Heavy Caterpillar Distances for Rooted Labeled Unordered Trees

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

Keywords: Heavy Caterpillar, Heavy Caterpillar Distance, Rooted Labeled Unordered Tree, Tree Edit Distance, Variations of Tree Edit Distance.

Abstract: In this paper, we introduce two *heavy caterpillar distances* between rooted labeled unordered trees (*trees*, for short) based on the edit distance between the *heavy caterpillars* obtained from the heavy paths in trees. Then, we show that the heavy caterpillar distances provide the upper bound of the edit distance for trees, can be computed in quadratic time under the unit cost function and are incomparable with other variations of the edit distance.

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* $\tau_{\rm TAI}$ (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). This statement also holds even if trees are binary or the maximum height of trees is at most 3 (Akutsu et al., 2013; Hirata et al., 2011).

Many variations of the edit distance have developed as more structurally sensitive distances as the minimum cost of the variations of the Tai mapping (Jiang et al., 1995; Kan et al., 2014; Kuboyama, 2007; Lu et al., 2001; Wang and Zhang, 2001; Yamamoto et al., 2014; Yoshino and Hirata, 2017; Zhang, 1996). In particular, the *alignment distance* $\tau_{\rm ALN}$ (Jiang et al., 1995) and the *segmental distance* $\tau_{\rm SG}$ (Kan et al., 2014) are the most general variations of $\tau_{\rm TAI}$, where $\tau_{\rm ALN}$ is incomparable with $\tau_{\rm SG}$, and the *isolated-subtree distance* $\tau_{\rm ILST}$ (Wang and Zhang, 2001) (or *constrained distance*) (Zhang, 1996) is the most general tractable variation of $\tau_{\rm TAI}$ (Yoshino and Hirata, 2017).

A *caterpillar* (*cf.* (Gallian, 2007)) is a tree transformed to a path after removing all the leaves in it. Recently, Muraka *et al.* (Muraka et al., 2018) have shown that the problem of computing the edit distance between caterpillars is tractable and the structural restriction of caterpillars provides the limitation of the tractability for computing the edit distance. Also Muraka *et al.* (Muraka et al., 2019) have developed the method to fast approximate the edit distance between caterpillars.

Hence, in this paper, we introduce new distances for trees by using the edit distance between the embedded caterpillars. Then, we focus on the *heavy path* (Sleator and Tarjan, 1983), which is a famous embedded path in a tree obtained by selecting vertices whose number of descendants is largest from the root. In particular, Demaine *et al.* (Demaine et al., 2009) have adopted the heavy path to analyze the time complexity of computing the edit distance for rooted labeled *ordered* trees.

In this paper, first we formulate a *heavy caterpillar* in a tree as the caterpillar whose backbone is the heavy path in the tree and whose set of leaves consists of all the adjacent vertices to the heavy path in the tree. Then, we introduce the following two *heavy caterpillar distances* $\tau_{\rm HC}$ and $\tau_{\widehat{\rm HC}}$ between trees.

The heavy caterpillar distance $\tau_{\rm HC}$ is formulated as the sum of the edit distance between heavy caterpillars and the cost of deleting and inserting the remained vertices not contained in the heavy caterpillars. On the other hand, the heavy caterpillar distance $\tau_{\widehat{\rm HC}}$ is formulated as the sum of the edit distance between heavy caterpillars and the cost of the Tai map-

Abe, N., Yoshino, T. and Hirata, K.
Heavy Caterpillar Distances for Rooted Labeled Unordered Trees.
DOI: 10.5220/0009095801980204
In *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2020)*, pages 198-204
ISBN: 978-989-758-397-1; ISSN: 2184-4313

ping obtained by repeating recursively, after selecting vertices (as leaves in heavy caterpillars) to bridge the Tai mapping between the heavy caterpillars, to compute the edit distance (the Tai mapping) between the heavy caterpillars of the complete subtree rooted by the selected vertices.

Then, in this paper, we show that the heavy caterpillar distances τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$ provide the upper bound of τ_{TAI}, that is, $\tau_{\mathrm{TAI}} \leq \tau_{\widehat{\mathrm{HC}}} \leq \tau_{\mathrm{HC}}$. For the maximum height h and the maximum number λ of leaves in given two trees, we can compute τ_{HC} in $O(h^2\lambda^3)$ time under the general cost function and in $O(h^2\lambda)$ time under the unit cost function, and $\tau_{\widehat{\mathrm{HC}}}(T_1,T_2)$ in $O(h^2\lambda^4)$ time under the general cost function and in $O(h^2\lambda^2)$ time under the unit cost function. Furthermore, we show that τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$ are incomparable with τ_{ILST}, τ_{ALN} and τ_{SG}. Hence, the heavy caterpillar distances τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$ provide another tractable variations of the edit distance τ_{TAI} incomparable with the isolated-subtree distance τ_{ILST}.

2 PRELIMINARIES

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 (V',E') such that $V' = \{v_1,\dots,v_k\}$, $v_1 = r$, $v_k = v$ 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 denote the set of all ancestors of v by $anc(v)$. 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 $ch(v)$ and that v is a ancestor of u by $u \leq v$. We call a node with no children a *leaf* and denote the set of all the leaves in T by $lv(T)$.

A *rooted path* P is a rooted tree $(\{v_1,\dots,v_n\},\{(v_i,v_{i+1}) \mid 1 \leq i \leq n-1\})$ such that $r(P) = v_1$. We call the node v_n (the leaf of P) an *endpoint* of P and denote it by $e(P)$.

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*. In particular, we denote the forest obtained by deleting v in $T[v]$ by $T(v)$.

Definition 1 (Caterpillar (*cf.*, (Gallian, 2007)))**.** 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 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 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 $\varepsilon \notin \Sigma$ 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

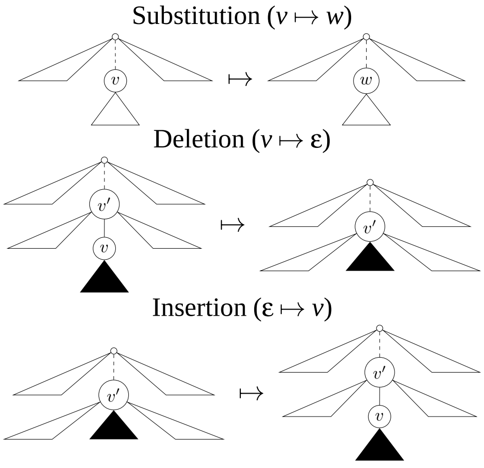

Figure 1: Edit operations for trees.

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 $E = e_1,\dots,e_k$ 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.

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

Theorem 1 ((Tai, 1979))**.** $\tau_{\mathrm{TAI}}(T_1,T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\mathrm{TAI}}(T_1,T_2)\}$.

Furthermore, we introduce the variations of Tai mappings. Whereas the alignment distance (Jiang et al., 1995) has first defined by using an alignment tree between two trees as the common supertree, it is known that the alignment distance coincides with the minimum cost of less-constrained mappings (Kuboyama, 2007). Hence, in this paper, we regard the less-constrained mapping as an alignable mapping and formulate the alignment distance as the minimum cost of alignable mappings.

Definition 5 (Variations of Tai mapping)**.** Let T_1 and T_2 be trees and $M \in \mathcal{M}_{\mathrm{TAI}}(T_1,T_2)$.

1. We say that M is an *alignable mapping* (Kuboyama, 2007) (or an *less-constrained mapping* (Lu et al., 2001)), denoted by $M \in \mathcal{M}_{\mathrm{ALN}}(T_1,T_2)$, if M satisfies the following condition:
$$\forall (v_1,w_1)(v_2,w_2)(v_3,w_3) \in M \\ \Big((v_1 \sqcup v_2 < v_1 \sqcup v_3) \Longrightarrow (w_2 \sqcup w_3 = w_1 \sqcup w_3)\Big).$$
Also we define an *alignment distance* $\tau_{\mathrm{ALN}}(T_1,T_2)$ (Jiang et al., 1995) as the minimum cost of all the alignable mappings, that is:
$$\tau_{\mathrm{ALN}}(T_1,T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\mathrm{ALN}}(T_1,T_2)\}.$$
2. We say that M is an *isolated-subtree mapping* (Wang and Zhang, 2001) (or a *constrained mapping* (Zhang, 1996)), denoted by $M \in \mathcal{M}_{\mathrm{ILST}}(T_1,T_2)$, if M satisfies the following condition:
$$\forall (v_1,w_1)(v_2,w_2)(v_3,w_3) \in M \\ \Big((v_3 < v_1 \sqcup v_2) \iff (w_3 < w_1 \sqcup w_2)\Big).$$
Also we define an *isolated-subtree distance* $\tau_{\mathrm{ILST}}(T_1,T_2)$ as the minimum cost of all the isolated-subtree mappings, that is:
$$\tau_{\mathrm{ILST}}(T_1,T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\mathrm{ILST}}(T_1,T_2)\}.$$
3. We say that M is a *segmental mapping* (Kan et al., 2014), denoted by $M \in \mathcal{M}_{\mathrm{SG}}(T_1,T_2)$, if M satisfies the following condition.
$$\forall (v,w) \in M^{-} \\ \left(\begin{array}{c} \exists (v',w') \in M\Big(\big(v' \in anc(v)\big) \wedge \big(w' \in anc(w)\big)\Big) \\ \Longrightarrow \big((par(v),par(w)) \in M\big) \end{array}\right).$$
Also we define a *segmental distance* $\tau_{\mathrm{SG}}(T_1,T_2)$ as the minimum cost of all the segmental mappings, that is:
$$\tau_{\mathrm{SG}}(T_1,T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\mathrm{SG}}(T_1,T_2)\}.$$

Furthermore, for distances τ_{A} and τ_{B}, we say that τ_{A} is *incomparable with* τ_{B} if there exist trees T_1, T_2, T_3 and T_4 such that $\tau_{\mathrm{A}}(T_1,T_2) < \tau_{\mathrm{B}}(T_1,T_2)$ and $\tau_{\mathrm{B}}(T_3,T_4) < \tau_{\mathrm{A}}(T_3,T_4)$.

Theorem 2 ((Kuboyama, 2007; Yoshino and Hirata, 2017))**.** *Let T_1 and T_2 be trees. Then, it holds that $\mathcal{M}_{\text{ILST}}(T_1,T_2) \subseteq \mathcal{M}_{\text{ALN}}(T_1,T_2) \subseteq \mathcal{M}_{\text{TAI}}(T_1,T_2)$ and $\mathcal{M}_{\text{SG}}(T_1,T_2) \subseteq \mathcal{M}_{\text{TAI}}(T_1,T_2)$. On the other hand, $\mathcal{M}_{\text{ILST}}(T_1,T_2)$ or $\mathcal{M}_{\text{ALN}}(T_1,T_2)$ is incomparable with $\mathcal{M}_{\text{SG}}(T_1,T_2)$ with respect to set inclusion.*

Theorem 2 implies $\tau_{\text{TAI}}(T_1,T_2) \leq \tau_{\text{ALN}}(T_1,T_2) \leq \tau_{\text{ILST}}(T_1,T_2)$ and $\tau_{\text{TAI}}(T_1,T_2) \leq \tau_{\text{SG}}(T_1,T_2)$ for every tree T_1 and T_2. On the other hand, τ_{ILST} or τ_{ALN} is incomparable with τ_{SG}. Furthermore, the following theorem is known for the problem of computing τ_{TAI} and its variations.

Theorem 3. *Let T_1 and T_2 be trees such that $n = \max\{|T_1|,|T_2|\}$ and $d = \min\{d(T_1),d(T_2)\}$.*

1. *The problem of computing $\tau_{\text{TAI}}(T_1,T_2)$ is* MAX SNP*-hard (Zhang and Jiang, 1994). 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 (Akutsu et al., 2013; Hirata et al., 2011).*
2. *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 (Jiang et al., 1995).*
3. *We can compute $\tau_{\text{ILST}}(T_1,T_2)$ in $O(n^2d)$ time (cf., (Yamamoto et al., 2014)).*
4. *The problem of computing $\tau_{\text{SG}}(T_1,T_2)$ is* MAX SNP*-hard. This statement holds even if both T_1 and T_2 are binary or the cost function is the unit cost function (Yamamoto et al., 2014).*

In contrast to Theorem 3, Muraka *et al.* (Muraka et al., 2018) have recently shown the following theorem of the edit distance for caterpillars.

Theorem 4 ((Muraka et al., 2018))**.** *Let C_1 and C_2 be caterpillars, $h = \max\{h(C_1),h(C_2)\}$ and $\lambda = \max\{|lv(C_1)|,|lv(C_2)|\}$. Then, 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.*

3 HEAVY CATERPILLAR DISTANCES

In this section, we introduce the *heavy caterpillar* in a tree, based on the *heavy path* (Sleator and Tarjan, 1983). Then, we formulate another variation of the edit distance as *heavy caterpillar distances* based on the edit distance for heavy caterpillars.

Definition 6 (Heavy path (Sleator and Tarjan, 1983))**.** Let T be a tree. For $v \in T$ and $w \in ch(v)$, w is a *heavy child* of v if $|T[w]|$ is maximum and denote it by $hv(v)$. A *heavy path* of T is the rooted path $(\{v_1,\dots,v_n\},\{(v_i,v_{i+1}) \mid 1 \leq i \leq n-1\})$ such that $v_1 = r(T)$, $v_{i+1} = hv(v_i)$ $(1 \leq i \leq n-1)$ and $v_n \in lv(T)$.

If there exist more than two heavy children of v, then we may name one of them arbitrary a heavy child of v. Then, based on the heavy path in a tree, we introduce the heavy caterpillar in a tree as follows.

Definition 7 (Heavy caterpillar)**.** Let T be a tree and P the heavy path of T. Then, we define the *heavy caterpillar* $hc(T) = (V,E)$ of T as follows.

$$
\begin{aligned}
V &= V(P) \cup \{w \in ch(v) \mid v \in V(P)\},\\
E &= E(P) \cup \{(v,w) \mid v \in V(P), w \in ch(v)\}.
\end{aligned}
$$

We denote the minimum cost Tai mapping between $C_1 = hc(T_1)$ and $C_2 = hc(T_2)$ by $M_{hc}(C_1,C_2)$. Then, the algorithm HvyCatMap in Algorithm 1 returns a Tai mapping based on the heavy caterpillars C_1 and C_2. We define the *heavy caterpillar mapping* between T_1 and T_2 as the mapping obtained from the algorithm HvyCatMap(T_1,T_2) and denote it by $M_{hc}(T_1,T_2)$.

```
procedure HvyCatMap(T1,T2)
    /* T1,T2 : trees */
    C1 ← hc(T1); C2 ← hc(T2); L1 ← lv(C1);
      L2 ← lv(C2); M ← Mhc(C1,C2);
    L ← {(v,w) ∈ M | v ∈ L1, w ∈ L2, T1(v) ≠
      ∅, T2(w) ≠ ∅};
    foreach (v,w) ∈ L do
        M1 ← HvyCatMap(T1[v],T2[w]);
          M ← M ∪ M1;
    return M;
```

Algorithm 1: HvyCatMap.

Definition 8 (Heavy caterpillar distances)**.** Let T_i be a tree, $C_i = hc(T_i)$ and $D_i = T_i \setminus C_i$ $(i = 1,2)$. Then, we define the *heavy caterpillar distances* $\tau_{\text{HC}}(T_1,T_2)$ and $\tau_{\widehat{\text{HC}}}(T_1,T_2)$ as follows.

$$
\begin{aligned}
&\tau_{\text{HC}}(T_1,T_2)\\
&\quad = \tau_{\text{TAI}}(C_1,C_2) + \sum_{v \in D_1} \gamma(v,\varepsilon) + \sum_{w \in D_2} \gamma(\varepsilon,w),\\
&\tau_{\widehat{\text{HC}}}(T_1,T_2) = \gamma(M_{hc}(T_1,T_2)).
\end{aligned}
$$

Theorem 5. *For trees T_1 and T_2, it holds that $\tau_{\text{TAI}}(T_1,T_2) \leq \tau_{\widehat{\text{HC}}}(T_1,T_2) \leq \tau_{\text{HC}}(T_1,T_2)$.*

Proof. For $C_i = hc(T_i)$ and $M' = M_{hc}(T_1,T_2) \setminus M_{hc}(C_1,C_2)$, since $\tau_{\text{TAI}}(C_1,C_2) = \gamma(M_{hc}(C_1,C_2))$, it holds that $\tau_{\widehat{\text{HC}}}(T_1,T_2) = \tau_{\text{TAI}}(C_1,C_2) + \gamma(M')$. If $M' = \emptyset$, then it holds that $\gamma(M') = \sum_{v \in D_1} \gamma(v,\varepsilon) + \sum_{w \in D_2} \gamma(\varepsilon,w)$, which implies that $\tau_{\widehat{\text{HC}}}(T_1,T_2) \leq \tau_{\text{HC}}(T_1,T_2)$.

In order to show that $\tau_{\mathrm{TAI}}(T_1,T_2) \leq \tau_{\widehat{\mathrm{HC}}}(T_1,T_2)$, it is sufficient to show that the heavy caterpillar mapping $M_{hc}(T_1,T_2)$ is a Tai mapping. If it is true, then it holds that $\tau_{\mathrm{TAI}}(T_1,T_2) \leq \gamma(M_{hc}(T_1,T_2))$.

Let $L^* = \{(v_1,w_1),\dots(v_k,w_k)\}$ be the union of all the L selected at line 2 in HvyCatMap in Algorithm 1 recursively, $v_0 = r(T_1)$ and $w_0 = r(T_2)$. Also let M_i be the output of HvyCatMap$(T_1[v_i],T_2[w_i])$ $(0 \leq i \leq k)$ and $M = M_0 \cup M_1 \cup \cdots \cup M_k$, where $M_0 = M_{hc}(C_1,C_2) \in \mathcal{M}_{\mathrm{TAI}}(T_1,T_2)$. Note that $M_i \in \mathcal{M}_{\mathrm{TAI}}(T_1[v_i],T_2[w_i])$, so $M_i \in \mathcal{M}_{\mathrm{TAI}}(T_1,T_2)$.

Since M_i is mutually distinct for every i and $M_i \in \mathcal{M}_{\mathrm{TAI}}(T_1[v_i],T_2[w_i])$, M satisfies the one-to-one condition. By the construction of L, $(M \setminus M_i) \cup \{(v_i,w_i)\}$ satisfies the ancestor condition for every $(v_i,w_i) \in L^*$, which implies that M satisfies the ancestor condition. Hence, it holds that $M \in \mathcal{M}_{\mathrm{TAI}}(T_1,T_2)$. Since $M = M_{hc}(T_1,T_2)$, it holds that $M_{hc}(T_1,T_2) \in \mathcal{M}_{\mathrm{TAI}}(T_1,T_2)$. ☐

Theorem 6. *Let T_1 and T_2 be trees, where $h = \max\{h(T_1),h(T_2)\}$ and $\lambda = \max\{|lv(T_1)|,|lv(T_2)|\}$. Then, we can compute $\tau_{\mathrm{HC}}(T_1,T_2)$ in $O(h^2\lambda^3)$ time under the general cost function and in $O(h^2\lambda)$ time under the unit cost function. Also we can compute $\tau_{\widehat{\mathrm{HC}}}(T_1,T_2)$ in $O(h^2\lambda^4)$ time under the general cost function and in $O(h^2\lambda^2)$ time under the unit cost function.*

Proof. Let $C_i = hc(T_i)$ $(i = 1,2)$. First, we can obtain C_i in $O(|T_i|) = O(h\lambda)$ time (Sleator and Tarjan, 1983). Since it is essential for computing $\tau_{\mathrm{HC}}(T_1,T_2)$ to compute $\tau_{\mathrm{TAI}}(C_1,C_2)$, the time complexity of computing τ_{HC} follows from Theorem 4.

Next, consider the number of recursive calls in HvyCatMap in Algorithm 1. For L^* in the proof of Theorem 5, we denote $L_1^* = \{v \in V(T_1) \mid (v,w) \in L^*\}$ and $L_2^* = \{w \in V(T_2) \mid (v,w) \in L^*\}$. Then, for every leaf $u \in lv(T_1) \setminus lv(C_1)$ (*resp.*, $u \in lv(T_2) \setminus lv(C_2)$), there exists exactly one $v \in L_1^*$ (*resp.*, $w \in L_2^*$) such that $T_1[v]$ (*resp.*, $T_2[w]$) called as HvyCatMap$(T_1[v],T_2[w])$ at line 4 in Algorithm 1 contains u. This statement implies that $|L^*| \leq \lambda$. Hence, the number of recursive calls is at most λ, so the statement of computing $\tau_{\widehat{\mathrm{HC}}}$ holds. ☐

In the remainder of this section, we assume that the cost function is the unit cost function. Then, we compare $\tau_{\widehat{\mathrm{HC}}}$ with the edit distance τ_{TAI} and its other variations τ_{ALN}, τ_{ILST} and τ_{ALN}.

Lemma 1. *There exist trees T_1 and T_2 such that $|T_1| = |T_2| = O(n)$, $\tau_{\mathrm{TAI}}(T_1,T_2) = O(1)$ but $\tau_{\widehat{\mathrm{HC}}}(T_1,T_2) = \Omega(n)$.*

Proof. Consider T_1 and T_2 illustrated in Figure 2. It is obvious that $|T_1| = |T_2| = 2n+1$. Also it holds that $\tau_{\mathrm{TAI}}(T_1,T_2) = 2$ because M_1 in Figure 2 is the minimum cost mapping for τ_{TAI}. Note that $\tau_{\mathrm{ILST}}(T_1,T_2) = \tau_{\mathrm{ALN}}(T_1,T_2) = \tau_{\mathrm{SG}}(T_1,T_2) = 2$.

On the other hand, by the definition of $\tau_{\widehat{\mathrm{HC}}}$, we construct the mapping with cost 0 between $hc(T_1)$ and $hc(T_2)$, that is, the second child of the root in T_1 (labeled by a) is corresponding to the third child of the root in T_2 (labeled by a) and the third child of the root in T_1 (labeled by b) is to the second child of the root in T_2 (labeled by b). Then, M_2 in Figure 2 is the minimum cost mapping for $\tau_{\widehat{\mathrm{HC}}}$. Hence, it holds that $\tau_{\widehat{\mathrm{HC}}}(T_1,T_2) = 2n-4$. ☐

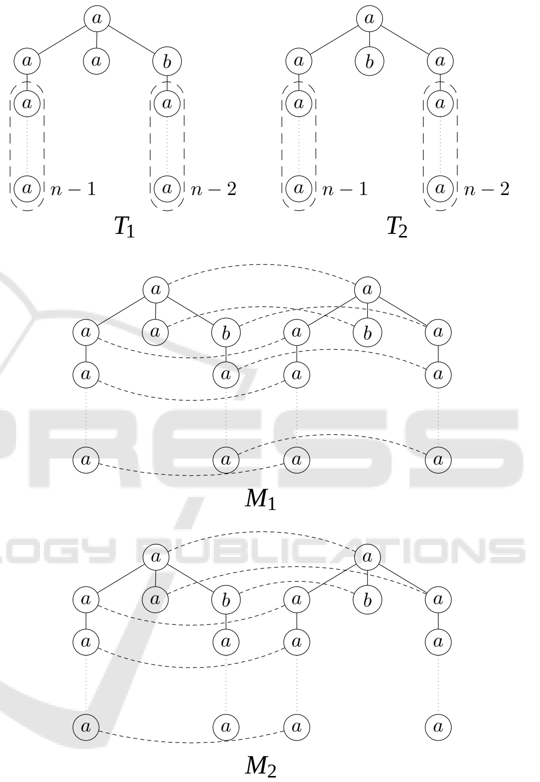

Figure 2: Trees T_1 and T_2 in Lemma 1 and the minimum cost mappings M_1 for τ_{TAI} and M_2 for $\tau_{\widehat{\mathrm{HC}}}$.

Lemma 2. *There exist trees T_1 and T_2 such that $|T_1| = |T_2| = O(n)$, $\tau_{\mathrm{TAI}}(T_1,T_2) = \tau_{\widehat{\mathrm{HC}}}(T_1,T_2) = O(1)$ but $\tau_{\mathrm{ILST}}(T_1,T_2) = \Omega(n)$.*

Proof. Consider T_1 and T_2 illustrated in Figure 3. It is obvious that $|T_1| = |T_2| = 2n+1$. Since T_1 and T_2 are caterpillars, it holds that $\tau_{\mathrm{TAI}}(T_1,T_2) = \tau_{\widehat{\mathrm{HC}}}(T_1,T_2) = \tau_{\mathrm{HC}}(T_1,T_2) = 1$. Note that $\tau_{\mathrm{ALN}}(T_1,T_2) = 1$ and $\tau_{\mathrm{SG}}(T_1,T_2) = 3$.

On the other hand, the minimum cost isolated-subtree mapping maps $r_1 = r(T_1)$ to $r_2 = r(T_2)$, $n+1$ children of r_1 to $n+1$ children of r_2, so the number of the remained (non-mapped) vertices is $n-1+n = 2n-1$. Hence, it holds that $\tau_{\mathrm{ILST}}(T_1,T_2) = 2n-1$. ☐

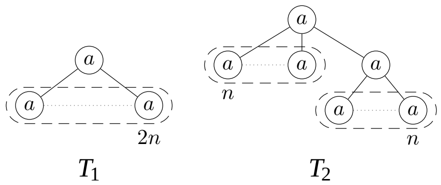

Figure 3: Trees T_1 and T_2 in Lemma 2.

Lemma 3. *There exist trees T_1 and T_2 such that $|T_1| = |T_2| = O(n)$, $\tau_{\mathrm{TAI}}(T_1,T_2) = \tau_{\widehat{\mathrm{HC}}}(T_1,T_2) = O(1)$ but $\tau_{\mathrm{ALN}}(T_1,T_2) = \Omega(n)$.*

Proof. Consider trees T_1 and T_2 in Figure 4. It is obvious that $|T_1| = |T_2| = 2n+2$. Since T_1 is transformed to T_2 by inserting a vertex labeled d in T_2 after deleting a vertex labeled by d in T_1, it holds that $\tau_{\mathrm{TAI}}(T_1,T_2) = 2$. Since T_1 and T_2 are caterpillars, it also holds that $\tau_{\mathrm{HC}}(T_1,T_2) = \tau_{\widehat{\mathrm{HC}}}(T_1,T_2) = 2$. Note that $\tau_{\mathrm{SG}}(T_1,T_2) = 4$.

On the other hand, the minimum cost alignable mapping maps a vertex labeled by b (*resp.*, c) in T_1 to a vertex labeled by c (*resp.*, b) in T_2 injectively. Then, it holds that $\tau_{\mathrm{ALN}}(T_1,T_2) = 2n$. Also it holds that $\tau_{\mathrm{ILST}}(T_1,T_2) = 2n$. □

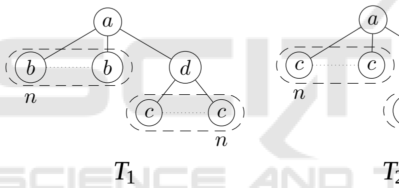

Figure 4: Trees T_1 and T_2 in Lemma 3.

Lemma 4. *There exist trees T_1 and T_2 such that $|T_1| = |T_2| = O(n)$, $\tau_{\mathrm{TAI}}(T_1,T_2) = \tau_{\widehat{\mathrm{HC}}}(T_1,T_2) = O(1)$ but $\tau_{\mathrm{SG}}(T_1,T_2) = \Omega(n)$.*

Proof. Consider T_1 and T_2 illustrated in Figure 5 (*cf.*, (Kan et al., 2014)) and let $C_i = hc(T_i)$ $(i = 1,2)$. It is obvious that $|T_1| = 4n$ and $|T_2| = 4n-2$. Also it holds that $\tau_{\mathrm{TAI}}(T_1,T_2) = 2$.

For C_1 and C_2 in Figure 5, it holds that $\tau_{\mathrm{HC}}(T_1,T_2) = \tau_{\mathrm{TAI}}(C_1,C_2) + 2n = 2n+2$. Since the minimum cost mapping for $\tau_{\mathrm{TAI}}(C_1,C_2)$ maps the rightmost vertex v in C_1 to the rightmost vertex w in C_2, $hc(T_1,T_2)$ maps the children of v in T_1 to the children of w in T_2 injectively. Hence, it holds that $\tau_{\widehat{\mathrm{HC}}}(T_1,T_2) = \tau_{\mathrm{TAI}}(C_1,C_2) = 2$. Note that $\tau_{\mathrm{ILST}}(T_1,T_2) = \tau_{\mathrm{ALN}}(T_1,T_2) = 2$.

On the other hand, since the minimum cost segmental mapping maps to the path with $n-1$ vertices and its n children and the vertex and its n children in T_2, the number of remained (i.e., non-mapped) vertices is $n+1$ in T_1 and $n-1$ in T_2, so it holds that $\tau_{\mathrm{SG}}(T_1,T_2) = 2n$. □

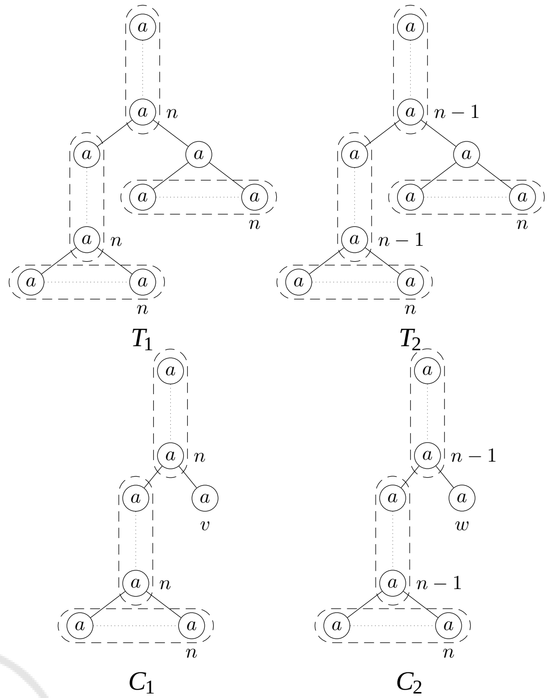

Figure 5: Trees T_1, T_2, C_1 and C_2 in Lemma 4.

Lemmas 2, 3 and 4 imply the following theorem.

Theorem 7. *The distances τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$ are incomparable with the distances τ_{ALN}, τ_{ILST} and τ_{SG}.*

By incorporating Theorem 6 and 7, we can conclude that the heavy caterpillar distances τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$ are tractable variations of the edit distance τ_{TAI} incomparable with the isolated-subtree distance τ_{ILST}.

4 CONCLUSION

In this paper, we have introduced heavy the caterpillar distances τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$ and shown that they provide the upper bound of the edit distance τ_{TAI}, they are tractable, in particular, quadratic-time computable under the unit cost function, and incomparable with other variations of τ_{TAI} presented by (Yoshino and Hirata, 2017). Since τ_{ILST} is the most general tractable variation of τ_{TAI} (Yoshino and Hirata, 2017), τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$ are another tractable variations of τ_{TAI} incomparable with τ_{ILST}.

Concerned with Lemma 1, it is possible to avoid this problem to compute the edit distance (the Tai mapping) between heavy caterpillars by considering the occurrences of labels in the descendants. It is a future work whether or not we can design a new method to avoid to this problem.

The heavy caterpillar distances τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$ are defined by $M_{hc}(C_1,C_2)$ and $M_{hc}(T_1,T_2)$ as operational, whereas other variations of τ_{TAI} are based on the declarative definition of the Tai mapping. Then, it is a future work whether or not to give the declarative definition of τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$.

In general, we cannot determine the heavy path and then the heavy caterpillar uniquely. Then, it is a future work to design the method to select the heavy path and the heavy caterpillar uniquely appropriate to τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$.

Finally, after improving that the heavy caterpillar distances τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$ are determined uniquely, it is an important future work to give experimental results to compare τ_{HC} and $\tau_{\widehat{\mathrm{HC}}}$ with the isolated-subtree distance τ_{ILST} for real data.

ACKNOWLEDGMENTS

This work is partially supported by Grant-in-Aid for Scientific Research 17H00762, 16H02870 and 16H01743 from the Ministry of Education, Culture, Sports, Science and Technology, Japan. The authors would like to thank anonymous referees of ICPRAM'20 for valueable comments to revise the submitted version of this paper.

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.

Demaine, E. D., Mozes, S., Rossman, B., and Weimann, O. (2009). An optimal decomposition algorithm for tree edit distance. *ACM Trans. Algo.*, 6.

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.

Jiang, T., Wang, L., and Zhang, K. (1995). Alignment of trees – an alternative to tree edit. *Theoret. Comput. Sci.*, 143:137–148.

Kan, T., Higuchi, S., and Hirata, K. (2014). Segmental mapping and distance for rooted ordered labeled trees. *Fundam. Inform.*, 132:1–23.

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

Lu, C. L., Su, Z.-Y., and Yang, C. Y. (2001). A new measure of edit distance between labeled trees. In *Proc. COCOON'01 (LNCS 2108)*, pages 338–348.

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

Muraka, K., Yoshino, T., and Hirata, K. (2019). Vertical and horizontal distances to approximate edit distance for rooted labeled caterpillars. In *Proc. ICPRAM'19*, pages 590–597.

Sleator, D. D. and Tarjan, R. E. (1983). A data structure for dynamoic trees. *J. Comput. Sys. Sci.*, 26:362–391.

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

Wang, J. T. L. and Zhang, K. (2001). Finding similar consensus between trees: An algorithm and a distance hierarchy. *Pattern Recog.*, 34:127–137.

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. (1996). A constrained edit distance between unordered labeled trees. *Algorithmica*, 15:205–222.

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