Segmental Mapping and Distance for Rooted Labeled Ordered Trees*

Tomohiro Kan[1], Shoichi Higuchi[1], and Kouichi Hirata[2]

[1] Graduage School of Computer Science and Systems Engineering
[2] Department of Artificial Intelligence & Biomedical Informatics R&D Center
Kyushu Institute of Technology Kawazu 680-4, Iizuka 820-8502, Japan
{kan,syou_hig,hirata}@dumbo.ai.kyutech.ac.jp

Abstract. In this paper, as a variation of a Tai mapping between trees, we introduce a *segmental* mapping to preserve the parent-children relationship as possible. Then, we show that the segmental mapping provides a new hierarchy for the classes of Tai mappings in addition to a well-known one. Also we show that the *segmental distance* as the minimum cost of segmental mappings is a metric. Finally, we design the algorithm to compute the segmental distance in quadratic time and space.

1 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. In this paper, we formulate such data as *rooted labeled ordered trees* (*trees*, for short) and then focus on distance measures between trees.

The most famous distance measure between trees is the *edit distance* [5]. The edit distance is formulated as the minimum cost to transform from a tree to another tree by applying *edit operations* of a *substitution*, a *deletion* and an *insertion* to trees. It is known that the edit distance is closely related to the notion of a *Tai mapping* (Tai) [5], which is a one-to-one node correspondence between trees preserving ancestor and sibling relations. The minimum cost of possible mappings coincides with the edit distance [5]. After introducing the edit distance, the time complexity to compute it has been improved as $O(n^3)$ time [2], where n is the maximum number of nodes in given two trees.

While the edit distance is the standard measure for comparing trees, it is too general for several applications. Therefore, more structural sensitive variations of the edit distance such as the *top-down* (or *degree*-1) distance [1,4], the *degree*-2 distance [10], the *accordant* distance [3], the *isolated-subtree* (or *constrained*) distance [7,9] and the *bottom-up* distance [6] are required for these applications. Such variations are formulated as the minimum cost of restricted mappings such

* This work is partially supported by Grand-in-Aid for Scientific Research 22240010, 24240021 and 24300060 from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

K.-M. Chao, T.-s. Hsu, and D.-T. Lee (Eds.): ISAAC 2012, LNCS 7676, pp. 485–494, 2012.

as *top-down* (Top), *degree*-2 (Dg2), *accordant* (Acc), *isolated-subtree* (IsSt) and *bottom-up* (Bot) mappings, respectively, and computed in $O(n^2)$ time[1].

It is known that these mappings provide the hierarchy described in Figure 1 (left) [3,7] as a Hasse diagram. This diagram claims that if $M \in A$ then $M \in B$ for a mapping M, a lower class A and an upper class B in Figure 1 (left).

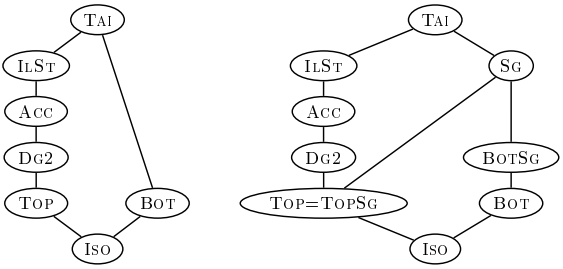

Fig. 1. A mapping hierarchy [3,7] (left) and a new mapping hierarchy (right)

In the above mappings, the parent-children relationship is just preserved by both top-down and bottom-up mappings, which are too restricted. On the other hand, it is sometimes important in several applications such as the function determination of glycan data, the parse trees of programs, the trace patterns of procedure calls and the change detection of XML documents (*cf.*, [1,3,6]).

As the generalization of top-down and bottom-up mappings, we introduce a *segmental mapping* (Sg) preserving the parent-children relationship as possible. The segmental mapping requires that, for every pair of nodes in a mapping, if the mapping contains a pair of the ancestors of the nodes, then it always contains the pair of the parents of the nodes. Also we formulate *top-down* and *bottom-up segmental mappings* (TopSg and BotSg) that are segmental mappings always containing the pair of the roots and the pair of leaves as descendants, respectively.

In this paper, first we show that Sg, TopSg and BotSg provide a new hierarchy in Figure 1 (right). Next, we show that the *segmental distance* and the *bottom-up segmental distance* as the minimum cost of Sg and BotSg are metrics. Finally, we design the algorithm to compute the segmental distance in $O(n^2)$ time and space.

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 $E(T)$, respectively. Also 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 $\emptyset$.

[1] While Valiente [6] has first introduced a bottom-up distance, his distance does not allow the substitution. Then, his distance is an indel distance, which runs in $O(n)$ time, rather than an edit distance, which runs in $O(n^2)$ time. See [8].

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. 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. In this paper, 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 w' such that $w' \leq w$, $u \leq w'$ and $v \leq w'$.

A *leaf* is a node having no children. We denote the set of all leaves in T by $lv(T)$. The *degree* of a node $v \in V(T)$, denoted by $deg(v)$, is the number of children of v. A (*complete*) *subtree of* T *rooted by* v, denoted by $T(v)$, is a tree consisting of v and all of the descendants of v.

We say that a rooted tree is *ordered* if a left-to-right order among siblings is given. For a rooted ordered tree T, a node v in T and its children $v_1, \ldots, v_i$, the *preorder traversal* of $T(v)$ is obtained by visiting v and then recursively visiting $T(v_k)$ $(1 \leq k \leq i)$ in order. Similarly, the *postorder traversal* of $T(v)$ is obtained by first visiting $T(v_k)$ $(1 \leq k \leq i)$ and then visiting v. The *preorder* (*resp.*, *postorder*) *number* of $v \in T$ is the number of nodes preceding v in the preorder (*resp.* postorder) traversal of T and denote it by $pre(v)$ (*resp.*, $post(v)$). The nodes *to the left of* $v \in T$ is the set of nodes $u \in T$ satisfying that (1) $pre(u) \leq pre(v)$ and (2) $post(u) \leq post(v)$. If u is to the left of v, then v is to the *right* of u. We denote that u is to the left of v by $u \preceq v$.

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 ordered tree a tree simply. A(n *ordered*) *forest* is a sequence of trees. We denote a forest consisting of trees $T_1, \ldots, T_m$ by $[T_1, \ldots, T_m]$.

Definition 1 (Edit operations). We define *edit operations* of a tree T as follows. See Figure 2.

1. *Substitution*: Change the label of the node v in T.
2. *Deletion*: Delete a non-root 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'.
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 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* $\gamma : (\Sigma_\varepsilon \times \Sigma_\varepsilon - \{(\varepsilon, \varepsilon)\}) \mapsto \mathbf{R}$ 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 $\gamma(l_1, l_3) \leq \gamma(l_1, l_2) + \gamma(l_2, l_3)$.

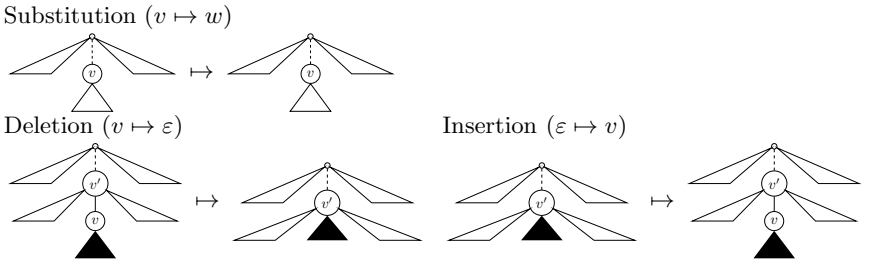

Fig. 2. Edit operations for trees

Definition 2 (Edit distance). 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* $\tau(T_1, T_2)$ between trees T_1 and T_2 is defined as follows:

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

Definition 3 (Mapping). Let T_1 and T_2 be trees. We say that a triple (M, T_1, T_2) (or simply M when there is no confusion) is a *Tai mapping* (a *mapping*, for short) between T_1 and T_2, which we denote by $M \in \textsc{Tai}$, 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 three conditions.

1. $v_1 = v_2$ iff $w_1 = w_2$. 2. $v_1 \leq v_2$ iff $w_1 \leq w_2$. 3. $v_1 \preceq v_2$ iff $w_1 \preceq w_2$.

Let M be a mapping between T_1 and T_2. Let I and J be the sets of nodes in T_1 and T_2 but not in M. Then, the *cost* of M is given as follows.

$$\gamma(M) = \sum_{(v,w) \in M} \gamma(v \mapsto w) + \sum_{v \in I} \gamma(v \mapsto \varepsilon) + \sum_{w \in J} \gamma(\varepsilon \mapsto w).$$

Theorem 1 (Tai [5]). $\tau(T_1, T_2) = \min\{\gamma(M) \mid M \in \textsc{Tai}\}$.

Trees T_1 and T_2 are *isomorphic*, denoted by $T_1 \equiv T_2$, if there exists a mapping M between T_1 and T_2 such that $\gamma(M) = 0$, which we denote by $M \in \textsc{Iso}$.

Definition 4 (Variations). Let T_1 and T_2 be trees and $M \subseteq V(T_1) \times V(T_2)$ a mapping between T_1 and T_2. Also we denote $M - \{(r(T_1), r(T_2))\}$ by M^-.

1. We say that M is an *isolated-subtree mapping* [7] (or a *constrained mapping* [9]), denoted by $M \in \textsc{IlSt}$, if M satisfies the following condition.
$$\forall (v_1, w_1), (v_2, w_2), (v_3, w_3) \in M \Big(v_3 < v_1 \sqcup v_2 \iff w_3 < w_1 \sqcup w_2 \Big).$$
2. We say that M is an *accordant mapping* [3], denoted by $M \in \textsc{Acc}$, if M satisfies the following condition.
$$\forall (v_1, w_1), (v_2, w_2), (v_3, w_3) \in M \left(\begin{array}{l} v_1 \sqcup v_2 = v_1 \sqcup v_3 \\ \iff w_1 \sqcup w_2 = w_1 \sqcup w_3 \end{array} \right).$$

3. We say that M is a *degree-2 mapping* [10], denoted by $M \in \text{DG2}$, if M satisfies the following condition.
$$\forall (v_1, w_1), (v_2, w_2) \in M^{-}\Big((v_1 \sqcup v_2, w_1 \sqcup w_2) \in M\Big).$$
4. We say that M is a *top-down mapping* [1,4], denoted by $M \in \text{TOP}$, if M satisfies the following condition.
$$\forall (v, w) \in M^{-}\Big((par(v), par(w)) \in M\Big).$$
5. We say that M is a *bottom-up mapping* [3,6,8][2], denoted by $M \in \text{BOT}$, if M satisfies the following condition.
$$\forall (v, w) \in M \begin{pmatrix} \forall v' \in T_1(v) \exists w' \in T_2(w)\Big((v', w') \in M\Big) \\ \wedge \forall w' \in T_2(w) \exists v' \in T_1(v)\Big((v', w') \in M\Big) \end{pmatrix}.$$

Also we define the *top-down distance* $\tau_{\top}(T_1, T_2)$ as $\min\{\gamma(M) \mid M \in \text{TOP}\}$.

Example 1. Consider the mappings M_i $(1 \leq i \leq 6)$ in Figure 3. Then, it holds that $M_1 \in \text{TOP}$; $M_2 \notin \text{TOP}$ but $M_2 \in \text{DG2}$; $M_3 \notin \text{DG2}$ but $M_3 \in \text{ACC}$; $M_4 \notin \text{ACC}$ but $M_4 \in \text{ILST}$; $M_5 \notin \text{ILST}$ but $M_5 \in \text{TAI}$. Also it holds that $M_6 \in \text{BOT}$ but $M_6 \notin \text{ILST}$. Furthermore, it holds that $M_i \notin \text{BOT}$ $(1 \leq i \leq 5)$.

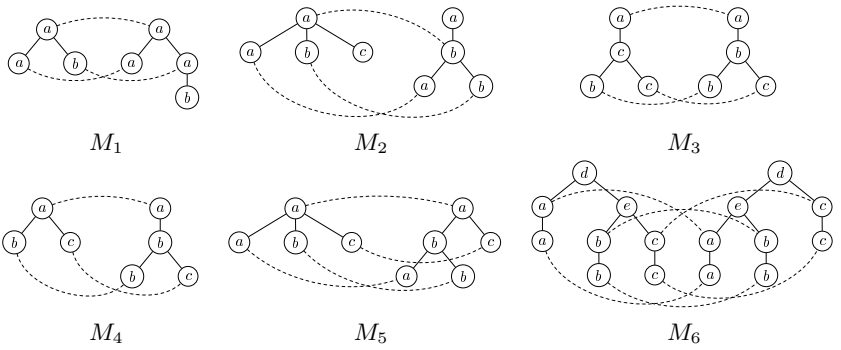

Fig. 3. Mappings M_i $(1 \leq i \leq 6)$ in Example 1

3 Segmental Mapping and Distance

In this section, we introduce a *segmental mapping* and a *segmental distance*.

Definition 5 (Segmental mapping). Let T_1 and T_2 be trees and $M \subseteq V(T_1) \times V(T_2)$ a mapping between T_1 and T_2.

[2] While Valiente [6] has introduced a bottom-up mapping that requires an isolated-subtree mapping, his algorithm computes one that is not an isolated-subtree distance. Then, we adopt the revised definition of a bottom-up mapping [3,8].

1. We say that M is a *segmental* mapping, denoted by $M \in \textsc{Sg}$, if M satisfies the following condition.
$$\forall (v,w) \in M^{-} \left(\begin{array}{l} \Big(\Big((v',w') \in M \Big) \wedge \Big(v' \in anc(v) \Big) \wedge \Big(w' \in anc(w) \Big) \Big) \\ \quad \Longrightarrow (par(v), par(w)) \in M \end{array} \right).$$
2. We say that M is a *top-down segmental* mapping, denoted by $M \in \textsc{TopSg}$, if M is a segmental mapping such that $(r(T_1), r(T_2)) \in M$.
3. We say that M is a *bottom-up segmental* mapping, denoted by $M \in \textsc{BotSg}$, if M is a segmental mapping satisfying the following condition.
$$\forall (v,w) \in M \left(\begin{array}{l} \exists (v',w') \in M \left(\begin{array}{l} \Big(v \in anc(v') \Big) \wedge \Big(w \in anc(w') \Big) \\ \quad \wedge \Big(v' \in lv(T_1) \Big) \wedge \Big(w' \in lv(T_2) \Big) \end{array} \right) \\ \vee \Big(\Big(v \in lv(T_1) \Big) \wedge \Big(w \in lv(T_1) \Big) \Big) \end{array} \right).$$

Example 2. Consider the mappings M_i $(7 \leq i \leq 9)$ in Figure 4. For M_7, it holds that $M_7 \in \textsc{Top}$, $M_7 \in \textsc{TopSg}$, $M_7 \in \textsc{BotSg}$ and $M_7 \in \textsc{Sg}$ but $M_7 \notin \textsc{Bot}$. For M_8, it holds that $M_8 \in \textsc{BotSg}$ and $M_8 \in \textsc{Sg}$ but $M_8 \notin \textsc{Top}$ and $M_8 \notin \textsc{TopSg}$. For M_9, it holds that $M_9 \in \textsc{Sg}$ but $M_9 \notin \textsc{BotSg}$, $M_9 \notin \textsc{TopSg}$ and $M_9 \notin \textsc{Top}$. Also it holds that $M_9 \notin \textsc{IlSt}$. Furthermore, for M_3 and M_6 in Example 1, it holds that $M_3 \in \textsc{IlSt}$ but $M_3 \notin \textsc{Sg}$; $M_6 \in \textsc{BotSg}$ and $M_6 \in \textsc{Sg}$ but $M_6 \notin \textsc{IlSt}$.

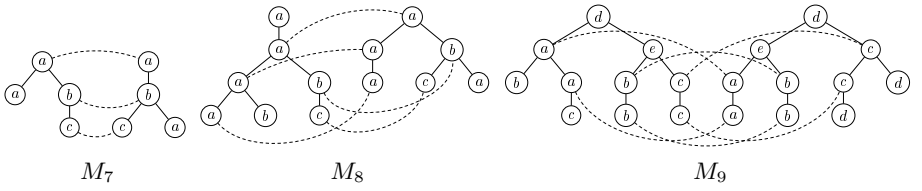

Fig. 4. Mappings M_i $(7 \leq i \leq 9)$ in Example 2

Theorem 2 (Mapping hierarchy). *The mapping hierarchy illustrated in Figure 1 (right) in Section 1 holds. that is:*

1. $\textsc{Top} = \textsc{TopSg} \subset \textsc{Sg} \subset \textsc{Tai}$ *and* $\textsc{Bot} \subset \textsc{BotSg} \subset \textsc{Sg} \subset \textsc{Tai}$.
2. $A \not\subseteq B$ *and* $B \not\subseteq A$ *for* $A \in \{\textsc{BotSg}, \textsc{Sg}\}$ *and* $B \in \{\textsc{Top}, \textsc{Dg2}, \textsc{Acc}, \textsc{IlSt}\}$.

Proof. The formula in Definition 5 implies that $\textsc{Top} = \textsc{TopSg}$. Other inclusion, properness and incomparability follow from Definition 5 and Example 2. □

For segmental mappings M_i $(i = 1, 2)$ between T_i and T_{i+1}, we define the *composition* $M_1 \circ M_2$ as $\{(u,w) \mid \exists v \in T_2 \text{ s.t. } (u,v) \in M_1 \text{ and } (v,w) \in M_2\}$. Then, we can show the following lemma from Definition 5 as similar as [9].

Lemma 1. *1. $M_1 \circ M_2$ is a segmental mapping between T_1 and T_3.*
2. For a cost function γ that is a metric, $\gamma(M_1 \circ M_2) \leq \gamma(M_1) + \gamma(M_2)$.

Definition 6 (Segmental distance). A *segmental distance* $\delta(T_1, T_2)$ and a *bottom-up segmental distance* $\delta_\perp(T_1, T_2)$ between T_1 and T_2 are defined as:

$$\delta(T_1, T_2) = \min\{\gamma(M) \mid M \in \textsc{Sg}\},\ \delta_\perp(T_1, T_2) = \min\{\gamma(M) \mid M \in \textsc{BotSg}\}.$$

Theorem 3. *Both δ and $\delta_\perp$ are metrics.*

Proof. It is sufficient to show the triangle inequality for δ. Let M_1 (*resp.*, M_2) be the minimum cost segmental mapping between T_1 and T_2 (*resp.*, between T_2 and T_3). By Lemma 1, it holds that $\delta(T_1, T_3) \leq \gamma(M_1 \circ M_2) \leq \gamma(M_1) + \gamma(M_2) = \delta(T_1, T_2) + \delta(T_2, T_3)$, so δ is a metric. Similarly, $\delta_\perp$ is also a metric □

4 Computing Segmental Distance

In this section, we identify a node in T_1 (*resp.*, T_2) with its postorder number i ($1 \leq i \leq |T_1|$) (*resp.*, j ($1 \leq j \leq |T_2|$)) of T_1 (*resp.*, T_2), where 0 denotes the empty tree. We denote the postorder number of the leftmost leaf of $T_1(i)$ (*resp.*, $T_2(j)$) by $ll(i)$ (*resp.*, $ll(j)$). Also let $F_1(i)$ (*resp.*, $F_2(j)$) denote the forest obtained by deleting i (*resp.*, j) from $T_1(i)$ (*resp.*, $T_2(j)$). Let $n = \max\{|T_1|, |T_2|\}$.

Let M be a segmental mapping between T_1 and T_2. Then, there exists at least one pair $(i, j) \in M$ such that $(i', j') \notin M$ for every ancestor i' of i in T_1 and every ancestor j' of j in T_2. We call such a pair a *maximal pair* of M and denote the set of all maximal pairs of M by P_M. Also, for every $(i, j) \in P_M$, we can obtain the subset $M_{(i,j)} \subseteq M$ such that $M_{(i,j)} = \{(i', j') \in M \mid i' \in T_1(i), j' \in T_2(j)\}$. We denote the set of nodes that are not descendants of every i (*resp.*, j) such that $(i, j) \in P_M$ by R^1_M (*resp.*, R^2_M). Then, the following equation is straightforward.

$$\gamma(M) = \sum_{(i,j)\in P_M} \gamma(M_{(i,j)}) + \sum_{v \in R^1_M} \gamma(v \mapsto \varepsilon) + \sum_{w \in R^2_M} \gamma(\varepsilon \mapsto w). \tag{1}$$

Lemma 2. *For every $(i, j) \in P_M$, $M_{(i,j)}$ is a top-down mapping between $T_1(i)$ and $T_2(j)$. Hence, it holds that $\gamma(M_{(i,j)}) \geq \tau_\top(T_1(i), T_2(j))$.*

Proof. For $(i', j') \in M_{(i,j)}$, it holds that $i' \leq i$ in T_1 and $j' \leq j$ in T_2. Since $M_{(i,j)}$ is a segmental mapping, there exists a sequence $(i'_1, j'_1), \ldots, (i'_a, j'_a)$ of pairs in $M_{(i,j)}$ such that $i'_1 = i$, $j'_1 = j$, $i'_a = i'$, $j'_a = j'$, $i'_b = par(i'_{b+1})$ and $j'_b = par(j'_{b+1})$ for $1 \leq b \leq a - 1$. This implies that $M_{(i,j)}$ is a top-down mapping. □

Lemma 3. *Let M^* be the minimum cost segmental mapping between T_1 and T_2. Then, the following equation holds.*

$$\delta(T_1, T_2) = \sum_{(i,j)\in P_{M^*}} \tau_\top(T_1(i), T_2(j)) + \sum_{v \in R^1_{M^*}} \gamma(v \mapsto \varepsilon) + \sum_{w \in R^2_{M^*}} \gamma(\varepsilon \mapsto w). \tag{2}$$

Proof. Since $\gamma(M^*) = \delta(T_1, T_2)$ and the minimality of $\gamma(M^*)$ implies that $\gamma(M^*_{(i,j)}) = \tau_\top(T_1(i), T_2(j))$, the equation (2) follows from the equation (1). □

procedure SegDist(T_1, T_2, γ)
  /* T_1, T_2 : trees, γ : cost function */
1 **for** $i = 1$ **to** $|T_1|$ **do**
2 **for** $j = 1$ **to** $|T_2|$ **do**
3 $TD[i, j] \leftarrow$ TopDownPair(i, j, γ);
4 $D[0, 0] \leftarrow 0$;
5 **for** $i = 1$ **to** $|T_1|$ **do**
6 $D[i, 0] \leftarrow D[i - 1, 0] + \gamma(i \mapsto \varepsilon)$;
7 **for** $j = 1$ **to** $|T_2|$ **do**
8 $D[0, j] \leftarrow D[0, j - 1] + \gamma(\varepsilon \mapsto j)$;
9 **for** $i = 1$ **to** $|T_1|$ **do**
10 **for** $j = 1$ **to** $|T_2|$ **do**
11 $D[i, j] \leftarrow \min \left\{ \begin{array}{l} D[i-1, j] + \gamma(i \mapsto \varepsilon),\ D[i, j-1] + \gamma(\varepsilon \mapsto j), \\ D[ll(i) - 1, ll(j) - 1] + TD[i, j] \end{array} \right\}$;
12 **output** $D[|T_1|, |T_2|]$;

procedure TopDownPair(i, j, γ)
  /* $i \in T_1$, $F_1(i) = [T_1(i_1), \ldots, T_1(i_m)]$, where $i_0 = 0$ */
  /* $j \in T_2$, $F_2(j) = [T_2(j_1), \ldots, T_2(j_n)]$, where $j_0 = 0$ */
13 $F[0, 0] \leftarrow 0$;
14 **for** $k = 1$ **to** m **do**
15 $F[i_k, 0] \leftarrow F[i_{k-1}, 0] + |T_1(i_k)| \times \gamma(i_k \mapsto \varepsilon)$;
16 **for** $l = 1$ **to** n **do**
17 $F[0, j_l] \leftarrow F[0, j_{l-1}] + |T_2(j_l)| \times \gamma(\varepsilon \mapsto j_l)$;
18 **for** $k = 1$ **to** m **do**
19 **for** $l = 1$ **to** n **do**
20 $F[i_k, j_l] \leftarrow \min \left\{ \begin{array}{l} F[i_{k-1}, j_l] + |T_1(i_k)| \times \gamma(i_k \mapsto \varepsilon), \\ F[i_k, j_{l-1}] + |T_2(j_l)| \times \gamma(\varepsilon \mapsto j_l), \\ F[i_{k-1}, j_{l-1}] + TD[i_k, j_l] \end{array} \right\}$;
21 **output** $F[i_m, j_n] + \gamma(i \mapsto j)$;

Algorithm 1. SegDist

The equation (2) claims that we can compute the segmental distance $\delta(T_1, T_2)$ by first computing the top-down distance $\tau_\top(T_1(i), T_2(j))$ for every pair $(i, j) \in T_1 \times T_2$ and then combining pairs such that the total cost of a mapping is minimum, which we achieve in $O(n^4)$ time by using a naive method [1,4]. In this paper, we design an $O(n^2)$ time algorithm SegDist in Algorithm 1.

Lemma 4. *For $i \in T_1$ and $j \in T_2$, the algorithm* TopDownPair(i, j, γ) *computes the top-down distance $\tau_\top(T_1(i), T_2(j))$ in $O(deg(i) \times deg(j))$ time.*

Proof. Let $i_1, \ldots, i_m$ be the children of i in T_1 and $j_1, \ldots, j_n$ the children of j in T_2, that is, let $F_1(i) = [T_1(i_1), \ldots, T_1(i_m)]$ and $F_2(j) = [T_2(j_1), \ldots, T_2(j_n)]$. Also let $I = \{i_1, \ldots, i_m\}$ and $J = \{j_1, \ldots, j_n\}$. Furthermore, since the for-loop of lines 1 and 2 in SegDist executes in postorder traversal, we can suppose

that $TD[i_a, j_b](= \tau_\top(T_1(i_a), T_2(j_b)))$ has been already computed for $1 \leq i_a < i$ $(1 \leq a \leq m)$ and $1 \leq j_b < j$ $(1 \leq b \leq n)$ when computing $\tau_\top(T_1(i), T_2(j))$.

Since $\gamma(i \mapsto j)$ in line 21 is the cost of the pair (i, j), which is contained from every top-down mapping between $T_1(i)$ and $T_2(j)$, we can obtain the top-down distance $\tau_\top(T_1(i), T_2(j))$ by adding $\gamma(i \mapsto j)$ to the combination of I and J providing the minimum cost. As the same discussion of [9], we can regard such a combination as the string edit distance between $i_1 \cdots i_m$ and $j_1 \cdots j_n$ under the cost function c such that $c(i_a, \varepsilon) = |T_1(i_a)| \times \gamma(i_a \mapsto \varepsilon) = \tau_\top(T_1(i_a), \emptyset)$, $c(\varepsilon, j_b) = |T_2(j_b)| \times \gamma(\varepsilon \mapsto j_b) = \tau_\top(\emptyset, T_2(j_a))$ and $c(i_a, j_b) = \tau_\top(T_1(i_a), T_2(j_b))$ for $1 \leq a \leq m$ and $1 \leq b \leq n$, each of which is a formula in line 20. It is obvious that the algorithm TopDownPair(i, j, γ) runs in $O(deg(i) \times deg(j))$ time. □

Theorem 4. *The algorithm* SegDist *computes the segmental distance* $\delta(T_1, T_2)$ *between* T_1 *and* T_2 *in* $O(n^2)$ *time and space.*

Proof. Let $F_1[i]$ (*resp.*, $F_2[j]$) be the forest of T_1 (*resp.*, T_2) constructing the nodes from 1 to i (*resp.*, from 1 to j) in postorder of T_1 (*resp.*, T_2). By the definition of ll, $ll(i)-1$ and $ll(j)-1$ are the left siblings of i in $F_1[i]$ and j in $F_2[j]$, that is, $F_1[i] = [\ldots, T_1(ll(i)-1), T_1(i)]$ and $F_2[j] = [\ldots, T_2(ll(j)-1), T_2(j)]$.

Suppose that $D[k, l]$ is the segmental distance between $F_1[k]$ and $F_2[l]$ for $1 \leq k \leq i$ and $1 \leq l \leq j$, and consider the segmental distance between $F_1[i]$ and $F_2[j]$. If j is inserted, then $D[i, j]$ is the sum of the segmental distance $D[i, j-1]$ between $F_1[i]$ and $F_2[j-1]$ and the cost $\gamma(\varepsilon \mapsto j)$ of the insertion of j. If i is deleted, then $D[i, j]$ is the sum of the segmental distance $D[i-1, j]$ between $F_1[i-1]$ and $F_2[j]$ and the cost $\gamma(i \mapsto \varepsilon)$ of the deletion of i. If i is substituted to j, then, by Lemma 3, $D[i, j]$ is the sum of the segmental distance $D[ll(i)-1, ll(j)-1]$ between $F_1[ll(i)-1]$ and $F_2[ll(j)-1]$ and the top-down distance $TD[i, j]$ between $T_1(i)$ and $T_2(j)$. Hence, $\delta(T_1, T_2)$ is given as $D[|T_1|, |T_2|]$.

The algorithm SegDist uses $O(|T_1| \times |T_2|)$ space. Also, by Lemma 4, the time complexity of the algorithm SegDist is given as $\sum_{i=1}^{|T_1|} \sum_{j=1}^{|T_2|} O(deg(i) \times deg(j)) + O(|T_1|) + O(|T_2|) + O(|T_1| \times |T_2|) \leq O\left(\sum_{i=1}^{|T_1|} deg(i) \times \sum_{j=1}^{|T_2|} deg(j)\right) + O(|T_1| \times |T_2|) \leq O(|T_1| \times |T_2|)$. □

Furthermore, we can design the algorithm to compute the bottom-up segmental distance $\delta_\perp(T_1, T_2)$ in $O(n^2)$ time and space, by adding the routine of determining that a current top-down mapping contains a pair of leaves when the third statement of $F[i_{k-1}, j_{l-1}] + TD[i_k, j_l]$ in line 20 is executed to SegDist.

Figure 5 illustrates distributions and the correlation diagrams to the edit distance τ of segmental, top-down and bottom-up distances for N-glycan data provided from KEGG[3]. Hence, the segmental distance preserves the parent-children relationship more than the top-down and the bottom-up distances nearer to τ.

[3] Kyoto Encyclopedia of Genes and Genomes, `http://www.kegg.jp/`. The number of N-glycan data is 2142, the average number of nodes is 11.09, the average number of labels is 5.43 and the average depth and degree are 5.38 and 2.07, respectively.

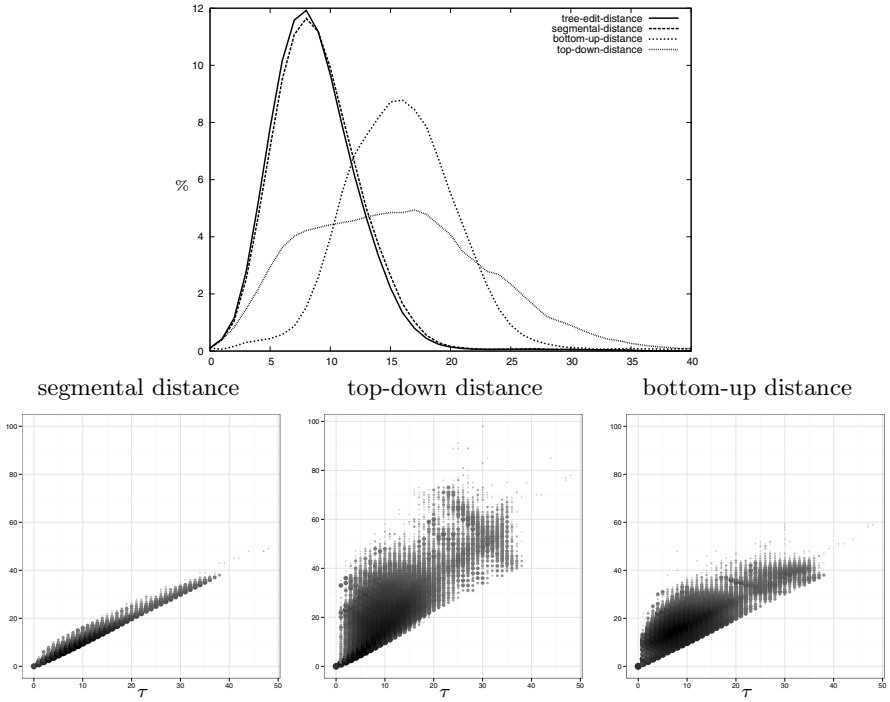

Fig. 5. The distributions (upper) and the correlation diagrams (lower) of segmental, top-down and bottom-up distances to an edit distance τ for N-glycan data

References

1. Chawathe, S.S.: Comparing hierarchical data in external memory. In: Proc. VLDB 1999, pp. 90–101 (1999)
2. Demaine, E.D., Mozes, S., Rossman, B., Weimann, O.: An optimal decomposition algorithm for tree edit distance. ACM Trans. Algorithms 6 (2009)
3. Kuboyama, T.: Matching and learning in trees. Ph.D thesis, University of Tokyo (2007), `http://tk.cc.gakushuin.ac.jp/doc/kuboyama2007phd.pdf`
4. Selkow, S.M.: The tree-to-tree editing problem. Inform. Process. Lett. 6, 184–186 (1977)
5. Tai, K.-C.: The tree-to-tree correction problem. J. ACM 26, 422–433 (1979)
6. Valiente, G.: An efficient bottom-up distance between trees. In: Proc. SPIRE 2001, pp. 212–219 (2001)
7. Wang, J.T.L., Zhang, K.: Finding similar consensus between trees: An algorithm and a distance hierarchy. Pattern Recog. 34, 127–137 (2001)
8. Yamamoto, Y., Hirata, K., Kuboyama, T.: A bottom-up edit distance between rooted labeled trees. In: Proc. LLLL 2011, pp. 26–33 (2011)
9. Zhang, K.: Algorithms for the constrained editing distance between ordered labeled trees and related problems. Pattern Recog. 28, 463–474 (1995)
10. Zhang, K., Wang, J.T.L., Shasha, D.: On the editing distance between undirected acyclic graph. Int. J. Found. Comput. Sci. 7, 43–58 (1995)