Proceedings of the Federated Conference on Computer Science and Information Systems pp. 433–440
DOI: 10.15439/2017F73
ISSN 2300-5963 ACSIS, Vol. 11

Anchored Alignment Distance between Rooted Labeled Unordered Trees

Takuya Yoshino, Yuma Ishizaka†
Graduate School of Computer Science and Systems Engineering
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820-8502, Japan
Email: {yoshino,y_ishizaka}@dumbo.ai.kyutech.ac.jp

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

***Abstract*—In this paper, we formulate an *anchored alignment distance* between rooted labeled unordered trees as the minimum cost of the anchored alignment whose anchoring is constructed from the minimum cost isolated-subtree mapping by adding the pairs of non-mapped leaves, and design the algorithm to compute it. Since this algorithm runs in exponential time with respect to the number of leaves in theoretical, we give experimental results for randomly generated trees and for N-glycan data with small degree as real data to evaluate the anchored alignment distance by comparing with the isolated-subtree distance and the alignment distance.**

I. Introduction

COMPARING tree-structured data such as HTML and XML data for web mining or DNA and glycan data for bioinformatics is one of the important tasks for data mining. The most famous distance measure between *rooted labeled unordered trees* (*trees*, for short) is the *edit distance* [6], [11], denoted by τ_{TAI}. The edit distance is formulated as the minimum cost of *edit operations*, consisting of a *substitution*, a *deletion* and an *insertion*, applied to transform from a tree to another tree.

It is known that the edit distance is closely related to the notion of a *Tai mapping* (*mapping*, for short) [11], which is a one-to-one node correspondence between trees preserving ancestor relations. Then, the minimum cost of possible Tai mappings coincides with the edit distance [11]. However, it is known that the problem of computing the edit distance between trees is MAX SNP-hard [18] even if they are binary [2].

An *alignment distance*, denoted by τ_{ALN}, is an alternative distance measure to compare trees [4]. The alignment distance is formulated as the minimum cost of an *alignment* between two trees obtained by first inserting nodes labeled with spaces into two trees such that the resulting trees have the same structure and then overlaying them. The alignment distance is an edit distance such that every insertion proceeds to deletions in operational.

Note first that, whereas the edit distance between strings coincides with the alignment distance between them, the edit distance between trees is different from the alignment distance between them in general (*cf.*, [6]); The edit distance is smaller than or equal to the alignment distance. The reason is to exist trees not preserving both cycle-free and ancestor relations when every deletion proceeds to insertions.

As another characterization of the alignment distance for trees, Kuboyama [6] has first formulated an *alignable mapping* as the variation of a Tai mapping whose minimum cost coincides with the alignment distance and shown that the alignable mapping coincides with a *less-constrained mapping* [7]. Furthermore, whereas the problem of computing the alignment distance is also MAX SNP-hard, it is tractable if the maximum degree of two trees are bounded by some constant D, where the detailed time complexity is $O(n^2 D!)$ time for the maximum number n of nodes in two trees [4].

In bioinformatics, Schiermer and Giegerich [10] have introduced an *anchored alignment* with respect to a Tai mapping, called an *anchoring*, in the context of forest alignments. The anchored alignment is an alignment (that is, a tree) which contains a node labeled by a pair of labels for every pair of nodes in the anchoring.

However, there arises a problem that an arbitrary anchoring between two trees does not always provide an anchored alignment, since an arbitrary Tai mapping is not always an alignable (that is, a less-constrained) mapping. In order to avoid this problem, Ishizaka *et al.* [3] have designed an efficient algorithm to compute the anchored alignment in $O(H|M|^2 + n)$ time if an anchoring M is less-constrained; returns "no" otherwise, where H is the maximum height of two trees.

In order to compute the anchored alignment, it is necessary to give an anchoring. In this paper, we construct an anchoring from the minimum cost *isolated-subtree* (or *constrained*) *mapping* [12], [16], [17], because the isolated-subtree mapping is the nearest mapping to the less-constrained mapping in a Tai mapping hierarchy [6], [15] and we can compute an *isolated-subtree distance* τ_{ILST} as the minimum cost of possible isolated-subtree mappings in $O(n^2 d)$ time, where d is the minimum of the degrees of two trees [13].

For the minimum cost isolated-subtree mapping M, we select the set M' of pairs of non-mapped leaves by M. Then, we formulate an *anchored alignment distance* τ_{ACH} as the minimum cost of the anchored alignment through an anchoring

†Current affiliation: Hitachi, Ltd.

*The author would like to express thanks for support by Grant-in-Aid for Scientific Research 17H00762, 16H02870, 16H01743 and 15K12102 from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

$M \cup M'$ if $M \cup M'$ is less-constrained; τ_{ILST} otherwise. We design the algorithm to compute τ_{ACH} in $O(n^2(d+H2^v))$ time, where v is the minimum number of leaves in two trees.

Since this algorithm runs in exponential time with respect to v in theoretical, we first give experimental results for randomly generated trees to evaluate the anchored alignment distance τ_{ACH} by comparing with the isolated-subtree distance τ_{ILST}. Here, in this experiment, we cannot compute the alignment distance τ_{ALN} of which time complexity is $O(n^2D!)$ within one day. Next, we give experimental results for N-glycan data as real data provided from KEGG [5] whose v is small to compute τ_{ACH} efficiently. Then, we compare τ_{ACH} with τ_{ALN} and τ_{ILST}, where it holds that $\tau_{\text{ALN}} \leq \tau_{\text{ACH}} \leq \tau_{\text{ILST}}$ in general. For N-glycan data, it holds that $\tau_{\text{ALN}} = \tau_{\text{ACH}} = \tau_{\text{ILST}}$ in more than 94% pairs and $\tau_{\text{ACH}} = \tau_{\text{ILST}}$ in more than 99% pairs. Furthermore, we investigate the pairs such that $\tau_{\text{ALN}} < \tau_{\text{ACH}} < \tau_{\text{ILST}}$ and $\tau_{\text{ALN}} = \tau_{\text{ACH}} < \tau_{\text{ILST}}$.

II. Preliminaries

A *tree* is a connected graph without cycles. For a tree $T = (V, E)$, we denote V and E by $V(T)$ and $E(T)$, respectively. 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 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 $r(T)$. For each node v in a rooted tree with the root r, let $UP_r(v)$ be the unique path from v to r. If $UP_r(v)$ has exactly k edges, then we say that the *height* of v is k and denote it by $h(v) = k$. We define $h(T) = \max\{h(v) \mid v \in T\}$ and call it the *height* of T.

The *parent* of $v(\neq r)$ 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 that v is an ancestor of u by $u < v$ that $u < v$ or $u = v$ by $u \leq v$. Also we denote neither $u \leq v$ nor $v \leq u$ by $u \# v$. 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 w' such that $w' \leq w$, $u \leq w'$ and $v \leq w'$.

We say that u is a *child* of v if v is the parent of u. The set of children of v is denoted by $ch(v)$. A *leaf* is a node having no children. We denote the set of all leaves in T by $lv(T)$. We define $d(v) = |ch(v)|$ and $d(T) = \max\{d(v) \mid v \in T\}$ and call them the *degree* of v and T, respectively.

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 $l(v)$. Let $\varepsilon \notin \Sigma$ denote a special *blank* symbol and $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.

Let $v \in T$ and $v_i, v_j \in ch(v)$ such that v_i (*resp.*, v_j) is the i-th (*resp.*, j-th) child of v. We say that v_i is *to the left of* v_j if $i \leq j$. Also, for every $u, v \in T$, we define a *sibling order* $u \preceq v$ if there exist $u', v' \in ch(u \sqcup v)$ such that $u \leq u'$, $v \leq v'$ and u' is to the left of v'. Hence, we say that a rooted tree is *ordered* if the sibling order $\preceq$ is fixed; *unordered* otherwise. In this paper, we call a rooted labeled unordered tree a *tree*.

Definition 1 (Edit operations [11]): The *edit operations* of a tree T are defined as follows.

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'.
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'.

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 $l_1 = \varepsilon$.

We define a *cost function* $\gamma : (\Sigma_\varepsilon \times \Sigma_\varepsilon - \{(\varepsilon, \varepsilon)\}) \mapsto \mathbf{R}^+$ on 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 $\gamma(l_1, l_3) \leq \gamma(l_1, l_2) + \gamma(l_2, l_3)$. We call the cost function that $\gamma(l_1, l_2) = 1$ if $l_1 \neq l_2$ a *unit cost function* and denote it by μ.

Definition 2 (Edit distance [11]): 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* $\tau^{\gamma}_{\text{TAI}}(T_1, T_2)$ between trees T_1 and T_2 under γ is defined as follows:

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

Definition 3 (Tai mapping [11]): Let T_1 and T_2 be trees and $M \subseteq V(T_1) \times V(T_2)$. We say that a triple (M, T_1, T_2) is a *Tai mapping* between T_1 and T_2 if 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. Also we denote the set of all the Tai mappings between T_1 and T_2 by $\mathcal{M}_{\text{TAI}}(T_1, T_2)$.

We denote the sets $\{v \in T_1 \mid (v, w) \in M\}$ and $\{w \in T_2 \mid (v, w) \in M\}$ by $M|_1$ and $M|_2$, respectively. For $M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)$, the *cost* $\gamma(M)$ of M is given as:

$$\gamma(M) = \sum_{(v,w)\in M} \gamma(v, w) + \sum_{v \in T_1 - M|_1} \gamma(v, \varepsilon) + \sum_{w \in T_2 - M|_2} \gamma(\varepsilon, w).$$

Theorem 1 (Tai [11]): $\tau^{\gamma}_{\text{TAI}}(T_1, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)\}$.

Definition 4 (Less constrained and isolated-subtree mappings): Let T_1 and T_2 be trees and $M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)$.

1) We say that M is a *less-constrained mapping* [7], denoted by $M \in \mathcal{M}_{\text{LESS}}(T_1, T_2)$, if M satisfies that, for every $(v_1, w_1), (v_2, w_2), (v_3, w_3) \in M$:
$$v_1 \sqcup v_2 < v_1 \sqcup v_3 \Longrightarrow w_2 \sqcup w_3 = w_1 \sqcup w_3.$$
Or equivalently [6]:
$$w_1 \sqcup w_2 < w_1 \sqcup w_3 \Longrightarrow v_2 \sqcup v_3 = v_1 \sqcup v_3.$$
2) We say that M is an *isolated-subtree mapping* [12] (or a *constrained mapping* [16], [17]), denoted by

$M \in \mathcal{M}_{\text{ILST}}(T_1, T_2)$, if M satisfies that, for every $(v_1, w_1), (v_2, w_2), (v_3, w_3) \in M$:

$$v_3 < v_1 \sqcup v_2 \iff w_3 < w_1 \sqcup w_2.$$

As similar as Theorem 1, we formulate a *less-constrained distance* $\tau^{\gamma}_{\text{LESS}}(T_1, T_2)$ and an *isolated-subtree distance* $\tau^{\gamma}_{\text{ILST}}(T_1, T_2)$ as follows:

$$\tau^{\gamma}_{\text{LESS}}(T_1, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\text{LESS}}(T_1, T_2)\},$$
$$\tau^{\gamma}_{\text{ILST}}(T_1, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\text{ILST}}(T_1, T_2)\}.$$

For $\mathtt{A} \in \{\text{TAI}, \text{LESS}, \text{ILST}\}$, we define the set $\mathcal{M}^*_{\mathtt{A}}(T_1, T_2, \gamma)$ of all the minimum cost mappings between T_1 and T_2 under a cost function γ as follows.

$$\mathcal{M}^*_{\mathtt{A}}(T_1, T_2, \gamma) = \text{argmin}\{\gamma(M) \mid M \in \mathcal{M}_{\mathtt{A}}(T_1, T_2)\}.$$

Jiang *et al.* [4] have introduced an alignment distance as an alternative distance measure to compare trees, which is based on an alignment. Here, for two trees T_1 and T_2, we say that T_1 and T_2 are *isomorphic without labels* if there exists a bijection ϕ from $V(T_1)$ to $V(T_2)$, called an *isomorphism*, satisfying that $u \leq v$ iff $\phi(u) \leq \phi(v)$.

Definition 5 (Alignment [4]): Let T_1 and T_2 be trees. Then, an *alignment* between T_1 and T_2 is a tree $\mathcal{T}$ obtained by the following steps.

1) Insert new nodes labeled by ε into T_1 and T_2 so that the resulting trees T_1' and T_2' are isomorphic without labels and $l(\phi(v)) \neq \varepsilon$ whenever $l(v) = \varepsilon$ for an isomorphism ϕ from T_1' to T_2' and every node $v \in T_1'$.
2) Set $\mathcal{T}$ to a tree T_1' obtained by relabeling a label $l(v)$ for every node $v \in T_1'$ with $(l(v), l(\phi(v)))$. (Note that $(\varepsilon, \varepsilon) \notin \mathcal{T}$.)

Let $\mathcal{A}(T_1, T_2)$ denote the set of all possible alignments between T_1 and T_2. The *cost* $\gamma(\mathcal{T})$ of an alignment $\mathcal{T}$ is the sum of the costs of all labels in $\mathcal{T}$.

Definition 6 (Alignment distance [4]): Let T_1 and T_2 be trees and γ a cost function. Then, an *alignment distance* $\tau^{\gamma}_{\text{ALN}}(T_1, T_2)$ between T_1 and T_2 under γ is defined as follows.

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

In operational, the alignment distance is an edit distance such that every insertion proceeds to deletions [4]. Furthermore, the following theorem is known.

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

1) ([6], [7]) $\mathcal{M}_{\text{ILST}}(T_1, T_2) \subseteq \mathcal{M}_{\text{LESS}}(T_1, T_2) \subseteq \mathcal{M}_{\text{TAI}}(T_1, T_2)$, which implies that $\tau^{\gamma}_{\text{TAI}}(T_1, T_2) \leq \tau^{\gamma}_{\text{LESS}}(T_1, T_2) \leq \tau^{\gamma}_{\text{ILST}}(T_1, T_2)$. The equation does not always hold in general.
2) ([3], [6]) $\tau^{\gamma}_{\text{ALN}}(T_1, T_2) = \tau^{\gamma}_{\text{LESS}}(T_1, T_2)$.
3) ([2], [18]) The problem of computing $\tau^{\gamma}_{\text{TAI}}(T_1, T_2)$ is MAX SNP-hard, even if T_1 and T_2 are binary trees.
4) ([4]) The problem of computing $\tau^{\gamma}_{\text{ALN}}(T_1, T_2)$ is MAX SNP-hard. On the other hand, if D is bounded by some constant, then we can compute $\tau^{\gamma}_{\text{ALN}}(T_1, T_2)$ in $O(nmD!)$ time.
5) ([13]) We can compute $\tau^{\gamma}_{\text{ILST}}(T_1, T_2)$ in $O(nmd)$ time.

III. Anchored Alignment Distance

Let T_1 and T_2 be trees and $M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)$ called an *anchoring*. Then, Schiermer and Giegerich [10] have introduced an *anchored alignment* between T_1 and T_2 *through* M, which we call in this paper, as an alignment $\mathcal{T}$ containing a node labeled by $(l(v), l(w))$ for every $(v, w) \in M$. We denote it by $ach(T_1, T_2, M)$.

Note that an arbitrary anchoring does not always provide an anchored alignment, since $M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)$ whenever $M \in \mathcal{M}_{\text{LESS}}(T_1, T_2)$ but the converse direction does not hold in general (Theorem 2.1). In order to avoid this problem, Ishizaka *et al.* [3] have formulated an *anchored alignment problem* to output an anchored alignment $\mathcal{T}$ between T_1 and T_2 through M if $\mathcal{T}$ exists; return "no" otherwise. Also they have designed an efficient algorithm, called ACHALN in this paper, to solve the problem by using the following cover sequence.

For $M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)$ and $(v, w) \in M$, let $S_1(v) = V(T_1[v]) \cap M|_1$ and $S_2(w) = V(T_2[v]) \cap M|_2$, where $T[v]$ denotes the complete subtree of T rooted at $v \in T$. Also, by denoting $UP_{r_1}(v)$ (*resp.*, $UP_{r_2}(w)$) as a sequence $[r_1, \ldots, v]$ (*resp.*, $[r_2, \ldots, w]$) for $r_1 = r(T_1)$ (*resp.*, $r_2 = r(T_2)$), the *cover sequence* of v in T_1 (*resp.*, w in T_2), denoted by $C_1(v)$ (*resp.*, $C_2(w)$), is a sequence $[S_1(r_1), \ldots, S_1(v)]$ (*resp.*, $[S_2(r_2), \ldots, S_2(w)]$). We say that $C_1(v)$ and $C_2(w)$ are *incomparable* if there exist $s_1 \in C_1(T_1)$ and $s_2 \in C_2(T_2)$ such that neither $s_1 \subseteq s_2$ nor $s_2 \subseteq s_1$.

Then, the outline of the algorithm ACHALN is illustrated as follows.

1) For every $(v, w) \in M$, construct cover sequences $C_1(v)$ and $C_2(w)$.
2) If there exists $(v, w) \in M$ such that $C_1(v)$ and $C_2(w)$ are incomparable, then set $ach(T_1, T_2, M)$ to $\emptyset$.
3) Otherwise:
   a) For every $(v, w) \in M$, align $C_1(v)$ and $C_2(w)$ as $C_1'(v)$ and $C_2'(w)$ and construct a path $P(v, w)$ by pairing each element of $C_1'(v)$ and $C_2'(w)$.
   b) Set $ach(T_1, T_2, M)$ to a tree constructed from merging every path $P(v, w)$.

Theorem 3 (Ishizaka et al. [3]): We can solve the anchored alignment problem in $O(H|M|^2 + n + m)$ time, where $n = |T_1|$, $m = |T_2|$ and $H = \max\{h(T_1), h(T_2)\}$.

Then, we can formulate an *anchored alignment distance through* M as follows.

Definition 7 (Anchored alignment distance through mapping): Let T_1 and T_2 be trees, $M \in \mathcal{M}_{\text{TAI}}(T_1, T_2)$ and γ a cost function. Then, we define an *anchored alignment distance* $\tau^{\gamma}_{\text{ACH}}(T_1, T_2, M)$ between T_1 and T_2 *through* M under γ as follows.

$$\tau^{\gamma}_{\text{ACH}}(T_1, T_2, M) = \begin{cases} \gamma(ach(T_1, T_2, M)), & \text{if } ach(T_1, T_2, M) \neq \emptyset, \\ |T_1| + |T_2|, & \text{otherwise.} \end{cases}$$

By Theorem 2.2, $M \in \mathcal{M}_{\text{LESS}}(T_1, T_2)$ iff $ach(T_1, T_2, M) \neq \emptyset$. The statements 1 and 2 in ACHALN can determine whether or not $M \in \mathcal{M}_{\text{LESS}}(T_1, T_2)$ in $O(H|M|)$ time. Also, by Theorem 2.1, $M \in \mathcal{M}_{\text{LESS}}(T_1, T_2)$ whenever $M \in \mathcal{M}_{\text{ILST}}(T_1, T_2)$.

Theorem 4: For trees T_1 and T_2 and a cost function γ, suppose that $M_1 \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \gamma)$ and $M_2 \in \mathcal{M}^*_{\text{LESS}}(T_1, T_2, \gamma)$. If $M_1 \subset M_2$, then, for every $(v, w) \in M_2 \setminus M_1$, there exist $(v_1, w_1), (v_2, w_2) \in M_1$ satisfying each of the following statements.

1) $v < v_1 \sqcup v_2$ and $w \# w_1 \sqcup w_2$.
2) $v \# v_1 \sqcup v_2$ and $w < w_1 \sqcup w_2$.

Proof: Suppose that neither the statements 1 nor 2 holds. Then, for every $(v_1, w_1), (v_2, w_2) \in M_1$, it holds that (1) $v < v_1 \sqcup v_2$ and one of $w \leq w_1 \sqcup w_2$, $w = w_1 \sqcup w_2$ or $w_1 \sqcup w_2 \leq w$ and (2) $w < w_1 \sqcup w_2$ and one of $v \leq v_1 \sqcup v_2$, $v = v_1 \sqcup v_2$ or $v_1 \sqcup v_2 \leq v$. By the ancestor condition, it holds that $v < v_1 \sqcup v_2 \iff w < w_1 \sqcup w_2$, which implies that $M_2 \in \mathcal{M}_{\text{ILST}}(T_1, T_2)$. Since $M_1 \subset M_2$ and $M_1 \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \gamma)$, it is a contradiction. ■

Theorem 5: There exist trees T_1 and T_2 and a cost function γ such that neither $M_1 \subset M_2$ nor $M_2 \subset M_1$ for $M_1 \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \gamma)$ and $M_2 \in \mathcal{M}^*_{\text{LESS}}(T_1, T_2, \gamma)$. This statement also holds even if trees T_1 and T_2 are unlabeled (or equivalently unique-labeled).

Proof: Let μ be the unit cost function. First consider the trees T_1 and T_2 in Figure 1 (left). Figure 1 (right) illustrates $M_1 \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \mu)$ and $M_2 \in \mathcal{M}^*_{\text{LESS}}(T_1, T_2, \mu)$. Then, it holds that $\mu(M_1) = 3$ and $\mu(M_2) = 2$, but neither $M_1 \subset M_2$ nor $M_2 \subset M_1$.

Also consider the unique-labeled trees T_3 and T_4 in Figure 2 (left). Figure 2 (right) illustrates $M_3 \in \mathcal{M}^*_{\text{ILST}}(T_3, T_4, \mu)$ and $M_4 \in \mathcal{M}^*_{\text{LESS}}(T_3, T_4, \mu)$. Then, it holds that $\mu(M_3) = 4$ and $\mu(M_4) = 2$, but neither $M_3 \subset M_4$ nor $M_4 \subset M_3$. ■

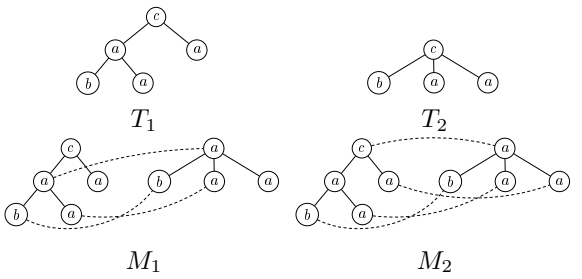

Fig. 1. Trees T_1 and T_2 (upper), $M_1 \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \mu)$ and $M_2 \in \mathcal{M}^*_{\text{LESS}}(T_1, T_2, \mu)$ (lower) in the proof of Theorem 5.

Theorem 5 claims that the minimum cost less-constrained mapping is not always comparable with the minimum cost isolated-subtree mapping (as set inclusion). On the other hand, $\mathcal{M}_{\text{ILST}}$ is the nearest mapping class to $\mathcal{M}_{\text{LESS}}$ in a Tai mapping hierarchy [6], [15] and τ_{ILST} is the most general tractable variation of τ_{TAI} [13]. Hence, in this paper, we construct candidates of an anchoring by adding pairs of non-mapped leaves to $M \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \gamma)$ by Theorem 4.

For $M \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \gamma)$, let $\overline{M}$ be a complement of M, that is, $\{(v, w) \in T_1 \times T_2 \mid (v, w) \notin M\}$. A *total leaf mapping* of M, denoted by $lm(M)$, is defined as $\overline{M} \cap (lv(T_1) \times lv(T_2))$ and we call $M' \subseteq lm(M)$ (possibly $M' = \emptyset$) a *leaf mapping* of M. Whereas every leaf mapping is always a Tai mapping, $M \cup M'$ is not always a Tai mapping. Then, for $M \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \gamma)$ and $M' \subseteq lm(M)$, by applying the algorithm ACHALN for an anchoring $M \cup M'$ as input, we can obtain the anchored alignment $ach(T_1, T_2, M \cup M')$.

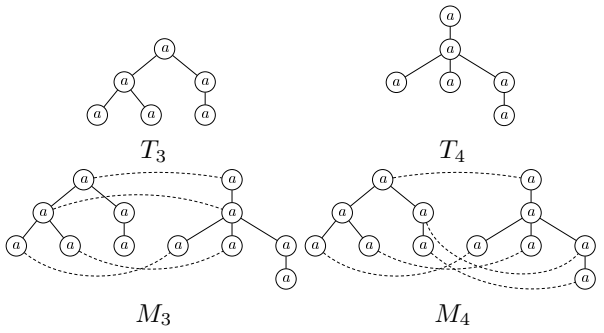

Fig. 2. Unique-labeled trees T_3 and T_4 (upper), $M_3 \in \mathcal{M}^*_{\text{ILST}}(T_3, T_4, \mu)$ and $M_4 \in \mathcal{M}^*_{\text{LESS}}(T_3, T_4, \mu)$ (lower) in the proof of Theorem 5.

Definition 8 (Anchored alignment distance): Let T_1 and T_2 be trees and and γ a cost function. Then, an *anchored alignment distance* $\tau^{\gamma}_{\text{ACH}}(T_1, T_2)$ between T_1 and T_2 under γ is defined as follows.

$$\tau^{\gamma}_{\text{ACH}}(T_1, T_2) = \min \left\{ \gamma(ach(T_1, T_2, N)) \;\middle|\; \begin{array}{l} M \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \gamma), \\ M' \subseteq lm(M), \\ N = M \cup M', \\ N \in \mathcal{M}_{\text{LESS}}(T_1, T_2) \end{array} \right\}.$$

If no M' such that $M' \subseteq lm(M)$ and $M \cup M' \in \mathcal{M}_{\text{LESS}}(T_1, T_2)$ exists, then it holds that $\tau^{\gamma}_{\text{ACH}}(T_1, T_2) = \tau^{\gamma}_{\text{ILST}}(T_1, T_2) = \gamma(ach(T_1, T_2, M))$, by regarding M' as $\emptyset$ and since $M \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \gamma)$. Hence, we can avoid to the case that $|T_1| + |T_2|$ in Definition 7 and it holds that $\tau^{\gamma}_{\text{ALN}}(T_1, T_2) \leq \tau^{\gamma}_{\text{ACH}}(T_1, T_2) \leq \tau^{\gamma}_{\text{ILST}}(T_1, T_2)$.

For every alignment $\mathcal{T}$, we can construct a mapping which consists of a pair (v, w) for every node $(l(v), l(w)) \in \mathcal{T}$. We call it an *alignable* mapping constructed from $\mathcal{T}$ [6]. Then, we denote the set of all the alignable mappings constructed from $ach(T_1, T_2, N)$ such that $M \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \gamma)$, $M' \subseteq lm(M)$, $N = M \cup M'$, $N \in \mathcal{M}_{\text{LESS}}(T_1, T_2)$ and the cost is minimum under γ by $\mathcal{M}^*_{\text{ACH}}(T_1, T_2, \gamma)$.

Theorem 6: We can compute $\tau^{\gamma}_{\text{ACH}}(T_1, T_2)$ in $O(nm(d + H2^v))$ time, where $n = |T_1|$, $m = |T_2|$, $d = \min\{d(T_1), d(T_2)\}$, $H = \max\{h(T_1), h(T_2)\}$ and $v = \min\{|lv(T_1)|, |lv(T_2)|\}$.

Proof: Consider the algorithm ACHALNDIST in Algorithm 1. Here, the algorithm $\text{ILST}(T_1, T_2, \gamma)$ in line 1 returns a pair of the isolated-subtree distance $\tau^{\gamma}_{\text{ILST}}(T_1, T_2)$ and its minimum cost isolated-subtree mapping in $\mathcal{M}^*_{\text{ILST}}(T_1, T_2, \gamma)$ [13], which runs in $O(nmd)$ time. By line 4, we ignore the case that $M \cup M'$ is not less-constrained. By Definition 8, if no $M' (\neq \emptyset)$ such that $M \cup M' \in \mathcal{M}_{\text{LESS}}(T_1, T_2)$ exists, then it holds that $\tau^{\gamma}_{\text{ACH}}(T_1, T_2) = \tau^{\gamma}_{\text{ILST}}(T_1, T_2)$ as the case that $M' = \emptyset$, realized by the selection of the minimum value of d and $\gamma(\mathcal{T})$ in lines 1 and 5. Since the running time of line 3 is $O(H|M|^2 + n + m) = O(mnH)$ by Theorem 3 and the

number of M' in line 2 is at most 2^v, the time complexity also holds. ■

procedure ACHALNDIST(T_1, T_2, γ)
/* T_1, T_2 : tree, γ: cost function */
1 $(d, M) \leftarrow$ ILST(T_1, T_2, γ);
/* $d = \tau^{\gamma}_{\text{ILST}}(T_1, T_2)$, $M \in \mathcal{M}^{*}_{\text{ILST}}(T_1, T_2, \gamma)$ */
2 **foreach** $M' \subseteq lm(M)$ s.t. $M \cup M' \in \mathcal{M}_{\text{TAI}}(T_1, T_2)$ **do**
3 $\mathcal{T} \leftarrow$ ACHALN($T_1, T_2, M \cup M'$);
/* $\mathcal{T} = \emptyset$ if $M \cup M'$ is not less-constrained */
4 **if** $\gamma(\mathcal{T}) > 0$ **then**
/* $\gamma(\mathcal{T}) = 0 \iff \mathcal{T} = \emptyset$ */
5 $d \leftarrow \min\{d, \gamma(\mathcal{T})\}$;
6 **output** d;

Algorithm 1: ACHALNDIST.

IV. EXPERIMENTAL RESULTS

In this section, we assume a cost function is always the unit cost function μ, so the subscript of a cost function is omitted. Also the computer environment is that CPU is Intel Xeon E51650 v3 (3.50GHz), RAM is 1GB and OS is Ubuntsu Linux (64bit).

A. Randomly generated trees

The running time of the algorithm ACHALNDIST is exponential with respect to the number of leaves in Theorem 6, that is, $O(nm(d + H2^v))$ time. Here, v depends on the number of M' in line 2 of the algorithm ACHALNDIST, which is the minimum value of $|lv(T_1)| - |(M|_1) \cap lv(T_1)|$ and $|lv(T_2)| - |(M|_2) \cap lv(T_2)|$. Then, τ_{ACH} is possible to be computed efficiently between trees if such a value is small.

First, we evaluate the above situation by using randomly generated trees. In this experiment, by using the algorithm PTC [8], we generate 10 rooted labeled trees with from 100 to 200 nodes when varying the maximum degree for 2, 3, 4, 5 and 10 and then compute τ_{ACH} and τ_{ILST} for all of the pairs of 10 trees, that is, 45 pairs. Table I illustrates the running time to compute τ_{ACH} and τ_{ILST}, the average value of τ_{ACH} and τ_{ILST} and the number of different pairs of τ_{ILST} and τ_{ACH}. Note that, under this setting, we cannot compute τ_{ALN} even if the maximum degree is 2 within one day.

TABLE I
THE RUNNING TIME TO COMPUTE τ_{ACH} AND τ_{ILST}, THE AVERAGE VALUE OF τ_{ACH} AND τ_{ILST} AND THE NUMBER OF THE DIFFERENT PAIRS OF τ_{ILST} AND τ_{ACH}.

max. degree	2	3	4	5	10
τ_{ACH} time (ms)	926	1,720	14,221	14,892	71,399
τ_{ILST} time (ms)	719	635	609	545	552
τ_{ACH} average	121.93	130.87	133.30	126.51	133.89
τ_{ILST} average	121.93	131.22	133.20	127.60	136.00
$\tau_{\text{ACH}} < \tau_{\text{ILST}}$	0	10	21	25	34
	0%	22.22%	46.67%	55.56%	75.56%

Table I shows that, when the maximum degree is increasing, whereas the average value is independent from the maximum degree, the running time is increasing exponentially and the pairs such that $\tau_{\text{ACH}} < \tau_{\text{ILST}}$ is increasing.

B. N-glycan data

Next, as real data for trees with small v, we adopt N-glycan data provided from KEGG [5] and evaluate τ_{ACH} by comparing with τ_{ILST} and τ_{ALN} and their mappings in more detail.

Here, the number of N-glycan data is $2,142$ and then the number of pairs is $2,293,011$. Furthermore, Table II illustrates the minimum, the maximum and the average values of the number of nodes, the number of leaves, the degree and the height of N-glycan data.

TABLE II
THE MINIMUM, THE MAXIMUM AND THE AVERAGE VALUES OF THE NUMBER OF NODES, THE NUMBER OF LEAVES, THE DEGREE AND THE HEIGHT OF N-GLYCAN DATA.

	min.	max.	ave.
nodes	2	38	11.0696
leaves	1	12	3.2876
degree	1	3	2.0724
height	1	5	5.3838

Table III illustrates the running time to compute τ_{ALN}, τ_{ACH} and τ_{ILST} for all the pairs of N-glycan data.

TABLE III
THE RUNNING TIME TO COMPUTE τ_{ALN}, τ_{ACH} AND τ_{ILST}.

distance	time(ms)
τ_{ALN}	50,503,659
τ_{ACH}	595,188
τ_{ILST}	274,425

Table III shows that, for N-glycan data, the total running time of computing τ_{ACH} is not so large and nearer the running time of computing τ_{ILST} whose complexity is $O(mnd)$ time than the running time of computing τ_{ALN} whose complexity is $O(nmD!)$ time. The total running time of computing τ_{ACH} is less than thrice of the total running time of computing τ_{ILST}.

Table IV illustrates the number of pairs for every inequality between τ_{ALN}, τ_{ACH} and τ_{ILST}. Note that $\tau_{\text{ALN}} \leq \tau_{\text{ACH}} \leq \tau_{\text{ILST}}$.

In contrast to Table I, Table IV shows that the number of pairs that $\tau_{\text{ALN}} = \tau_{\text{ACH}} = \tau_{\text{ILST}}$ is $2,116,005$, which is 94.4612% for all the pairs, and the number of pairs that $\tau_{\text{ACH}} = \tau_{\text{ILST}}$ is $2,275,260$, which is 99.2259% for all the pairs. Also, the number of pairs that $\tau_{\text{ALN}} = \tau_{\text{ACH}} < \tau_{\text{ILST}}$, which improve τ_{ILST} as τ_{ACH} is $17,144$, which is 0.7477% for all the pairs. On the other hand, the number of pairs that $\tau_{\text{ALN}} < \tau_{\text{ACH}}$, which is corresponding to Theorem 5, is $109,862$, which is 4.7912% pairs for all the pairs.

Concerned with Table III and IV, Table V illustrates the number of M' satisfying the condition of line 2 in the algorithm ACHALNDIST for all the pairs.

Table V claims that no M' in line 2 in the algorithm ACHALNDIST is selected in $2,275,201$ pairs, which is

TABLE IV
THE NUMBER OF PAIRS FOR EVERY INEQUALITY BETWEEN τ_{ALN}, τ_{ACH} AND τ_{ILST}.

inequality	#pairs	%
$\tau_{\text{ALN}} = \tau_{\text{ACH}} = \tau_{\text{ILST}}$	2,166,005	94.4612
$\tau_{\text{ALN}} < \tau_{\text{ACH}} = \tau_{\text{ILST}}$	109,255	4.7647
$\tau_{\text{ALN}} = \tau_{\text{ACH}} < \tau_{\text{ILST}}$	17,144	0.7477
$\tau_{\text{ALN}} < \tau_{\text{ACH}} < \tau_{\text{ILST}}$	607	0.0265

inequality	#pairs	%
$\tau_{\text{ALN}} < \tau_{\text{ILST}}$	127,006	5.5388
$\tau_{\text{ALN}} < \tau_{\text{ACH}}$	109,862	4.7912
$\tau_{\text{ACH}} < \tau_{\text{ILST}}$	17,751	0.7741

TABLE V
THE NUMBER OF M' SATISFYING THE CONDITION OF LINE 2 IN ACHALNDIST.

$\#M'$	#pairs
0	2,275,201
1	11,107
2	3,699
3	2,408
4	372

$\#M'$	#pairs
5	110
6	88
7	2
8	2

$\#M'$	#pairs
11	1
12	15
20	3
42	3

99.2233% for all the pair. Then, the number of M' is too smaller than the theoretical worst case $O(2^v)$, which implies the experimental efficiency of the algorithm ACHALNDIST illustrated in Table III. This is also the reason why the number of pairs that $\tau_{\text{ACH}} = \tau_{\text{ILST}}$ is very close to the number of all the pairs illustrated in Table IV. Furthermore, for the 24 pairs such that $\#M' \geq 8$, it holds that $\tau_{\text{ALN}} = \tau_{\text{ACH}} < \tau_{\text{ISLT}}$.

For the three cases in Table IV that (1) $\tau_{\text{ALN}} < \tau_{\text{ACH}} < \tau_{\text{ISLT}}$, (2) $\tau_{\text{ALN}} < \tau_{\text{ACH}} = \tau_{\text{ISLT}}$ and (3) $\tau_{\text{ALN}} = \tau_{\text{ACH}} < \tau_{\text{ISLT}}$, Table VI summarizes the average and the maximum values of difference in the inequalities and the pairs whose difference is maximum. Here, the subscript of the glycan number denotes its number of nodes. Table VI claims that the number of nodes in the pairs in the above inequalities is not always large, that is, near to 38 and at most one tree in the pairs is large.

TABLE VI
THE AVERAGE AND THE MAXIMUM VALUES OF DIFFERENCE IN THE INEQUALITIES AND THE PAIRS WHOSE DIFFERENCE IS MAXIMUM.

case	inequality	ave.	max.	#pairs	pairs
(1)	$\tau_{\text{ALN}} < \tau_{\text{ACH}}$	1.0644	7	2	(G06867_{28},G11335_{19}), (G06867_{28},G11339_{20})
(2)	$\tau_{\text{ACH}} < \tau_{\text{ILST}}$	1.0319	4	4	(G03669_{17},G04570_{11}), (G04186_{20},G04570_{11}), (G04570_{11},G04972_{19}), (G04570_{11},G06997_{18})
(3)	$\tau_{\text{ALN}} < \tau_{\text{ACH}}$	1.0115	3	1	(G04045_{36},G05896_{19})
	$\tau_{\text{ACH}} < \tau_{\text{ILST}}$	1.0537	3	4	(G04191_{18},G04570_{11}), (G04206_{37},G04570_{11}), (G04570_{11},G11846_{38}), (G04570_{11},G11847_{37})
	$\tau_{\text{ALN}} < \tau_{\text{ILST}}$	2.0659	5	3	(G04206_{37},G04570_{11}), (G04570_{11},G11846_{38}), (G04570_{11},G11847_{37})

Consider the glycan G04570_{11}, which occurs most frequently in Table VI. Then, the glycans of G04191_{18}, G04206_{37}, G11846_{38} and G11847_{37} consist of all the pairs with G04570_{11} satisfying that $\tau_{\text{ALN}} < \tau_{\text{ACH}} < \tau_{\text{ILST}}$. All the 4 pairs coincide with the pairs that $\tau_{\text{ACH}} < \tau_{\text{ILST}}$ in case (3) in Table VI.

Figure 3 illustrates the glycans $T_1 = \text{G04191}_{18}$ and $T_2 = \text{G04570}_{11}$, and the minimum cost mappings $M_1 \in \mathcal{M}^*_{\text{LESS}}(T_1, T_2, \mu)$, $M_2 \in \mathcal{M}^*_{\text{ACH}}(T_1, T_2, \mu)$ and $M_3 \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \mu)$. Here, nodes with the different shapes or colors represent different stereochemistry in glycan structures, so we treat them as different labels. Note that $\tau_{\text{ALN}}(T_1, T_2) = 9$, $\tau_{\text{ACH}}(T_1, T_2) = 10$ and $\tau_{\text{ILST}}(T_1, T_2) = 13$.

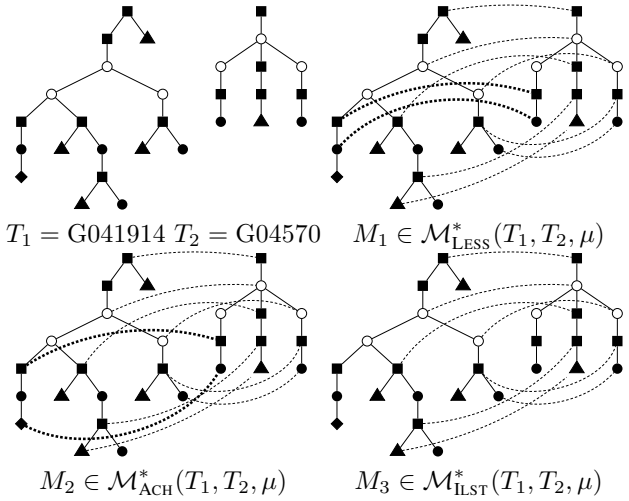

Fig. 3. The glycans $T_1 = \text{G04191}$ and $T_2 = \text{G04570}$, and the minimum cost mappings $M_1 \in \mathcal{M}^*_{\text{LESS}}(T_1, T_2, \mu)$, $M_2 \in \mathcal{M}^*_{\text{ACH}}(T_1, T_2, \mu)$ and $M_3 \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \mu)$.

In Figure 3, we depict the difference between M_1, M_2 and M_3 by thick lines. Then, for $M_3 \in \mathcal{M}^*_{\text{ILST}}(T_1, T_2, \mu)$ as input, the algorithm ACHALNDIST returns M_2 by adding not only the pair of leaves whose labels are different as an anchoring, depicted as the lower thick line, but also the pair of their ancestors, depicted as the upper thick line, to M_3.

On the other hand, since the node in T_1 in the pair in M_1 depicted by the lower thick line is not a leaf in T_1, the algorithm ACHALNDIST cannot find $M_1 \in \mathcal{M}^*_{\text{LESS}}(T_1, T_2, \mu)$. The algorithm ACHALNDIST cannot replace a leaf in pairs given as an anchoring with its ancestor.

Finally, consider the successful cases such that $\tau_{\text{ALN}} = \tau_{\text{ACH}} < \tau_{\text{ILST}}$, that is, the case (2) in Table VI. Figure 4 illustrates the mappings $M_i \in \mathcal{M}^*_{\text{ACH}}(T_i, T, \mu)$ $(1 \leq i \leq 4)$ for $T_1 = \text{G03669}$, $T_2 = \text{G04186}$, $T_3 = \text{G04972}$, $T_4 = \text{G06997}$ and $T = \text{G04570}$. Then, every M_i is obtained by adding the pairs depicted by thick lines to the mapping in $\mathcal{M}^*_{\text{ILST}}(T_i, T, \mu)$.

In contrast to Figure 3, the algorithm ACHALNDIST succeeds to find the minimum cost less-constrained mappings in Figure 4. The reason is that the pair of leaves given as an anchoring is also contained in the minimum cost less-

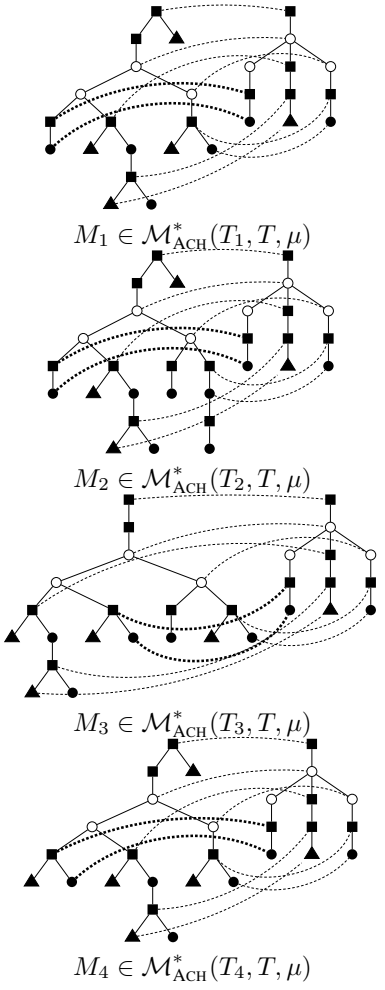

Fig. 4. The mappings $M_i \in \mathcal{M}^*_{\text{ACH}}(T_i, T, \mu)$ $(1 \leq i \leq 4)$ for $T_1 =$ G03669, $T_2 =$ G04186, $T_3 =$ G04972, $T_4 =$ G06997 and $T =$ G04570.

constrained mapping, which is not necessary to replace a leaf in pairs given as an anchoring with its ancestor.

V. Conclusion and Future Works

In this paper, we have formulated the anchored alignment distance τ_{ACH} based on the minimum cost isolated-subtree mapping and designed the algorithm to compute τ_{ACH}. Then, we have given experimental results for randomly generated trees and for N-glycan data to evaluate τ_{ACH} by comparing with τ_{ILST} and τ_{ALN}.

In particular, for N-glycan data, the running time of computing τ_{ACH} have been much smaller than the theoretical worst case and it has been nearer to the running time of computing τ_{ILST} than one of computing τ_{ALN}. The reason is that the number of leaves in N-glycan data is not large. On the other hand, the number of pairs that $\tau_{\text{ALN}} < \tau_{\text{ACH}}$ is larger than one that $\tau_{\text{ALN}} = \tau_{\text{ACH}} < \tau_{\text{ILST}}$, but even the former is less than 5%. It holds that $\tau_{\text{ALN}} = \tau_{\text{ACH}} = \tau_{\text{ILST}}$ in more than 94% pairs. Furthermore, concerned with Figure 3 and 4, we have just observed the improvement that $\tau_{\text{ACH}} < \tau_{\text{ILST}}$ by adding at most two pairs of nodes along a path to the minimum cost isolated-subtree mapping.

Hence, it is a future work to analyze whether or not there are cases that at least three pairs are added by containing some branches for other data.

Concerned with Section IV-B, Mori *et al.* [9] and Yoshino *et al.* [14] have designed the algorithms to compute unordered tree edit distance τ_{TAI} exactly for a part of N-glycan data. Section IV-B claims that the number of pairs that $\tau_{\text{ALN}} < \tau_{\text{ACH}}$ and $\tau_{\text{ALN}} < \tau_{\text{ACH}}$ is much smaller than the number of pairs that $\tau_{\text{ALN}} = \tau_{\text{ACH}}$ and $\tau_{\text{ALN}} = \tau_{\text{ILST}}$. Then, the number of pairs that $\tau_{\text{TAI}} < \tau_{\text{ALN}}$ is possible be much smaller than the number of pairs that $\tau_{\text{TAI}} = \tau_{\text{ALN}}$. In fact, all of the less-constrained mappings in Figure 3 and 4 are the minimum cost Tai mappings, and then it holds that $\tau_{\text{ALN}} = \tau_{\text{TAI}}$ in all the cases. On the other hand, as similar as Figure 2, we provide an example of the unique-labeled trees T_1 and T_2, $M_1 \in \mathcal{M}^*_{\text{TAI}}(T_1, T_2, \mu)$ and $M_2 \in \mathcal{M}^*_{\text{LESS}}(T_1, T_2, \mu)$ in Figure 5 such that, for a unit cost function μ, $\tau^{\mu}_{\text{TAI}}(T_1, T_2) = 2 < 4 = \tau^{\mu}_{\text{ALN}}(T_1, T_2)$.

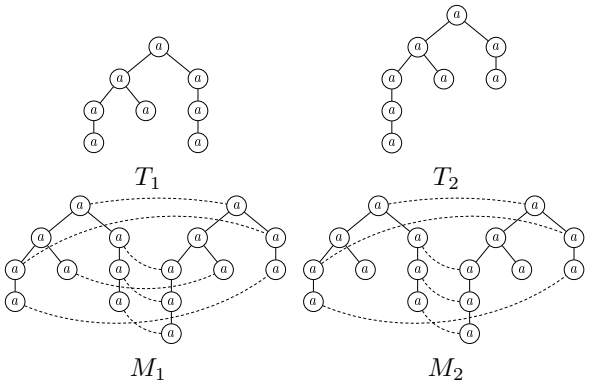

Fig. 5. Trees T_1 and T_2, $M_1 \in \mathcal{M}^*_{\text{TAI}}(T_1, T_2, \mu)$ and $M_2 \in \mathcal{M}^*_{\text{LESS}}(T_1, T_2, \mu)$.

Hence, it is a future work to investigate whether or not the difference between τ_{TAI} and τ_{ALN} exists for N-glycan data and other experimental data.

Concerned with Theorem 4 and 5, it is a future work to investigate the properties of less-constrained mappings to construct an anchoring. In particular, in order to find the minimum cost less-constrained mapping, it is necessary to replace a leaf in pairs given as an anchoring with its ancestor as illustrated in Figure 3. This replacement is possible to be essential for the intractability of the problem of computing τ_{ALN} [4]. Furthermore, concerned with Theorem 6 and Section IV-A, to apply the algorithm ACHALNDIST to trees with many leaves, it is necessary to decrease the number of leaf mappings, by using the number of connected components in the mapping [1], for example.

Hence, it is a future work to improve leaf mappings, to introduce other mappings instead of leaf mappings or to improve the definition of τ_{ACH} and the algorithm ACHALNDIST independent from leaf mappings.

REFERENCES

[1] P. Ferraro, C. Godin: *Optimal mappings with minimum number of connected components in tree-to-tree comparison problems*, J. Algo. **48**, 385–406, 2003. DOI: 10.1016/S0196-6774(03)00079-8.

[2] K. Hirata, Y. Yamamoto, T. Kuboyama: *Improved* MAX SNP-*hard results for finding an edit distance between unordered trees*, Proc. CPM 2011, LNCS **6661**, 402–415, 2011. DOI: 10.1007/978-3-642-21458-5_34.

[3] Y. Ishizaka, T. Yoshino, K. Hirata: *Anchored alignment problem for rooted labeled trees*, New Frontiers in Artificial Intelligence, LNAI **9067**, 296–309, 2015. DOI 10.1007/978-3-662-48119-6_22.

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

[5] KEGG: *Kyoto Encyclopedia of Genes and Genomes*, `http://www.kegg.jp/`.

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

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

[8] S. Luke, L. Panait: *A survey and comparison of tree generation algorithms*, Proc. GECCO'01, 81–88, 2001.

[9] T. Mori, T. Tamura, D. Fukagawa, A. Takasu, E. Tomita, T. Akutsu: *A clique-based method using dynamic programming for computing edit distance between unordered trees*, J. Comput. Bio. **19**, 1089–1104, 2012. DOI: 10.1089/cmb.2012.0133.

[10] S. Schiermer, R. Giegerich: *Forest alignment with affine gaps and anchors, applied in RNA structure comparison*, Theoret. Comput. Sci. **483**, 51–67, 2013. DOI: 10.1016/j.tcs.2012.07.040.

[11] K.-C. Tai: *The tree-to-tree correction problem*, J. ACM **26**, 422–433, 1979. DOI: 10.1145/322139.322143.

[12] J. T. L. Wang, K. Zhang: *Finding similar consensus between trees: An algorithm and a distance hierarchy*, Pattern Recog. **34**, 127–137, 2001. DOI: 10.1016/S0031-3203(99)00199-5.

[13] Y. Yamamoto, K, Hirata, T. Kuboyama: *Tractable and intractable variations of unordered tree edit distance*, Internat. J. Found. Comput. Sci. **25**, 307–329, 2014. DOI: 10.1142/S0129054114500154.

[14] T. Yoshino, S. Higuchi, K. Hirata: *A dynamic programming A* algorithm for computing unordered tree edit distance*, Proc. IIAI AAI '13, 135–140, 2013. DOI: 10.1109/IIAI-AAI.2013.71.

[15] T. Yoshino, K. Hirata: *Tai mapping hierarchy for rooted labeled trees through common subforest*, Theory of Comput. Sys. **60**, 759–783, 2017. DOI: 10.1007/s00224-016-9705-1.

[16] K. Zhang: *Algorithms for the constrained editing distance between ordered labeled trees and related problems*, Pattern Recog. **28**, 463–474, 1995. DOI: 10.1016/0031-3203(94)00109-Y.

[17] K. Zhang: *A constrained edit distance between unordered labeled trees*, Algorithmica **15**, 205-222, 1996. DOI: 10.1007/BF01975866.

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