSIS'18*, pages 245–252.

Selkow, S. M. (1977). The tree-to-tree editing problem. *Inform. Process. Lett.*, 6:184–186.

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

Ukita, Y., Yoshino, T., and Hirata, K. (2021). Caterpillar alignment distance for rooted labeled caterpillars: Distance based on alignments required to be caterpillars. In *Recent advance in computational optimization*, pages 111–134.

Valiente, G. (2001). An efficient bottom-up distance between trees. In *Proc. SPIRE'01*, pages 212–219.

Yamamoto, Y., Hirata, K., and Kuboyama, T. (2014). Tractable and intractable variations of unordered tree edit distance. *Internat. J. Found. Comput. Sci.*, 25:307–329.

Yoshino, T. and Hirata, K. (2017). Tai mapping hierarchy for rooted labeled trees through common subforest. *Theory of Comput. Sys.*, 60:769–787.

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

Zhang, K., Wang, J., and Shasha, D. (1996). On the editing distance between undirected acyclic graphs. *Internat. J. Found. Comput. Sci.*, 7:43–58.

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